**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 «**Реализация базы данных “Стажировка в IT-компании” с применением технологии резервного копирования и восстановления данных»

|  |  |  |
| --- | --- | --- |
| **Исполнитель** |  | |
| Студент 3 курса группы1 | подпись, дата | Д. В. Финский  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: Финский Даниил Вадимович

**Тема: «**Реалзация базы данных “Стажировка в IT-компании” 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, менторский состав, кандидаты, песочницы);
* Изменение информации о менторах, кандидатах, песочницах;
* Вывод отчетов о группе и успеваемости кандидатов;
* Вывод данных о кандидатах.

**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](#_Toc103198560)

[1. Постановка задачи 6](#_Toc103198561)

[2. Разработка модели базы данных 7](#_Toc103198562)

[3. Разработка необходимых объектов 8](#_Toc103198563)

[3.1. Таблицы 8](#_Toc103198564)

[3.2. Пользователи 11](#_Toc103198565)

[3.3. Хранимые процедуры 11](#_Toc103198566)

[4. Описание процедур импорта и экспорта 12](#_Toc103198567)

[5. Тестирование производительности 14](#_Toc103198568)

[6. Описание технологии и ее применения в базе данных 16](#_Toc103198569)

[7. Руководство пользователя 19](#_Toc103198570)

[Заключение 21](#_Toc103198571)

[Cписок используемых источников 22](#_Toc103198572)

[ПРИЛОЖЕНИЕ А 23](#_Toc103198573)

[ПРИЛОЖЕНИЕ Б 26](#_Toc103198574)

[ПРИЛОЖЕНИЕ В 27](#_Toc103198575)

[ПРИЛОЖЕНИЕ Г 35](#_Toc103198576)

[ПРИЛОЖЕНИЕ Д 37](#_Toc103198577)

## **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. 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 «Стажировка в IT-компании» с применением технологии резервного копирования.

База данных «Стажировка в IT-компании» 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 «Стажировка в IT-компании» существенно упрощает работу сотрудников IT-компании, а также предоставляет возможность администраторам и менторам 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, менторский состав, кандидаты, песочницы);
* изменение информации о менторах, кандидатах, песочницах;
* вывод отчетов о группе и успеваемости кандидатов;
* вывод данных о кандидатах.

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 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.

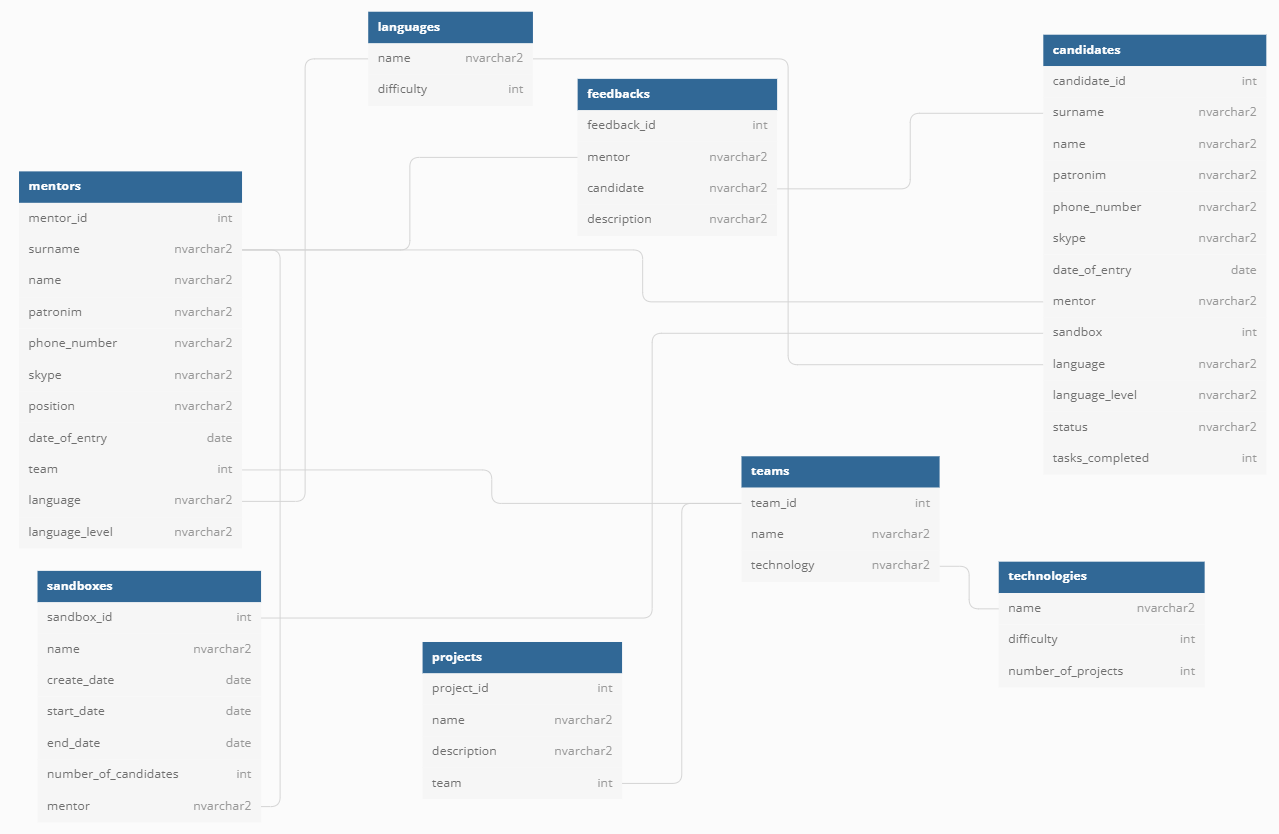


Рисунок 2.1 – Диаграмма базы данных

База данных приложения состоит из 8 таблиц, взаимосвязанных между собой внешними ключами: technologies, teams, projects, languages, mentors, sandboxes, candidates, feedbacks.

# **Разработка необходимых объектов**

# **Таблицы**

Для реализации базы данных «Стажировка в IT-компании» было разработано 8 таблиц: technologies, teams, projects, languages, mentors, sandboxes, candidates, feedbacks.

Таблица technologies представляет список технологий, используемых компанией (таблица 3.1):

Таблица 3.1 – Столбцы таблицы technologies

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| name | Название технологии | nvarchar2 |
| difficulty | Сложность изучения технологии | int |
| number\_of\_projects | Число проектов, использующих эту технологию | int |

Таблица teams представляет список команд, занимающимися проектами в компании (таблица 3.2):

Таблица 3.2 – Столбцы таблицы teams

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| team\_id | Идентификатор команды | int |
| name | Наименование команды | nvarchar2 |
| technology | Технология, используемая командой, внешний ключ | nvarchar2 |

Таблица projects представляет список проектов, которыми занимаются команды в компании (таблица 3.3):

Таблица 3.3 – Столбцы таблицы projects

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| project\_id | Идентификатор проекта | int |
| name | Наименование проекта | nvarchar2 |
| description | Описание проекта | nvarchar2 |
| team | Команда, занимающаяся данным проектом, внешний ключ | int |

Таблица languages представляет список иностранных языков, которыми владеют менторы и кандидаты (таблица 3.4):

Таблица 3.4 – Столбцы таблицы languages

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| name | Наименование языка | nvarchar2 |
| difficulty | Сложность изучения языка | int |

Таблица mentors представляет список менторов компании (таблица 3.5):

Таблица 3.5 – Столбцы таблицы mentors

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| mentor\_id | Идентификатор ментора | int |
| surname | Фамилия ментора | nvarchar2 |
| name | Имя ментора | nvarchar2 |
| patronim | Отчество ментора | nvarchar2 |
| phone\_number | Телефонный номер ментора | nvarchar2 |
| skype | Логин ментора в Skype | nvarchar2 |
| position | Должность ментора в компании | nvarchar2 |
| date\_of\_entry | Дата приема на работу ментора | date |
| team | Команда, в которой работает ментор, внешний ключ | int |
| language | Иностранный язык, которым владеет ментор, внешний ключ | nvarchar2 |
| language\_level | Уровень владения иностранным языком ментора | nvarchar2 |

Таблица sandboxes представляет список песочниц (групп кандидатов) компании (таблица 3.6):

Таблица 3.6 – Столбцы таблицы sandboxes

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| sandbox\_id | Идентификатор песочницы | int |
| name | Наименование песочницы | nvarchar2 |
| create\_date | Дата создания песочницы | date |
| start\_date | Дата начала работы песочницы | date |
| end\_date | Дата окончания работы песочницы | date |

Продолжение таблицы 3.6.

|  |  |  |
| --- | --- | --- |
| number\_of\_candidates | Число кандидатов в песочнице | int |
| mentor | Ментор, закрепленный за песочницей, внешний ключ | nvarchar2 |

Таблица candidates представляет список кандидатов, находящихся или находившихся в песочнице (таблица 3.7):

Таблица 3.7 – Столбцы таблицы candidates

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| candidate\_id | Идентификатор кандидата | int |
| surname | Фамилия кандидата | nvarchar2 |
| name | Имя кандидата | nvarchar2 |
| patronim | Отчество кандидата | nvarchar2 |
| phone\_number | Телефонный номер кандидата | nvarchar2 |
| skype | Логин кандидата в Skype | nvarchar2 |
| date\_of\_entry | Дата вступления кандидата в песочницу | date |
| mentor | Ментор, закрепленный за кандидатом, внешний ключ | nvarchar2 |
| sandbox | Песочница, к которой относится кандидат, внешний ключ | int |
| language | Иностранный язык, которым владеет кандидатом, внешний ключ | nvarchar2 |
| language\_level | Уровень владения иностранным языком кандидата | nvarchar2 |
| status | Статус кандидата в песочнице (прошел, не прошел, проходит) | nvarchar2 |
| tasks\_completed | Число выполненных заданий кандидатом | int |

Таблица feedbacks представляет список отзывов, которые оставляют менторы о кандидатах (таблица 3.8):

Таблица 3.8 – Столбцы таблицы feedback

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| feedback\_id | Идентификатор отзыва | int |

Продолжение таблицы 3.8.

|  |  |  |
| --- | --- | --- |
| mentor | Ментор, оставивший отзыв, внешний ключ | nvarchar2 |
| candidate | Кандидат, о котором был оставлен отзыв, внешний ключ | nvarchar2 |
| description | Сам отзыв | 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 3 пользователя: администратор, ментор и кандидат. Каждый пользователь имеет разные привилегии в соответствии с его статусом в компании.

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 ментора;
* добавление/удаление отзыва;
* добавление/удаление группы;
* обновление информации о кандидате;
* обновление информации о группе;
* экспорт и импорт таблицы в формат xml;
* просмотр информации о кандидате;
* 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 export\_projects\_to\_xml  is  DOC DBMS\_XMLDOM.DOMDocument;  XDATA XMLTYPE;  CURSOR XMLCUR IS  SELECT XMLELEMENT("projects",  XMLAttributes('http://www.w3.org/2001/XMLSchema' AS "xmlns:xsi",  'http://www.oracle.com/Users.xsd' AS "xsi:nonamespaceSchemaLocation"),  XMLAGG(XMLELEMENT("projects",  xmlelement("project\_id", projects.project\_id),  xmlelement("name", projects.name),  xmlelement("description", projects.description),  xmlelement("team", projects.team)  ))) from projects;  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/projects.xml');  END;  begin  export\_projects\_to\_xml();  end; |

Листинг 4.1 – Создание процедуры export\_projects\_to\_xml

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  add\_team('test', NULL);  end loop;  end; |

Листинг 5.1 – Анонимный блок для заполнения таблицы большим количеством данных

Таким образом, было добавлено 100000 строк в таблицу teams.

После этого в другом анонимном блоке, который содержит в себе дополнительно переменную для отслеживания времени выполнения, был сделан запрос, который должен возвращать в результате 30000 строк. Скрипт данного блока представлен на листинге 5.2.

|  |
| --- |
| declare  t1 number := dbms\_utility.get\_time();  begin  for r in (select \* from teams where team\_id > 40000 and team\_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.

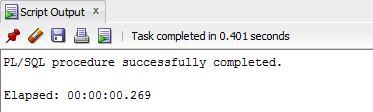


Рисунок 5.3 – Результат отбора данных

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

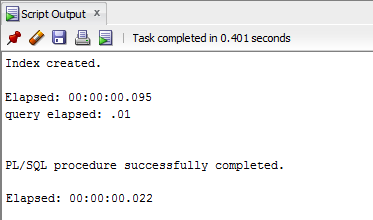


Рисунок 5.4 – Результат после создания индекса

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.

Резервирование файлов базы данных: горячее полное резервирование БД и холодное резервирование БД.

Горячее резервирование:

* может выполняться в состоянии СУБД OPEN;
* может выполняться только при включенном режиме архивирования журналов.

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.

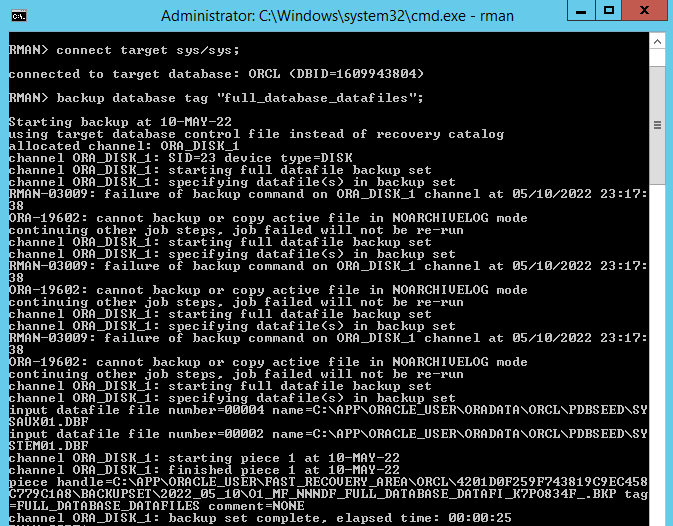


Рисунок 6.1 – Резервирование файлов данных

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

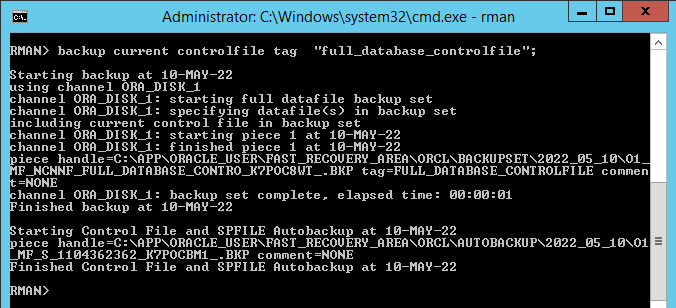


Рисунок 6.2 – Резервирование управляющих файлов

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 8 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 mentors. Результат выполнения процедуры представлен на рисунке 7.1.

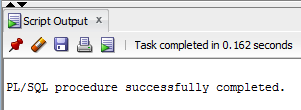


Рисунок 7.1 – Выполненная процедура

Проверим результат, выполнив запрос к самой таблице mentors. 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).

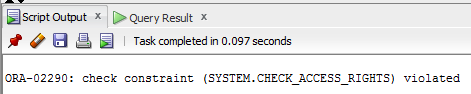


Рисунок 7.3 – Результат выполнения процедуры с некорректными входными параметрами

Пользователь также может изменять имеющиеся данные. Проверим это также на таблице mentors. Для начала создадим новую строку, у которой в столбце position пропишем «c# developer». Создадим мы ее с также с помощью процедур. Результат выполнения представлен на рисунке 7.4.



Рисунок 7.4 – Создание строчки в таблице mentors

А теперь изменим значение столбца position на, «BSTU developer». В итоге в таблице у нас будет уже измененная строка (рисунок 7.5).

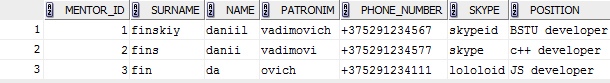


Рисунок 7.5 – Измененная строка в таблице

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

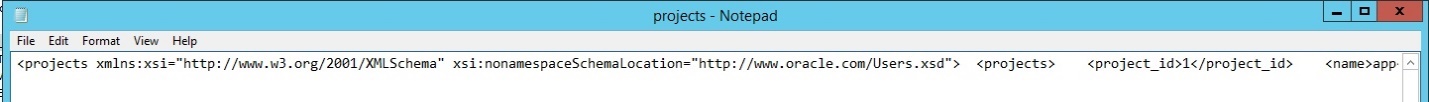


Рисунок 7.6 – Экспортированный файл projects.xml

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 «Стажировка в IT-компании». 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, менторский состав, кандидаты, песочницы);
* изменение информации о менторах, кандидатах, песочницах;
* вывод отчетов о группе и академической успеваемости кандидатов;
* вывод данных о кандидатах.

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**

Таблицы и последовательности

|  |
| --- |
| create table technologies  (name nvarchar2(20) primary key,  difficulty int,  number\_of\_projects int,  constraint check\_difficulty1 check (difficulty in (1, 2, 3, 4, 5)));  create sequence teams\_id\_seq;  create table teams  (team\_id int default teams\_id\_seq.nextval not null,  name nvarchar2(50),  technology nvarchar2(20),  constraint team\_pk primary key (team\_id));  create sequence projects\_id\_seq;  create table projects  (project\_id int default projects\_id\_seq.nextval not null,  name nvarchar2(50),  description nvarchar2(300),  team int,  constraint project\_pk primary key (project\_id));  create table languages  (name nvarchar2(30) primary key,  difficulty int,  constraint check\_difficulty2 check (difficulty in (1, 2, 3, 4, 5)));  create sequence mentors\_id\_seq;  create table mentors  (mentor\_id int default mentors\_id\_seq.nextval not null,  surname nvarchar2(50) unique,  name nvarchar2(50),  patronim nvarchar2(50),  phone\_number nvarchar2(20),  skype nvarchar2(30),  position nvarchar2(50),  date\_of\_entry date,  team int,  language nvarchar2(30),  language\_level nvarchar2(5),  constraint mentor\_pk primary key (mentor\_id),  constraint check\_language\_level1 check (language\_level in ('A1', 'A2', 'B1', 'B2', 'C1', 'C2')));  create sequence sandboxes\_id\_seq;  create table sandboxes  (sandbox\_id int default sandboxes\_id\_seq.nextval not null,  name nvarchar2(50),  create\_date date,  start\_date date,  end\_date date,  number\_of\_candidates int,  mentor nvarchar2(50),  constraint sandbox\_pk primary key (sandbox\_id));  create sequence candidates\_id\_seq;  create table candidates  (candidate\_id int default candidates\_id\_seq.nextval not null,  surname nvarchar2(50) unique,  name nvarchar2(50),  patronim nvarchar2(50),  phone\_number nvarchar2(20),  skype nvarchar2(30),  date\_of\_entry date,  mentor nvarchar2(50),  sandbox int,  language nvarchar2(30),  language\_level nvarchar2(5),  status nvarchar2(30),  tasks\_completed int,  constraint candidate\_pk primary key (candidate\_id),  constraint check\_language\_level2 check (language\_level in ('A1', 'A2', 'B1', 'B2', 'C1', 'C2')),  constraint check\_status check (status in ('not started', 'started', 'completed')));  create sequence feedback\_id\_seq;  create table feedbacks  (feedback\_id int default feedback\_id\_seq.nextval not null,  mentor nvarchar2(50),  candidate nvarchar2(50),  description nvarchar2(300),  constraint feedback\_pk primary key (feedback\_id));  alter table teams add constraint team\_fk foreign key (technology) references technologies(name); --foreign keys  alter table projects add constraint project\_fk foreign key (team) references teams(team\_id);  alter table mentors add constraint mentor\_fk1 foreign key (team) references teams(team\_id);  alter table mentors add constraint mentor\_fk2 foreign key (language) references languages(name);  alter table sandboxes add constraint sandbox\_fk foreign key (mentor) references mentors(surname);  alter table candidates add constraint candidate\_fk1 foreign key (mentor) references mentors(surname);  alter table candidates add constraint candidate\_fk2 foreign key (sandbox) references sandboxes(sandbox\_id);  alter table candidates add constraint candidate\_fk3 foreign key (language) references languages(name);  alter table feedbacks add constraint feedback\_fk1 foreign key (mentor) references mentors(surname);  alter table feedbacks add constraint feedback\_fk2 foreign key (candidate) references candidates(surname); |

ПРИЛОЖЕНИЕ Б

Пользователи и роли

|  |
| --- |
| create role sandbox\_admin;  grant all on technologies to sandbox\_admin;  grant all on teams to sandbox\_admin;  grant all on projects to sandbox\_admin;  grant all on languages to sandbox\_admin;  grant all on mentors to sandbox\_admin;  grant all on sandboxes to sandbox\_admin;  grant all on candidates to sandbox\_admin;  grant all on feedbacks to sandbox\_admin;  create role sandbox\_mentor;  grant select on technologies to sandbox\_mentor;  grant select on teams to sandbox\_mentor;  grant select on projects to sandbox\_mentor;  grant select on languages to sandbox\_mentor;  grant select on mentors to sandbox\_mentor;  grant all on sandboxes to sandbox\_mentor;  grant all on candidates to sandbox\_mentor;  grant all on feedbacks to sandbox\_mentor;  create role sandbox\_candidate;  grant select on candidates to sandbox\_candidate;  create user fin\_admin identified by Pa$$w0rd;  grant sandbox\_admin to fin\_admin;  create user fin\_mentor identified by Pa$$w0rd;  grant sandbox\_mentor to fin\_mentor;  create user fin\_candidate identified by Pa$$w0rd;  grant sandbox\_candidate to fin\_candidate; |

ПРИЛОЖЕНИЕ В

Процедуры

|  |
| --- |
| create or replace procedure add\_technology(iname technologies.name%type, --technologies --?????  idifficulty technologies.difficulty%type,  inumber\_of\_projects technologies.number\_of\_projects%type)  is  begin  insert into technologies (name, difficulty, number\_of\_projects)  values (iname, idifficulty, inumber\_of\_projects);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure delete\_technology(iname technologies.name%type)  is  begin  delete from technologies where name = iname;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure delete\_technology\_difficulty(idifficulty technologies.difficulty%type)  is  begin  delete from technologies where difficulty = idifficulty;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure update\_technology(iname technologies.name%type,  idifficulty technologies.difficulty%type,  inumber\_of\_projects technologies.number\_of\_projects%type)  is  begin  update technologies set  difficulty = idifficulty,  number\_of\_projects = inumber\_of\_projects  where name = iname;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure add\_team(iname teams.name%type, --teams  itechnology teams.technology%type)  is  begin  insert into teams (name, technology)  values (iname, itechnology);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure update\_team(iteam\_id teams.team\_id%type,  iname teams.name%type,  itechnology teams.technology%type)  is  begin  update teams set  name = iname,  technology = itechnology  where team\_id = iteam\_id;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure delete\_team(iteam\_id teams.team\_id%type)  is  begin  delete from teams where team\_id = iteam\_id;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure delete\_team\_name(iname teams.name%type)  is  begin  delete from teams where name = iname;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure add\_project(iname projects.name%type, --projects  idescription projects.description%type,  iteam projects.team%type)  is  begin  insert into projects (name, description, team)  values (iname, idescription, iteam);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure update\_project(iproject\_id projects.project\_id%type,  iname projects.name%type,  idescription projects.description%type,  iteam projects.team%type)  is  begin  update projects set  name = iname,  description = idescription,  team = iteam  where project\_id = iproject\_id;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure add\_language(iname languages.name%type, --languages  idifficulty languages.difficulty%type)  is  begin  insert into languages (name, difficulty)  values (iname, idifficulty);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure add\_mentor(isurname mentors.surname%type, --mentors  iname mentors.name%type,  ipatronim mentors.patronim%type,  iphone\_number mentors.phone\_number%type,  iskype mentors.skype%type,  iposition mentors.position%type,  idate\_of\_entry mentors.date\_of\_entry%type,  iteam mentors.team%type,  ilanguage mentors.language%type,  ilanguage\_level mentors.language\_level%type)  is  begin  insert into mentors (surname, name, patronim, phone\_number, skype, position, date\_of\_entry, team, language, language\_level)  values (isurname, iname, ipatronim, iphone\_number, iskype, iposition, idate\_of\_entry, iteam, ilanguage, ilanguage\_level);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure update\_mentor(imentor\_id mentors.mentor\_id%type,  isurname mentors.surname%type,  iname mentors.name%type,  ipatronim mentors.patronim%type,  iphone\_number mentors.phone\_number%type,  iskype mentors.skype%type,  iposition mentors.position%type,  idate\_of\_entry mentors.date\_of\_entry%type,  iteam mentors.team%type,  ilanguage mentors.language%type,  ilanguage\_level mentors.language\_level%type)  is  begin  update mentors set  surname = isurname,  name = iname,  patronim = ipatronim,  phone\_number = iphone\_number,  skype = iskype,  position = iposition,  date\_of\_entry = idate\_of\_entry,  team = iteam,  language = ilanguage,  language\_level = ilanguage\_level  where mentor\_id = imentor\_id;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure add\_sandbox(iname sandboxes.name%type, --sandboxes  icreate\_date sandboxes.create\_date%type,  istart\_date sandboxes.start\_date%type,  iend\_date sandboxes.end\_date%type,  inumber\_of\_candidates sandboxes.number\_of\_candidates%type,  imentor sandboxes.mentor%type)  is  begin  insert into sandboxes (name, create\_date, start\_date, end\_date, number\_of\_candidates, mentor)  values (iname, icreate\_date, istart\_date, iend\_date, inumber\_of\_candidates, imentor);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure update\_sandbox(isandbox\_id sandboxes.sandbox\_id%type,  iname sandboxes.name%type,  icreate\_date sandboxes.create\_date%type,  istart\_date sandboxes.start\_date%type,  iend\_date sandboxes.end\_date%type,  inumber\_of\_candidates sandboxes.number\_of\_candidates%type,  imentor sandboxes.mentor%type)  is  begin  update sandboxes set  name = iname,  create\_date = icreate\_date,  start\_date = istart\_date,  end\_date = iend\_date,  number\_of\_candidates = inumber\_of\_candidates,  mentor = imentor  where sandbox\_id = isandbox\_id;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure add\_candidate(isurname candidates.surname%type, --candidates  iname candidates.name%type,  ipatronim candidates.patronim%type,  iphone\_number candidates.phone\_number%type,  iskype candidates.skype%type,  idate\_of\_entry candidates.date\_of\_entry%type,  imentor candidates.mentor%type,  isandbox candidates.sandbox%type,  ilanguage candidates.language%type,  ilanguage\_level candidates.language\_level%type,  istatus candidates.status%type,  itasks\_completed candidates.tasks\_completed%type)  is  begin  insert into candidates (surname, name, patronim, phone\_number, skype, mentor, sandbox, language, language\_level, status, tasks\_completed)  values (isurname, iname, ipatronim, iphone\_number, iskype, imentor, isandbox, ilanguage, ilanguage\_level, istatus, itasks\_completed);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure update\_candidate(icandidate\_id candidates.candidate\_id%type,  isurname candidates.surname%type,  iname candidates.name%type,  ipatronim candidates.patronim%type,  iphone\_number candidates.phone\_number%type,  iskype candidates.skype%type,  idate\_of\_entry candidates.date\_of\_entry%type,  imentor candidates.mentor%type,  isandbox candidates.sandbox%type,  ilanguage candidates.language%type,  ilanguage\_level candidates.language\_level%type,  istatus candidates.status%type,  itasks\_completed candidates.tasks\_completed%type)  is  begin  update candidates set  surname = isurname,  name = iname,  patronim = ipatronim,  phone\_number = iphone\_number,  skype = iskype,  date\_of\_entry = idate\_of\_entry,  mentor = imentor,  sandbox = isandbox,  language = ilanguage,  language\_level = ilanguage\_level,  status = istatus,  tasks\_completed = itasks\_completed  where candidate\_id = icandidate\_id;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure add\_feedbacks(imentor feedbacks.mentor%type, --feedbacks  icandidate feedbacks.candidate%type,  idescription feedbacks.description%type)  is  begin  insert into feedbacks (mentor, candidate, description)  values (imentor, icandidate, idescription);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure delete\_feedback(ifeedback\_id feedbacks.feedback\_id%type)  is  begin  delete from feedbacks where feedback\_id = ifeedback\_id;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure delete\_feedback\_mentor(imentor feedbacks.mentor%type)  is  begin  delete from feedbacks where mentor = imentor;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure delete\_feedback\_candidate(icandidate feedbacks.candidate%type)  is  begin  delete from feedbacks where candidate = icandidate;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure update\_feedback(ifeedback\_id feedbacks.feedback\_id%type,  imentor feedbacks.mentor%type,  icandidate feedbacks.candidate%type,  idescription feedbacks.description%type)  is  begin  update feedbacks set  mentor = imentor,  candidate = icandidate,  description = idescription  where feedback\_id = ifeedback\_id;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  declare procedure view\_candidate\_info\_by\_id(icandidate\_id candidates.candidate\_id%type) -- select procedures  is  cursor candidates\_cursor is select candidate\_id,  surname,  name,  patronim,  phone\_number,  skype,  date\_of\_entry,  mentor,  sandbox,  language,  language\_level,  status,  tasks\_completed  from candidates where candidate\_id = icandidate\_id;  begin  for any\_candidate in candidates\_cursor  loop  dbms\_output.put\_line(any\_candidate.surname || ' ' ||  any\_candidate.name || ' ' ||  any\_candidate.patronim || ' ' ||  any\_candidate.phone\_number || ' ' ||  any\_candidate.skype || ' ' ||  any\_candidate.date\_of\_entry || ' ' ||  any\_candidate.mentor || ' ' ||  any\_candidate.sandbox || ' ' ||  any\_candidate.language || ' ' ||  any\_candidate.language\_level || ' ' ||  any\_candidate.status || ' ' ||  any\_candidate.tasks\_completed || ' ');  end loop;  end;  begin  view\_candidate\_info\_by\_id(1);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  declare procedure view\_sandbox\_info\_by\_id(isandbox\_id sandboxes.sandbox\_id%type)  is  cursor sandboxes\_cursor is select sandbox\_id,  name,  create\_date,  start\_date,  end\_date,  number\_of\_candidates,  mentor  from sandboxes where sandbox\_id = isandbox\_id;  begin  for any\_sandbox in sandboxes\_cursor  loop  dbms\_output.put\_line(any\_sandbox.sandbox\_id || ' ' ||  any\_sandbox.name || ' ' ||  any\_sandbox.create\_date || ' ' ||  any\_sandbox.start\_date || ' ' ||  any\_sandbox.end\_date || ' ' ||  any\_sandbox.number\_of\_candidates || ' ' ||  any\_sandbox.mentor || ' ');  end loop;  end;  begin  view\_sandbox\_info\_by\_id(1);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  declare procedure completed\_tasks\_by\_id(icandidate\_id candidates.candidate\_id%type)  is  cursor candidates\_cursor1 is select candidate\_id,  tasks\_completed  from candidates where candidate\_id = icandidate\_id;  begin  for any\_candidate1 in candidates\_cursor1  loop  dbms\_output.put\_line(any\_candidate1.candidate\_id || ' ' ||  any\_candidate1.tasks\_completed || ' ');  end loop;  end;  begin  completed\_tasks\_by\_id(1);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end; |

ПРИЛОЖЕНИЕ Г

Экспорт и импорт в XML

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

ПРИЛОЖЕНИЕ Д

Тестирования производительности базы данных

|  |
| --- |
| -- delete teams;  set timing on serveroutput on  begin  for Lcntr IN 1..100000  loop  add\_team('test', NULL);  end loop;  end;  select \* from teams;  declare  t1 number := dbms\_utility.get\_time();  begin  for r in (select \* from teams where team\_id > 40000 and team\_id < 70000) loop null; end loop;  dbms\_output.put\_line('query elapsed: '||(dbms\_utility.get\_time() - t1)/100);  end;  create index test\_teams\_index on teams(team\_id, name);  declare  t1 number := dbms\_utility.get\_time();  begin  for r in (select \* from teams where team\_id > 40000 and team\_id < 70000) loop null; end loop;  dbms\_output.put\_line('query elapsed: '||(dbms\_utility.get\_time() - t1)/100);  end;  drop index test\_teams\_index;  select \* from teams; |