МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПЕУБЛИКИ 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

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023 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: Шкабров Данила Сергеевич

**Тема:** Реализация базы данных для покупки билетов на хоккейные матчи с реализацией технологии маскирования данных

**1. Срок сдачи студентом законченной работы**: «10» мая 2023 г.

**2. Исходные данные к проекту:**

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

* хэширование паролей в БД;
* вход и авторизация пользователей;
* просмотр расписания и результатов матчей;
* просмотр статистики команд в группах
* покупка билета с выбором места определенной цены и переносом информации о брони в БД;
* администратор может:
* изменение данных о статистике команд и сохранение их в БД;
* добавление результатов матча дописывая и сохраняя их в БД;

**2.2. Требования.**

* База данных должна быть реализована в СУБД Oracle 12c.
* 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 | 22.02.2023 |  |
| 2 | Аналитический обзор литературы по теме проекта | 03.03.2023 |  |
| 3 | Изучение требований, определение вариантов использования | 10.03.2023 |  |
| 4 | Анализ и проектирование модели базы данных. Описание информационных объектов и ограничений целостности. | 24.03.2023 |  |
| 5 | Создание необходимых объектов | 07.04.2023 |  |
| 6 | Импорт и экспорт данных | 14.04.2023 |  |
| 7 | Описание используемой технологии | 28.04.2023 |  |
| 8 | Тестирование производительности | 02.05.2023 |  |
| 9 | Оформление пояснительной записки | 05.05.2023 |  |
| 10 | Сдача проекта | 10.05.2023 |  |

**5. Дата выдачи задания** «15» февраля 2023 г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Д.В. Сазонова*

(подпись)

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

(дата и подпись студента)

**Оглавление**

[Введение 5](#_Toc134797972)

[1 Постановка задачи 6](#_Toc134797973)

[2 Разработка модели базы данных 7](#_Toc134797974)

[3 Разработка необходимых объектов 8](#_Toc134797975)

[3.1 Таблицы 8](#_Toc134797976)

[3.2 Пользователи 9](#_Toc134797977)

[3.3 Хранимые процедуры 10](#_Toc134797978)

[4 Описание процедур экспорта и импорта 11](#_Toc134797979)

[5 Тестирование производительности 13](#_Toc134797980)

[6 Описание технологии и ее применения в базе данных 15](#_Toc134797981)

[7 Руководство пользователя 17](#_Toc134797982)

[Cписок используемых источников 20](#_Toc134797983)

[ПРИЛОЖЕНИЕ А 21](#_Toc134797984)

[ПРИЛОЖЕНИЕ Б 23](#_Toc134797985)

[ПРИЛОЖЕНИЕ В 24](#_Toc134797986)

[ПРИЛОЖЕНИЕ Г 30](#_Toc134797987)

[ПРИЛОЖЕНИЕ Д 33](#_Toc134797988)

## **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 реализация базы данных для покупки билетов на хоккейные матчи с реализацией технологии маскирования данных.

База данных будет содержать соответствующие таблицы 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 для покупки билетов на хоккейные матчи существенно упрощает фанатам просматривать результаты и покупать билеты, 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 (игры, команды, пользователи, билеты);
* изменение информации о командах, играх;
* покупка билета на матч;
* отображение статистик команд по группам.

Должны быть выполнены следующие требования:

* пароли для входа пользователя должны хэшироваться;
* отображение арены для брони места;
* 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 19c. 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.

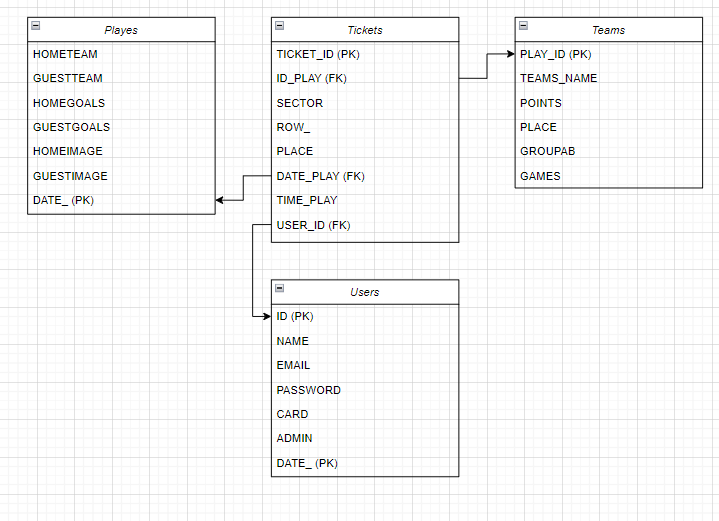


Рисунок 2.1 – Диаграмма базы данных

База данных приложения состоит из 4 таблицы, взаимосвязанных между собой внешними ключами: Plays, Tickets, Teams, Users.

# **Разработка необходимых объектов**

* 1. **Таблицы**

Для реализации базы данных «Хоккейная лига» было разработано 4 таблицы: Plays, Tickets, Teams, Users.

В таблице 3.1 проиллюстрирована структура таблицы «Playes», которая содержит информацию об играх. В данной таблице поля HOMETEAM и GUESTTEAM, где представлены в nvarchar. Поля HOMEGOALS и GUESTGOALS хранит голы матчей, HOMEIMAGE и GUESTIMAGE – хранит путь до картинок, DATE – хранит дату игры, первичный ключ.

Таблица 3.1 – Столбцы таблицы Playes

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| HOMETEAM | Команда хозяев | nvarchar2 |
| GUESTTEAM | Команда гостей | nvarchar2 |
| HOMEGOALS | Кол-во голов команда хозяев | int |
| GUESTGOALS | Кол-во голов команда гостей | int |
| HOMEIMAGE | Изображение команда хозяев | Blob |
| GUESTIMAGE | Изображение команда гостей | Blob |
| DATE\_ | Дата матча | Date |

В таблице 3.2 проиллюстрирована структура таблицы «Teams», которая содержит информацию о командах. В данной таблице поле Play\_ID, является первичным ключом. Поле TEAMS\_NAME хранит имя команды, POINTS – очки, PLACE – место в таблице, GROUPAB — содержит информацию о группе, в которой команда, GAMES – количество игр.

Таблица 3.2 – Столбцы таблицы Teams

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| PLAY\_ID | Идентификатор команды | int |
| TEAMS\_NAME | Название команды | nvarchar2 |
| POINTS | Кол-во очков команды | int |
| PLACE | Место команды | int |
| GROUPAB | Группа команды | nvarchar2 |
| GAMES | Кол-во игр команды | int |

В таблице 3.3 показана структура таблицы «Users», которая содержит информацию о пользователях. В данной таблице поле ID, является первичным ключом. Поле NAME хранит имя пользователя, PASSWORD – пароль, EMAIL – адрес электронной почты, CARD — номер платежной карты, ADMIN – соответственно роль (Yes для администратора, NULL для пользователя).

Таблица 3.3 – Столбцы таблицы Users

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| ID | Идентификатор пользователя | int |
| NAME | Имя пользователя | nvarchar2 |
| EMAIL | Email пользователя | nvarchar2 |
| PASSWORD | Пароль пользователя | nvarchar2 |
| CARD | Карта пользователя | nvarchar2 |
| ADMIN | Роль администратора | nvarchar2 |

В таблице 3.4 изображена структура таблицы «Tickets», содержащая информацию о билетах. Первичный ключ TICKET\_ID – идентификатор билета, ID\_PLAY хранит уникальный идентификатор игры, SECTOR – Сектор, ROW –ряд, PLACE –место, DATE\_PLAY – дата матча, TIME\_PLAY – время матча, USER\_ID – id пользователя, ID\_TICKET – ID билета.

Таблица 3.4 – Столбцы таблицы Tickets

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| TICKET\_ID | Идентификатор билета | int |
| ID\_PLAY | Идентификатор игры | int |
| SECTOR | Сектор арены | int |
| ROW\_ | Ряд сектора | int |
| PLACE | Место в ряду | int |
| DATE\_PLAY | Дата матча | date |
| TIME\_PLAY | Время матча | timestamp |
| USER\_ID | Идентификатор пользователя | int |

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

**3.2 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 2 пользователя: администратор и пользователь (фанат). Каждый администратор и пользователь имеют разные привилегии в соответствии с их статусами.

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

3.3 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/обновление пользователя (фаната);
* добавление/обновление администратора;
* добавление/обновление информации об игры;
* добавление/удаление команды;
* обновление информации о команде;
* добавление/удаление билета;
* экспорт и импорт таблицы в формат xml;
* просмотр информации о группе;
* просмотр информации о заполняемости в секторе на матче;
* просмотр информации об игре;
* просмотр информации о результатах игры.

Все хранимые процедуры представлены в приложении В данной записки.

# **Описание процедур экспорта и импорта**

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

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 PLAYES. Пример реализации функции экспорта представлен в листинге 4.1.

|  |
| --- |
| SELECT file\_name FROM dba\_data\_files;  create or replace directory UTLDATA AS 'D:/BSTU';  create or replace procedure ExportPlayesToXml is  DOC DBMS\_XMLDOM.DOMDocument;  XDATA XMLTYPE;  CURSOR XMLCUR IS SELECT XMLELEMENT("PLAYES",  XMLAttributes('http://www.w3.org/2001/XMLSchema' AS "xmlns:xsi", 'http://www.oracle.com/Users.xsd' AS "xsi:nonamespaceSchemaLocation"),  XMLAGG(XMLELEMENT("PLAYES",  xmlelement("HOMETEAM", PLAYES.HOMETEAM),  xmlelement("GUESTTEAM", PLAYES.GUESTTEAM),  xmlelement("HOMEGOALS", PLAYES.HOMEGOALS),  xmlelement("GUESTGOALS", PLAYES.GUESTGOALS),  xmlelement("HOMEIMAGE", PLAYES.HOMEIMAGE),  xmlelement("GUESTIMAGE", PLAYES.GUESTIMAGE),  xmlelement("DATE", PLAYES.DATE\_)))) from PLAYES;  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/PLAYES.xml');  END; |

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, т. к. 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  AddFans('TEST', 'test@gmail.com', '112233', 1111111111111111);  end loop;  end; |

Листинг 5.1 – Анонимный блок для заполнения таблицы большим количеством данных

Таким образом было добавлено 100000 строк в таблицу USERS.

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 USERS where ID > 50000 and 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.

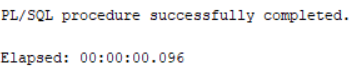


Рисунок 5.3 – Результат отбора данных

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

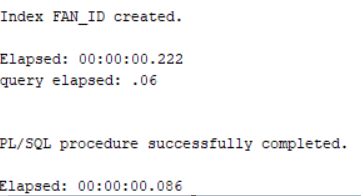


Рисунок 5.4 – Результат после создания индекса

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 данной работе используется маскирование паролей пользователей используя метод хэширования MD-5. Для начала мы покажем, как выглядит пароль в таблице без использования процедуры для хэширования пароля (рисунок 6.1).

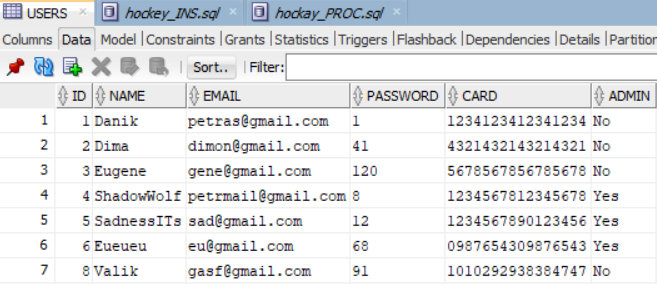


Рисунок 6.1 – Данные о пользователе с паролем без маскирования

А теперь мы применим нашу процедуру к пользователям и захэшируем их пароль. Результат на рисунке 6.2

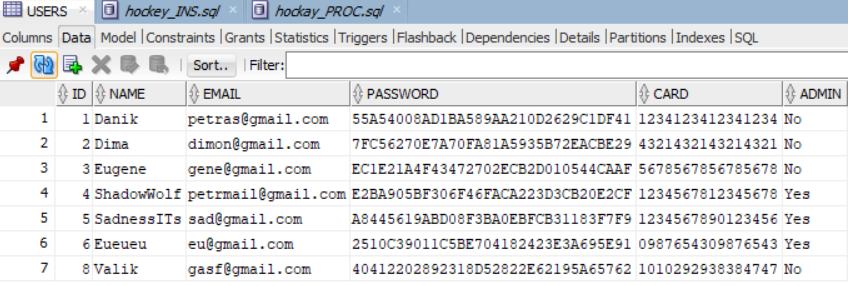


Рисунок 6.2 – Данные о пользователе с хэшированным паролем

Процедура с хэшированием показана на листинге 6.3

|  |
| --- |
| CREATE OR REPLACE PROCEDURE hash\_password  IS  BEGIN  FOR rec IN (SELECT ID, PASSWORD FROM USERS)  LOOP  UPDATE users  SET PASSWORD = RAWTOHEX(DBMS\_CRYPTO.HASH(rec.PASSWORD, DBMS\_CRYPTO.HASH\_MD5))  WHERE ID = rec.ID;  END LOOP;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  END; |

Листинг 6.3 – Процедура для хэширования паролей пользователей

Таким образом, обеспечивается защита пользователей от возможности взлома их аккаунтов, защищая их данные с помощью способа маскирования паролей пользователей, а конкретно хэшированием MD-5.

# **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 4 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 USERS. Результат выполнения процедуры представлен на рисунке 7.1.

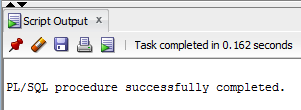


Рисунок 7.1 – Выполненная процедура

Проверим результат, выполнив select-запрос к самой таблице USERS. 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).

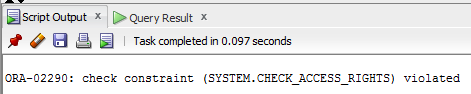


Рисунок 7.3 – Результат выполнения процедуры с некорректными входными параметрами

Пользователь также может изменять имеющиеся данные. Проверим это на таблице TEAMS. Для начала создадим новую строку, у которой в столбце POINTS пропишем 3. Создадим мы ее с также с помощью процедур. Результат выполнения представлен на рисунке 7.4.

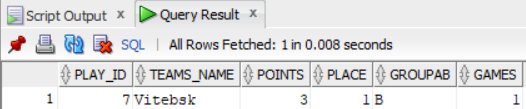


Рисунок 7.4 – Создание строчки в таблице Admins

А теперь изменим значение столбца POINTS на, например, 6. В итоге в таблице у нас будет уже измененная строка (рисунок 7.5).

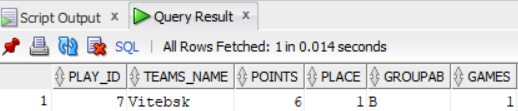


Рисунок 7.5 – Измененная строка в таблице

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



Рисунок 7.6 – Экспортированный файл PLAYES.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 для покупки билетов на хоккейные матчи с реализацией технологии маскирования данных. 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 (администраторы и пользователи (фанаты));
* изменение информации об администраторах, фанатов, игр и команд;
* вывод данных о заполняемости в секторе на матче, информация о группе, о результатах игры и будущих игр.

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

Была реализована технология «маскирования паролей с помощью хэширования».

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: 01.05.2023.
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: 01.05.2023.
3. Генерируем MD5 HASH на Oracle PL/SQL [Электронный ресурс]. Режим доступа: https://babakov.net/blog/oracledb/313.html. Дата доступа: 02.05.2023.

# **ПРИЛОЖЕНИЕ А**

Создание таблиц и последовательностей

|  |
| --- |
| CREATE SEQUENCE USERS\_ID\_SEQ;  CREATE TABLE USERS (  ID INTEGER DEFAULT USERS\_ID\_SEQ.NEXTVAL NOT NULL,  NAME NVARCHAR2(50) NOT NULL,  EMAIL NVARCHAR2(50) NOT NULL,  PASSWORD NVARCHAR2(50) NOT NULL,  CARD NVARCHAR2(50) NOT NULL,  ADMIN NVARCHAR2(50) DEFAULT 'No',  CONSTRAINT USERS\_PK PRIMARY KEY(ID)  );  CREATE SEQUENCE PLAYES\_ID\_SEQ;  CREATE TABLE PLAYES (  HOMETEAM NVARCHAR2(50) NOT NULL,  GUESTTEAM NVARCHAR2(50) NOT NULL,  HOMEGOALS INTEGER DEFAULT 0 NOT NULL,  GUESTGOALS INTEGER DEFAULT 0 NOT NULL,  HOMEIMAGE BLOB,  GUESTIMAGE BLOB,  DATE\_ DATE,  CONSTRAINT PLAYES\_PK PRIMARY KEY(DATE\_)  );  CREATE SEQUENCE TEAMS\_ID\_SEQ;  CREATE TABLE TEAMS (  PLAY\_ID INTEGER DEFAULT TEAMS\_ID\_SEQ.NEXTVAL NOT NULL,  TEAMS\_NAME NVARCHAR2(50) NOT NULL,  POINTS INTEGER DEFAULT 0 NOT NULL,  PLACE INTEGER DEFAULT 0 NOT NULL,  GROUPAB NVARCHAR2(50) NOT NULL,  GAMES INTEGER DEFAULT 0 NOT NULL,  CONSTRAINT CHECK\_GROUPAB CHECK (GROUPAB in ('A', 'B')),  CONSTRAINT TEAMS\_PK PRIMARY KEY(PLAY\_ID)  );  CREATE SEQUENCE TICKETS\_ID\_SEQ;  CREATE TABLE TICKETS (  TICKET\_ID INTEGER DEFAULT TICKETS\_ID\_SEQ.NEXTVAL NOT NULL,  ID\_PLAY INTEGER,  SECTOR INTEGER NOT NULL,  ROW\_ INTEGER NOT NULL,  PLACE INTEGER NOT NULL,  DATE\_PLAY DATE,  TIME\_PLAY TIMESTAMP,  USER\_ID INTEGER,  CONSTRAINT TICKETS\_PK PRIMARY KEY(TICKET\_ID)  );  ALTER TABLE TICKETS ADD CONSTRAINT PLAY\_FK FOREIGN KEY (ID\_PLAY) REFERENCES TEAMS(PLAY\_ID);  ALTER TABLE TICKETS ADD CONSTRAINT DATE\_FK FOREIGN KEY (DATE\_PLAY) REFERENCES PLAYES(DATE\_);  ALTER TABLE TICKETS ADD CONSTRAINT USER\_FK FOREIGN KEY (USER\_ID) REFERENCES USERS(ID);  ALTER TABLE TICKETS MODIFY TIME\_PLAY TIMESTAMP;  --DROP TABLE USERS;  --DROP TABLE PLAYES;  --DROP TABLE TEAMS;  --DROP TABLE TICKETS; |

ПРИЛОЖЕНИЕ Б

Создание пользователей и ролей

|  |
| --- |
| alter session set "\_ORACLE\_SCRIPT"=true;  CREATE ROLE ADMIN\_ROLE;  GRANT ALL ON USERS TO ADMIN\_ROLE;  GRANT ALL ON PLAYES TO ADMIN\_ROLE;  GRANT ALL ON TEAMS TO ADMIN\_ROLE;  GRANT ALL ON TICKETS TO ADMIN\_ROLE;  CREATE ROLE FAN\_ROLE;  GRANT ALL ON PLAYES TO FAN\_ROLE;  GRANT ALL ON TEAMS TO FAN\_ROLE;  GRANT ALL ON TICKETS TO FAN\_ROLE;  -- users  CREATE USER Dan IDENTIFIED BY dss9;  GRANT ADMIN\_ROLE TO Dan;  CREATE USER Dima IDENTIFIED BY dss;  GRANT FAN\_ROLE TO Dima; |

ПРИЛОЖЕНИЕ В

Создание хранимых процедур

|  |
| --- |
| -- procedures  -- logins  create or replace procedure AddFan(IID USERS.ID%type, ILOGIN\_NAME USERS.NAME%type, IEMAIL USERS.EMAIL%type, IPASSWORD USERS.PASSWORD%type, ICARD USERS.CARD%type)  is  begin  insert into USERS (ID, NAME, EMAIL, PASSWORD, CARD)  values (IID, ILOGIN\_NAME, IEMAIL, ORA\_HASH(IPASSWORD, 128), ICARD);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure AddFans(ILOGIN\_NAME USERS.NAME%type, IEMAIL USERS.EMAIL%type, IPASSWORD USERS.PASSWORD%type, ICARD USERS.CARD%type)  is  begin  insert into USERS (NAME, EMAIL, PASSWORD, CARD)  values (ILOGIN\_NAME, IEMAIL, ORA\_HASH(IPASSWORD, 128), ICARD);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure UpdateFan(IID USERS.ID%type, ILOGIN\_NAME USERS.NAME%type, IEMAIL USERS.EMAIL%type, IPASSWORD USERS.PASSWORD%type, ICARD USERS.CARD%type)  is  begin  update USERS set  ID = IID,  NAME = ILOGIN\_NAME,  EMAIL = IEMAIL,  PASSWORD = ORA\_HASH(IPASSWORD, 128),  CARD = ICARD;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  -- admins  create or replace procedure AddAdmin(IID USERS.ID%type, ILOGIN\_NAME USERS.NAME%type, IEMAIL USERS.EMAIL%type, IPASSWORD USERS.PASSWORD%type,  ICARD USERS.CARD%type, IADMIN USERS.ADMIN%type)  is  begin  insert into USERS (ID, NAME, EMAIL, PASSWORD, CARD, ADMIN)  values (IID, ILOGIN\_NAME, IEMAIL, ORA\_HASH(IPASSWORD, 128), ICARD, IADMIN);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure UpdateAdmin(IID USERS.ID%type, ILOGIN\_NAME USERS.NAME%type, IEMAIL USERS.EMAIL%type, IPASSWORD USERS.PASSWORD%type,  ICARD USERS.CARD%type, IADMIN USERS.ADMIN%type)  is  begin  update USERS set  ID = IID,  NAME = ILOGIN\_NAME,  EMAIL = IEMAIL,  PASSWORD = ORA\_HASH(IPASSWORD, 128),  CARD = ICARD,  ADMIN = IADMIN;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  -- playes  create or replace procedure AddPlay (IHOMETEAM PLAYES.HOMETEAM%type, IGUESTTEAM PLAYES.GUESTTEAM%type, IDATE PLAYES.DATE\_%type)  is  begin  insert into PLAYES (HOMETEAM, GUESTTEAM, DATE\_)  values (IHOMETEAM, IGUESTTEAM, IDATE);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure UpdatePlays (IHOMEGOALS PLAYES.HOMEGOALS%type, IGUESTGOALS PLAYES.GUESTGOALS%type, IDATE PLAYES.DATE\_%type)  is  begin  update PLAYES set  HOMEGOALS = IHOMEGOALS,  GUESTGOALS = IGUESTGOALS  WHERE DATE\_ = IDATE;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  -- teams  create or replace procedure AddTeam (IPLAY\_ID TEAMS.PLAY\_ID%type, ITNAME TEAMS.TEAMS\_NAME%type, IGROUP TEAMS.GROUPAB%type)  is  begin  insert into TEAMS (PLAY\_ID, TEAMS\_NAME, GROUPAB)  values (IPLAY\_ID, ITNAME, IGROUP);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure UpdateTeams(IPLAY\_ID TEAMS.PLAY\_ID%type, IPOINTS TEAMS.POINTS%type, IPLACE TEAMS.PLACE%type, IGAMES TEAMS.GAMES%type)  is  begin  update TEAMS set  POINTS = IPOINTS,  PLACE = IPLACE,  GAMES = IGAMES  WHERE PLAY\_ID = IPLAY\_ID;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure DeleteTeam (IPLAY\_ID TEAMS.PLAY\_ID%type)  is  begin  delete from TEAMS where PLAY\_ID = IPLAY\_ID;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  -- tickets  create or replace procedure AddTicket (ITICKET\_ID TICKETS.TICKET\_ID%type, IID\_PLAY TICKETS.ID\_PLAY%type, ISECTOR TICKETS.SECTOR%type, IROW TICKETS.ROW\_%type,  IPLACE TICKETS.PLACE%type, IDATE\_PLAY TICKETS.DATE\_PLAY%type, ITIME\_PLAY TICKETS.TIME\_PLAY%type, IUSER\_ID TICKETS.USER\_ID%type)  is  begin  insert into TICKETS (TICKET\_ID, ID\_PLAY, SECTOR, ROW\_, PLACE, DATE\_PLAY, TIME\_PLAY, USER\_ID)  values (ITICKET\_ID, IID\_PLAY, ISECTOR, IROW, IPLACE, IDATE\_PLAY, ITIME\_PLAY, IUSER\_ID);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  create or replace procedure DeleteTicket (ITICKET\_ID TICKETS.TICKET\_ID%type)  is  begin  delete from TICKETS where TICKET\_ID = ITICKET\_ID;  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  -- others  declare procedure ViewGroupInfo(IGROUP TEAMS.GROUPAB%type)  is  cursor groupab\_cursor is select TEAMS\_NAME, POINTS, PLACE, GROUPAB, GAMES  from TEAMS where GROUPAB = IGROUP;  begin  for any\_group in groupab\_cursor  loop  dbms\_output.put\_line(any\_group.TEAMS\_NAME || ' ' ||  any\_group.POINTS || ' ' || any\_group.PLACE || ' ' ||  any\_group.GROUPAB || ' ' || any\_group.GAMES || ' ');  end loop;  end;  begin  ViewGroupInfo('A');  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  declare procedure ViewSectorOccupancy (ISECTOR TICKETS.SECTOR%type)  is  cursor sector\_occupancy\_cursor is select TICKET\_ID, SECTOR, ROW\_, PLACE, DATE\_PLAY, USER\_ID  from TICKETS where SECTOR = ISECTOR;  begin  for any\_sector\_occupancy in sector\_occupancy\_cursor  loop  dbms\_output.put\_line(any\_sector\_occupancy.TICKET\_ID || ' ' ||  any\_sector\_occupancy.SECTOR || ' ' ||  any\_sector\_occupancy.ROW\_ || ' ' ||  any\_sector\_occupancy.PLACE || ' ' ||  any\_sector\_occupancy.DATE\_PLAY || ' ' ||  any\_sector\_occupancy.USER\_ID || ' ');  end loop;  end;  begin  ViewSectorOccupancy(3);  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  declare procedure ViewGame (IDATE PLAYES.DATE\_%type)  is  cursor game\_cursor is select HOMETEAM, GUESTTEAM, DATE\_  from PLAYES where DATE\_ = IDATE;  begin  for any\_game in game\_cursor  loop  dbms\_output.put\_line(any\_game.HOMETEAM || ' ' ||  any\_game.GUESTTEAM || ' ' ||  any\_game.DATE\_ || ' ');  end loop;  end;  begin  ViewGame(to\_date('2023/05/13', 'yyyy/mm/dd'));  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  declare procedure ViewResult (IDATE PLAYES.DATE\_%type)  is  cursor result\_cursor is select HOMETEAM, HOMEGOALS, GUESTTEAM, GUESTGOALS, DATE\_  from PLAYES where DATE\_ = IDATE;  begin  for any\_result in result\_cursor  loop  dbms\_output.put\_line(any\_result.HOMETEAM || ' ' ||  any\_result.HOMEGOALS || ' ' ||  any\_result.GUESTTEAM || ' ' ||  any\_result.GUESTGOALS || ' ' ||  any\_result.DATE\_ || ' ');  end loop;  end;  begin  ViewResult(to\_date('2023/05/11', 'yyyy/mm/dd'));  exception  when others  then DBMS\_OUTPUT.put\_line(sqlerrm);  end;  CREATE OR REPLACE PROCEDURE hash\_password  IS  BEGIN      FOR rec IN (SELECT ID, PASSWORD FROM USERS)      LOOP      UPDATE users      SET PASSWORD = RAWTOHEX(DBMS\_CRYPTO.HASH(rec.PASSWORD, DBMS\_CRYPTO.HASH\_MD5))      WHERE ID = rec.ID;    END LOOP;      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 'D:/BSTU';  create or replace procedure ExportPlayesToXml  is  DOC DBMS\_XMLDOM.DOMDocument;  XDATA XMLTYPE;  CURSOR XMLCUR IS  SELECT XMLELEMENT("PLAYES",  XMLAttributes('http://www.w3.org/2001/XMLSchema' AS "xmlns:xsi",  'http://www.oracle.com/Users.xsd' AS "xsi:nonamespaceSchemaLocation"),  XMLAGG(XMLELEMENT("PLAYES",  xmlelement("HOMETEAM", PLAYES.HOMETEAM),  xmlelement("GUESTTEAM", PLAYES.GUESTTEAM),  xmlelement("HOMEGOALS", PLAYES.HOMEGOALS),  xmlelement("GUESTGOALS", PLAYES.GUESTGOALS),  xmlelement("HOMEIMAGE", PLAYES.HOMEIMAGE),  xmlelement("GUESTIMAGE", PLAYES.GUESTIMAGE),  xmlelement("DATE", PLAYES.DATE\_)  ))) from PLAYES;  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/PLAYES.xml');  END;  begin  ExportPlayesToXml();  end;  -- drop procedure ExportPlayesToXml;  ----------------------------------------------------------------  -- xml import  create or replace procedure ImportPlayesFromXml  IS  L\_CLOB CLOB;  L\_BFILE BFILE := BFILENAME('UTLDATA', 'PLAYES.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 PLAYES%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,  'HOMETEAM/text()',et.HOMETEAM);  Dbms\_Xslprocessor.Valueof(V\_Current\_Node,  'GUESTTEAM/text()',et.GUESTTEAM);  Dbms\_Xslprocessor.Valueof(V\_Current\_Node,  'HOMEGOALS/text()',et.HOMEGOALS);  dbms\_xslprocessor.valueof(v\_current\_node,  'GUESTGOALS/text()',et.GUESTGOALS);  dbms\_xslprocessor.valueof(v\_current\_node,  'HOMEIMAGE/text()',et.HOMEIMAGE);  dbms\_xslprocessor.valueof(v\_current\_node,  'GUESTIMAGE/text()',et.GUESTIMAGE);  dbms\_xslprocessor.valueof(v\_current\_node,  'DATE/text()',et.DATE\_);  insert into PLAYES (HOMETEAM, GUESTTEAM, HOMEGOALS, GUESTGOALS, HOMEIMAGE, GUESTIMAGE, DATE\_)  values (et.HOMETEAM, et.GUESTTEAM, et.HOMEGOALS, et.GUESTGOALS, et.HOMEIMAGE, et.GUESTIMAGE, et.DATE\_);  end loop;  DBMS\_LOB.FREETEMPORARY(L\_CLOB);  DBMS\_XMLPARSER.FREEPARSER(P);  DBMS\_XMLDOM.FREEDOCUMENT(V\_DOC);  commit;  END;  begin  ImportPlayesFromXml();  end;  select \* from PLAYES; |

ПРИЛОЖЕНИЕ Д

Скрипт тестирования производительности базы данных

|  |
| --- |
| -- testing 100000 rows  -- delete Logins;  set timing on serveroutput on  begin  for Lcntr IN 1..100000  loop  AddFans('TEST', 'test@gmail.com', '112233', 1111111111111111);  end loop;  end;  select \* from USERS;  declare  t1 number := dbms\_utility.get\_time();  begin  for r in (select \* from USERS where ID > 50000 and ID < 70000) loop null; end loop;  dbms\_output.put\_line('query elapsed: '||(dbms\_utility.get\_time() - t1)/100);  end;  create index fan\_id on USERS(ID, Password);  declare  t1 number := dbms\_utility.get\_time();  begin  for r in (select \* from USERS where ID > 50000 and ID < 70000) loop null; end loop;  dbms\_output.put\_line('query elapsed: '||(dbms\_utility.get\_time() - t1)/100);  end;  drop index fan\_id;  select \* from USERS; |