**УО «Белорусский государственный технологический университет»**

Факультет **информационных технологий**

Кафедра **информационных систем и технологий**

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

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОЙ РАБОТЫ**

**по дисциплине «**Администрирование баз данных и приложений»

**тема «**Разработка базы данных “Автошкола” с применением технологии резервного копирования и восстановления данных»

|  |  |  |
| --- | --- | --- |
| **Исполнитель** |  | |
| Студент 3 курса группы1 | подпись, дата | Д. В. Божко  инициалы и фамилия |
|  |  |  |
| **Руководитель** |  |  |
| Ассистент | подпись, дата | М. В. Колмаков  инициалы и фамилия |

|  |  |
| --- | --- |
| **Курсовая работа защищена с оценкой** | **\_\_\_\_\_\_\_\_\_\_** |
| **Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  (подпись) | М. В. Колмаков  инициалы и фамилия |

**Минск 2022**

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра информационных систем и технологий

Утверждаю

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Смелов

подпись инициалы и фамилия

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

**ЗАДАНИЕ**

**к курсовому проектированию**

**по дисциплине** "Администрирование баз данных и приложений"

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

Группа: 1

Студент: Божко Денис Владимирович

**Тема: «**Разработка базы данных «Автошкола» с применением технологии резервного копирования и восстановления данных»

**1. Срок сдачи студентом законченной работы**: «06» мая 2022 г.

**2. Исходные данные к проекту:**

**2.1**. Функционально должны быть выполнены следующие задачи:

* Управление базой данных (администраторский состав, преподавательский состав, ученики, автомобили);
* Изменение информации о преподавателях, учениках, группах;
* Вывод отчетов о группе и академической успеваемости учеников;
* Вывод данных об учениках.

**2.2. Требования.**

* База данных должна быть реализована в СУБД Oracle 19c.
* Доступ к данным должен осуществляться только через соответствующие процедуры.
* Количество объектов БД (таблиц, представлений, индексов, пользователей и пр.) регламентируется задачей.
* 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.
* Листинги проекта должны содержать комментарии.

**3. Содержание расчетно-пояснительной записки**

* Введение
* Постановка задачи
* Проектирование базы данных.
* Разработка объектов базы данных
* Описание процедур импорта и экспорта
* Тестирование производительности
* Описание технологии и ее применения в базе данных
* Краткое описание приложения для демонстрации
* Руководство пользователя
* Заключение
* Список используемых источников
* Приложения

**4. Форма представления выполненного курсового проекта:**

* + Пояснительная записка оформляется в MS Word.
  + Оформление пояснительной записки должно соответствовать требованиям к оформлению пояснительной записки для курсовых работ.
  + Листинги всех скриптов представляются в приложении.
  + К записке необходимо приложить DVD-диск, который должен содержать пояснительную записку, листинги и файлы базы данных.

#### Календарный план

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Введение | 06.03.2022 |  |
| 2 | Аналитический обзор литературы по теме проекта | 12.03.2022 |  |
| 3 | Изучение требований, определение вариантов использования | 18.03.2022 |  |
| 4 | Анализ и проектирование модели базы данных. Описание информационных объектов и ограничений целостности. | 25.03.2022 |  |
| 5 | Создание необходимых объектов | 11.04.2022 |  |
| 6 | Импорт и экспорт данных | 17.04.2022 |  |
| 7 | Описание используемой технологии | 23.04.2022 |  |
| 8 | Тестирование производительности | 27.04.2022 |  |
| 9 | Оформление пояснительной записки | 01.05.2022 |  |
| 10 | Сдача проекта | 06.05.2022 |  |

**5. Дата выдачи задания** «17» февраля 2022 г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *М.В. Колмаков*

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата и подпись студента)

**Оглавление**

[Введение 5](#_Toc103018998)

[1. Постановка задачи 6](#_Toc103018999)

[2. Разработка модели базы данных 7](#_Toc103019000)

[3. Разработка необходимых объектов 8](#_Toc103019001)

[3.1. Таблицы 8](#_Toc103019002)

[3.2. Пользователи 12](#_Toc103019003)

[3.3. Хранимые процедуры 12](#_Toc103019004)

[4. Описание процедур импорта и экспорта 13](#_Toc103019005)

[5. Тестирование производительности 15](#_Toc103019006)

[6. Описание технологии и ее применения в базе данных 17](#_Toc103019007)

[7. Руководство пользователя 20](#_Toc103019008)

[Заключение 22](#_Toc103019009)

[Cписок используемых источников 23](#_Toc103019010)

[ПРИЛОЖЕНИЕ А 24](#_Toc103019011)

[ПРИЛОЖЕНИЕ Б 27](#_Toc103019012)

[ПРИЛОЖЕНИЕ В 28](#_Toc103019013)

[ПРИЛОЖЕНИЕ Г 36](#_Toc103019014)

[ПРИЛОЖЕНИЕ Д 38](#_Toc103019015)

## **Введение**

Любая организация нуждается в своевременном доступе к информации. Ценность информации в современном мире очень высока. Роль распорядителей информации в современном мире чаще всего выполняют базы данных. Базы данных обеспечивают надежное хранение информации, в структурированном виде и своевременный доступ к ней. Практически любая современная организация нуждается в базе данных, удовлетворяющей те или иные потребности по хранению, управлению и администрированию данных.

За последние несколько лет наблюдается тенденция к усложнению структур данных. Простые виды информации, представимой в форме чисел и текстовых строк, не утратив своей значимости, дополняются сегодня многочисленными мультимедийными документами, графическими образами, хронологическими рядами, процедурными, или активными, данными и мириадами прочих сложных информационных форм.

На сегодняшний день на рынке представлено множество технологий доступа к данным и серверов баз данных, каждое, из которых имеет свои отличительные черты. Современные приложения обработки данных ориентированы на работу с большим количеством пользователей, на их удаленность от места расположения основного сервера БД.

Темой данного курсового проекта является разработка базы данных «Автошкола» с применением технологии резервного копирования.

База данных «Автошкола» будет содержать соответствующие таблицы, процедуры, функции и предназначена для взаимодействия с источником данных. Взаимодействие подразумевает получение данных, их представление в определенном формате для просмотра пользователем, редактирование в соответствии с реализованными в программе бизнес-алгоритмами и возврат обработанных данных обратно в базу данных.

База данных «Автошкола» существенно упрощает работу сотрудников автошколы, а также предоставляет возможность администраторам своевременно вносить необходимые изменения.

# **Постановка задачи**

Задача проекта: разработать архитектуру базы данных, создать процедуры и функции, взаимодействие с которыми будет понятно любому пользователю.

Функционально должны быть выполнены следующие задачи:

* управление базой данных (администраторский состав, преподавательский состав, ученики, автомобили);
* изменение информации о преподавателях, учениках, группах;
* вывод отчетов о группе и академической успеваемости учеников;
* вывод данных об учениках.

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.

# **Разработка модели базы данных**

Первым этапом курсового проекта будет создание логически взаимосвязанных таблиц. Реализовывать мы их будем в СУБД Oracle DataBase 12c [1]. 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.

Диаграмма базы данных, спроектированной в ходе разработки показана на рисунке 2.1.

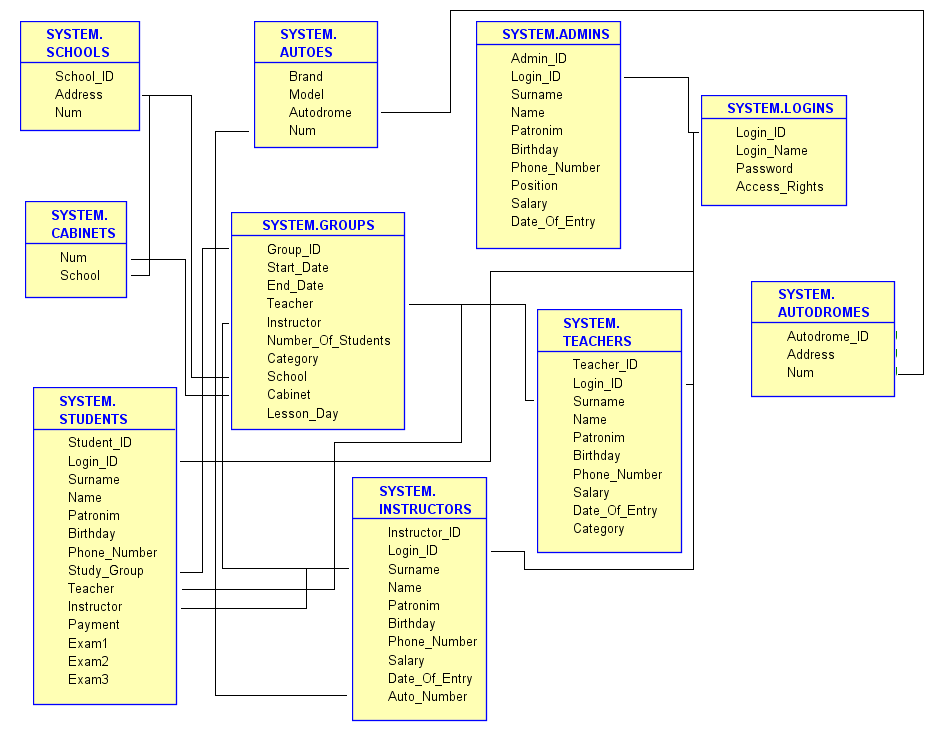


Рисунок 2.1 – Диаграмма базы данных

База данных приложения состоит из 10 таблиц, взаимосвязанных между собой внешними ключами: Logins, Admins, Autoes, Autodromes, Teachers, Instructors, Schools, Cabinets, Groups, Students.

# **Разработка необходимых объектов**

# **Таблицы**

Для реализации базы данных «Автошкола» было разработано 10 таблиц: Logins, Admins, Autoes, Autodromes, Teachers, Instructors, Schools, Cabinets, Groups, Students.

Таблица Logins представляет список логинов и паролей для всей пользователей (таблица 3.1):

Таблица 3.1 – Столбцы таблицы Logins

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| Login\_ID | Идентификатор логина | int |
| Login\_Name | Значение логина | nvarchar2 |
| Password | Пароль | nvarchar2 |
| Access\_Rights | Уровень доступа | nvarchar2 |

Таблица Admins представляет список администраторов автошколы (таблица 3.2):

Таблица 3.2 – Столбцы таблицы Admins

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| Admin\_ID | Идентификатор администратора | int |
| Login\_ID | Идентификатор логина администратора, внешний ключ | int |
| Surname | Фамилия администратора | nvarchar2 |
| Name | Имя администратора | nvarchar2 |
| Patronim | Отчество администратора | nvarchar2 |
| Birthday | Дата рождения администратора | date |
| Phone\_Number | Номер телефона администратора | nvarchar2 |
| Position | Занимаемая позиция в администраторском составе | nvarchar2 |
| Salary | Заработная плата администратора | int |
| Date\_Of\_Entry | Дата приема на работу администратора | date |

Таблица Autoes представляет список автомобилей, имеющихся в распоряжении автошколы (таблица 3.3):

Таблица 3.3 – Столбцы таблицы Autoes

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| Brand | Марка автомобиля | nvarchar2 |
| Model | Модель автомобиля | nvarchar2 |
| Autodrome | Автодром, где стоит автомобиль, внешний ключ | int |
| Num | Номер автомобиля | nvarchar2 |

Таблица Autodromes представляет список автодромов, где автошкола хранит автомобили (таблица 3.4):

Таблица 3.4 – Столбцы таблицы Autodromes

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| Autodrome\_ID | Идентификатор автодрома | int |
| Address | Адрес автодрома | nvarchar2 |
| Num | Номер автодрома | int |

Таблица Teachers представляет список преподавателей автошколы (таблица 3.5):

Таблица 3.5 – Столбцы таблицы Teachers

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| Teacher\_ID | Идентификатор преподавателя | int |
| Login\_ID | Идентификатор логина преподавателя, внешний ключ | int |
| Surname | Фамилия преподавателя | nvarchar2 |
| Name | Имя преподавателя | nvarchar2 |
| Patronim | Отчество преподавателя | nvarchar2 |
| Birthday | Дата рождения преподавателя | date |
| Phone\_Number | Номер телефона преподавателя | nvarchar2 |
| Category | Категория, которой обучает преподаватель | nvarchar2 |
| Salary | Заработная плата преподавателя | int |
| Date\_Of\_Entry | Дата приема на работу преподавателя | date |

Таблица Instructors представляет список инструкторов автошколы (таблица 3.6):

Таблица 3.6 – Столбцы таблицы Instructors

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| Instructor\_ID | Идентификатор инструктора | int |
| Login\_ID | Идентификатор логина инструктора, внешний ключ | int |
| Surname | Фамилия инструктора | nvarchar2 |
| Name | Имя инструктора | nvarchar2 |
| Patronim | Отчество инструктора | nvarchar2 |
| Birthday | Дата рождения инструктора | date |
| Phone\_Number | Номер телефона инструктора | nvarchar2 |
| Auto\_Number | Номер автомобиля, закрепленного за инструктором, внешний ключ | nvarchar2 |
| Salary | Заработная плата инструктора | int |
| Date\_Of\_Entry | Дата приема на работу инструктора | date |

Таблица Schools представляет здания или автошколы, в которых может происходить обучение (таблица 3.7):

Таблица 3.7 – Столбцы таблицы Schools

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| School\_ID | Идентификатор здания | int |
| Address | Адрес здания | nvarchar2 |
| Num | Номер здания | int |

Таблица Cabinets представляет список кабинетов в зданиях атвошколы (таблица 3.8):

Таблица 3.8 – Столбцы таблицы Cabinets

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| Num | Номер кабинеты | int |
| School | Здание, в котором находится кабинет, внешний ключ | nvarchar2 |

Таблица Groups представляет группы студентов, обучающихся в автошколе (таблица 3.9):

Таблица 3.9 – Столбцы таблицы Groups

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| Group\_ID | Идентификатор группы | int |
| Start\_Date | Начало обучения группы | date |
| End\_Date | Конец обучения группы | date |
| Teacher | Преподаватель группы, внешний ключ | nvarchar2 |
| Instructor | Инструктор группы, внешний ключ | nvarchar2 |
| Number\_Of\_Students | Число студентов в группе | int |
| Category | Категория, на которую происходит обучение в группе | nvarchar2 |
| School | Здание, в котором занимается группа, внешний ключ | nvarchar2 |
| Cabinet | Кабинет, в котором занимается группа, внешний ключ | int |
| Lesson\_Day | День недели, в который занимается группа | nvarchar2 |

Таблица Students представляет студентов, обучающихся в автошколе (таблица 3.10):

Таблица 3.10 – Столбцы таблицы Teachers

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| Student\_ID | Идентификатор студента | int |
| Login\_ID | Идентификатор логина студента, внешний ключ | int |
| Surname | Фамилия студента | nvarchar2 |
| Name | Имя студента | nvarchar2 |
| Patronim | Отчество студента | nvarchar2 |
| Birthday | Дата рождения студента | date |
| Phone\_Number | Номер телефона студента | nvarchar2 |
| Study\_Group | Группа, в которой обучается студент | int |
| Teacher | Преподаватель студента, внешний ключ | nvarchar2 |
| Instructor | Инструктор студента, внешний ключ | nvarchar2 |
| Payment | Сумма, которую выплатил студент | int |

Продолжение таблицы 3.10

|  |  |  |
| --- | --- | --- |
| Exam1 | Результат первого экзамена студента | nvarchar2 |
| Exam2 | Результат второго экзамена студента | nvarchar2 |
| Exam3 | Результат третьего экзамена студента | nvarchar2 |

Скрипты создания таблиц и последовательностей представлены в приложении А данной записки.

# **Пользователи**

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

При проектировании базы данных было использовано 4 пользователя: администратор, инструктор, преподаватель и студент. Каждый пользователь имеет разные привилегии в соответствии с его статусом в автошколе.

Все скрипты создания пользователей и их ролей представлены в приложении Б данной записки.

* 1. Хранимые процедуры

Хранимая процедура – объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. Таким образом, процедура будет принимать соответствующие аргументы при каждом ее вызове. Все хранимые процедуры, созданные в данном курсовом проекте, содержат обработку исключений для того, чтобы ограничить пользователя от возможных ошибок.

При разработке курсового проекта было создано множество процедур для следующих целей:

* выборка данных для таблиц;
* добавление/удаление сотрудника;
* добавление/удаление автодрома/автомобиля;
* добавление/удаление школы/кабинета;
* обновление информации о студенте;
* обновление информации о группе;
* экспорт и импорт таблицы в формат xml;
* просмотр информации о студенте;
* просмотр информации о группе.

Все хранимые процедуры представлены в приложении В данной записки.

# **Описание процедур импорта и экспорта**

В курсовом проекте был проведен импорт данных из XML файлов и экспорт данных в формат XML [2].

XML — это формат данных, используемый для обмена данными в форме, которая может быть легко использована и распространена. Часто возникает необходимость импортировать и экспортировать XML-файлы в Oracle, в данной курсовой работе используются пакеты DBMS\_XMLPARSER и DBMS\_XSLPROCESSOR для импорта xml, и DBMS\_XMLDOM для экспорта данных в xml формат. Для работы с файлами так же использовался пакет DBMS\_LOB. Данные пакеты были выбраны, потому что они обладают очень широким функционалом и гибкой настройкой xml документа. В данном курсовом проекте функции экспорта и импорта используются для таблицы Autoes. Пример реализации функции экспорта представлен в листинге 4.1.

|  |
| --- |
| --xml export  SELECT file\_name FROM dba\_data\_files;  create or replace directory UTLDATA AS 'C:/app';  create or replace procedure ExportAutoesToXml  is  DOC DBMS\_XMLDOM.DOMDocument;  XDATA XMLTYPE;  CURSOR XMLCUR IS  SELECT XMLELEMENT("Autoes",  XMLAttributes('http://www.w3.org/2001/XMLSchema' AS "xmlns:xsi",  'http://www.oracle.com/Users.xsd' AS "xsi:nonamespaceSchemaLocation"),  XMLAGG(XMLELEMENT("Autoes",  xmlelement("brand", Autoes.Brand),  xmlelement("model", Autoes.Model),  xmlelement("autodrome", Autoes.Autodrome),  xmlelement("num", Autoes.Num)  ))) from Autoes;  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/Autoes.xml');  END;  begin  ExportAutoesToXml();  end;  -- drop procedure ExportAutoesToXml; |

Листинг 4.1 – Создание процедуры ExportAutoesToXml

Полный код функций экспорта и импорта данных в XML представлен в приложении Г данной записки.

Для сохранения и управления XML-данными в реляционной таблице применяется специальный тип данных XMLType. Тип данных XMLType поставляется с набором специальных XML-методов, которые можно использовать для работы с объектами XMLType. Эти методы можно применять для выполнения как типичных операций в базе данных, наподобие проверки на предмет существования узла или извлечения узла, так и нескольких специальных операций, позволяющих получать доступ к XML-данными и манипулировать ими в виде части обычного SQL-оператора.

# **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 (листинг 5.1).

|  |
| --- |
| begin  for Lcntr IN 1..100000  loop  AddLogin('testing','100000','admin');  end loop;  end; |

Листинг 5.1 – Анонимный блок для заполнения таблицы большим количеством данных

Таким образом было добавлено 100000 строк в таблицу Logins.

После этого в другом анонимном блоке, который содержит в себе дополнительно переменную для отслеживания времени выполнения, был сделан запрос, который должен возвращать в результате 20000 строк. Скрипт данного блока представлен на листинге 5.2.

|  |
| --- |
| declare  t1 number := dbms\_utility.get\_time();  begin  for r in (select \* from Logins where login\_id > 50000 and login\_id < 70000) loop null; end loop;  dbms\_output.put\_line('query elapsed: '||(dbms\_utility.get\_time() - t1)/100);  end; |

Листинг 5.2 – Блок отбора данных

Далее после выполнения данного блока наша таблица будет содержать большое количество данных, и мы можем проанализировать время выполнения запроса. Также можно просмотреть план запроса используя стандартные средства Oracle, а именно кнопку на главной панели, предварительно выделив данный запрос. Покажем результат, в котором будет заметна разница во времени выполнения запроса до создания индекса на рисунке 5.3.

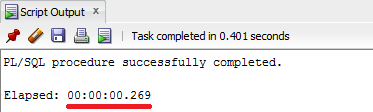


Рисунок 5.3 – Результат отбора данных

Теперь остаётся лишь создать индекс и протестировать время выполнения аналогичного запроса. Покажем это на рисунке 5.4.

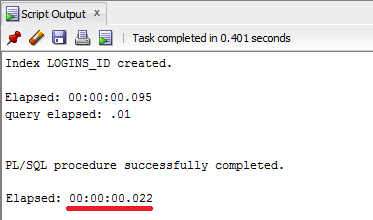


Рисунок 5.4 – Результат после создания индекса

Как видим, время запроса уменьшилось, но тем не менее даже без индекса наша база данных успешно прошла тест на производительность.

Полный скрипт тестирования базы данных на производительность представлен в приложении Д данной записки.

# **Описание технологии и ее применения в базе данных**

В процессе разработки проекта была использована технология: резервное копирование и восстановление данных.

Традиционный пользовательский метод резервного копирования состоит в применении команд операционной системы для копирования необходимых файлов в другое место и/или на ленточное устройство.

В процессе разработки проекта резервное копирование и восстановление данных происходило при помощи утилиты RMAN [3]. Резервное копирование файлов базы данных Oracle выполняться внутри базы данных посредством самого сервера баз данных. RMAN умеет делать резервные копии и копии образов файлов данных, управляющих файлов, архивных журналов повторного выполнения, файлов SPFILE и фрагментов резервных копий RMAN.

Возможности RMAN включают следующее:

* выполнение полного резервирования и резервирования изменений;
* выполнение холодного/горячего резервирования;
* обнаружение поврежденных блоков;
* параллельное выполнения операций ввода/вывода;
* автоматическое протоколирование операций копирования и восстановления.

С помощью RMAN можно выполнять инкрементное резервное копирование. Размер резервных копий в таком случае зависит не от размера базы данных, а скорее от уровня активности внутри нее, поскольку во время инкрементного резервного копирования не измененные блоки пропускаются.

Существует ряд сущностей, которые позволяют утилите RMAN выполнять ее функции в области резервного копирования и восстановления. В данном случае была использована целевая база данных (target database). Так называется база данных, в отношении которой RMAN выполняет резервное копирование. Все операции по резервному копированию и восстановлению осуществляются при помощи запускаемых в целевой базе данных серверных сеансов RMAN.

Подключаться к RMAN можно путем ввода в приглашении операционной системы команды rman. После этого будет появляться приглашение RMAN>, позволяющее вводить различные команды RMAN.

В курсовом проекте был использован метод холодного резервирования. Режим архивирования журналов выключен, СУБД выполняется в режиме NOMOUNT. Очевидный недостаток данного метода: из «холодной» резервной копии можно восстановить только то состояние базы данных, которое было в момент останова; транзакции, сделанные после рестарта базы, в «холодную» резервную копию не попадут;

Для резервирования файлов данных используется команда: BACKUP DATABASE TAG "FULL\_DATABASE\_DATAFILES". Результат можно посмотреть на рисунке 6.1.

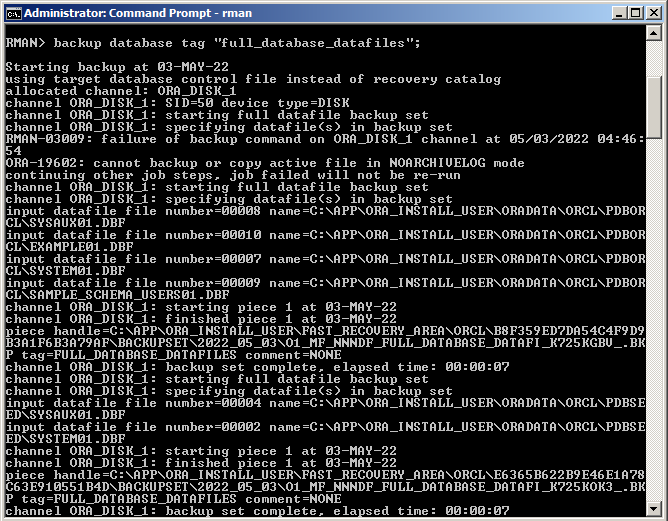


Рисунок 6.1 – Резервирование файлов данных

Для резервирования контрольных файлов используется команда: BACKUP CURRENT CONTROLFILE TAG "FULL\_DATABASE\_CONTROLFILE". Результат можно посмотреть на рисунке 6.2.

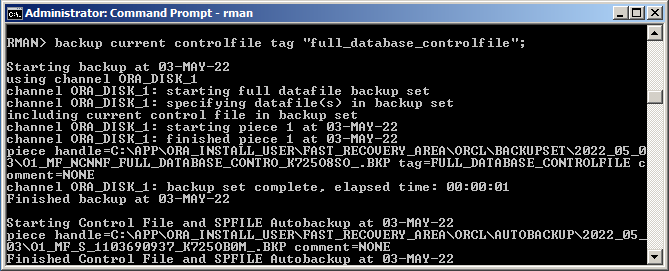


Рисунок 6.2 – Резервирование управляющих файлов

Чтобы получить информацию о созданных бэкапах нужно выполнить команду: RMAN> LIST BACKUP;

Для восстановления данных целевая БД должна находиться в состоянии NOMOUNT/ MOUNT/ OPEN в зависимости от характера восстановления, например:

* NOMOUNT: для восстановления контрольных файлов БД (фактически – СУБД)
* MOUNT: для восстановления БД целиком или табличного пространства SYSTEM
* OPEN: для восстановление табличных пространств, помимо SYSTEM (в этом случае перед процедурой восстановления само табличное пространство потребуется перевести в состояние OFFLINE).

# **Руководство пользователя**

По итогу нами была разработана полноценная база данных с таблицами и процедурами. Составим небольшое руководство пользователя о том, как грамотно использовать все объекты нашей БД.

База данных состоит из 10 таблиц. Созданные процедуры предоставляют пользователю основной функционал, а именно добавление, изменение и удаление объекта.

Пользователь может создавать объекты с нуля. Попробуем с помощью нашей процедуры добавить строчку в таблицу Logins. Результат выполнения процедуры представлен на рисунке 7.1.

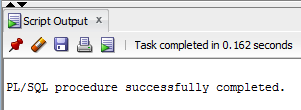


Рисунок 7.1 – Выполненная процедура

Проверим результат, выполнив select-запрос к самой таблице Logins. Получим результат, представленный на рисунке 7.2.

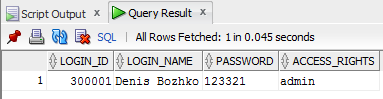


Рисунок 7.2 – Результат выполненной процедуры

В случае, если пользователь будет в процедуру передавать некорректные значения, ему будет выведено сообщение об ошибке (рисунок 7.3).

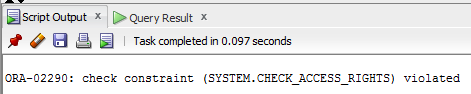


Рисунок 7.3 – Результат выполнения процедуры с некорректными входными параметрами

Пользователь также может изменять имеющиеся данные. Проверим это на таблице Admins. Для начала создадим новую строку, у которой в столбце Position пропишем Director. Создадим мы ее с также с помощью процедур. Результат выполнения представлен на рисунке 7.4.

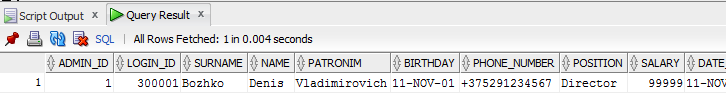


Рисунок 7.4 – Создание строчки в таблице Admins

А теперь изменим значение столбца Position на, например, Data Scientist. В итоге в таблице у нас будет уже измененная строка (рисунок 7.5).

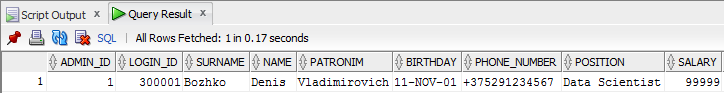


Рисунок 7.5 – Измененная строка в таблице

Пользователь может также проводить экпорт и импорт любой таблицы в формат XML. Итоговый файл для таблицы Autoes после экспорта представлен на рисунке 7.6.



Рисунок 7.6 – Экспортированный файл Autoes.xml

Исходя из проделанной работы мы можем отметить, что данная база данных правильно выполняет свои функции такие как защита от некорректного ввода данных, вывод необходимой информации, изменение строк, удаление и создание новых значений.

# **Заключение**

В процессе решения поставленной задачи была достигнута поставленная цель по созданию базы данных «Автошкола». В данной работе использовалось СУБД Oracle DataBase 12c. При разработке курсового проекта использовались объекты: таблицы, хранимые процедуры, индексы, функции, последовательности.

Основной целью курсового проекта стало проектирование базы данных для дальнейшей интеграции с приложением, которое помогло бы облегчить взаимодействие с базой данных посредством программного интерфейса. При разработке выполнены следующие пункты:

* управление базой данных (администраторский состав, преподавательский состав, ученики, автомобили);
* изменение информации о преподавателях, учениках, группах;
* вывод отчетов о группе и академической успеваемости учеников;
* вывод данных об учениках.

Приложение прошло тестирование при использовании в БД большого количество данных. Также были реализованы процедуры для импорта, экспорта данных в формат XML.

Была реализована технология «Резервное копирование и восстановление».

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

# **Cписок используемых источников**

1. Официальный сайт Oracle [Электронный ресурс]. Режим доступа: https://www.oracle.com/database/database-vault/index.html. Дата доступа: 01.05.2022.
2. Export and Import In Oracle XML DB Data [Электронный ресурс]. Режим доступа: https://docs.oracle.com/database/121/ADXDB/xdb26imp.htm#ADXDB3000. Дата доступа: 01.05.2022.
3. Getting Started with RMAN [Электронный ресурс]. Режим доступа: https://docs.oracle.com/cd/E11882\_01/backup.112/e10642/rcmquick.htm#BRADV89346. Дата доступа: 02.05.2022.

# **ПРИЛОЖЕНИЕ А**

Создание таблиц и последовательностей

|  |
| --- |
| -- tables and sequences creation  create sequence Logins\_ID\_SEQ; -- logins  create table Logins  (Login\_ID int default Logins\_ID\_SEQ.nextval not null,  Login\_Name nvarchar2(30),  Password nvarchar2(100),  Access\_Rights nvarchar2(10),  constraint Check\_Access\_Rights check (Access\_Rights in ('admin', 'teacher', 'instructor', 'student')),  constraint Login\_PK primary key (Login\_ID));  create sequence Admins\_ID\_SEQ; -- admins  create table Admins  (Admin\_ID int default Admins\_ID\_SEQ.nextval not null,  Login\_ID int not null,  Surname nvarchar2(50),  Name nvarchar2(50),  Patronim nvarchar2(50),  Birthday date,  Phone\_Number nvarchar2(20),  Position nvarchar2(50),  Salary int,  constraint Check\_Salary check (Salary > 0),  Date\_Of\_Entry date,  constraint Admin\_PK primary key (Admin\_ID));  create sequence Autodromes\_ID\_SEQ; -- autodromes  create table Autodromes  (Autodrome\_ID int default Autodromes\_ID\_SEQ.nextval not null,  Address nvarchar2(60) not null,  Num int unique,  constraint Autodrome\_PK primary key (Autodrome\_ID));  create table Autoes -- autoes  (Brand nvarchar2(40),  Model nvarchar2(40),  Autodrome int not null,  Num nvarchar2(15) primary key);  create sequence Instructors\_ID\_SEQ; -- instructors  create table Instructors  (Instructor\_ID int default Instructors\_ID\_SEQ.nextval not null,  Login\_ID int not null,  Surname nvarchar2(50) unique,  Name nvarchar2(50),  Patronim nvarchar2(50),  Birthday date,  Phone\_Number nvarchar2(20),  Salary int,  constraint Check\_Salary2 check (Salary > 0),  Date\_Of\_Entry date,  Auto\_Number nvarchar2(15) not null,  constraint Instructor\_PK primary key (Instructor\_ID));  create sequence Teachers\_ID\_SEQ; -- teachers  create table Teachers  (Teacher\_ID int default Teachers\_ID\_SEQ.nextval not null,  Login\_ID int not null,  Surname nvarchar2(50) unique,  Name nvarchar2(50),  Patronim nvarchar2(50),  Birthday date,  Phone\_Number nvarchar2(20),  Salary int,  constraint Check\_Salary3 check (Salary > 0),  Date\_Of\_Entry date,  Category nvarchar2(5),  constraint Check\_Category check (Category in ('A', 'A1', 'B', 'C', 'D', 'E')),  constraint Teacher\_PK primary key (Teacher\_ID));  create sequence Schools\_ID\_SEQ; -- schools  create table Schools  (School\_ID int default Schools\_ID\_SEQ.nextval not null,  Address nvarchar2(100) unique,  Num int,  constraint Check\_Num check (Num > 0),  constraint School\_PK primary key (School\_ID));  create table Cabinets -- cabinets  (Num int primary key,  School nvarchar2(100) not null,  constraint Check\_Num2 check (Num > 0));  create sequence Groups\_ID\_SEQ; -- groups  create table Groups  (Group\_ID int default Groups\_ID\_SEQ.nextval not null,  Start\_Date date,  End\_Date date,  Teacher nvarchar2(50) not null,  Instructor nvarchar2(50) not null,  Number\_Of\_Students int,  constraint Check\_Number\_Of\_Students check (Number\_Of\_Students > 0),  Category nvarchar2(5),  constraint Check\_Category2 check (Category in ('A', 'A1', 'B', 'C', 'D', 'E')),  School nvarchar2(100) not null,  Cabinet int not null,  Lesson\_Day nvarchar2(20),  constraint Check\_Day check (Lesson\_Day in ('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday')),  constraint Group\_PK primary key (Group\_ID));  create sequence Students\_ID\_SEQ; -- students  create table Students  (Student\_ID int default Students\_ID\_SEQ.nextval not null,  Login\_ID int not null,  Surname nvarchar2(50),  Name nvarchar2(50),  Patronim nvarchar2(50),  Birthday date,  Phone\_Number nvarchar2(20),  Study\_Group int not null,  Teacher nvarchar2(50) not null,  Instructor nvarchar2(50) not null,  Payment int,  constraint Check\_Payment check (Payment <= 1000 and Payment >= 0),  Exam1 nvarchar2(20),  constraint Check\_Exam1 check (Exam1 in ('passed', 'not passed')),  Exam2 nvarchar2(20),  constraint Check\_Exam2 check (Exam2 in ('passed', 'not passed')),  Exam3 nvarchar2(20),  constraint Check\_Exam3 check (Exam3 in ('passed', 'not passed')),  constraint Student\_PK primary key (Student\_ID));  alter table Admins add constraint Admin\_FK foreign key (Login\_ID) references Logins(Login\_ID); -- foreign keys  alter table Autoes add constraint Auto\_FK foreign key (Autodrome) references Autodromes(Num);  alter table Instructors add constraint Instructor\_FK foreign key (Login\_ID) references Logins(Login\_ID);  alter table Instructors add constraint Instructor\_FK2 foreign key (Auto\_Number) references Autoes(Num);  alter table Teachers add constraint Teacher\_FK foreign key (Login\_ID) references Logins(Login\_ID);  alter table Cabinets add constraint Cabinet\_FK foreign key (School) references Schools(Address);  alter table Groups add constraint Group\_FK foreign key (Teacher) references Teachers(Surname);  alter table Groups add constraint Group\_FK2 foreign key (Instructor) references Instructors(Surname);  alter table Groups add constraint Group\_FK3 foreign key (School) references Schools(Address);  alter table Groups add constraint Group\_FK4 foreign key (Cabinet) references Cabinets(Num);  alter table Students add constraint Student\_FK foreign key (Study\_Group) references Groups(Group\_ID);  alter table Students add constraint Student\_FK2 foreign key (Login\_ID) references Logins(Login\_ID);  alter table Students add constraint Student\_FK3 foreign key (Teacher) references Teachers(Surname);  alter table Students add constraint Student\_FK4 foreign key (Instructor) references Instructors(Surname);  --drop table Groups;  --drop table Cabinets;  --drop table Schools;  --drop table Teachers;  --drop table Instructors;  --drop table Autodromes;  --drop table Autoes;  --drop table Admins;  --drop table Logins; |

ПРИЛОЖЕНИЕ Б

Создание пользователей и ролей

|  |
| --- |
| -- roles  create role Admin\_Role;  grant all on Logins to Admin\_Role;  grant all on Admins to Admin\_Role;  grant all on Autodromes to Admin\_Role;  grant all on Autoes to Admin\_Role;  grant all on Instructors to Admin\_Role;  grant all on Teachers to Admin\_Role;  grant all on Schools to Admin\_Role;  grant all on Cabinets to Admin\_Role;  grant all on Groups to Admin\_Role;  grant all on Students to Admin\_Role;  create role Instructor\_Role;  grant all on Autoes to Instructor\_Role;  create role Teacher\_Role;  grant all on Groups to Teacher\_Role;  grant all on Students to Teacher\_Role;  create role Student\_Role;  grant select on Autodromes to Student\_Role;  grant select on Autoes to Student\_Role;  grant select on Schools to Student\_Role;  grant select on Cabinets to Student\_Role;  grant select on Groups to Student\_Role;  grant select on Students to Student\_Role;  -- users  create user Denis password expire;  grant Admin\_Role to Denis;  create user Anton password expire;  grant Instructor\_Role to Anton;  create user Makar password expire;  grant Teacher\_Role to Makar;  create user Daniil password expire;  grant Student\_Role to Daniil; |

ПРИЛОЖЕНИЕ В

Создание хранимых процедур

|  |
| --- |
| -- procedures  -- logins  create or replace procedure AddLogin(ILogin\_Name Logins.Login\_Name%type, IPassword Logins.Password%type, IAccess\_Rights  Logins.Access\_Rights%type)  is  begin  insert into Logins (Login\_Name, Password, Access\_Rights)  values (ILogin\_Name, IPassword, IAccess\_Rights);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  -- admins  create or replace procedure AddAdmin(ILogin\_ID Admins.Login\_ID%type,  ISurname Admins.Surname%type,  IName Admins.Name%type,  IPatronim Admins.Patronim%type,  IBirthday Admins.Birthday%type,  IPhone\_Number Admins.Phone\_Number%type,  IPosition Admins.Position%type,  ISalary Admins.Salary%type,  IDate\_Of\_Entry Admins.Date\_Of\_Entry%type)  is  begin  insert into Admins (Login\_ID, Surname, Name, Patronim, Birthday, Phone\_Number, Position, Salary, Date\_Of\_Entry)  values (ILogin\_ID, ISurname, IName, IPatronim, IBirthday, IPhone\_Number, IPosition, ISalary, IDate\_Of\_Entry);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure UpdateAdmin(ILogin\_ID Admins.Login\_ID%type,  ISurname Admins.Surname%type,  IName Admins.Name%type,  IPatronim Admins.Patronim%type,  IBirthday Admins.Birthday%type,  IPhone\_Number Admins.Phone\_Number%type,  IPosition Admins.Position%type,  ISalary Admins.Salary%type,  IDate\_Of\_Entry Admins.Date\_Of\_Entry%type)  is  begin  update Admins set  Login\_ID = ILogin\_ID,  Surname = ISurname,  Name = IName,  Patronim = IPatronim,  Birthday = IBirthday,  Phone\_Number = IPhone\_Number,  Position = IPosition,  Salary = ISalary,  Date\_Of\_Entry = IDate\_Of\_Entry;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  -- autodromes  create or replace procedure AddAutodrome(IAddress Autodromes.Address%type,  INum Autodromes.Num%type)  is  begin  insert into Autodromes (Address, Num)  values (IAddress, INum);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure DeleteAutodrome(IAutodrome\_ID Autodromes.Autodrome\_ID%type)  is  begin  delete from Autodromes where Autodrome\_ID = IAutodrome\_ID;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  -- autoes  create or replace procedure AddAuto(IBrand Autoes.Brand%type,  IModel Autoes.Model%type,  IAutodrome Autoes.Autodrome%type,  INum Autoes.Num%type)  is  begin  insert into Autoes (Brand, Model, Autodrome, Num)  values (IBrand, IModel, IAutodrome, INum);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure DeleteAuto(INum Autoes.Num%type)  is  begin  delete from Autoes where Num = INum;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  -- instructors  create or replace procedure AddInstructor(ILogin\_ID Instructors.Login\_ID%type,  ISurname Instructors.Surname%type,  IName Instructors.Name%type,  IPatronim Instructors.Patronim%type,  IBirthday Instructors.Birthday%type,  IPhone\_Number Instructors.Phone\_Number%type,  ISalary Instructors.Salary%type,  IDate\_Of\_Entry Instructors.Date\_Of\_Entry%type,  IAuto\_Number Instructors.Auto\_Number%type)  is  begin  insert into Instructors (Login\_ID, Surname, Name, Patronim, Birthday, Phone\_Number, Salary, Date\_Of\_Entry, Auto\_Number)  values (ILogin\_ID, ISurname, IName, IPatronim, IBirthday, IPhone\_Number, ISalary, IDate\_Of\_Entry, IAuto\_Number);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure UpdateInstructor(ILogin\_ID Instructors.Login\_ID%type,  ISurname Instructors.Surname%type,  IName Instructors.Name%type,  IPatronim Instructors.Patronim%type,  IBirthday Instructors.Birthday%type,  IPhone\_Number Instructors.Phone\_Number%type,  ISalary Instructors.Salary%type,  IDate\_Of\_Entry Instructors.Date\_Of\_Entry%type,  IAuto\_Number Instructors.Auto\_Number%type)  is  begin  update Instructors set  Login\_ID = ILogin\_ID,  Surname = ISurname,  Name = IName,  Patronim = IPatronim,  Birthday = IBirthday,  Phone\_Number = IPhone\_Number,  Salary = ISalary,  Date\_Of\_Entry = IDate\_Of\_Entry,  Auto\_Number = IAuto\_Number;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  -- teachers  create or replace procedure AddTeacher(ILogin\_ID Teachers.Login\_ID%type,  ISurname Teachers.Surname%type,  IName Teachers.Name%type,  IPatronim Teachers.Patronim%type,  IBirthday Teachers.Birthday%type,  IPhone\_Number Teachers.Phone\_Number%type,  ISalary Teachers.Salary%type,  IDate\_Of\_Entry Teachers.Date\_Of\_Entry%type,  ICategory Teachers.Category%type)  is  begin  insert into Teachers (Login\_ID, Surname, Name, Patronim, Birthday, Phone\_Number, Salary, Date\_Of\_Entry, Category)  values (ILogin\_ID, ISurname, IName, IPatronim, IBirthday, IPhone\_Number, ISalary, IDate\_Of\_Entry, ICategory);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure UpdateTeacher(ILogin\_ID Teachers.Login\_ID%type,  ISurname Teachers.Surname%type,  IName Teachers.Name%type,  IPatronim Teachers.Patronim%type,  IBirthday Teachers.Birthday%type,  IPhone\_Number Teachers.Phone\_Number%type,  ISalary Teachers.Salary%type,  IDate\_Of\_Entry Teachers.Date\_Of\_Entry%type,  ICategory Teachers.Category%type)  is  begin  update Teachers set  Login\_ID = ILogin\_ID,  Surname = ISurname,  Name = IName,  Patronim = IPatronim,  Birthday = IBirthday,  Phone\_Number = IPhone\_Number,  Salary = ISalary,  Date\_Of\_Entry = IDate\_Of\_Entry,  Category = ICategory;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  -- schools  create or replace procedure AddSchool(IAddress Schools.Address%type,  INum Schools.Num%type)  is  begin  insert into Schools (Address, Num)  values (IAddress, INum);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure DeleteSchool(ISchool\_ID Schools.School\_ID%type)  is  begin  delete from Schools where School\_ID = ISchool\_ID;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  -- cabinets  create or replace procedure AddCabinet(INum Cabinets.Num%type,  ISchool Cabinets.School%type)  is  begin  insert into Cabinets (Num, School)  values (INum, ISchool);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure DeleteCabinet(INum Cabinets.Num%type)  is  begin  delete from Cabinets where Num = INum;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  -- groups  create or replace procedure AddGroup(IStart\_Date Groups.Start\_Date%type,  IEnd\_Date Groups.End\_Date%type,  ITeacher Groups.Teacher%type,  IInstructor Groups.Instructor%type,  INumber\_Of\_Students Groups.Number\_Of\_Students%type,  ICategory Groups.Category%type,  ISchool Groups.School%type,  ICabinet Groups.Cabinet%type,  ILesson\_Day Groups.Lesson\_Day%type)  is  begin  insert into Groups (Start\_Date, End\_Date, Teacher, Instructor, Number\_Of\_Students, Category, School, Cabinet, Lesson\_Day)  values (IStart\_Date, IEnd\_Date, ITeacher, IInstructor, INumber\_Of\_Students, ICategory, ISchool, ICabinet, ILesson\_Day);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure UpdateGroup(IStart\_Date Groups.Start\_Date%type,  IEnd\_Date Groups.End\_Date%type,  ITeacher Groups.Teacher%type,  IInstructor Groups.Instructor%type,  INumber\_Of\_Students Groups.Number\_Of\_Students%type,  ICategory Groups.Category%type,  ISchool Groups.School%type,  ICabinet Groups.Cabinet%type,  ILesson\_Day Groups.Lesson\_Day%type)  is  begin  update Groups set  Start\_Date = IStart\_Date,  End\_Date = IEnd\_Date,  Teacher = ITeacher,  Instructor = IInstructor,  Number\_Of\_Students = INumber\_Of\_Students,  Category = ICategory,  School = ISchool,  Cabinet = ICabinet,  Lesson\_Day = ILesson\_Day;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  -- students  create or replace procedure AddStudent(ILogin\_ID Students.Login\_ID%type,  ISurname Students.Surname%type,  IName Students.Name%type,  IPatronim Students.Patronim%type,  IBirthday Students.Birthday%type,  IPhone\_Number Students.Phone\_Number%type,  IStudy\_Group Students.Study\_Group%type,  ITeacher Students.Teacher%type,  IInstructor Students.Instructor%type,  IPayment Students.Payment%type,  IExam1 Students.Exam1%type,  IExam2 Students.Exam2%type,  IExam3 Students.Exam3%type)  is  begin  insert into Students (Login\_ID, Surname, Name, Patronim, Birthday, Phone\_Number, Study\_Group, Teacher, Instructor, Payment, Exam1, Exam2, Exam3)  values (ILogin\_ID, ISurname, IName, IPatronim, IBirthday, IPhone\_Number, IStudy\_Group, ITeacher, IInstructor, IPayment, IExam1, IExam2, IExam3);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure UpdateStudent(ILogin\_ID Teachers.Login\_ID%type,  ISurname Students.Surname%type,  IName Students.Name%type,  IPatronim Students.Patronim%type,  IBirthday Students.Birthday%type,  IPhone\_Number Students.Phone\_Number%type,  IStudy\_Group Students.Study\_Group%type,  ITeacher Students.Teacher%type,  IInstructor Students.Instructor%type,  IPayment Students.Payment%type,  IExam1 Students.Exam1%type,  IExam2 Students.Exam2%type,  IExam3 Students.Exam3%type)  is  begin  update Students set  Login\_ID = ILogin\_ID,  Surname = ISurname,  Name = IName,  Patronim = IPatronim,  Birthday = IBirthday,  Phone\_Number = IPhone\_Number,  Study\_Group = IStudy\_Group,  Teacher = ITeacher,  Instructor = IInstructor,  Payment = IPayment,  Exam1 = IExam1,  Exam2 = IExam2,  Exam3 = IExam3;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  -- others  declare procedure ViewGroupInfo(IGroup\_ID Groups.Group\_ID%type)  is  cursor group\_cursor is select Group\_ID,  Start\_Date,  End\_Date,  Teacher,  Instructor,  Number\_Of\_Students,  Category  from Groups where Group\_ID = IGroup\_ID;  begin  for any\_group in group\_cursor  loop  dbms\_output.put\_line(any\_group.Group\_ID || ' ' ||  any\_group.Start\_Date || ' ' ||  any\_group.End\_Date || ' ' ||  any\_group.Teacher || ' ' ||  any\_group.Instructor || ' ' ||  any\_group.Number\_Of\_Students || ' ' ||  any\_group.Category || ' ');  end loop;  end;  begin  ViewGroupInfo(1);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  declare procedure ViewStudentsExams(IStudent\_ID Students.Student\_ID%type)  is  cursor student\_cursor is select Student\_ID,  Exam1,  Exam2,  Exam3  from Students where Student\_ID = IStudent\_ID;  begin  for any\_student in student\_cursor  loop  dbms\_output.put\_line(any\_student.Student\_ID || ' ' ||  any\_student.Exam1 || ' ' ||  any\_student.Exam2 || ' ' ||  any\_student.Exam3 || ' ');  end loop;  end;  begin  ViewStudentsExams(1);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  declare procedure ViewStudentsInfoByGroup(IGroup\_ID Students.Study\_Group%type)  is  cursor students\_cursor is select Student\_ID,  Surname,  Name,  Patronim,  Birthday,  Phone\_Number,  Study\_Group  from Students where Study\_Group = IGroup\_ID;  begin  for any\_student in students\_cursor  loop  dbms\_output.put\_line(any\_student.Student\_ID || ' ' ||  any\_student.Surname || ' ' ||  any\_student.Name || ' ' ||  any\_student.Patronim || ' ' ||  any\_student.Birthday || ' ' ||  any\_student.Phone\_Number || ' ' ||  any\_student.Study\_Group || ' ');  end loop;  end;  begin  ViewStudentsInfoByGroup(1);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end; |

ПРИЛОЖЕНИЕ Г

Функции экспорта и импорта данных

|  |
| --- |
| --xml export  SELECT file\_name FROM dba\_data\_files;  create or replace directory UTLDATA AS 'C:/app';  create or replace procedure ExportAutoesToXml  is  DOC DBMS\_XMLDOM.DOMDocument;  XDATA XMLTYPE;  CURSOR XMLCUR IS  SELECT XMLELEMENT("Autoes",  XMLAttributes('http://www.w3.org/2001/XMLSchema' AS "xmlns:xsi",  'http://www.oracle.com/Users.xsd' AS "xsi:nonamespaceSchemaLocation"),  XMLAGG(XMLELEMENT("Autoes",  xmlelement("brand", Autoes.Brand),  xmlelement("model", Autoes.Model),  xmlelement("autodrome", Autoes.Autodrome),  xmlelement("num", Autoes.Num)  ))) from Autoes;  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/Autoes.xml');  END;  begin  ExportAutoesToXml();  end;  -- drop procedure ExportAutoesToXml;  ----------------------------------------------------------------  -- xml import  create or replace procedure ImportAutoesFromXml  IS  L\_CLOB CLOB;  L\_BFILE BFILE := BFILENAME('UTLDATA', 'Autoes.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;  et Autoes%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,  'brand/text()',et.Brand);  Dbms\_Xslprocessor.Valueof(V\_Current\_Node,  'model/text()',et.Model);  Dbms\_Xslprocessor.Valueof(V\_Current\_Node,  'autodrome/text()',et.Autodrome);  dbms\_xslprocessor.valueof(v\_current\_node,  'num/text()',et.Num);  insert into Autoes (Brand, Model, Autodrome, Num)  values (et.Brand, et.Model, et.Autodrome, et.Num);  end loop;  DBMS\_LOB.FREETEMPORARY(L\_CLOB);  DBMS\_XMLPARSER.FREEPARSER(P);  DBMS\_XMLDOM.FREEDOCUMENT(V\_DOC);  commit;  END;  begin  ImportAutoesFromXml();  end;  select \* from Autoes; |

ПРИЛОЖЕНИЕ Д

Скрипт тестирования производительности базы данных

|  |
| --- |
| -- testing 100000 rows  -- delete Logins;  set timing on serveroutput on  begin  for Lcntr IN 1..100000  loop  AddLogin('testing','100000','admin');  end loop;  end;  select \* from Logins;  declare  t1 number := dbms\_utility.get\_time();  begin  for r in (select \* from Logins where login\_id > 50000 and login\_id < 70000) loop null; end loop;  dbms\_output.put\_line('query elapsed: '||(dbms\_utility.get\_time() - t1)/100);  end;  create index logins\_id on Logins(Login\_ID, Password);  declare  t1 number := dbms\_utility.get\_time();  begin  for r in (select \* from Logins where login\_id > 50000 and login\_id < 70000) loop null; end loop;  dbms\_output.put\_line('query elapsed: '||(dbms\_utility.get\_time() - t1)/100);  end;  drop index logins\_id;  select \* from Logins; |