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 «Сеть для командной работы» c применением шифрования и маскирования

Выполнил студент Костюкова Анна Олеговна

(Ф.И.О.)

Руководитель проекта ассистент Нистюк О.А.

(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)

Консультант: ассистент Нистюк О.А

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

Нормоконтролер: ассистент Нистюк О.А

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

Курсовой проект защищен с оценкой

Минск 2020

Реферат

Пояснительная записка 59 страниц, 29 рисунков, 14 литературных источников, 3 приложения.

Реализация технологии шифрование и маскирование для базы данных приложения «Сеть для командной работы».

Целью проекта является разработка базы для программного средства «Сеть для командной работы».

В первом разделе была обоснована задача курсового проекта и приведен обзор прототипов.

Во втором разделе приводится разработка базы данных.

В третьем разделе приводиться описание разработки необходимых объектов базы данных.

В четвёртом разделе описываются процедуры экспорта и импорта.

В пятом разделе приводится тестирование производительности.

В шестом разделе приводится описание технологии.

В заключении приведены результаты проделанной работы.

**Abstract**

Explanatory note 59 pages, 29 drawings, 14 literary sources, 3 appendices.

Implementation of encryption and masking technology for the database of the "network for team work" application.

The goal of the project is to develop a database for the "network for teamwork" software tool.

The first section explains the task of the course project and provides an overview of the prototypes.

The second section describes the development of the database.

The third section describes how to develop the necessary database objects.

The fourth section describes the export and import procedures.

The fifth section provides performance testing.

In the sixth section provides a description of the technology.

In conclusion, the results of the work done are presented.

# Содержание

[Содержание 2](#_Toc58794436)

[Введение 4](#_Toc58794437)

[1 Постановка задачи 5](#_Toc58794438)

[1.1 Обзор прототипов 5](#_Toc58794439)

[2 Разработка модели базы данных 9](#_Toc58794440)

[3 Разработка необходимых объектов 10](#_Toc58794441)

[3.1 Таблицы 10](#_Toc58794442)

[3.2 Процедуры 14](#_Toc58794443)

[3.3 Профиль безопасности 15](#_Toc58794444)

[3.4 Пользователи 16](#_Toc58794445)

[3.5 Триггеры 17](#_Toc58794446)

[3.6 Пакеты 17](#_Toc58794447)

[3.7 Представления 18](#_Toc58794448)

[4 Описание процедур импорта и экспорта 19](#_Toc58794449)

[5 Тестирование производительности 21](#_Toc58794450)

[6 Описание технологии 22](#_Toc58794451)

[6.1 Технология «Шифрование» 22](#_Toc58794452)

[6.2 Применение технологии «Шифрования» 24](#_Toc58794453)

[6.3 Технология «Маскирование» 25](#_Toc58794454)

[6.4 Применение технологии «Маскирование» 27](#_Toc58794455)

[Заключение 29](#_Toc58794456)

[Список используемых источников 30](#_Toc58794457)

[Приложение А 31](#_Toc58794458)

[Приложение Б 34](#_Toc58794459)

[Приложение В 55](#_Toc58794460)

Введение

На сегодняшний день цифровые технологии сильно изменили жизнь человека. Появилась возможность общаться из любой точки мира, обмениваться информацией и многое другое. Существует много инструментов для осуществления взаимодействия между людьми, в том числе и инструмент для эффективной работы предприятия. Такой инструмент называется системой управления проектами, который является совокупностью технических и организационных методов, поддерживающих управление проектами и повышающий эффективность их реализации.

Главной целью данной курсовой работы является разработка реляционной базы данных для программного обеспечения «Сеть для командной работы». База данных должна содержать необходимые данные о пользователе, командах, задачах, а также хранить сообщения, которыми обмениваются пользователи при выполнении поставленных целей.

В качестве СУБД для базы данных была выбрана Oracle Database 12c, в связи с ее производительностью и надежностью.

Для корректного выполнения операций над данными, с которыми работает приложение, необходимо чётко и правильно составить реляционную структуру. Задачи курсового проекта:

* Определить необходимости сущности;
* Создать таблицы и установить связи между ними;
* Реализовать процедуры для основных пользовательских действий с данными таких, как создание, удаление, редактирование и чтение;
* Применить технологию «Шифрование и маскирование»;
* Реализовать экспорт и импорт в определенном формате;
* Добиться приемлемой скорости работы базы данных.

1. Постановка задачи

Основной задачей курсового проекта является разработка базы данных для программного обеспечения «Сеть для командной работы». Главная идея этой сети заключается в том, что пользователи объединяются в команды для решения поставленных задах, могут обсуждать в ходе их выполнения, а также могут легко отследить, в каком темпе идёт работа.

Задачи курсового проекта:

* Определить необходимости сущности;
* Создать таблицы и установить связи между ними;
* Реализовать процедуры для основных пользовательских действий с данными таких, как создание, удаление, редактирование и чтение;
* Применить технологию «Шифрование и маскирование»;
* Реализовать экспорт и импорт в определенном формате;
* Добиться приемлемой скорости работы базы данных.

Функционально должны быть выполнены следующие задачи:

* Возможность авторизации по ролям;
* Возможность создавать, удалять и редактировать команду;
* Возможность добавлять и удалять пользователя в команду;
* Возможность прикреплять и удалять в задаче фотографии, видеоматериалы;
* Возможность осуществлять поиск пользователей по имени;
* Возможность раздавать и удалять задания для пользователей в команде;
* Возможность удалять и редактировать комментарии.
  1. Обзор прототипов

Trello – это сервис для командной работы, который позволяет планировать и публиковать текущие задачи, систематизировать их и следить за исполнением. Основан на японской системе досок канбан – удобного инструмента для организации работы.

Для начала работы перейти по адресу https://trello.com и авторизоваться (рисунок 1.1). Это можно сделать через существующий аккаунт или другим предложенным способом.

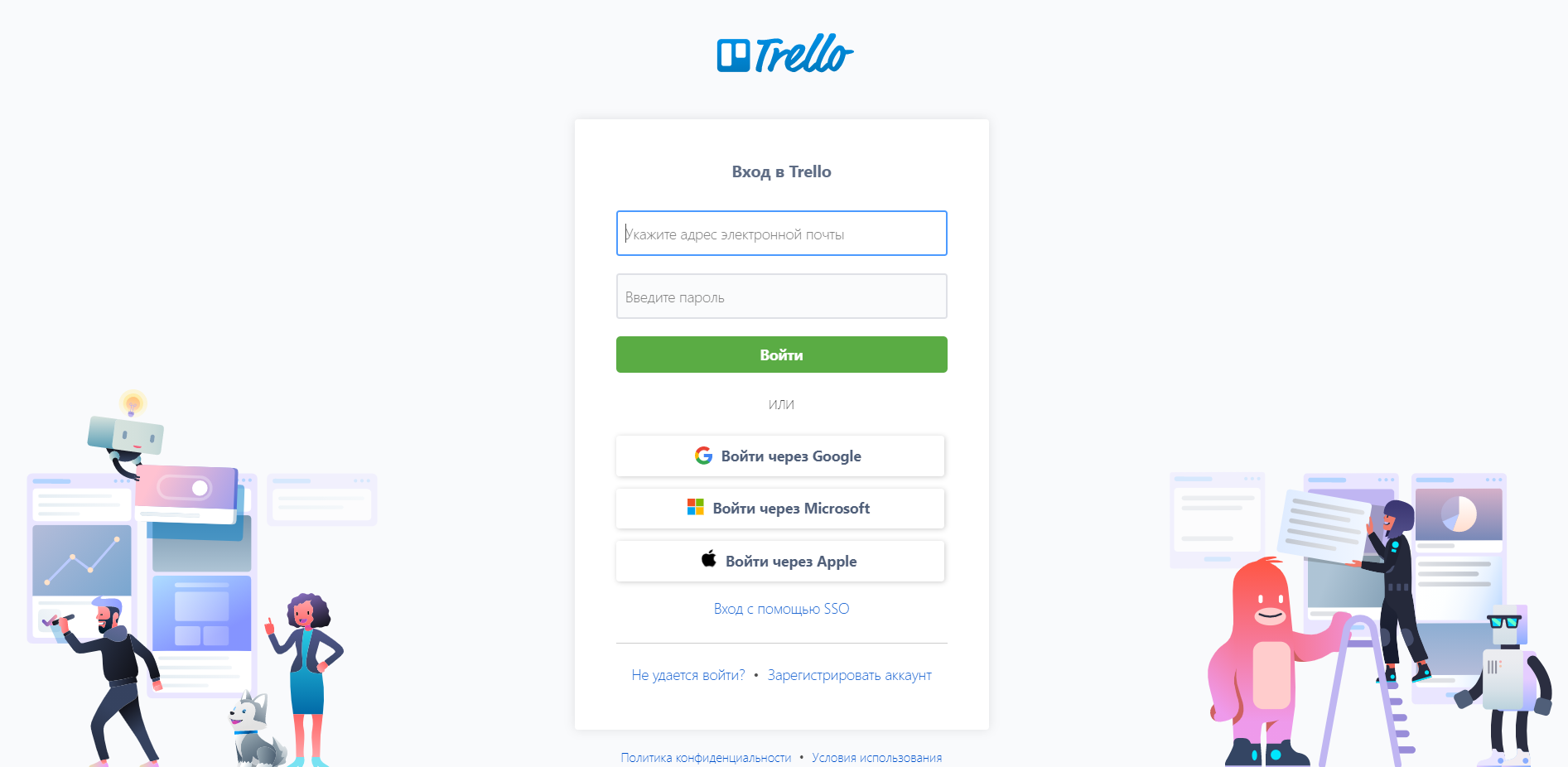


Рисунок 1.1 – Авторизация Trello

После авторизации открывается окно с персональными досками (рисунок 1.2).

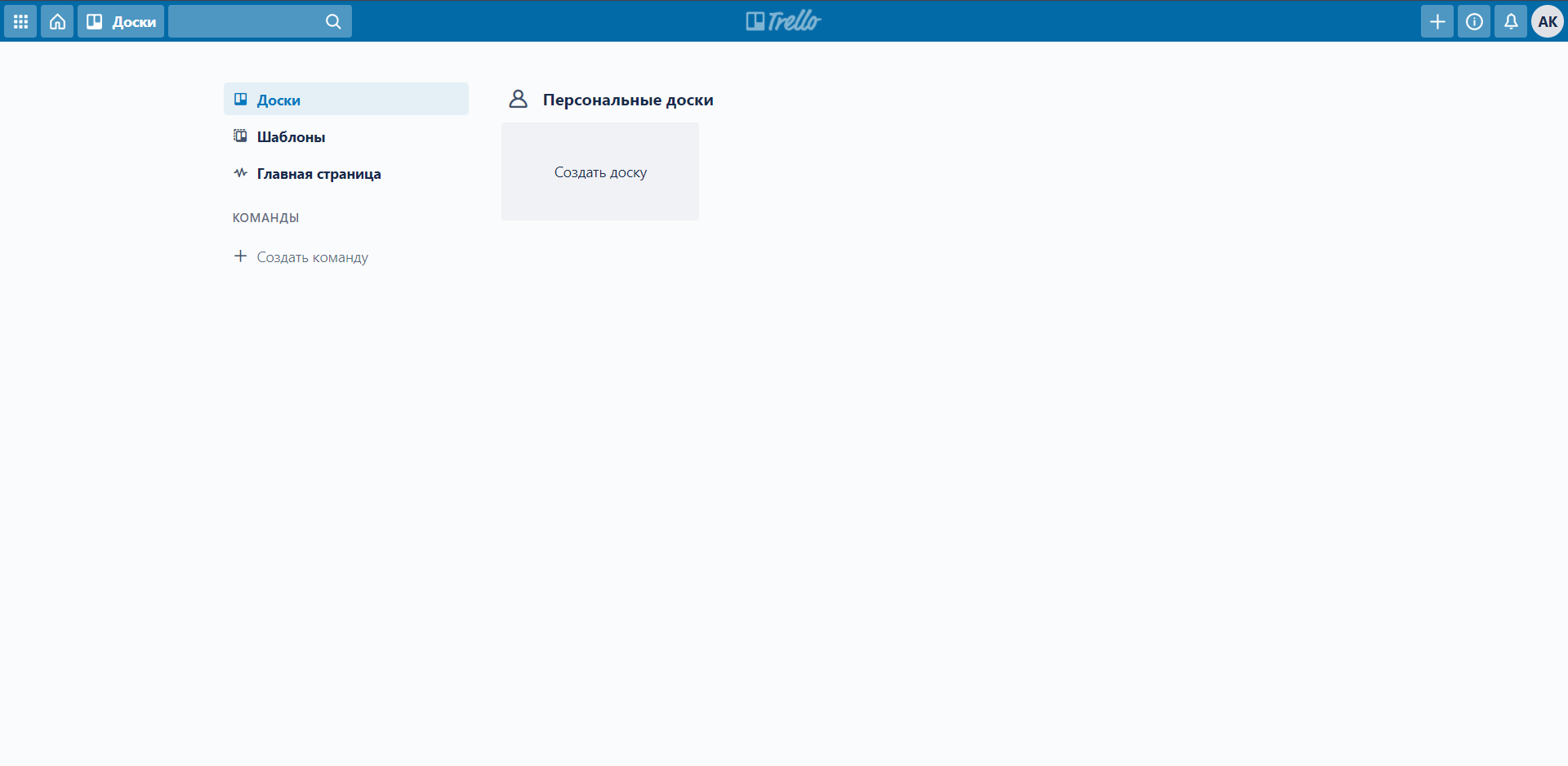


Рисунок 1.2 – Персональные доски

Весь сервис состоит из досок. Как правило, одна доска посвящена одному проекту (если в сервисе работает команда) или работе одного отдела, если речь идет о компании.

Здесь можно создать доску, для этого необходимо всего лишь ввести название доски, выбрать фон, а также выбрать тип доступа: приватная, доступная только для добавленных участников, или публичная, доступная для просмотра всем в Интернете (рисунок 1.3).

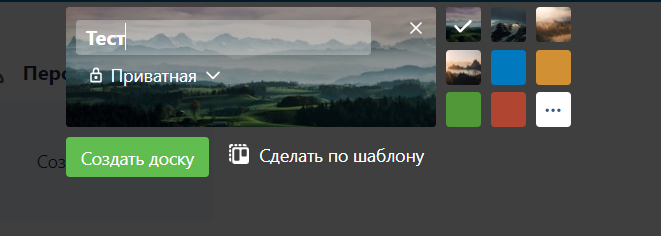


Рисунок 1.3 – Создание доски

После создание доски доступны несколько столбцов – это те самые задачи, которые выполняются в процессе работы (рисунок 1.4).

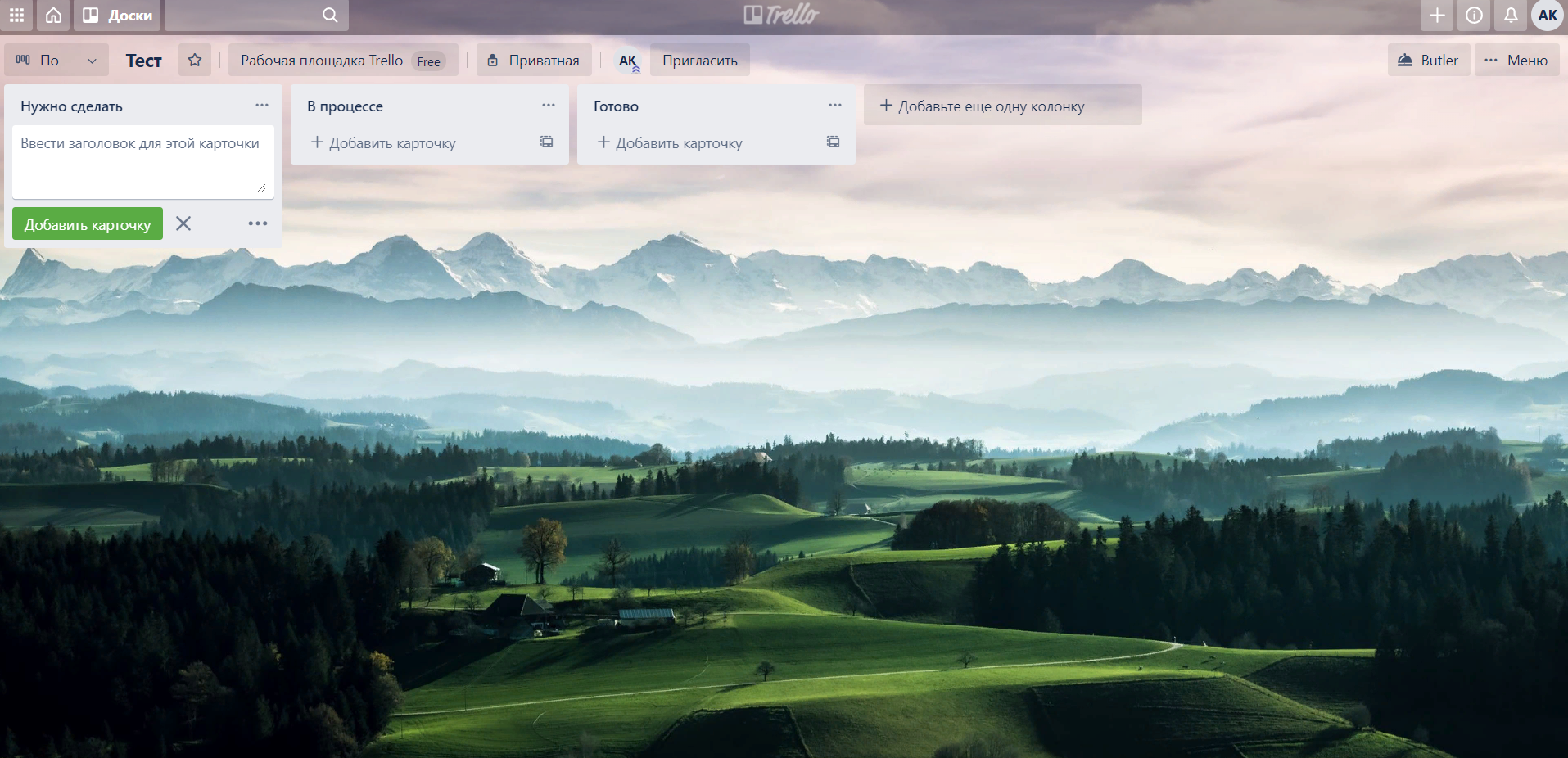


Рисунок 1.4 – Задачи

Каждая команда сама решает, какие списки задач и в каком порядке будут располагаться. Как правило, задачи разбиваются на:

* Запланированные;
* Текущие;
* Выполненные.

Карточки можно двигать как внутри одного списка, так и свободно перемещать между списками или досками. Списки тоже можно перемещать. Для любой задачи можно назначить людей, ответственных за ее выполнение. Trello предлагает множество полезных возможностей для оформления, настройки и управления своими функциональными элементами.

Что можно сделать с карточкой в Trello:

* Переименовать, заполнить описанием и редактировать текст с помощью простейших тегов Markdown;
* Присвоить метки, участников, срок выполнения, добавить файл или чек-лист;
* Добавить комментарии, смайлы, вложения, другие задачи, оповестить выбранных участников;
* Изменить положение блока в списке, перемещать его по спискам и другим доскам;
* Скопировать, следить за изменениями, заархивировать;
* Распечатать, экспортировать в JSON, поделиться ссылкой на карточку или ее почтовым адресом (письма будут появляться в виде комментариев);
* Удалить навсегда.

Списки тоже можно копировать, перемещать и архивировать. Меню с досками в Trello можно сделать фиксированным, а сами доски добавлять в «Избранные» и сортировать.

1. Разработка модели базы данных

Одним из важнейших аспектов при создании базы данных является грамотный анализ сферы применения. В результате создается модель данных, которая правильно отражает, как взаимодействовать с этими данными в целом, особенно с этой моделью. Данная модель предоставляет все возможности курсового проекта такие, как:

* Регистрация, авторизация, редактирование и удаление пользователей;
* Добавление, редактирование и удаление задач;
* Добавление, редактирование и удаление команд;
* Прикрепление фотографии и видеоматериалы к задаче;
* Назначение задачи на конкретных пользователей;
* Добавление пользователей в команду;
* Поиск пользователя по имени.

На рисунке 2.1 представлена диаграмма связей таблиц для необходимой базы данных.

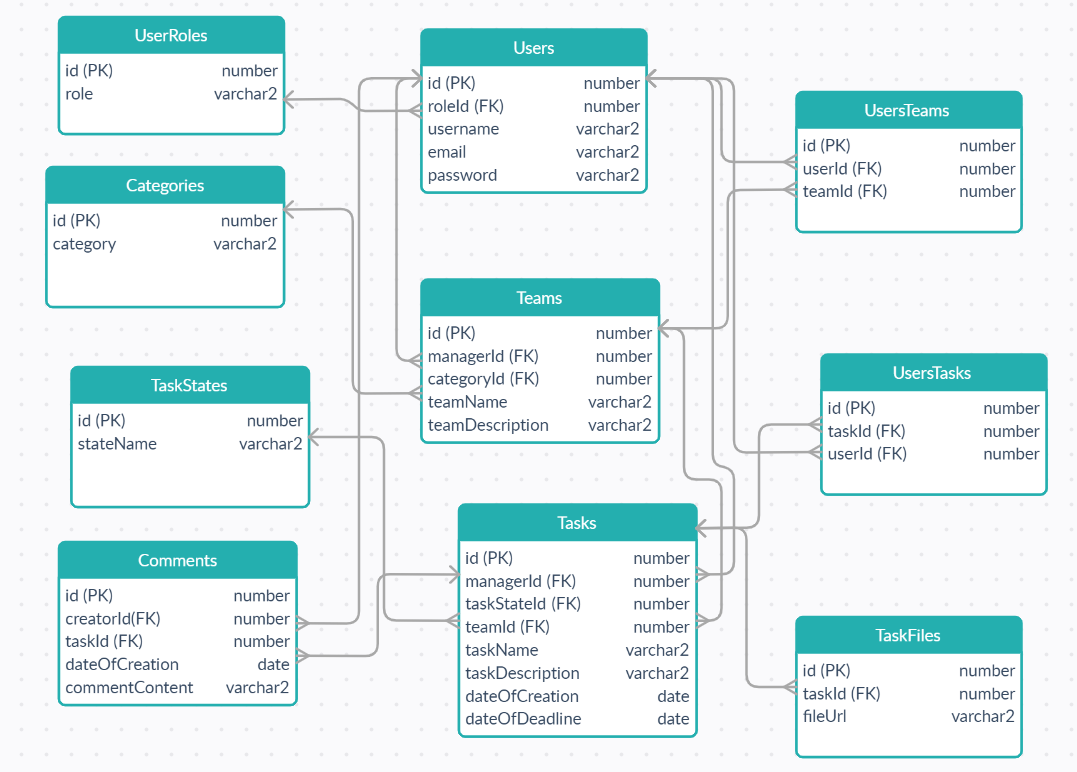


Рисунок 2.1 – Диаграмма базы данных

1. Разработка необходимых объектов

При разработке курсового проекта понадобились следующие объекты: таблицы, хранимые процедуры, профиль безопасности, пользователи, триггеры, пакеты, представления.

* 1. Таблицы

Таблицы – это основные единицы хранилища в базе данных Oracle. Таблица – это логическая сущность, которая делает чтение и манипуляции данных интуитивно понятными для пользователя, ограничивает и упорядочивает хранимую информацию, а также обеспечивают связанность за счет внешних ключей [1].

Для реализации базы данных для программного обеспечения «Сеть для командной работы» было разработано 10 таблиц.

Таблица UserRoles содержит возможные роли пользователей и включает 2 столбца:

* Столбец id является первичным ключом, тип данных number, инкрементируется при добавлении роли;
* Столбец role хранит роль пользователя, тип данных varchar2, максимальный размер 100 символов, уникальный, ненулевой.

Скрипт создания таблицы UserRoles изображен на рисунке 3.1.

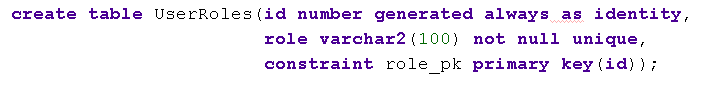


Рисунок 3.1 – Создание таблицы UserRoles

Таблица Users содержит информацию пользователей и включает 4 столбца:

* Столбец id является первичным ключом, тип данных number, инкрементируется при добавлении пользователя;
* Столбец username хранит имя пользователя, тип данных varchar2, максимальный размер 100 символов, уникальный, ненулевой;
* Столбец email хранит почту пользователя, тип данных varchar2, максимальный размер 320 символов, ненулевой;
* Столбец password хранит пароль в зашифрованном виде, тип данных varchar2, максимальный размер 100 символов, ненулевой;
* Столбец roleId является внешним ключом, идентификатор роли, тип данных number, ненулевой.

Скрипт создания таблицы Users изображен на рисунке 3.2.

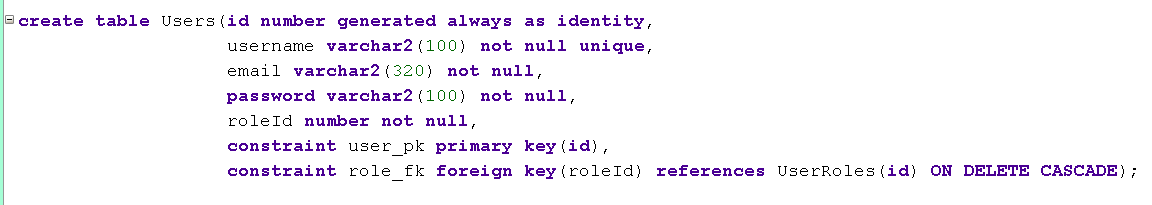


Рисунок 3.2 – Создание таблицы Users

Таблица Categories содержит категории команд и включает 2 столбца:

* Столбец id является первичным ключом, тип данных number, инкрементируется при добавлении категории;
* Столбец category хранит категорию команды, тип данных varchar2, максимальный размер 100 символов, уникальный, ненулевой.

Скрипт создания таблицы Categories изображен на рисунке 3.3.

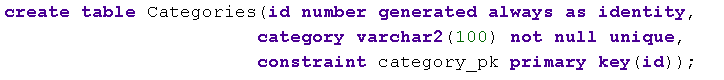


Рисунок 3.3 – Создание таблицы Categories

Таблица Teams содержит команды и включает 5 столбцов:

* Столбец id является первичным ключом, тип данных number, инкрементируется при добавлении команды;
* Столбец teamName хранит название команды, тип данных varchar2, максимальный размер 100 символов, уникальный, ненулевой;
* Столбец teamDescription хранит описание команды, тип данных varchar2, максимальный размер 1000 символов, необязателен для заполнения;
* Столбец managerId является внешним ключом, идентификатор пользователя, который создал команду, тип данных number, ненулевой;
* Столбец categoryId является внешним ключом, идентификатор категории команды, тип данных number, ненулевой.

Скрипт создания таблицы Teams изображен на рисунке 3.4.

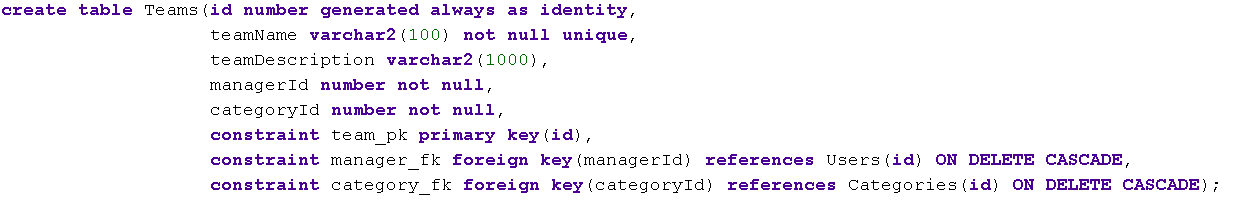


Рисунок 3.4 – Создание таблицы Teams

Таблица TaskStates содержит состояния задачи и включает 2 столбца:

* Столбец id является первичным ключом, тип данных number, инкрементируется при добавлении состояния задачи;
* Столбец stateName хранит состояние задачи, тип данных varchar2, максимальный размер 100 символов, уникальный, ненулевой.

Скрипт создания таблицы TaskStates изображен на рисунке 3.5.

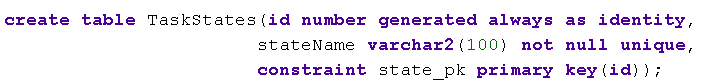


Рисунок 3.5 – Создание таблицы TaskStates

Таблица Tasks содержит команды и включает 7 столбцов:

* Столбец id является первичным ключом, тип данных number, инкрементируется при добавлении задачи;
* Столбец taskName хранит название задачи, тип данных varchar2, максимальный размер 100 символов, ненулевой;
* Столбец taskDescription хранит описание задачи, тип данных varchar2, максимальный размер 1000 символов, необязателен для заполнения;
* Столбец dateOfCreation хранит дату создания задачи, тип данных date, ненулевой;
* Столбец dateOfDeadline хранит дату завершения задачи, тип данных date, необязателен для заполнения;
* Столбец managerId хранит идентификатор пользователя, который создал задачу, тип данных number, ненулевой;
* Столбец teamId является внешним ключом, идентификатор команды, в которую добавлена задача, тип данных number, ненулевой;
* Столбец taskStateId является внешним ключом, идентификатор состояния задачи, тип данных number, ненулевой.

Скрипт создания таблицы Tasks изображен на рисунке 3.6.

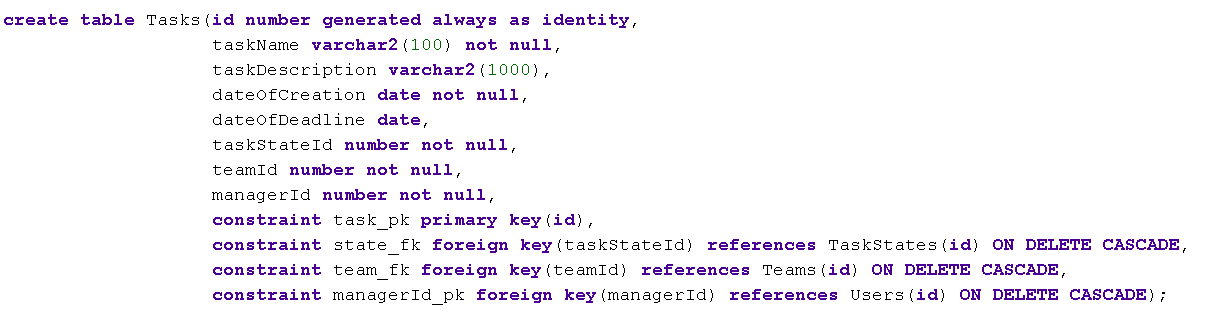


Рисунок 3.6 – Создание таблицы Tasks

Таблица Comments хранит комментарии к задаче и включает 5 столбцов:

* Столбец id является первичным ключом, тип данных number, инкрементируется при добавлении комментария;
* Столбец creatorId является внешним ключом, идентификатор пользователя, который добавил комментарий, тип данных number, ненулевой;
* Столбец dateOfCreation хранит дату добавления комментария, тип данных date, ненулевой;
* Столбец commentContent хранит содержимое комментария, тип данных varchar2, максимальный размер 4000 символов, ненулевой;
* Столбец taskId является внешним ключом, идентификатор задачи, к которой добавлен комментарий, тип данных number, ненулевой.

Скрипт создания таблицы Comments изображен на рисунке 3.7.

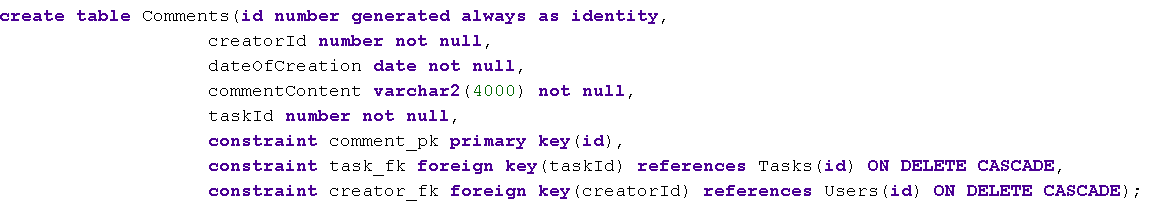


Рисунок 3.7 – Создание таблицы Comments

Таблица TaskFiles содержит прикрепленные фотографии или видеоматериалы к задаче и включает 3 столбца:

* Столбец id является первичным ключом, тип данных number, инкрементируется при добавлении документа;
* Столбец fileUrl хранит ссылку на документ, прикрепленный к задаче, тип данных varchar2, максимальный размер 4000 символов, ненулевой;
* Столбец taskId является внешним ключом, идентификатор задачи, к которой прикреплен документ, тип данных number, ненулевой.

Скрипт создания таблицы TaskFiles изображен на рисунке 3.8.

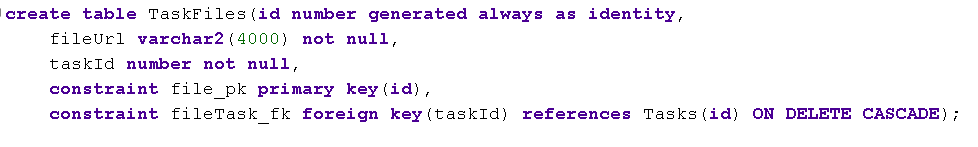


Рисунок 3.8 – Создание таблицы TaskFiles

Таблица UsersTasks является реализацией связующей таблицы между пользователями и их задачами (связь многие ко многим) и включает 3 столбца:

* Столбец id является первичным ключом, тип данных number, инкрементируется при добавлении пользователя в задачу;
* Столбец userId является внешним ключом, хранит идентификатор пользователя, который записан в задачу, тип данных number, ненулевой;
* Столбец taskId является внешним ключам, хранит идентификатор задачи, тип данных number, ненулевой.

Скрипт создания таблицы UsersTasks изображен на рисунке 3.9.

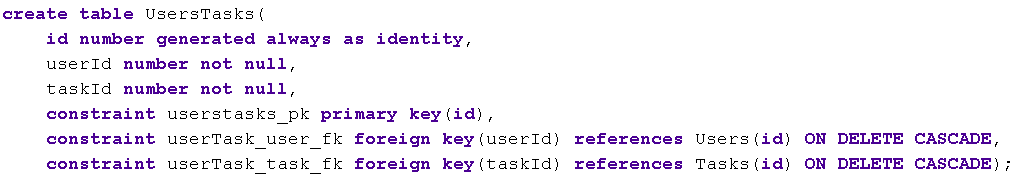


Рисунок 3.9 – Создание таблицы UsersTasks

Таблица UsersTeams является реализацией связующей таблицы между пользователями и их командами (связь многие ко многим) и включает 3 столбца:

* Столбец id является первичным ключом, тип данных number, инкрементируется при добавлении пользователя в команду;
* Столбец userId является внешним ключом, хранит идентификатор пользователя, который записан в команду, тип данных number, ненулевой;
* Столбец teamId является внешним ключам, хранит идентификатор команды, тип данных number, ненулевой.

Скрипт создания таблицы UsersTeams изображен на рисунке 3.10.

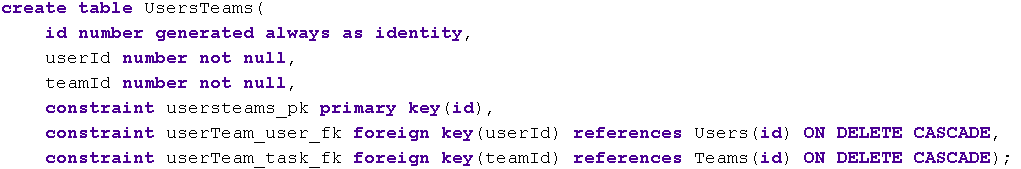


Рисунок 3.10 – Создание таблицы UsersTeams

* 1. Процедуры

Процедура представляет собой модуль, выполняющий одно или несколько действий. Поскольку вызов процедуры в PL/SQL является отдельным исполняемым оператором, блок кода PL/SQL может состоять только из вызова процедуры. Процедуры относятся к числу ключевых компонентов модульного кода, обеспечивающих оптимизацию и повторное использование программной логики [2].

Для реализации базы данных для программного обеспечения «Сеть для командной работы» было разработано 46 процедур, которые реализуют функции CRUD [3].

Процедуры, для манипуляции данными пользователей, находятся в файле UserService.sql:

* CheckUser – проверка на наличие пользователя в базе данных;
* AuthorizeUser – авторизация пользователя;
* encryptPassword – шифрование пароля пользователя;
* RegistrationUser – регистрация пользователя;
* ChangeUserName – изменение имени пользователя;
* ChangePassword – изменение пароля пользователя;
* DeleteUser – удаление пользователя;
* SearchUserByName – поиск пользователя по его имени.

Процедуры, для манипуляции данными команд, находятся в файле TeamService.sql:

* CreateTeam – создание команды и автоматическое добавление пользователя, который создает, в команду;
* DeleteTeam – удаление одной команды;
* DeleteAllTeams – удаление всех команд пользователя, который их создавал;
* UpdateTeam – изменение информации о команде;
* AddUserInTeam – добавить других пользователей в команду;
* DeleteUserFromTeam – удалить пользователя из команды.

Процедуры, для манипуляции данными задач, находятся в файле TaskService.sql:

* CreateTask – создание задача и установка состояния «В процессе» по умолчанию;
* CompletedTask – изменение состояния задачи на «Выполнено»;
* CancelledTask – изменение состояния задачи на «Отменено»;
* InProgressTask – изменение состояния задачи на «В процессе»;
* DeleteTask –удаление задачи;
* DeleteTasksInTeam – удаление всех задач в команде;
* AddUserInTask – добавление пользователя в задачу;
* DeleteUserFromTask – удаление пользователя из задачи;
* UpdateTask – изменение информации о задаче;
* AddFilesToTask – добавление фотографии или видеоматериала к задаче;
* DeleteFilesFromTask – удаление фотографии или видеоматериала в задаче.

Процедуры, для манипуляции данными комментария, находятся в файле CommentService.sql:

* AddComment – создание комментария;
* DeleteComment – удаления комментария;
* DeleteAllComments – удаление всех комментариев пользователя;
* DeleteCommentsInTask – удаление всех комментариев к задаче;
* UpdateComment – изменение комментария.
  1. Профиль безопасности

Профиль – это коллекция атрибутов, связанных с использованием ресурсов и паролей, которая может быть назначена пользователю. Несколько пользователей могут разделять один и тот же профиль, и в базе данных Oracle может существовать неограниченное количество профилей. Профили накладывают жесткие ограничения на потребление ресурсов различными пользователями в базе данных и помогают ограничивать число одновременно отрытых пользователем сеансов, их продолжительность и использование ЦП и других ресурсов [4].

Скрипт создания профиля безопасности представлен на рисунке 3.11.

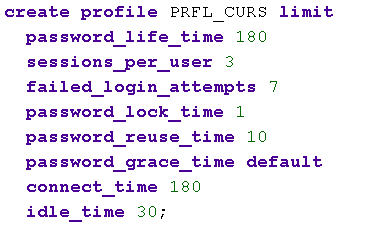


Рисунок 3.11 – Создание профиля безопасности

* 1. Пользователи

Пользователь БД – это физическое или юридическое лицо, которое имеет доступ к БД и пользуется услугами информационной системы для получения информации. Это могут быть случайные пользователи, обращающиеся к базе данных время от времени, за получение некоторой информации, а могут быть регулярные пользователи. На каждом этапе развития БД (проектирование, реализация, эксплуатация, модернизация и развитие, полная реорганизация) с ней связаны разные категории пользователей [5].

В данном курсовом проекте созданы 3 пользователя: администратор базы данных, администратор приложения и пользователь приложения. Администратор базы данных имеет все привилегии. Администратор приложения имеет доступ ко всем процедурам и представлениям, может просматривать всю информацию базы данных, управлять объектами, созданные пользователями. Пользователь приложения имеет доступ к процедурам и может просматривать только доступную ему информацию.

Скрипт создания пользователь представлен на рисунке 3.12.

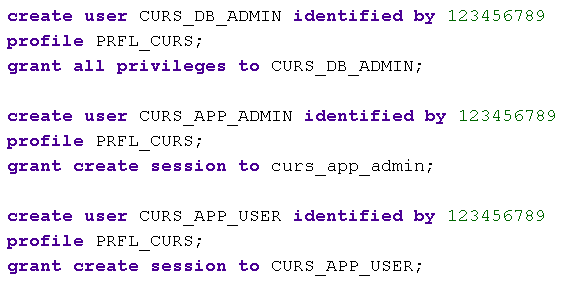


Рисунок 3.12 – Создание пользователей

* 1. Триггеры

Триггеры PL/SQL – это именованные программные блоки, выполняемые в ответ на происходящие в базе данных Oracle события такие, как добавление, изменение, удаление. 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 [6].

В данном курсовом проекте для всех таблиц были реализованы after триггеры, реагирующие на операции INSERT, UPDATE, DELETE. В результате вызова триггера он выводит строку изменения в DBMS\_OUTPUT.

Скрипт создания триггеров находится в приложении А.

* 1. Пакеты

Пакет 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. Спецификация – это интерфейс для ваших приложений.

В спецификации пакета объявляются типы, переменные, константы, исключения, курсоры и подпрограммы, доступные для использования. Тело пакета полностью определяет курсоры и подпрограммы и реализует спецификацию [7].

В курсовом проекте разработано 5 пакетов: UserPackage – пакет с процедурами, связанные с пользователями, TeamPackage – пакет с процедурами, связанные с командами, TaskPackage – пакет с процедурами, связанные с задачами, CommentPackage – пакет с процедурами, связанные с комментариями, ViewPackage – пакет с процедурами, связанные с выводом данных.

Скрипты создания пакет находится в приложении Б.

* 1. Представления

Представление – 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 [8].

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

В процессе выполнения курсового проекта были созданы представления для каждой таблицы (рисунок 3.13).

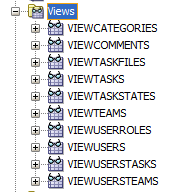


Рисунок 3.13 – Предстваления

1. Описание процедур импорта и экспорта

В любой типичной организации информация хранится в нескольких форматах, часть которой, возможно, помещается в реляционные базы данных, но большая часть все-таки хранится за пределами этих баз данных. Не помещаемая в базы данных информация может храниться в формате конкретных приложений, например, электронных таблиц Excel. Сохранение ее вместо этого в формате XML позволяет упрощать процесс получения доступа и обновления неструктурированных данных организации [9].

XML – расширяемый язык разметки. Спецификация XML описывает XML-документы и частично описывает поведение XML-процессоров (программ, читающих XML-документы и обеспечивающих доступ к их содержимому). XML разрабатывался как язык с простым формальным синтаксисом, удобный для создания и обработки документов программами и одновременно удобный для чтения и создания документов человеком, с подчёркиванием нацеленности на использование в Интернете. Язык называется расширяемым, поскольку он не фиксирует разметку, используемую в документах: разработчик волен создать разметку в соответствии с потребностями к конкретной области, будучи ограниченным лишь синтаксическими правилами языка [10].

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 использовался пакет DBMS\_LOB. Данные пакеты были выбраны, потому что они обладают очень широким функционалом и гибкой настройкой xml документа. Функции экспорта, импорта используются для таблицы Users, так как эта таблица является важнейшей в базе данных. Пример выходного xml файла можно посмотреть на рисунке 4.1.

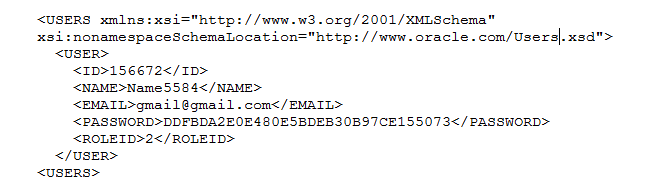


Рисунок 4.1 – Выходной файл xml

После изменим файл, чтобы у нас добавились новые пользователи (рисунок 4.2).

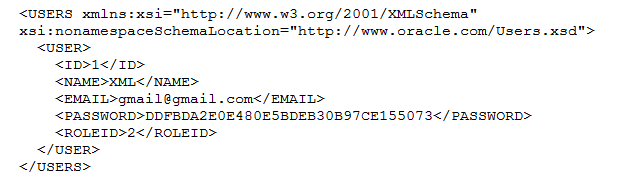


Рисунок 4.2 – Изменненный xml файл для импорта

На рисунке 4.3 изображен пользователь, добавленный после импорта.



Рисунок 4.3 – Добавленный пользователь

Скрипты функций импорта, экспорта представлены в приложении В.

1. Тестирование производительности

Тестирование производительности показывается на таблице Users. Данная таблица заполнена на 153847 строк (рисунок 5.1).

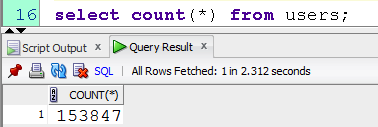


Рисунок 5.1 – Количество строк в таблице Users

Уникальные индексы основаны на уникальном столбце. Когда накладывается ограничение уникальности на столбец таблицы, Oracle автоматически создает уникальные индексы по этим столбцам. В данной таблице уникальным является столбец username. Выполняя запрос, мы получим следующую стоимость (рисунок 5.2).

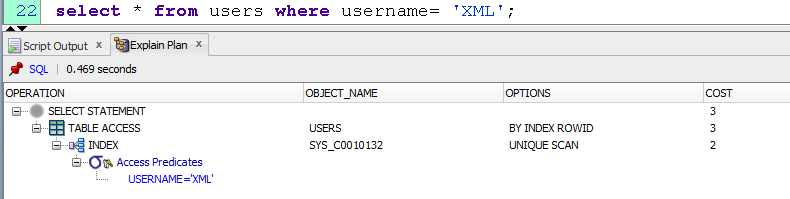


Рисунок 5.2 – Стоимость запроса

Таким образом, можно сделать, что построенная система удовлетворяет требованию по быстродействию.

1. Описание технологии

В данной курсовой работе были применена технология «Шифрование и маскирование».

6.1 Технология «Шифрование»

В простейшем определении шифрование представляет собой «маскировку» данных, или их преобразование, при котором данные не могут использоваться посторонними.

Oracle поддерживает шифрование сетевых данных посредством своей опции Advanced Security. Для обеспечения шифрования данных Oracle предлагает два пакета PL/SQL, один из которых, DBMS\_OBFUSCATION\_TOOLKIT, является более старым. Этот пакет позволяет шифровать данные по алгоритму DES. Для обеспечения наивысшего уровня безопасности этот пакет инструментов поддерживает тройное DES-шифрование. Кроме того, он поддерживает использование безопасного криптографического хеша MD5.

Для шифрования и дешифрации данных можно также применять более новый пакет шифрования PL/SQL по имени DBMS\_CRYPTO. По сравнению с DBMS\_OBFUSCATION\_TOOLKIT пакет DBMS\_CRYPTO предлагает более широкое множество шифровальных и криптографических алгоритмов для обеспечения повышенной безопасности и проще в использовании. Этот пакет предназначен для замены более старого пакета DBMS\_OBFUSCATION\_TOOLKIT. Но какой бы пакет не использовался, придется управлять ключами шифрования данных, что является нетривиальной задачей. Часто нужно создавать представления, которые облегчают дешифрацию и шифрование данных, что увеличивает количество задач управления. Кроме того, согласно рекомендациям Oracle, шифрованные данные нельзя индексировать, что в некоторых случаях снижает ценность обоих этих пакетов шифрования.

Существует также и третий, более простой подход: шифрование с помощью функции прозрачного шифрования данных. Далее будет показано, как зашифровать табличные данные Oracle с использованием Oracle Wallet (Бумажник Oracle) для хранения ключей шифрования. При этом можно также индексировать столбцы шифрованной таблицы, тем самым избавляясь от одного из наибольших недостатков применения пакетов шифрования Oracle.

Одной из важнейших проблем при построении инфраструктуры шифрования является построение эффективной системы управления ключами. Если злоумышленник получит доступ к ключам шифрования, то зашифрованные данные окажутся под угрозой независимо от сложности алгоритма. С другой стороны, некоторые пользователи должны иметь доступ к ключам для своей работы, и он должен быть достаточно простым для нормальной работы приложения. Проблема заключается в определении оптимального соотношения между простотой доступа и безопасностью ключей.

Существует немало распространенных, пользующихся коммерческой поддержкой алгоритмов шифрования. Все алгоритмы, поддерживаемые Oracle для приложений PL/SQL, относятся к категории алгоритмов с закрытым ключом (иногда называемых симметричными алгоритмами).

В Oracle чаще всего используются следующие алгоритмы:

* DES (Data Encryption Standard). Традиционно алгоритм DES занимал ведущие позиции в области шифрования. Он был разработан более 20 лет назад для Национального бюро стандартов (позднее переименованного в Национальный институт стандартов и технологий), и с тех пор был принят в качестве стандарта ISO. Об алгоритме DES и его истории можно рассказать очень много, но моей задачей является не описание алгоритма, а краткое описание его применения. Алгоритму DES необходим 64-разрядный ключ, но 8 бит ключа не используются. Чтобы подобрать ключ, злоумышленнику придется перебрать до 72 057 594 037 927 936 комбинаций. Возможностей DES было достаточно в течение нескольких десятилетий, но сейчас он постепенно уходит в прошлое. Современные мощные компьютеры способны перебрать даже огромное число комбинаций, необходимое для взлома ключа DES.
* DES3. В этой схеме, базирующейся на исходном алгоритме DES, данные шифруются дважды или трижды (в зависимости от режима вызова). DES3 использует 128- или 192-разрядный ключ; его длина определяется количеством проходов. Надежность алгоритма DES3 тоже была приемлемой в течение некоторого времени, но сейчас и этот алгоритм постепенно устаревает и не обеспечивает защиты от целенаправленных атак.
* AES. В ноябре 2001 года был одобрен новый стандарт AES (Advanced Encryption Standard), вступивший в силу в мае 2002 года. Полный текст стандарта можно найти по этому адресу.

Пакет DBMS\_CRYPTO появился в Oracle10g. В более ранних версиях пакет DBMS\_OBFUSCATION\_TOOLKIT предоставлял похожую (но не идентичную) функциональность. Старый пакет все еще остается доступным, но сейчас он считается устаревшим, и вместо него рекомендуется использовать новый пакет.

Для выполнения шифрования кроме входных данных необходимы еще четыре компонента:

* ключ шифрования;
* алгоритм шифрования;
* метод заполнения;
* метод сцепления.

Ключ шифрования предоставляете вы, а остальные компоненты предоставляет Oracle. Выбор осуществляется при помощи соответствующих констант пакета DBMS\_CRYPTO [11].

6.2 Применение технологии «Шифрования»

Для реализации технологии применяется функция DBMS\_CRYPTO.ENCRYPT. Функция получает четыре аргумента:

* src – исходные данные, подлежащие шифрованию должны иметь тип данных raw.
* key – ключ шифрования, тип данных raw. Длина ключа должна соответствовать вы­бранному алгоритму. Например, для алгоритма DES она должна быть не менее 64 бит.
* typ – определение трех компонентов алгоритм, механизм заполнения и метод сце­пления в виде суммы соответствующих констант.
* iv – необязательный вектор инициализации, еще один компонент схемы шифрования, затрудняющий анализ «закономерностей» в зашифрованном тексте.

В курсовой работе будут использоваться:

* алгоритм – AES c 128-разрядным ключом;
* метод сцепления – CBC;
* механизм заполнения – PKCS#5.

На рисунке 6.1 продемонстрирована процедура, которая применяют функцию шифрования для маскирования паролей от пользовательских аккаунтов.

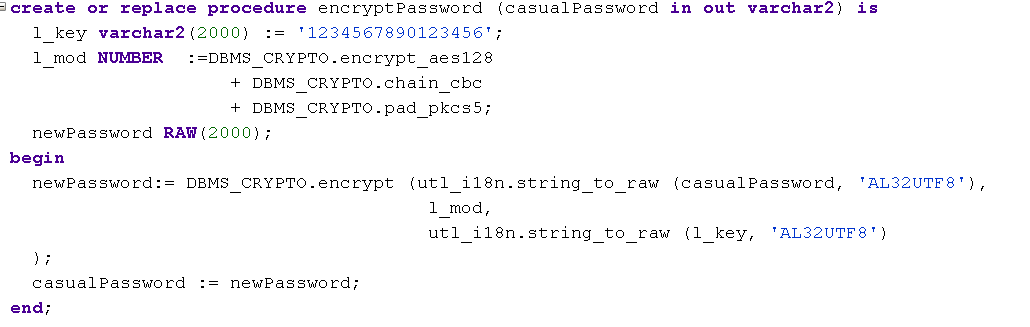


Рисунок 6.1 – Процедура шифрования пароля

Данная процедура принимает пароль в качестве единственного параметра и возвращает зашифрованный (рисунок 6.2-6.3).

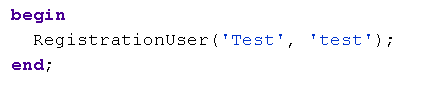


Рисунок 6.2 – Вызов процедуры шифрования



Рисунок 6.3 – Зашифрованный пароль

6.3 Технология «Маскирование»

Маскирование данных – это один из методов защиты конфиденциальной информации от несанкционированного доступа, при котором используются различные методы модификации данных, требующих защиты. Метод чаще всего используется при работе с базами данных (в том числе с помощью веб-приложений). Маскирование позволяет скрыть часть данных либо при создании копии базы данных, либо в ответе на запрос какой-либо информации, сохраняя общую информационную структуру и оставляя доступной ту часть данных, которая необходима для работы.

Применение маскировки данных наиболее распространено в процессе [разработки приложений](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F). При этом общепринятой практикой является использование производственных данных на всех этапах разработки: при создании приложений и [расширений](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D1%88%D0%B8%D1%80%D0%B5%D0%BD%D0%B8%D0%B5_(%D0%9F%D0%9E)), на этапах [тестирования](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) и [отладки](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BB%D0%B0%D0%B4%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D1%8B).

Основной проблемой с точки зрения руководства предприятий и организаций является то, что разработчики приложений не всегда подвергаются проверке корпоративными службами безопасности прежде, чем получают доступ к производственным данным. Подобная практика может стать причиной возникновения серьёзных уязвимостей в системе безопасности, поскольку данные могут быть скопированы неавторизованными пользователями, а меры безопасности на различных этапах производства легко обойти.

Существует несколько видов маскирования данных:

* статическое маскирование – это необратимый процесс замены критичных данных на реалистичные, основанный на заданных правилах, при котором данные преобразуются в одном направлении, а первоначальные данные не могут быть получены, извлечены или восстановлены, в отличие от шифрования и токенизации, которые позволяют обратить процесс преобразования данных, тем самым увеличивая риск утечки критичной информации.;
* динамическое маскирование – предусматривает подмену критичных данных в режиме реального времени при обращении к производственной базе данных. Реальные данные не покидают базу данных; они заменяются на этапе запроса, например, на полностью реалистичные, без промежуточной записи.

Данные, подвергнутые маскировке, должны соответствовать следующим критериям:

* Замаскированные данные должны быть понятными с точки зрения логики приложения. Например, рассмотрим ситуацию, когда необходимо замаскировать элементы почтовых адресов, а названия городов заменить другими названиями. Если приложение имеет возможность проверки почтового индекса или поиска по почтовому индексу, то маскировка не должна мешать корректному выполнению этих функций. Это же касается алгоритмов проверки номеров платёжных карт, страховых свидетельств и т. д.
* Маскировка должна полностью исключать возможность восстановления реальных производственных данных из замаскированных. Например, может быть общеизвестным, что в организации работают 10 руководителей высшего звена, зарплата которых составляет свыше $300,000. Если в замаскированную базу данных отдела кадров включено 10 значений из указанного числового диапазона (свыше $300,000), то злоумышленник может восстановить оставшиеся сведения методом реверс-инжиниринга. Поэтому маскировка данных должна проводиться таким образом, чтобы гарантировать защиту записей, содержащих персональные сведения, а не просто отдельных элементов в разрозненных полях и таблицах.

Методы маскировки данных:

– Замена является одним из самых эффективных способов маскировки, позволяющим сохранить исходный внешний вид данных. Например, если исходная таблица БД содержит записи с информацией о клиентах, то реальные имена и фамилии можно заменить именами и фамилиями, взятыми из специально созданного (подготовленного) файла. Так, на первом этапе маскировки все имена клиентов могут заменяться произвольными мужскими именами, а на втором этапе можно произвести вставку женских имён в ячейки, соответствующие клиентам-женщинам (с помощью фильтрации списка клиента по ячейке с указанием пола). Применение подобного подхода к маскировке позволяет обеспечить должную анонимность записей и сохранить половое соотношение клиентов в замаскированной таблице. Важно, что база данных при этом выглядит реалистично, а факт маскировки информации не является очевидным.

– Перемешивание – очень распространённый способ маскировки данных. Он схож с методом замены, но при перемешивании данные для замены берутся из той же колонки таблицы, что и исходные данные. Однако, маскировка с использованием лишь одного этого метода имеет серьёзные недостатки. Злоумышленник, имеющий доступ к части реальных сведений, может восстановить остальные данные путём анализа методом «что, если?». Кроме того, перемешивание можно обратить вспять, дешифровав его алгоритм.

– Метод дисперсии (разброса) применяется при работе c полями БД, содержащими финансовую информацию и даты. Этот метод заключается в отклонении замаскированного числового значения от исходного на определённую величину. Например, при маскировке ячеек, содержащих данные о зарплате сотрудников, отклонение от исходного значения может составлять 10%, поэтому замаскированная информация выглядит вполне реалистично и логично.

– Шифрование – это наиболее сложный способ маскировки данных. Алгоритм шифрования обычно предполагает наличие «ключа», необходимого для дешифровки и просмотра исходных данных. Шифрование также может сопровождаться преобразованием исходных данных в бинарный вид, что способно вызвать проблемы в работе приложений. Для выявления и устранения конфликтов внутри приложений необходимо проводить тестирование с передачей исходной информации тестировщикам, а это, в свою очередь, предполагает проверку задействованных в тестировании IT-специалистов службой безопасности [12].

6.4 Применение технологии «Маскирование»

Для реализации технологии в курсовой работе используется вызов процедуры, представленная на рисунке 6.4.

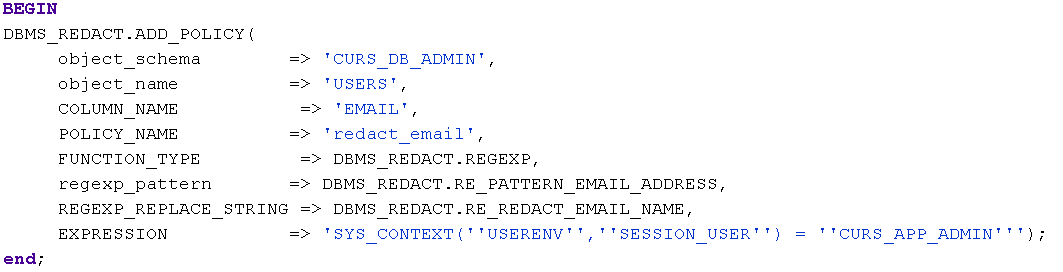


Рисунок 6.4 – Создание политики выполнения

Тут используется пакет DBMS\_REDACT, который был представлен в Oracle database версии 12c. эта функциональность также была перенесена в выпуск 11.2.0.4. Пакет используется для определения политик, которые управляют тем, какие данные отображаются пользователям приложения в зависимости от ситуации [13].

Чтобы отредактировать данные только для пользователя CURS\_APP\_ADMIN используется выражение 'SYS\_CONTEXT(''USERENV'',''SESSION\_USER'') = ''CURS\_APP\_ADMIN'''). Функция SYS\_CONTEXT используется для получения информации о состоянии среды окружения Oracle [14].

После создания политики выполнения у пользователя CURS\_APP\_ADMIN будет скрыто имя в почте (рисунок 6.5).

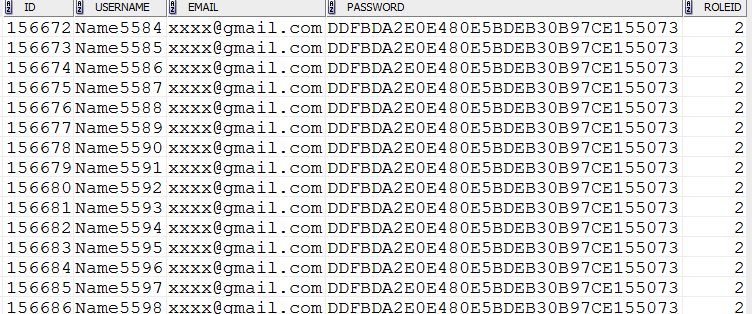


Рисунок 6.5 – Скрытие почты

Таким образом, можно скрывать данные от пользователей, которые не имеют нужные права на просмотр.

Заключение

В процессе выполнения курсовой работы была достигнута поставленная цель по созданию базы данных для программного обеспечения «Сеть для командной работы». Были созданы необходимые объекты: таблицы, процедуры, пакеты, профиль безопасности, пользователи, триггеры, представления. Удалось обеспечить большую гибкость базы данных за счет добавление процедур в пакеты.

При разработке проекта выполнены следующие пункты:

* Возможность авторизации по ролям;
* Возможность создавать, удалять и редактировать команду;
* Возможность добавлять и удалять пользователя в команду;
* Возможность прикреплять и удалять в задаче фотографии, видеоматериалы;
* Возможность осуществлять поиск пользователей по имени;
* Возможность раздавать и удалять задания для пользователей в команде;
* Возможность удалять и редактировать комментарии.

Была изучена технология «Шифрование и маскирование» на примере шифрования пароля пользователя и скрытия данных от посторонних. В ходе реализации технологии рассмотрены системные пакеты DBMS\_CRYPTO и DBMS\_REDACT.

В результате выполнения курсового проекта были получены навыки по взаимодействию с документами в формате xml посредством системных пакетов Oracle таких, как DBMS\_XMLDOM, DBMS\_XMLPARSER, DBMS\_XSLPROCESSOR.

По результатам выполнения программы можно сделать вывод о том, что разработанная программа работает должным образом и требования технического задания полностью выполнены.

Список используемых источников

1. Patches IT Community [Электронный ресурс]: Дата доступа: 23.11.2020. Режим доступа – https://oracle-patches.com/oracle/prof/3002-таблицы-oracle
2. Patches IT Community [Электронный ресурс]: Дата доступа: 23.11.2020. Режим доступа – https://oracle-patches.com/ db/sql/3611-процедуры-pl-sql-программирование-на-примерах
3. Свободная энциклопедия Википедия [Электронный ресурс]. Дата доступа: 23.11.2020. Режим доступа – https://ru.wikipedia.org/wiki/CRUD
4. Patches IT Community [Электронный ресурс]: Дата доступа: 23.11.2020. Режим доступа – https://oracle-patches.com/ oracle/prof/3043-пользователи-oracle-управление-и-безопасность-базы-данных
5. Студопедия [Электронный ресурс]. Дата доступа: 23.11.2020. Режим доступа – https://studopedia.su/16\_42650\_polzovateli-baz-dannih-administrator-bazi-dannih-ego-funktsii.html
6. Patches IT Community [Электронный ресурс]: Дата доступа: 23.11.2020. Режим доступа – https://oracle-patches.com/db/sql/3645-что-такое-триггеры-pl-sql-код,-выполняемый-по-событию
7. Oracle PL/SQL учебник [Электронный ресурс]: Дата доступа: 23.11.2020. Режим доступа – https://oracleplsql.ru/contents-oracle-plsql.html
8. Patches IT Community [Электронный ресурс]: Дата доступа: 23.11.2020. Режим доступа – https://oracle-patches.com/ oracle/begin/3016-представления-oracle-database
9. Patches IT Community [Электронный ресурс]: Дата доступа: 23.11.2020. Режим доступа – https://oracle-patches.com/ oracle/prof/3126-oracle-xml-db
10. Свободная энциклопедия Википедия [Электронный ресурс]. Дата доступа: 23.11.2020. Режим доступа –https://ru.wikipedia.org/wiki/XML
11. Patches IT Community [Электронный ресурс]: Дата доступа: 23.11.2020. Режим доступа – https://oracle-patches.com db/sql/3800-shifrovanie-i-deshifrovanie-v-pl-sql-dlya-bd-oracle-na-primere
12. Свободная энциклопедия Википедия [Электронный ресурс]. Дата доступа: 23.11.2020. Режим доступа – https://ru.wikipedia.org/wiki/Маскировка\_данных
13. Qualogy [Электронный ресурс]. Дата доступа: 23.11.2020. Режим доступа – https://www.qualogy.com/techblog/it-development-and-operations/redacting-data-from-the-oracle-database
14. Oracle PL/SQL учебник [Электронный ресурс]: Дата доступа: 23.11.2020. Режим доступа – https://oracleplsql.ru/sys\_context-function.html

Приложение А

--users triggers

create or replace trigger AfterUserInsert

after insert on users

begin

Dbms\_Output.Put\_Line('USER INSERTED');

end;

Create Or Replace Trigger AfterUserUpdate

after update on users

Begin

Dbms\_Output.Put\_Line('USER UPDATED');

end;

Create Or Replace Trigger AfterUserDelete

after delete on users

Begin

Dbms\_Output.Put\_Line('USER DELETED');

end;

--tasks triggers

Create Or Replace Trigger AfterTaskinsert

after insert on tasks

Begin

Dbms\_Output.Put\_Line('TASK INSERTED');

end;

Create Or Replace Trigger AfterTaskUpdate

after update on tasks

Begin

Dbms\_Output.Put\_Line('TASK UPDATED');

end;

Create Or Replace Trigger AfterTaskDelete

after delete on tasks

Begin

Dbms\_Output.Put\_Line('TASK DELETED');

End;

--teams triggers

Create Or Replace Trigger Afterteaminsert

after insert on teams

Begin

Dbms\_Output.Put\_Line('TEAM INSERTED');

end;

Create Or Replace Trigger Afterteamupdate

after update on teams

Begin

Dbms\_Output.Put\_Line('TEAM UPDATED');

end;

Create Or Replace Trigger Afterteamdelete

after delete on teams

Begin

Dbms\_Output.Put\_Line('TEAM DELETED');

End;

--comments triggers

Create Or Replace Trigger Aftercommentinsert

after insert on comments

Begin

Dbms\_Output.Put\_Line('COMMENT INSERTED');

end;

Create Or Replace Trigger Aftercommentupdate

after update on comments

Begin

Dbms\_Output.Put\_Line('COMMENT UPDATED');

end;

Create Or Replace Trigger Aftercommentdelete

after delete on comments

Begin

Dbms\_Output.Put\_Line('COMMENT DELETED');

End;

--usertasks triggers

Create Or Replace Trigger Afterusersteamsinsert

after insert on usersteams

Begin

Dbms\_Output.Put\_Line('ADD USER TO TEAM');

end;

Create Or Replace Trigger Afteruersteamsdelete

after delete on usersteams

Begin

Dbms\_Output.Put\_Line('DELETE USER FROM TEAM');

End;

--userstasks triggers

Create Or Replace Trigger Afteruserstasksinsert

after insert on userstasks

Begin

Dbms\_Output.Put\_Line('ADD USER TO TASK');

end;

Create Or Replace Trigger Afteruserstasksdelete

after delete on userstasks

Begin

Dbms\_Output.Put\_Line('DELETE USER FROM TASK');

End;

Приложение Б

create or replace package viewpackage as

procedure teamsbyuser(inuserid in number, cursor in out sys\_refcursor);

procedure viewteamsbyuser(inuserid in number);

procedure commentsintask(intaskid in number, cursor in out sys\_refcursor);

procedure viewcommentsintask(intaskid in number);

procedure tasksbyteam(inteamid in number, cursor in out sys\_refcursor);

procedure viewtasksbyteam(inteamid in number);

procedure usersinteam(inteamid in number, cursor in out sys\_refcursor);

procedure viewusersinteam(inteamid in number);

procedure usersintask(intaskid in number, cursor in out sys\_refcursor);

procedure viewusersintask(intaskid in number);

procedure filesintask(intaskid in number, cursor in out sys\_refcursor);

procedure viewfilesintask(intaskid in number);

procedure getuserinformation (inuserid in number, cursor in out sys\_refcursor);

procedure ViewUserInformaion(inUserId IN NUMBER);

end ViewPackage;

create or replace package body viewpackage as

--all user teams

procedure TeamsByUser(inUserId in number,

cursor in out sys\_refcursor)

is

BEGIN

open cursor for

select ut.userId, t.id, t.teamName, t.teamdescription, t.categoryId, ca.category, t.managerid, u.username

from UsersTeams ut inner join Teams t on ut.teamId = t.id

inner join Categories ca on ca.id = t.categoryid

inner join Users u on u.id = t.managerid

where ut.userId = inUserId;

end;

procedure ViewTeamsByUser(INUSERID IN NUMBER)

is

REFCUR SYS\_REFCURSOR;

TYPE RECORDTYPE IS RECORD (COL1 NUMBER, COL2 NUMBER, COL3 VARCHAR(2000), COL4 VARCHAR(4000), COL5 NUMBER, COL6 VARCHAR(2000), COL7 NUMBER, COL8 VARCHAR(2000));

outtable RECORDTYPE;

BEGIN

TeamsByUser(INUSERID, REFCUR);

LOOP

FETCH REFCUR INTO outtable;

EXIT WHEN REFCUR%NOTFOUND;

DBMS\_OUTPUT.PUT('userId:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL1);

DBMS\_OUTPUT.PUT('teamId:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL2);

DBMS\_OUTPUT.PUT('teamName:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL3);

DBMS\_OUTPUT.PUT('teamDescription:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL4);

DBMS\_OUTPUT.PUT('categoryId:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL5);

DBMS\_OUTPUT.PUT('category:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL6);

DBMS\_OUTPUT.PUT('managerId:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL7);

DBMS\_OUTPUT.PUT('username:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL8);

DBMS\_OUTPUT.PUT\_LINE('');

END LOOP;

CLOSE REFCUR;

END;

--all comments in task

procedure CommentsInTask(inTaskId in number,

cursor in out sys\_refcursor)

is

BEGIN

open cursor for

select ut.id, co.creatorId, u.userName, co.commentContent, co.dateOfCreation

from UsersTasks ut inner join Users u on ut.userId = u.id

inner join Comments co on co.creatorId = u.id

where ut.taskId = inTaskId;

end;

procedure ViewCommentsInTask(inTaskId IN NUMBER)

is

REFCUR SYS\_REFCURSOR;

TYPE RECORDTYPE IS RECORD (COL1 NUMBER, COL2 NUMBER, COL3 VARCHAR(2000), COL4 VARCHAR(4000), COL5 DATE);

outtable RECORDTYPE;

BEGIN

CommentsInTask(inTaskId, REFCUR);

LOOP

FETCH REFCUR INTO outtable;

EXIT WHEN REFCUR%NOTFOUND;

DBMS\_OUTPUT.PUT('taskId:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL1);

DBMS\_OUTPUT.PUT('creatorId:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL2);

DBMS\_OUTPUT.PUT('userName:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL3);

DBMS\_OUTPUT.PUT('commentcontent:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL4);

DBMS\_OUTPUT.PUT('dateOfCreation:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL5);

DBMS\_OUTPUT.PUT\_LINE('');

END LOOP;

CLOSE REFCUR;

END;

--all tasks in team

procedure TasksByTeam(inTeamId in number,

cursor in out sys\_refcursor)

is

BEGIN

open cursor for

select t.id, ta.id, ta.taskname, ta.taskdescription, ta.dateofcreation, ta.dateofdeadline, ta.taskstateid, st.statename, ta.managerid, u.username

from Tasks ta inner join Teams t on ta.teamId = t.id

inner join Taskstates st on ta.taskstateid = st.id

inner join Users u on ta.managerid = u.id

where t.id = inTeamId;

end;

procedure ViewTasksByTeam(inTeamId IN NUMBER)

is

REFCUR SYS\_REFCURSOR;

TYPE RECORDTYPE IS RECORD (COL1 NUMBER, COL2 NUMBER, COL3 VARCHAR(100), COL4 VARCHAR(4000), COL5 DATE, COL6 DATE, COL7 NUMBER, COL8 VARCHAR(100), COL9 NUMBER, COL10 VARCHAR(100));

outtable RECORDTYPE;

BEGIN

TasksByTeam(inTeamId, REFCUR);

LOOP

FETCH REFCUR INTO outtable;

EXIT WHEN REFCUR%NOTFOUND;

DBMS\_OUTPUT.PUT('teamId:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL1);

DBMS\_OUTPUT.PUT('taskId:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL2);

DBMS\_OUTPUT.PUT('taskName:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL3);

DBMS\_OUTPUT.PUT('taskDescription:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL4);

DBMS\_OUTPUT.PUT('dateOfcreation:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL5);

DBMS\_OUTPUT.PUT('dateOfDeadline:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL6);

DBMS\_OUTPUT.PUT('taskStateId:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL7);

DBMS\_OUTPUT.PUT('stateName:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL8);

DBMS\_OUTPUT.PUT('managerId:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL9);

DBMS\_OUTPUT.PUT('userName:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL10);

DBMS\_OUTPUT.PUT\_LINE('');

END LOOP;

CLOSE REFCUR;

end;

--all users in team

procedure UsersInTeam(inTeamId in number,

cursor in out sys\_refcursor)

is

BEGIN

open cursor for

select t.id, t.teamName, u.id, u.userName

from UsersTeams ut inner join Teams t on ut.teamId = t.id

inner join Users u on u.id = ut.userId

where t.id = inTeamId;

end;

procedure ViewUsersInTeam(inTeamId IN NUMBER)

is

REFCUR SYS\_REFCURSOR;

TYPE RECORDTYPE IS RECORD (COL1 NUMBER, COL2 VARCHAR(100), COL3 NUMBER, COL4 VARCHAR(100));

outtable RECORDTYPE;

BEGIN

UsersInTeam(inTeamId, REFCUR);

LOOP

FETCH REFCUR INTO outtable;

EXIT WHEN REFCUR%NOTFOUND;

DBMS\_OUTPUT.PUT('teamId:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL1);

DBMS\_OUTPUT.PUT('teamName:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL2);

DBMS\_OUTPUT.PUT('userId:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL3);

DBMS\_OUTPUT.PUT('username:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL4);

DBMS\_OUTPUT.PUT\_LINE('');

END LOOP;

CLOSE REFCUR;

END;

--all users in task

procedure UsersInTask(inTaskId in number,

cursor in out sys\_refcursor)

is

BEGIN

open cursor for

select t.id, t.taskName, u.id, u.userName

from UsersTasks ut inner join Tasks t on ut.taskId = t.id

inner join Users u on u.id = ut.userId

where t.id = inTaskId;

end;

procedure ViewUsersInTask(inTaskId IN NUMBER)

is

REFCUR SYS\_REFCURSOR;

TYPE RECORDTYPE IS RECORD (COL1 NUMBER, COL2 VARCHAR(100), COL3 NUMBER, COL4 VARCHAR(100));

outtable RECORDTYPE;

BEGIN

UsersInTask(inTaskId, REFCUR);

LOOP

FETCH REFCUR INTO outtable;

EXIT WHEN REFCUR%NOTFOUND;

DBMS\_OUTPUT.PUT('taskId:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL1);

DBMS\_OUTPUT.PUT('taskName:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL2);

DBMS\_OUTPUT.PUT('userId:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL3);

DBMS\_OUTPUT.PUT('username:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL4);

DBMS\_OUTPUT.PUT\_LINE('');

END LOOP;

CLOSE REFCUR;

END;

--all files in task

procedure FilesInTask(inTaskId in number,

cursor in out sys\_refcursor)

is

BEGIN

open cursor for

select t.id, t.taskName, f.fileUrl

from TaskFiles f inner join Tasks t on f.taskId = t.id

where t.id = inTaskId;

end;

procedure ViewFilesInTask(inTaskId IN NUMBER)

is

REFCUR SYS\_REFCURSOR;

TYPE RECORDTYPE IS RECORD (COL1 NUMBER, COL2 VARCHAR(100), COL3 VARCHAR(4000));

outtable RECORDTYPE;

BEGIN

FilesInTask(inTaskId, REFCUR);

LOOP

FETCH REFCUR INTO outtable;

EXIT WHEN REFCUR%NOTFOUND;

DBMS\_OUTPUT.PUT('taskId:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL1);

DBMS\_OUTPUT.PUT('taskName:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL2);

DBMS\_OUTPUT.PUT('url:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL3);

DBMS\_OUTPUT.PUT\_LINE('');

END LOOP;

CLOSE REFCUR;

END;

--get user information

procedure GetUserInformation (inuserid in number, cursor in out sys\_refcursor)

is

begin

open cursor for select u.id, u.username, u.email, u.roleid, rol.role

from users u inner join userroles rol on rol.id=u.roleid where u.id=inuserid;

end;

procedure ViewUserInformaion(inUserId IN NUMBER)

is

refcur sys\_refcursor;

TYPE RECORDTYPE IS RECORD (COL1 NUMBER, COL2 VARCHAR(100), COL3 VARCHAR(320), COL4 number, COL5 varchar2(100));

outtable RECORDTYPE;

begin

GetUserInformation(inUserId, REFCUR);

LOOP

FETCH REFCUR INTO outtable;

EXIT WHEN REFCUR%NOTFOUND;

DBMS\_OUTPUT.PUT('userId:');

dbms\_output.put\_line(outtable.col1);

DBMS\_OUTPUT.PUT('username:');

dbms\_output.put\_line(outtable.col2);

DBMS\_OUTPUT.PUT('email:');

dbms\_output.put\_line(outtable.col3);

dbms\_output.put('roleid:');

dbms\_output.put\_line(outtable.col4);

dbms\_output.put('role:');

DBMS\_OUTPUT.PUT\_LINE(OUTTABLE.COL5);

DBMS\_OUTPUT.PUT\_LINE('');

END LOOP;

close refcur;

end;

end viewpackage;

create or replace package commentpackage as

procedure AddComment(inUserid in number,

incommentcontent in varchar2,

intaskid in number);

procedure deletecomment(incommentid in number);

procedure deleteallcomments(inuserid in number);

procedure deletecommentsintask(intaskid in number);

procedure UpdateComment (inCommentContent in varchar2);

end CommentPackage;

create or replace package body commentpackage as

--add comment

procedure AddComment(inUserid in number,

inCommentContent in varchar2,

inTaskId in number)

is

begin

insert into Comments(creatorId,

dateOfCreation,

commentContent,

taskId)

values(inUserId,

Sysdate,

inCommentContent,

inTaskId);

commit;

exception when others

then dbms\_output.put\_line(sqlcode||' '||sqlerrm);

end;

--delete comment

procedure DeleteComment(inCommentId in number)

is

begin

delete from Comments where id = inCommentId;

commit;

exception when others

then dbms\_output.put\_line(sqlcode||' '||sqlerrm);

end;

--delete all comments in all tasks

procedure DeleteAllComments(inUserid in number)

is

begin

delete from Comments where creatorId = inUserId;

commit;

exception when others

then dbms\_output.put\_line(sqlcode||' '||sqlerrm);

end;

--delete all comments in task

procedure DeleteCommentsInTask(inTaskId in number)

is

begin

delete from Comments where taskId = inTaskId;

commit;

exception when others

then dbms\_output.put\_line(sqlcode||' '||sqlerrm);

end;

--alter comment

procedure UpdateComment (inCommentContent in varchar2)

is

begin

update Comments set commentContent = inCommentContent;

commit;

exception when others

then dbms\_output.put\_line(sqlcode||' '||sqlerrm);

end;

end commentpackage;

create or replace package taskpackage as

procedure CreateTask (inTeamId in number,

inTaskName in varchar2,

inDescription in varchar2,

indateofdeadline in date,

inmanagerid in number);

procedure completedtask(intaskid in number);

procedure cancelledtask(intaskid in number);

procedure inprogresstask(intaskid in number);

procedure deletetask (intaskid in number);

procedure deletetasksinteam (inteamid in number);

procedure AddUserInTask (inUserId in number, inTaskId in number);

procedure UpdateTask (inTaskId in number,

inTaskName in varchar2,

inDescription in varchar2,

indateofdeadline in date,

inteamid in number);

procedure addfilestotask(intaskid in number, infileurl in varchar2);

procedure deletefilesfromtask(intaskid in number, infileurl in varchar2);

procedure DeleteUserFromTask (inUserId in number, inTaskId in number);

end TaskPackage;

create or replace package body taskpackage as

--add task

procedure CreateTask (inTeamId in number,

inTaskName in varchar2,

inDescription in varchar2,

inDateOfDeadline in date,

inManagerId in number)

is

ctask number;

cursor c1 is select t.id from Tasks t

where t.taskname = inTaskName and t.id = inTeamId;

begin

open c1;

fetch c1 into ctask;

if c1%found then

raise\_application\_error(-20001,'Task already exists');

end if;

insert into tasks(taskName,

taskDescription,

dateOfCreation,

dateOfDeadline,

taskStateId,

teamid,

managerid)

values(inTaskName,

inDescription,

Sysdate,

inDateOfDeadline,

1,

inTeamId,

inManagerid);

commit;

close c1;

exception when others

then dbms\_output.put\_line(sqlcode||' '||sqlerrm);

end;

--completed task

procedure CompletedTask(inTaskId in number)

is

begin

update Tasks set taskstateid = 2 where id = inTaskId;

commit;

exception when others

then dbms\_output.put\_line(sqlcode||' '||sqlerrm);

end;

--cancelled task

procedure CancelledTask(inTaskId in number)

is

begin

update Tasks set taskstateid = 3 where id = inTaskId;

commit;

exception when others

then dbms\_output.put\_line(sqlcode||' '||sqlerrm);

end;

--inProgress task

procedure InProgressTask(inTaskId in number)

is

begin

update Tasks set taskstateid = 1 where id = inTaskId;

commit;

exception when others

then dbms\_output.put\_line(sqlcode||' '||sqlerrm);

end;

--delete task

procedure DeleteTask (inTaskId in number)

is

begin

delete from Tasks where id = inTaskId;

commit;

exception when others

then dbms\_output.put\_line(sqlcode||' '||sqlerrm);

end;

--delete all tasks in team

procedure DeleteTasksInTeam (inTeamId in number)

is

begin

delete from Tasks where teamId = inTeamId;

commit;

exception when others

then dbms\_output.put\_line(sqlcode||' '||sqlerrm);

end;

--add user in task

procedure AddUserInTask (inUserId in number, inTaskId in number)

is

cuser number;

cursor c1 is select u.id from UsersTasks u

where u.userId = inUserId and u.taskId = inTaskId;

begin

open c1;

fetch c1 into cuser;

if c1%found then

raise\_application\_error(-20001,'User already in task');

end if;

insert into UsersTasks (userId, taskId) values(inUserId, inTaskId);

commit;

close c1;

exception when others

then dbms\_output.put\_line(sqlcode||' '||sqlerrm);

end;

--update task

procedure UpdateTask (inTaskId in number,

inTaskName in varchar2,

inDescription in varchar2,

inDateOfDeadline in date,

inTeamId in number)

is

ctask number;

cursor c1 is select t.id from Tasks t

where t.taskname = inTaskName and t.id = inTeamId;

begin

open c1;

fetch c1 into ctask;

if c1%found then

raise\_application\_error(-20001,'Task already exists');

end if;

update tasks set taskName = inTaskName,

taskDescription = InDescription,

dateOfDeadline = inDateOfDeadline

where id = inTaskId;

commit;

close c1;

exception when others

then dbms\_output.put\_line(sqlcode||' '||sqlerrm);

end;

--add pic or video to task

procedure AddFilesToTask(inTaskId in number, inFileUrl in varchar2)

is

begin

insert into TaskFiles (fileUrl, taskId) values (inFileUrl, inTaskId);

commit;

exception when others

then dbms\_output.put\_line(sqlcode||' '||sqlerrm);

end;

--delete doc from task

procedure DeleteFilesFromTask(inTaskId in number, inFileUrl in varchar2)

is

begin

delete from TaskFiles where fileUrl = inFileUrl and taskId = inTaskId;

commit;

exception when others

then dbms\_output.put\_line(sqlcode||' '||sqlerrm);

end;

--delete user from task

procedure DeleteUserFromTask (inUserId in number, inTaskId in number)

is

cuser number;

cursor c1 is select u.id from UsersTasks u

where u.userId = inUserId and u.taskId = inTaskId;

begin

open c1;

fetch c1 into cuser;

if c1%notfound then

raise\_application\_error(-20001,'User not found');

end if;

delete from UsersTasks where userId = inUserId and taskId = inTaskId;

commit;

close c1;

exception when others

then dbms\_output.put\_line(sqlcode||' '||sqlerrm);

end;

end TaskPackage;

create or replace package TeamPackage as

procedure CreateTeam (inUserid in number,

inTeamName in varchar2,

indescription in varchar2,

incategoryid in number);

procedure deleteteam (inteamid in number);

procedure deleteallteams (inuserid in number);

procedure UpdateTeam (newTeamName in varchar2,

newdescription in varchar2,

newcategoryid in number);

procedure adduserinteam (inuserid in number, inteamid in number);

procedure deleteuserfromteam (inuserid in number, inteamid in number);

end TeamPackage;

create or replace package body teampackage as

--create team

procedure CreateTeam (inUserid in number,inTeamName in varchar2,

inDescription in varchar2,inCategoryid in number)

is

cid number;

begin

insert into teams(managerid,

teamName,

teamDescription,

categoryid)

values (inUserid,

inTeamName,

inDescription,

inCategoryid) returning id into cid;

insert into UsersTeams (userId, teamid) values(inUserId, cid);

commit;

end;

--delete team

procedure DeleteTeam (inTeamid in number)

is

begin

delete from teams where id = inTeamid;

commit;

end;

--delete all teams

procedure DeleteAllTeams (inUserid in number)

is

begin

delete from teams where managerId = inUserid;

commit;

end;

--update team

procedure UpdateTeam (newTeamName in varchar2,newDescription in varchar2,

newCategoryid in number)

is

begin

update teams set teamName = newTeamName,

teamDescription = newDescription,

categoryid = newCategoryid;

commit;

end;

--add user in team

procedure AddUserInTeam (inUserId in number, inTeamId in number)

is

cuser number;

cursor c1 is select u.id from UsersTeams u

where u.userId = inUserId and u.teamId = inTeamId;

begin

open c1;

fetch c1 into cuser;

if c1%found then

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

end if;

insert into UsersTeams (userId, teamId) values(inUserId, inTeamId);

commit;

close c1;

end;

--delete user from team

procedure DeleteUserFromTeam (inUserId in number, inTeamId in number)

is

cuser number;

cursor c1 is select u.id from UsersTeams u

where u.userId = inUserId and u.teamId = inTeamId;

begin

open c1;

fetch c1 into cuser;

if c1%notfound then

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

end if;

delete from UsersTeams where userId = inUserId and teamId = inTeamId;

commit;

close c1;

end;

end TeamPackage;

create or replace package UserPackage as

procedure checkuser (inusername in varchar2, inpassword in varchar2, outuserid out number);

procedure encryptPassword (casualPassword in out varchar2);

procedure authorizeuser (inusername in varchar2, inpassword in varchar2);

procedure RegistrationUser (inUsername in varchar2, inEmail in varchar2, inPassword in varchar2);

procedure ChangeUserName (inUserId in number, inUsername in varchar2);

procedure ChangePassword (inUserId in number, inPassword in varchar2);

procedure ChangeEmail (inUserId in number, inEmail in varchar2);

procedure DeleteUser (inUserId in number);

procedure searchuserbyname (inusername in varchar2, outuserid out number);

procedure GetUserInformation (inUserId in number, cursor in out sys\_refcursor);

end UserPackage;

create or replace package body UserPackage as

procedure CheckUser (inUsername in varchar2, inPassword in varchar2, outUserId out number)

is

cuser number;

cursor c1 is select u.id from Users u where u.username = inUsername and u.password = inPassword;

begin

open c1;

fetch c1 into cuser;

if c1%notfound then

raise\_application\_error(-20001,'User not found');

end if;

outUserId := cuser;

close c1;

exception when others

then dbms\_output.put\_line(sqlcode||' '||sqlerrm);

end;

--encrypt password

procedure encryptPassword (casualPassword in out varchar2) is

l\_key varchar2(2000) := '1234567890123456';

l\_mod NUMBER :=DBMS\_CRYPTO.encrypt\_aes128

+ DBMS\_CRYPTO.chain\_cbc

+ DBMS\_CRYPTO.pad\_pkcs5;

newPassword RAW(2000);

begin

newPassword:= DBMS\_CRYPTO.encrypt (utl\_i18n.string\_to\_raw (casualPassword, 'AL32UTF8'),

l\_mod,

utl\_i18n.string\_to\_raw (l\_key, 'AL32UTF8')

);

casualpassword := newpassword;

end;

--authorization users

procedure AuthorizeUser (inUsername in varchar2, inPassword in varchar2)

is

cpassw varchar2(2000);

cuserid number;

begin

cpassw := inPassword;

encryptpassword(cpassw);

checkuser(inusername,cpassw, cuserid);

dbms\_output.put\_line(cuserid);

end;

--registration user

procedure RegistrationUser (inUsername in varchar2, inEmail in varchar2, inPassword in varchar2)

is

cuser number;

cursor c1 is select u.id from Users u where u.username = inUsername;

newPassword varchar(2000);

begin

newPassword :=inPassword;

encryptpassword(newPassword);

open c1;

fetch c1 into cuser;

if c1%found then

raise\_application\_error(-20001,'User already exists');

end if;

insert into Users (username, email, password, roleId) values(inUsername, inEmail, newPassword, 2);

commit;

close c1;

exception when others

then dbms\_output.put\_line(sqlcode||' '||sqlerrm);

end;

--change username

procedure ChangeUserName (inUserId in number, inUsername in varchar2)

is

cuser number;

cursor c1 is select u.id from Users u where u.username = inUsername;

begin

open c1;

fetch c1 into cuser;

if c1%found then

raise\_application\_error(-20001,'Username already exists');

end if;

update Users set userName = inUserName where id = inUserId;

close c1;

exception when others

then dbms\_output.put\_line(sqlcode||' '||sqlerrm);

end;

--change password

procedure ChangePassword (inUserId in number, inPassword in varchar2)

is

passw varchar2(2000);

begin

passw := inPassword;

encryptPassword(cpassw);

update Users set password = passw where id = inUserId;

end;

--change email

procedure ChangeEmail (inUserId in number, inEmail in varchar2)

is

begin

update Users set email = inEmail where id = inUserId;

close c1;

end;

--delete user

procedure DeleteUser (inUserId in number)

is

begin

delete from Users where id = inUserId;

end;

--search user by username

procedure SearchUserByName (inUserName in varchar2, outUserId out number)

is

cuser number;

cursor c1 is select u.id from Users u where u.username = inUsername;

begin

open c1;

fetch c1 into cuser;

if c1%notfound then

raise\_application\_error(-20001,'User not found');

end if;

outUserId := cuser;

close c1;

exception when others

then dbms\_output.put\_line(sqlcode||' '||sqlerrm);

end;

--get user information

procedure GetUserInformation (inUserId in number, cursor in out sys\_refcursor) is

begin

open cursor for

select u.id, u.userName, u.email, u.roleId from Users u where u.id = inUserId;

end;

end UserPackage;

Приложение В

create or replace directory UTLDATA AS 'C:/app/xml/';

create or replace procedure ExportUsersToXml is

DOC DBMS\_XMLDOM.DOMDocument;

XDATA XMLTYPE;

CURSOR XMLCUR IS

SELECT XMLELEMENT("USERS",

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

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

XMLAGG(XMLELEMENT("USER",

XMLELEMENT("ID",U.ID),

xmlelement("NAME",u.username),

xmlelement("EMAIL",u.email),

xmlelement("PASSWORD",u.password),

XMLELEMENT("ROLEID",U.roleId)

))

) FROM USERS 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/users.xml');

END;

begin

ExportUsersToXml();

end;

create or replace procedure ImportUsersFromXml

IS

L\_CLOB CLOB;

L\_BFILE BFILE := BFILENAME('UTLDATA', 'users.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 users%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;

P := Dbms\_Xmlparser.Newparser;

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

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

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

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

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

LOOP

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

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

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

Dbms\_Xslprocessor.Valueof(V\_Current\_Node,

'NAME/text()',U.USERNAME);

Dbms\_Xslprocessor.Valueof(V\_Current\_Node,

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

dbms\_xslprocessor.valueof(v\_current\_node,

'PASSWORD/text()',U.password);

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

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

insert into Users(Username,Email,password,Roleid)

values(U.USERNAME,U.EMAIL,U.password,U.ROLEID);

end loop;

DBMS\_LOB.FREETEMPORARY(L\_CLOB);

DBMS\_XMLPARSER.FREEPARSER(P);

DBMS\_XMLDOM.FREEDOCUMENT(V\_DOC);

commit;

END;

BEGIN

ImportUsersFromXml();

END;