МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1–40 05 01 Информационные системы и технологии

Специализация Издательско-полиграфический комплекс

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

Реализация базы данных для индивидуального тренера с использованием технологии мониторинга состояния СУБД

Выполнил студент Кохнюк А. С.

(Ф.И.О.)

Руководитель проекта ассистент Нистюк О. А.

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

Заведующий кафедрой 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 2023

Реферат

Пояснительная записка курсового проекта содержит 37 страниц пояснительной записки, 12 иллюстраций, 6 источников литературы, 2 приложения.

ORACLE 12c.

Цель курсового проекта: разработка базы данных для индивидуального тренера с применением технологии мониторинга состояния СУБД.

Первая глава описывает аналитический обзор.

Вторая глава описывает постановку задачи курсового проекта.

Третья глава описывает процесс создания и настройки базы данных, с которой будет происходить взаимодействие.

Четвертая глава описывает процесс создания объектов базы данных.

Пятая глава описывает процедуры импорта и экспорта данных в формате XML.

В шестой главе приведено тестирование производительности базы данных.

Седьмая глава описывает технологию, которая применена в базе данных.

В восьмой главе приведено руководство пользователя по взаимодействию с приложением и описаны базовые тестовые случаи.

В заключении описаны результаты проделанной работы.

Оглавление

[Введение 4](#_Toc134820716)

[1 Аналитический обзор литературы 5](#_Toc134820717)

[2 Постановка задачи 7](#_Toc134820718)

[3 Проектирование базы данных 8](#_Toc134820719)

[4 Разработка объектов базы данных 10](#_Toc134820720)

[4.1 Таблицы 10](#_Toc134820721)

[4.2 Процедуры 12](#_Toc134820722)

[5 Описание процедур импорта и экспорта 14](#_Toc134820723)

[5.1 Описание процедуры импорта 14](#_Toc134820724)

[5.2 Описание процедуры экспорта 15](#_Toc134820725)

[6 Тестирование производительности 16](#_Toc134820726)

[7 Описание технологии и ее применение в базе данных 17](#_Toc134820727)

[8 Руководство пользователя 19](#_Toc134820728)

[Заключение 21](#_Toc134820729)

[Список используемых источников 22](#_Toc134820730)

[Приложение А 23](#_Toc134820731)

[Приложение Б 25](#_Toc134820732)

Введение

В настоящее время огромное количество компаний используют персональные компьютеры для хранения и обработки любого рода информации. Эта информация содержится в базах данных. Базы данных играют важную роль в развивающемся мире технологий. С их помощью легко хранить, структурировать и организовывать большие объемы информации. Данные из базы данных могут быть получены очень быстро, что делает их отличным средством для хранения больших объемов информации, а также обеспечивают их защиту. Все, с чем мы ежедневно взаимодействуем в жизни, видимо, зафиксировано в какой-то базе данных.

Работа с базами данных является необходимым навыком при работе с компьютером, и специалисты в этой области становятся все более востребованными. Основные идеи современной информационной методологии основаны на представлении, согласно которому информация должна формироваться в базы данных с задачей отображения динамически меняющегося мира и удовлетворения всех потребностей пользователей в информации. Базы данных формируются и работают под управлением специального программного обеспечения, называемого системами управления базами данных.

В соответствии с заданием курсового проекта для проектирования базы данных используется система управления базами данных Oracle 12c.

1. Аналитический обзор литературы

Анализ требований — это процесс сбора требований к программному обеспечению, их систематизации, документирования, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе разработки программного обеспечения.

Цель анализа требований в проектах — получить максимум информации, уточнить рамки проекта, оценить возможные риски. На этом этапе происходит идентификация принципиальных требований методологического и технологического характера, формулируются цели и задачи проекта, а также определяются критические факторы успеха, которые впоследствии будут использоваться для оценки результатов внедрения. Определение и описание требований — шаги, которые во многом определяют успех всего проекта, поскольку именно они влияют на все остальные этапы.

«Фитнес Бро» - приложение, помогающее людям заниматься спортом. Каталог упражнений позволяет ознакомиться и выбрать наилучшие упражнения для тренировки. Любимые упражнения могут быть сохранены в отдельном месте, чтобы всегда иметь к ним быстрый доступ. Секундомер поможет засекать время между подходами и не даст тебе "остыть". Добавлен раздел напоминаний - теперь сложно забыть выпить витамин. Ниже на рисунке 1.1 представлен внешний вид приложения.

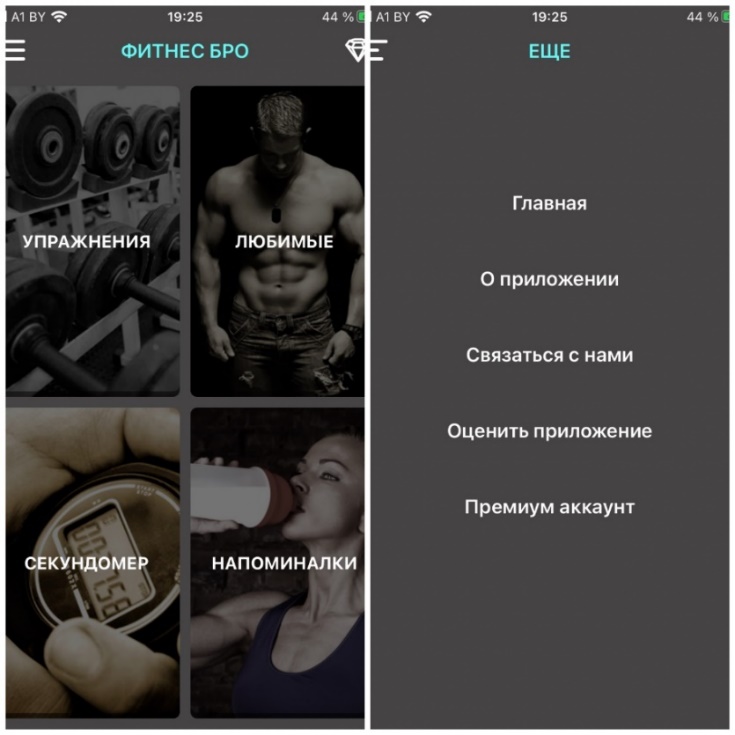


Рисунок 1.1 – Приложение «Фитнес Бро»

Приложение «Fitabillity» - авторская программа персональных тренировок, разработанная известным в спортивных кругах тренером по функциональной подготовке, реабилитологом Марией Буровой. Это уникальная сбалансированная система самостоятельных тренировок, основанная на десятилетнем профессиональном опыте действующего тренера и международных медицинских разработках. Приложение дает возможность выбора тренировки на какую-либо отдельную часть тела, где можно скачать видео с инструкциями тренировок. Ниже на рисунке 1.2 представлен внешний вид приложения.

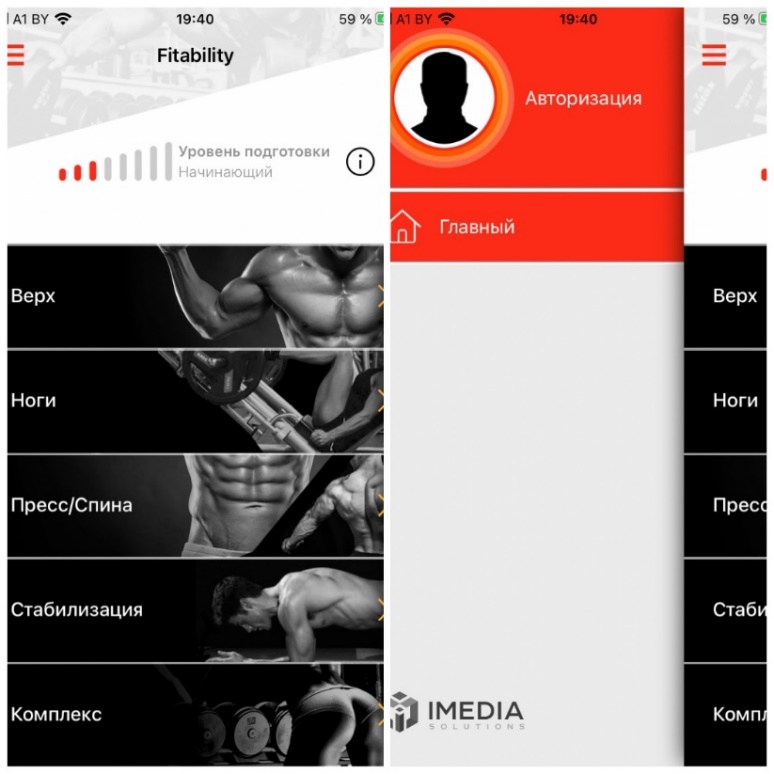


Рисунок 1.2 – Приложение «Fitabillity»

«Freeletics» - фитнес-приложение, дающее возможность заниматься с лучшим цифровым тренером где угодно. ПО представляет из себя систему, состоящую из 22 тренировки с собственным весом, 30 видов упражнений с изображениями, а также 20 аудиосессий. Ниже на рисунке 1.3 представлен внешний вид приложения.

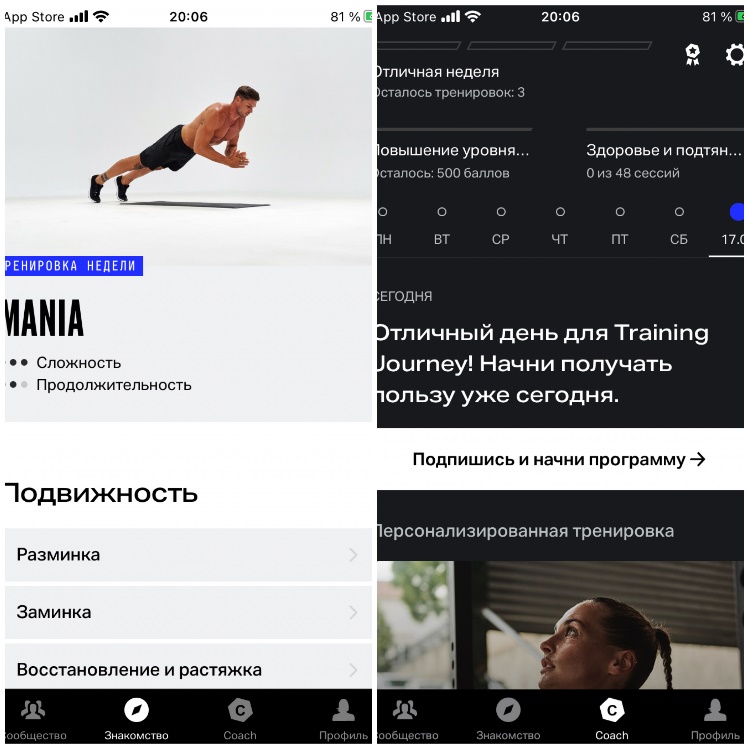


Рисунок 1.3 – Приложение «Freeletics»

2. Постановка задачи

В общем, задачу можно сформулировать следующим образом: проанализировав тему проекта, выбрать необходимые сущности, которые будут отражать хранимые данные. Далее охарактеризовать эти сущности с помощью соответствующих атрибутов, с корректно подобранным типом данных. Следующим шагом является проектирование реляционных таблиц, их последующая нормализация, в ходе которой могут появится новые таблицы, исходя из избыточности первоначальных, «сырых» данных. Далее, таблицы, которые представляют собой определённые сущности хранимых данных, связать между собой, тем самым получив полноценную реляционную структуру.

После проведения необходимых корректировок по спроектированной модели данных, нужно провести анализ таблиц. Необходимо с помощью инструментов, поставляемых СУБД MySQL и языка T-SQL реализовать логику взаимодействия с данными, исходя из целей проекта.

Дополнительно, необходимо реализовать процесс по импорту и экспорту данных в формат XML. Так же следует после создании базы данных заполнить ее необходимым количеством данных, а именно не менее 100000 строк.

1. Проектирование базы данных

Проектирование баз данных — процесс создания схемы базы данных и определения необходимых ограничений целостности.

Основные задачи проектирования базы данных:

* обеспечение хранения в БД всей необходимой информации;
* обеспечение возможности получения данных по всем необходимым запросам;
* сокращение избыточности и дублирования данных;
* обеспечение целостности базы данных.

Проектирование базы данных проводится в два этапа: концептуальное и логическое проектирование.

Концептуальное проектирование — построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня абстракции. Такая модель создаётся без ориентации на какую-либо конкретную СУБД и модель данных. Основными понятиями модели являются: сущность, связь и атрибут.

Сущность – это реальный или представляемый объект, информация о котором должна сохраняться и быть доступна.

Связь – это графически изображаемая ассоциация, устанавливаемая между двумя сущностями. Эта ассоциация обычно является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь).

Атрибут сущности − это любая деталь, которая служит для уточнения, идентификации, классификации, числовой характеристики или выражения состояния сущности.

В рамках этого этапа была создана модель, которая включает 5 сущностей:

* пользователи;
* тренировки;
* диеты;
* избранные тренировки;
* избранные диеты;
* завершенные тренировки.

В ходе проектирования должны быть выполнены следующие требования:

* доступ к данным должен осуществляться только через соответствующие процедуры;
* должен быть проведен импорт данных из XML файлов, экспорт данных в формат XML;
* необходимо протестировать производительность базы данных на таблице, содержащей не менее 100 000 строк, и внести изменения в структуру в случае необходимости. Необходимо проанализировать планы запросов к таблице;
* применить технологию базы данных согласно выбранной теме: подробно описать применяемые системные пакеты, утилиты или технологии; показать применение указанной технологии в базе данных.

1. Разработка объектов базы данных

Первым этапом курсового проекта будет создание логически взаимосвязанных таблиц. Чтобы составить визуальную взаимосвязанную структуру базы данных, нам необходимо продумать, какая информация будет храниться в этих таблицах, после этого создать связи с помощью первичных и внешних ключей.

Логическое проектирование — создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных. Для реляционной модели данных логическая модель — набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи. На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД. Логическая модель базы данных представлена на рисунке 3.1.

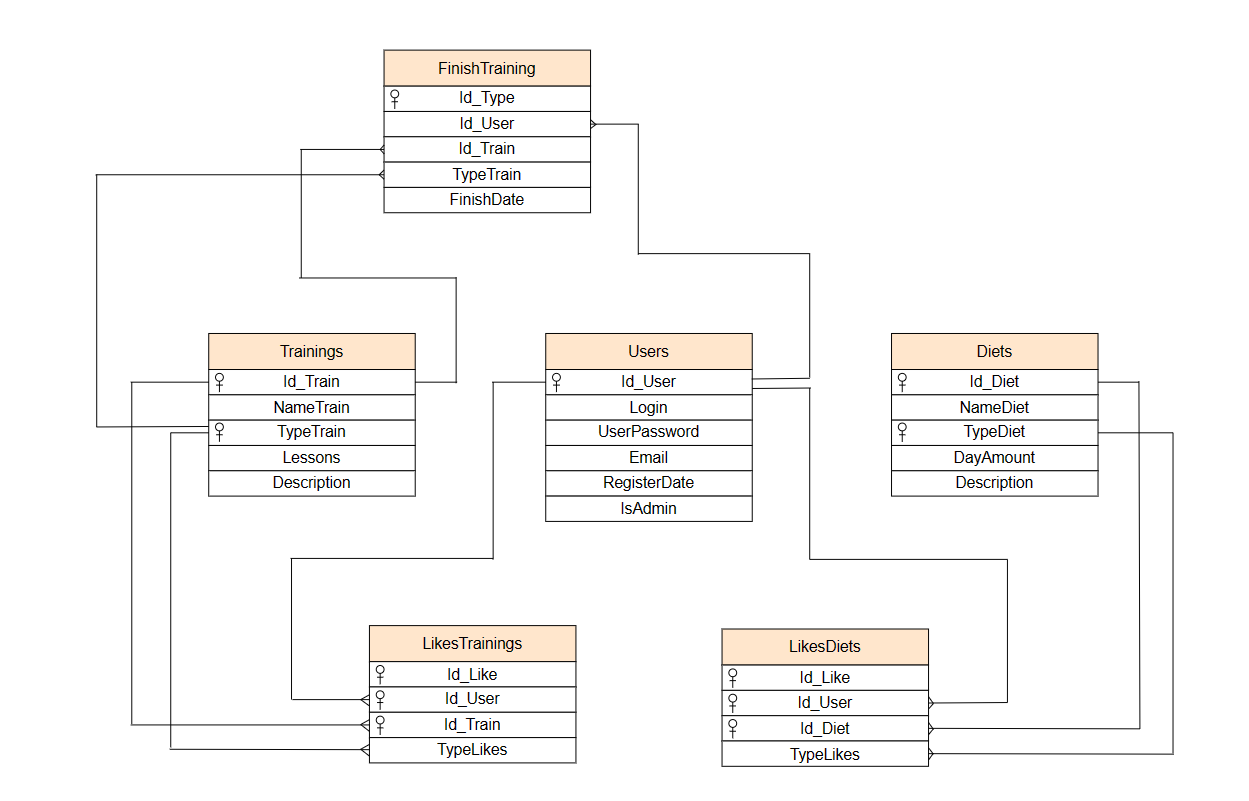


Рис. 3.1 – Логическая модель базы данных

* 1. Таблицы

Центральным объектом любой базы данных является связанная совокупность таблиц. Таблицы ограничивают и упорядочивают хранимую информацию, а также обеспечивают связанность за счет внешних ключей.

Всего в базе данных содержится 6 таблиц:

* Users;
* Diets;
* Trainings;
* LikesTrainings;
* LikesDiets;
* FinishTraining.

Таблица Users представляет список пользователей, состоит из столбцов, которые представлены в таблице 3.1. Поле «IsAdmin» - метка, которая показывает роль пользователя (1 - администратор, 0 - обычный пользователь):

Таблица 3.1 – Таблица «Users»

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| Id\_User | Идентификатор пользователя | Number |
| Login | Логин | Varchar |
| UserPassword | Пароль | Varchar |
| Email | Электронная почта | Varchar |
| RegisterDate | Дата регистрации | Date |
| IsAdmin | Роль пользователя | Number |

Таблица «Diets» содержит в себе список диет (таблица 3.2):

Таблица 3.2 – Таблица «Diets»

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| Id\_Diets | Идентификатор диеты | Number |
| NameDiet | Название диеты | Varchar |
| TypeDiet | Тип диеты | Varchar |
| DayAmount | Количество дней диеты | Varchar |
| Description | Описание диеты | Varchar |

Таблица «Trainigs» содержит в себе список тренировок (таблица 3.3):

Таблица 3.3 – Таблица «Trainigs»

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| Id\_Train | Идентификатор тренировки | Number |
| NameTrain | Название тренировки | Varchar |
| TypeTrain | Тип тренировки | Varchar |
| Description | Описание тренировки | Varchar |
| Lessons | Количество уроков | Number |

Таблица «LikesTrainings» представляет из себя список избранных тренировок (таблица 3.4):

Таблица 3.4 – Таблица «LikesTrainings»

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| Id\_Like | Идентификатор избранного | Number |
| Id\_User | Идентификатор пользователя | Number |
| Id\_Train | Идентификатор тренировки | Number |
| TypeLikes | Тип тренировки | Varchar |

Таблица «LikesDiets» представляет из себя список избранных тренировок (таблица 3.5):

Таблица 3.5 – Таблица «LikesDiets»

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| Id\_Like | Идентификатор избранного | Number |
| Id\_User | Идентификатор пользователя | Number |
| Id\_Diet | Идентификатор диеты | Number |
| TypeLikes | Тип диеты | Varchar |

Таблица «FinishTraining» содержит в себе список пройденных тренировок пользователя (таблица 3.6):

Таблица 3.6 – Таблица «FinishTraining»

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| Id\_Type | Идентификатор завершенной тренировки | Number |
| Id\_User | Идентификатор пользователя | Number |
| Id\_Train | Идентификатор тренировки | Number |
| TypeTrain | Тип тренировки | Varchar |
| FinishDate | Дата прохождения тренировки | Date |

Листинги создания таблиц представлены в приложении А.

* 1. Процедуры

Хранимая процедура – это поименованный код на языке T-SQL. Хранимая процедура может быть создана с помощью CREATE, изменена с помощью ALTER и удалена с помощью оператора DROP. Процедура может принимать входные и формировать выходные параметры. Результатом ее выполнения может быть целочисленное значение, которое возвращается к точке вызова оператором RETURN, либо один или более результирующих наборов, сформированных операторами SELECT, либо содержимое стандартного выходного потока, полученного при выполнении операторов PRINT. Вызов процедуры осуществляется оператором EXECUTE (EXEC). В хранимых процедурах допускается применение основных DDL, DML и TCL-операторов, конструкций TRY/CATCH, курсоров, временных таблиц. Листинг SQL-кода для создания процедур, выполняющих вставку информации, находится в приложении Б.

Процедуры, разработанные в рамках курсового проекта, выполняющие чтение информации:

* getAllUsers – получить всех пользователей;
* getDefiniteDateUsers – вывести пользователя, который зарегистрирован позже определенной даты;
* getUsersLogin – получить пользователя по логину;
* getAllTrainings – получить все тренировки;
* getAllDiets – получить все диеты;
* getAllLikesTrains – получить все избранные тренировки;
* getAllLikesTrainUser – получить все избранные тренировки определенного пользователя;
* getAllLikesDiets – получить все избранные диеты;
* getAllLikesDietUser – получить все избранные диеты определенного пользователя;
* getAllFinishTrain – получить все завершенные тренировки;
* getAllFinishTrainUser – получить все завершенные тренировки определенного пользователя.

Процедуры, разработанные в рамках курсового проекта, выполняющие вставку информации:

* addUser – добавить пользователя;
* addTraining – добавить новую тренировку;
* addDiet – добавить новую диету;
* addLikesTraining – добавить тренировку в избранное;
* addLikesTraining - добавить диету в избранное.

Процедуры, разработанные в рамках курсового проекта, выполняющие изменение информации в базе данных:

* changeTrainById – изменить название тренировки по ее индентификатору;
* changeTrainByName – изменить количество упражнений тренировки по ее названию;
* changeDietById – изменить тип диеты по ее идентификатору;
* changeDietByName – изменить описание диеты по ее названию.

Процедуры, разработанные в рамках курсового проекта, выполняющие удаление записей из базы данных:

* deleteUserByID – удалить пользователя по его идентификатору;
* deleteTrainById – удалить тренировку по ее идентификатору;
* deleteDietById – удалить диету по ее идентификатору;
* deleteLikesTrainById – удалить тренировку из избранного по ее идентификатору;
* deleteLikesDietById – удалить диету из избранного по ее идентификатору;
* deleteFinishTrainById – удалить завершенную тренировку по ее идентификатору.
* deleteFinishTrainByIdUser – удалить завершенную тренировку по идентификатору пользователя.

Листинги создания таблиц представлены в приложении Б.

2. Описание процедур импорта и экспорта

Одной из задач по разработке инфраструктуры базы данных являлась реализация процедур, которые будут осуществлять экспорт и импорт данных в формате XML. XML — это формат данных, используемый для обмена данными в форме, которая может быть легко использована и распространена.

* 1. Описание процедуры импорта

Для импорта данных из XML-файла необходимо создать процедуру, которая представлена на листинге 5.1 после чего выполнить.

|  |
| --- |
| CREATE OR REPLACE PROCEDURE ImportXMLUsers(FileName VARCHAR2)  AS  FileContent XMLTYPE;  IdUser number(10) ;  LoginUser NVARCHAR2(50);  UserPass NVARCHAR2(50);  EmailUser NVARCHAR2(50);  RegDate DATE;  AdminOrNot number(10);  i INT DEFAULT 1;  BEGIN  FileContent := ReadFileForImport(FileName);  LOOP  BEGIN  SELECT EXTRACTVALUE(FileContent, '/data/row['||i||']/ID\_USER') INTO IdUser FROM DUAL;  SELECT EXTRACTVALUE(FileContent, '/data/row['||i||']/LOGIN') INTO LoginUser FROM DUAL;  SELECT EXTRACTVALUE(FileContent, '/data/row['||i||']/USERPASSWORD') INTO UserPass FROM DUAL;  SELECT EXTRACTVALUE(FileContent, '/data/row['||i||']/EMAIL') INTO EmailUser FROM DUAL;  SELECT TO\_DATE(EXTRACTVALUE(FileContent, '/data/row['||i||']/REGISTERDATE'), 'yyyy-mm-dd') INTO RegDate FROM DUAL;  SELECT EXTRACTVALUE(FileContent, '/data/row['||i||']/ISADMIN') INTO AdminOrNot FROM DUAL;  IF IdUser IS NULL THEN EXIT;  END IF;  insert into users2(Id\_User, Login, UserPassword, Email, RegisterDate, IsAdmin)  values (IdUser, LoginUser, UserPass, EmailUser, RegDate, AdminOrNot);  i := i + 1;  END;  END LOOP;  exception  when others then dbms\_output.put\_line('Error');  END; |

Листинг 5.1 – Процедура импорта данных из XML-файла

Для того, чтобы получить строку из файла, была создана вспомогательная функция под названием ReadFileForImport и ее код можно увидеть на листинге 5.2.

|  |
| --- |
| CREATE OR REPLACE FUNCTION ReadFileForImport(FileName NVARCHAR2)  RETURN XMLTYPE  AS  FileContent CLOB;  Buffer CLOB;  FileDescriptor UTL\_FILE.FILE\_TYPE;  BEGIN  FileDescriptor := UTL\_FILE.FOPEN('INFILEXML', FileName, 'R');  LOOP  BEGIN  UTL\_FILE.GET\_LINE(FileDescriptor, Buffer);  FileContent := FileContent || Buffer;  EXCEPTION WHEN NO\_DATA\_FOUND THEN EXIT;  END;  END LOOP;  RETURN XMLTYPE(FileContent);  END; |

Листинг 5.2 – Процедура для чтения файла

В купе эти функции дают возможность импортировать данные в таблицу Users.

* 1. Описание процедуры экспорта

Возможность экспорта данных из БД предоставлена самим IDE и не нуждается в дополнительном вмешательстве с функциональной стороны. Экспорт можно осуществить при помощи нажатия на схему данных правой кнопкой мыши, затем необходимо выбрать пункт меню «Export Data to Files», далее выбираем формат XML и директорию, после чего объекты для экспорта и далее можно нажать на кнопку «Export to Files». Дальнейшие действия не требуются и можно уже использовать файлы в формате XML для последующей обработки.

1. Тестирование производительности

Для проверки производительности базы данных необходимо заполнить ее большим количеством различных данных и узнать время выполнения одного запроса.

Для данной задачи мы создали анонимный блок, и так как работаем именно с действующей базой данных, то можем использовать готовую процедуру для вставки данных в таблицу. Разработанный анонимный блок позволяет добавить большое количество строк за одно выполнение представлен на рисунке 6.1.

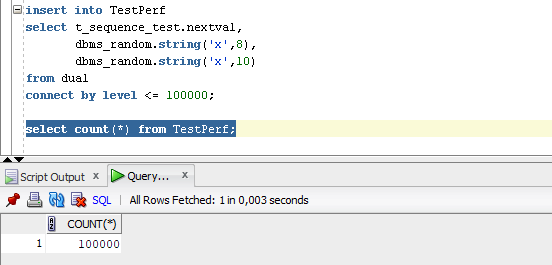


Рисунок 6.1 – Заполнение таблицы строками

Таким образом было добавлено 100000 строк в таблицу.

Построим план запроса по всем элементам таблицы и посмотрим время его выполнения (рисунок 6.2)

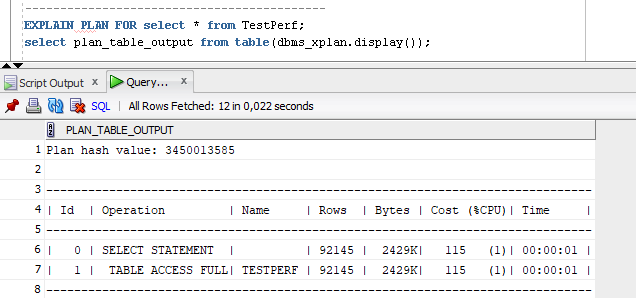


Рисунок 6.2 – План запроса

Как видим наша база данных успешно прошла тест на производительность.

1. Описание технологии и ее применение в базе данных

В процессе разработки проекта была использована технология мониторинга состояния СУБД.

Real Time SQL Monitoring - это инструмент, доступный в Oracle SQL Developer, который позволяет отслеживать выполнение SQL-запросов в реальном времени и обнаруживать проблемы во время выполнения. Он предоставляет подробную информацию о том, как выполняется запрос, и позволяет быстро выявлять узкие места и другие проблемы.

Для того чтобы включить Real Time SQL Monitoring в SQL Developer, необходимо выполнить следующее:

Выполнить SQL-запрос, выполняющий вставку строк в таблицу “TestPerf” который мы хотим отслеживать (рисунок 7.1).

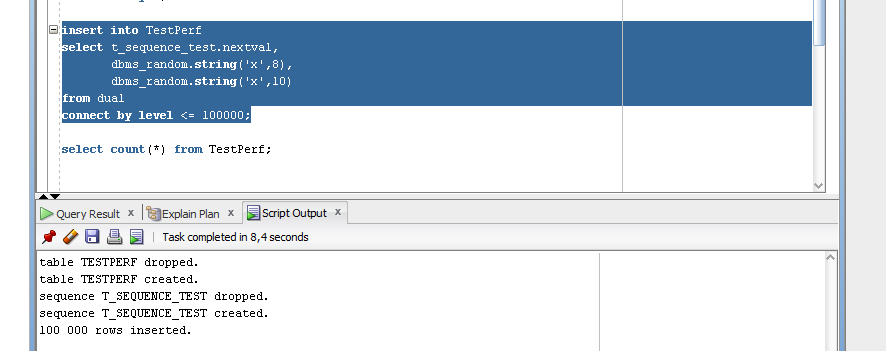


Рисунок 7.1 – SQL-запрос

После того, как запрос начнет выполняться, открываем окно "Real Time SQL Monitoring", нажав на кнопку "Real Time SQL Monitoring" в SQL Developer (рисунок 7.2). В окне Real Time SQL Monitoring мы видим детальную информацию о выполнении запроса. Мы можем просмотреть статистику выполнения запроса, план выполнения, активные сессии и многое другое.

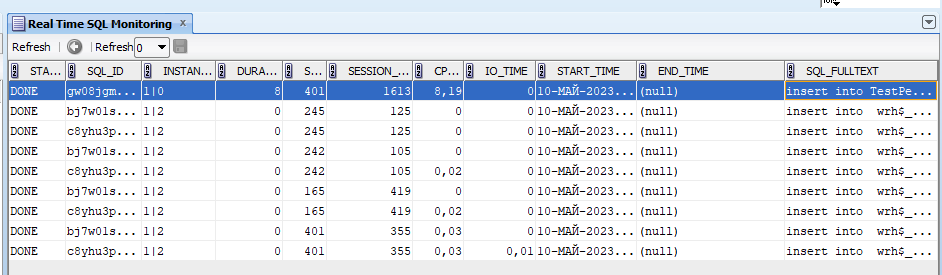


Рисунок 7.2 – Окно Real Time SQL Monitoring

Нажав правой кнопкой мыши по запросу, открывается более детальное окно (рисунок 7.3).

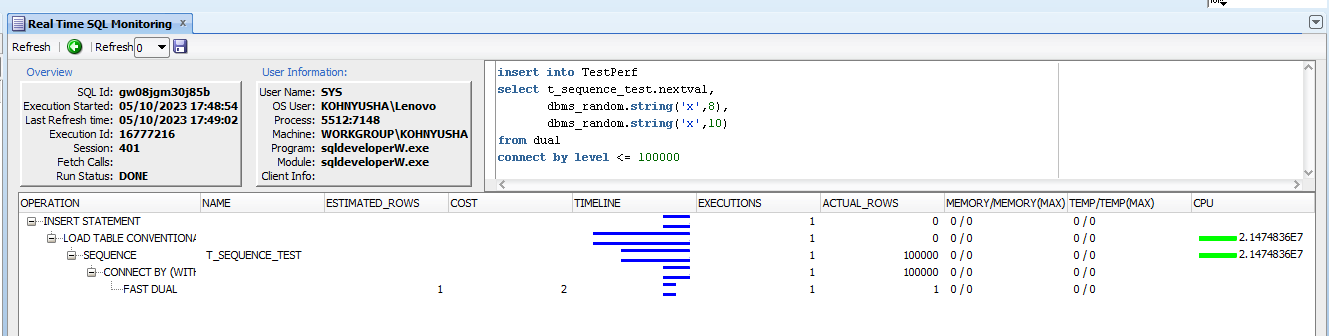


Рисунок 7.3 – Детальное окно выолненного запроса

Real Time SQL Monitoring позволяет быстро выявлять узкие места в выполнении запроса и оптимизировать его для улучшения производительности в будущем. Он также позволяет мониторить запросы с высокой степенью детализации и обнаруживать проблемы, которые могут быть пропущены другими инструментами мониторинга.

1. Руководство пользователя

Для проверки работоспособности, необходимо в первую очередь установить систему управления базы данных Oracle Database 12c, указав порт, SID. Далее задать пароли для системных пользователей.

Для подключения создаем новое подключение к базе данных от имени администратора (рисунок 8.1). Для этого создаем юзера, выдаем ему необходимые привилегии и подключаемся.

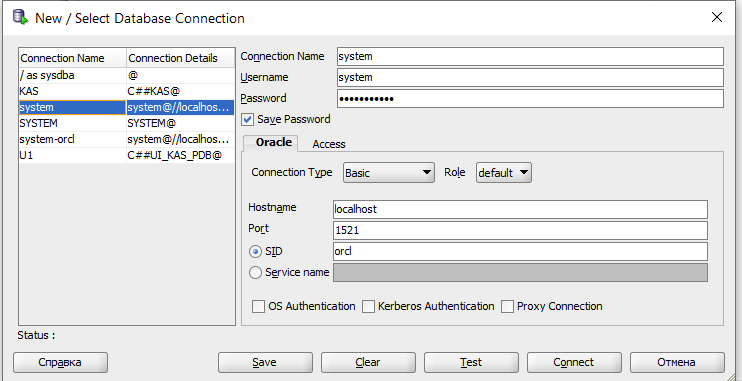


Рисунок 8.1 – Создание соединения под администратором

После подключение под администратора, можно приступать к выполнению процедур, представленных в приложении Б.

Для работы под обычным пользователем, создаем новое подключение (рисунок 8.2). Имя пользователя – «KAS».

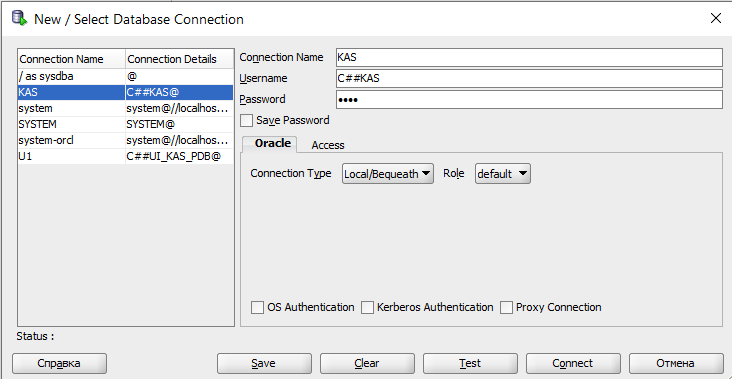


Рисунок 8.2 – Создание соединения под пользователем

Обычному пользователю доступны все нижеперечисленные процедуры:

* getAllTrainings – получить все тренировки;
* getAllDiets - получить все диеты;
* getAllLikesTrainUser – получить избранные тренировки;
* getAllLikesDietUser – получить избранные диеты;
* getAllFinishTrainUser – получить завершенные тренировки;
* addLikesDiet – добавить диету в избранное;
* addLikesTraining – добавить тренировку в избранное;
* deleteLikesTrainById – удалить тренировку из избранного по ее id;
* deleteLikesDietById – удалить диету из избранного по ее id.

Заключение

В ходе выполнения данной работы подробно были закреплены знания для проектирования баз данных, с учётом большого количества хранимых параметров, учётом множества связей и использование процедур и последовательностей для осуществления взаимодействий с таблицами.

В процессе проектирования была достигнута поставленная цель по созданию базы данных «Индивидуальный тренер». В данной работе использовалась СУБД Oracle Database 12c. Было проведено тестирование производительности в котором анализ времени выполнения показал, что база данных работает быстро, даже не в реалистичных условиях.

Была реализована технология «Мониторинг состояния СУБД». В соответствии с полученным результатом можно сделать вывод, что требования технического задания выполнены в полном объёме.

Список используемых источников

1. Фейерштейн С., Прибыл Б. Oracle PL/SQL. Для профессионалов. 6-е изд. [Электронный ресурс] /SQL – Режим доступа: https://obuchalochka.org/20201105126592/oracle-pl-sql-dlya-professionalov-feiershtein-s-pribil-b-2015 - Дата доступа: 21.04.2023.
2. Том Кайт. Oracle для профессионалов. [Электронный ресурс] /SQL – Режим доступа: https:///777russia.ru/book/uploads/РАЗНОЕ/Oracle Для Профессионалов 2 Тома.pdf - Дата доступа: 21.04.2023.
3. Свободная энциклопедия Википедия [Электронный ресурс] /CRUD – Режим доступа: https://ru.wikipedia.org/wiki/CRUD ­– Дата доступа: 21.04.2023.
4. Свободная энциклопедия Википедия [Электронный ресурс] /SQL – Режим доступа: https://ru.wikipedia.org/wiki/SQL ­– Дата доступа: 21.04.2023.
5. Microsoft [Электронный ресурс] / Transact-SQL Reference (Database Engine) – Режим доступа: https://docs.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-2017 ­– Дата доступа: 21.04.2023.

Свободная энциклопедия Википедия [Электронный ресурс] /XML – Режим доступа: https://ru.wikipedia.org/wiki/XML ­– Дата доступа: 21.04.2023.

Приложение А

|  |
| --- |
| create table Users  (  Id\_User number(10) primary key,  Login varchar(50) not null,  UserPassword varchar(50) not null,  Email varchar(50) not null,  RegisterDate date not null,  IsAdmin number(10) default(0)  );  create table Trainings  (  Id\_Train number(10) not null,  NameTrain varchar(50) not null,  TypeTrain varchar(100) not null,  Lessons number(10) not null,  Description varchar(400) not null,  primary key (Id\_Train, TypeTrain)  );  create table Diets  (  Id\_Diet number(10) not null,  NameDiet varchar(20) not null,  TypeDiet varchar(20) not null,  DayAmount number(10) not null,  Description varchar(200) not null,  primary key (Id\_Diet, TypeDiet)  );  create table LikesTrainings  (  Id\_Like number(10) generated by default as identity,  Id\_User number(10) not null,  Id\_Train number(10) not null,  TypeLikes varchar(20) not null,  primary key (Id\_User, Id\_Like, TypeLikes),  constraint fk\_id\_user foreign key (Id\_User) references Users(Id\_User),  constraint fk\_type\_t foreign key (Id\_Train, TypeLikes) references Trainings(Id\_Train, TypeTrain)  );  create table LikesDiets  (  Id\_Like number(10) generated by default as identity,  Id\_User number(10) not null,  Id\_Diet number(10) not null,  TypeLikes varchar(20) not null,  primary key (Id\_User, Id\_Like, TypeLikes),  constraint fk\_id\_user2 foreign key (Id\_User) references Users(Id\_User),  constraint fk\_type\_d foreign key (Id\_Diet, TypeLikes) references Diets(Id\_Diet, TypeDiet)  );  create table FinishTraining  (  Id\_Type number(10) generated by default as identity,  Id\_User number(10) not null,  Id\_Train number(10) not null,  TypeTrain varchar(20) not null,  FinishDate date not null,  constraint fk\_id\_user3 foreign key (Id\_User) references Users(Id\_User),  constraint fk\_type\_training foreign key (Id\_Train, TypeTrain) references Trainings(Id\_Train, TypeTrain)  ); |

Приложение Б

|  |
| --- |
| create or replace procedure getAllUsers  is  cursor curs is select users.Login from users;  Login users.Login%type;  begin  open curs;  loop  fetch curs into Login;  exit when curs%notfound;  dbms\_output.put\_line(Login);  end loop;  close curs;  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  getAllUsers();  end;  create or replace procedure getDefiniteDateUsers(DefiniteDate users.registerdate%type)  is  cursor curs is select users.id\_user, users.Login, users.registerdate  from users  where users.registerdate >= DefiniteDate;  id users.id\_user%type;  Login users.Login%type;  reg users.registerdate%type;  begin  open curs;  loop  fetch curs into id, Login, reg;  exit when curs%notfound;  dbms\_output.put\_line(id||' '||Login||' '||reg);  end loop;  close curs;  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  getDefiniteDateUsers('20.04.2023');  end;  create or replace procedure getUsersLogin(loginuser users.login%type)  is  cursor curs is select users.id\_user, users.Login, users.registerdate  from users  where users.login = loginuser;  id users.id\_user%type;  Login users.Login%type;  reg users.registerdate%type;  begin  open curs;  loop  fetch curs into id, Login, reg;  exit when curs%notfound;  dbms\_output.put\_line(id||' '||Login||' '||reg);  end loop;  close curs;  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  getUsersLogin('Admin');  end;  create or replace procedure addUser (id\_user users.id\_user%type, login users.login%type, userpassword users.userpassword%type, email users.email%type, registerdate users.registerdate%type)  is  begin  insert into users(id\_user, login, userpassword, email, registerdate) values (id\_user, login, userpassword, email, registerdate);  SYS.dbms\_output.put\_line('User was inserted');  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  addUser(t\_sequence\_users.nextval, 'login123', 'userpassword', 'email', '06.05.2023');  end;  create or replace procedure deleteUserByID(id USERS.id\_user%type)  is  begin  delete from users where users.id\_user = id;  SYS.dbms\_output.put\_line('User was deleted');  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  deleteUserByID(22);  end;  create or replace procedure deleteUserByLogin(login USERS.login%type)  is  begin  delete from users where users.login = login;  SYS.dbms\_output.put\_line('User was deleted');  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  deleteUserByLogin('NewUser5JU');  end;  create or replace procedure getAllTrainings  is  cursor curs is select trainings.nametrain from trainings;  nametrain trainings.nametrain%type;  begin  open curs;  loop  fetch curs into nametrain;  exit when curs%notfound;  dbms\_output.put\_line(nametrain);  end loop;  close curs;  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  getAllTrainings();  end;  create or replace procedure addTraining (id\_training trainings.id\_train%type, nametrain trainings.nametrain%type, typetrain trainings.typetrain%type, lessons trainings.lessons%type, description trainings.description%type)  is  begin  insert into trainings(id\_train, nametrain, typetrain, lessons, description) values (id\_training, nametrain, typetrain, lessons, description);  SYS.dbms\_output.put\_line('Training was inserted');  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  addTraining(t\_sequence\_trainings.nextval, 'Новая тренировка', '5-Разминка', 20, 'Не спеши сразу же переходить к упражнениям.');  end;  create or replace procedure changeTrainById(nametraining trainings.nametrain%type, id trainings.id\_train%type)  is  begin  update trainings SET nametrain = nametraining where id\_train = id;  SYS.dbms\_output.put\_line('Training was udpdate');  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  ChangeTrainById('UpdateName', 16);  end;  create or replace procedure changeTrainByName(lessonstrain trainings.lessons%type, nametraining trainings.nametrain%type)  is  begin  update trainings SET lessons = lessonstrain where nametrain = nametraining;  SYS.dbms\_output.put\_line('Training was udpdate');  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  ChangeTrainByName(100, 'UpdateName');  end;  create or replace procedure deleteTrainById(id trainings.id\_train%type)  is  begin  delete from trainings where trainings.id\_train = id;  SYS.dbms\_output.put\_line('Training was deleted');  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  deleteTrainById(23);  end;  create or replace procedure getAllDiets  is  cursor curs is select diets.namediet from diets;  namediet diets.namediet%type;  begin  open curs;  loop  fetch curs into namediet;  exit when curs%notfound;  dbms\_output.put\_line(namediet);  end loop;  close curs;  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  getAllDiets();  end;  create or replace procedure addDiet(id\_diet diets.id\_diet%type, namediet diets.namediet%type, typediet diets.typediet%type, dayamount diets.dayamount%type, description diets.description%type)  is  begin  insert into diets(id\_diet, namediet, typediet, DayAmount, description) values (id\_diet, namediet, typediet, DayAmount, description);  SYS.dbms\_output.put\_line('Diet was inserted');  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  addDiet(t\_sequence\_diets.nextval, 'Новая', '8-Ужин', 7, 'Результат уже через неделею!');  end;  create or replace procedure changeDietById(typed diets.typediet%type, id diets.id\_diet%type)  is  begin  update diets SET typediet = typed where id\_diet = id;  SYS.dbms\_output.put\_line('Diet was udpdate');  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  ChangeDietById('10-Ужин', 4);  end;  create or replace procedure changeDietByName(namedi diets.namediet%type, descr diets.description%type)  is  begin  update diets SET description = descr where namediet = namedi;  SYS.dbms\_output.put\_line('Diet was udpdate');  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  changeDietByName('Куриное филе и овощи', 'Новое описание, Новое описание, Новое описание');  end;  create or replace procedure deleteDietById(id diets.id\_diet%type)  is  begin  delete from diets where diets.id\_diet = id;  SYS.dbms\_output.put\_line('Diet was deleted');  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  deleteDietById(25);  end;  create or replace procedure getAllLikesTrains  is  cursor curs is select likestrainings.id\_like, likestrainings.typelikes from likestrainings;  id\_like likestrainings.id\_like%type;  typelikes likestrainings.typelikes%type;  begin  open curs;  loop  fetch curs into id\_like, typelikes;  exit when curs%notfound;  dbms\_output.put\_line(id\_like ||' '|| typelikes);  end loop;  close curs;  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  getAllLikesTrains();  end;  create or replace procedure getAllLikesTrainUser(id\_u likestrainings.id\_user%type)  is  cursor curs is select likestrainings.id\_like, likestrainings.typelikes from likestrainings  where id\_user = id\_u;  id\_like likestrainings.id\_like%type;  typelikes likestrainings.typelikes%type;  begin  open curs;  loop  fetch curs into id\_like, typelikes;  exit when curs%notfound;  dbms\_output.put\_line(id\_like ||' '|| typelikes);  end loop;  close curs;  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  getAllLikesTrainUser(5);  end;  create or replace procedure addLikesTraining  (  id\_user diets.id\_diet%type,  id\_train diets.namediet%type,  typelikes diets.typediet%type  )  is  begin  insert into likestrainings(id\_user, id\_train, typelikes) values (id\_user, id\_train, typelikes);  SYS.dbms\_output.put\_line('LikesTraining was inserted');  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  create or replace procedure deleteLikesTrainById(id likestrainings.id\_train%type)  is  begin  delete from likestrainings where likestrainings.id\_train = id;  SYS.dbms\_output.put\_line('Training was deleted from favorites');  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  deleteLikesTrainById(7);  end;  create or replace procedure getAllLikesDiets  is  cursor curs is select likesdiets.id\_like, likesdiets.typelikes from likesdiets;  id\_like likesdiets.id\_like%type;  typelikes likesdiets.typelikes%type;  begin  open curs;  loop  fetch curs into id\_like, typelikes;  exit when curs%notfound;  dbms\_output.put\_line(id\_like ||' '|| typelikes);  end loop;  close curs;  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  getAllLikesDiets();  end;  create or replace procedure getAllLikesDietUser(id\_u likesdiets.id\_user%type)  is  cursor curs is select likesdiets.id\_like, likesdiets.typelikes from likesdiets  where id\_user = id\_u;  id\_like likesdiets.id\_like%type;  typelikes likesdiets.typelikes%type;  begin  open curs;  loop  fetch curs into id\_like, typelikes;  exit when curs%notfound;  dbms\_output.put\_line(id\_like ||' '|| typelikes);  end loop;  close curs;  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  getAllLikesDietUser(3);  end;  create or replace procedure addLikesDiet  (  id\_user diets.id\_diet%type,  id\_diet diets.namediet%type,  typelikes diets.typediet%type  )  is  begin  insert into likesdiets(id\_user, id\_diet, typelikes) values (id\_user, id\_diet, typelikes);  SYS.dbms\_output.put\_line('LikesDiet was inserted');  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  create or replace procedure deleteLikesDietById(id likesdiets.id\_diet%type)  is  begin  delete from likesdiets where likesdiets.id\_diet = id;  SYS.dbms\_output.put\_line('Diet was deleted from favorites');  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  deleteLikesDietById(8);  end;  create or replace procedure getAllFinishTrain  is  cursor curs is select finishtraining.id\_type, finishtraining.typetrain from finishtraining;  id\_type finishtraining.id\_type%type;  typetrain finishtraining.typetrain%type;  begin  open curs;  loop  fetch curs into id\_type, typetrain;  exit when curs%notfound;  dbms\_output.put\_line(id\_type ||' '|| typetrain);  end loop;  close curs;  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  getAllFinishTrain();  end;  create or replace procedure getAllFinishTrainUser(id\_u finishtraining.id\_user%type)  is  cursor curs is select finishtraining.id\_type, finishtraining.typetrain from finishtraining  where id\_user = id\_u;  id\_type finishtraining.id\_type%type;  typetrain finishtraining.typetrain%type;  begin  open curs;  loop  fetch curs into id\_type, typetrain;  exit when curs%notfound;  dbms\_output.put\_line(id\_type ||' '|| typetrain);  end loop;  close curs;  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  getAllFinishTrainUser(9);  end;  create or replace procedure deleteFinishTrainById(id finishtraining.id\_train%type)  is  begin  delete from finishtraining where finishtraining.id\_train = id;  SYS.dbms\_output.put\_line('Training was deleted from completed');  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  deleteFinishTrainById(12);  end;  create or replace procedure deleteFinishTrainByIdUser(id finishtraining.id\_user%type)  is  begin  delete from finishtraining where finishtraining.id\_user = id;  SYS.dbms\_output.put\_line('Training was deleted from completed');  exception  when others  then SYS.dbms\_output.put\_line(sqlcode || sqlerrm);  end;  begin  deleteFinishTrainByIdUser(10);  end; |