МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий
Кафедра Информационные системы и технологии
Специальность 1–40 01 01 Программное обеспечение информационных технологий

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ НА ТЕМУ:

«Реализация базы данных магазина музыкальных инструментов с применением технологии полнотекстового поиска в БД»

Выполнил студент Пшенко А.Ф.
(Ф.И.О.)

Руководитель работы асс. Нистюк Ольга Александровна
(учен. степень, звание, должность, Ф.И.О., подпись)

И.о. зав. кафедрой ст. преп. Блинова Е.А.
(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовая работа защищена с оценкой _____

Минск 2023

Содержание

Введение

В современном мире люди на регулярной основе посещают различные магазины, поэтому приложения для магазинов стали очень актуальными и востребованными, поскольку позволяют пользователям удобно и быстро выбирать, заказывать и оплачивать товары, не выходя из дома.

Целью курсового проектирования является разработка базы данных для магазина музыкальных инструментов.

В базе данных разграничены возможности администратора и пользователя.

Пользователь может выполнять поиск по инструментам, резервировать инструменты, оставлять и просматривать отзывы, просматривать все инструменты и популярные инструменты.

Администратор имеет все возможности клиента, а также может управлять инструментами: добавлять, изменять и удалять инструменты, производить анализ продукции: просматривать количество проданных инструментов по периодам и информацию о них.

В ходе выполнения курсового проектирования будут решены следующие задачи:

– анализ литературы по теме работы;
– изучение требований и определение вариантов использования;
– анализ и проектирование модели базы данных и ограничений целостности;
– создание необходимых объектов;
– реализация импорта и экспорта данных;
– описание используемой технологии;
– тестирование производительности;
– создание руководства пользователя.

Целями базы данных являются хранение, организация и обеспечение безопасности данных, а задачами являются структуризация данных в таблицы и использование индексов для ускоренного доступа к данным.

1 Постановка задачи

1.1 Обзор аналогичных решений

На сегодняшний день существует большое количество интернет-магазинов по продаже музыкальных инструментов.

В качестве первого аналога был рассмотрен «Musicmarket.by». Он позволяет выполнять поиск по инструментам, просматривать каталог, выполнять фильтрацию и сортировку по критериям, резервировать инструмент, оставлять отзывы, а также сравнивать инструменты между собой. Клиент может взаимодействовать с сайтом как в режиме гостя, так и авторизованного пользователя.

В качестве второго аналогичного решения был рассмотрен интернет-магазин «Popmusic.ru». Функционал схож с «Musicmarket.by»: присутствует возможность выполнить поиск и фильтрацию по определенным критериям, добавить и удалить товар из корзины, перейти на страницу с описанием конкретного товара, прочитать отзывы, а также оставить свой. К минусам можно отнести отсутствие возможности сравнения товаров и недостаточно функциональную фильтрацию по критериям.

«Piano.by» – еще один известный интернет-магазин музыкальных инструментов. Присутствует возможность зарегистрироваться/авторизоваться или взаимодействовать с сайтом в режиме гостя, выполнить поиск по каталогу и фильтрацию, добавление и удаление товара из корзины, а также сравнение нескольких товаров по определенным критериям. Пользователь может оформить заказ, просмотреть отзывы к товарам и оставить свои.

1.2 Спецификация требований

База данных должна быть реализована в СУБД Oracle. Подключение к базе данных должно осуществляться при помощи обычного пользователя и администратора. Доступ к данным должен осуществляться через хранимые процедуры, права на выполнение которых должны быть выданы нужным пользователям.

Должен быть реализован импорт данных из JSON-файлов, экспорт данных в формат JSON. 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.

Необходимо использовать технологию полнотекстового поиска.

Обычному пользователю должна быть доступна возможность выполнять поиск по инструментам, резервировать инструменты, оставлять и просматривать отзывы, просматривать все инструменты и популярные инструменты.

Администратору должна быть доступна возможность использовать любые процедуры для работы с базой данных, в том числе добавление, удаление и изменение инструментов.

Диаграмма вариантов использования представлена на рисунке 1.1.

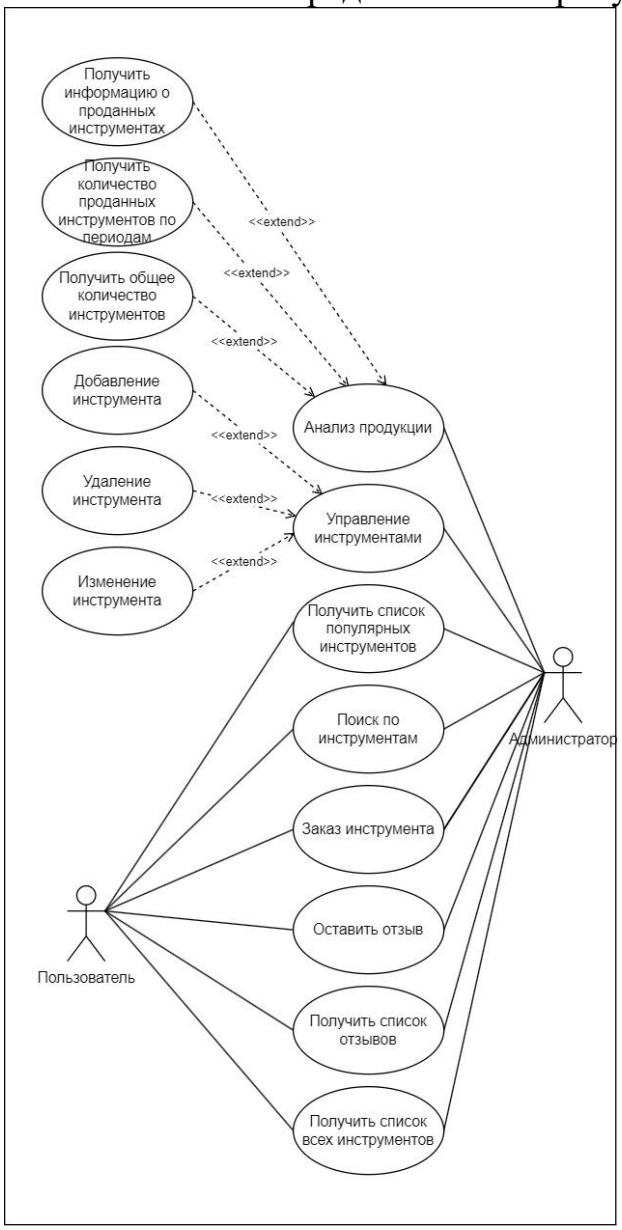

Рисунок 1.1 – Диаграмма вариантов использования базы данных

На диаграмме вариантов использования применяются два типа основных сущностей: варианты использования и группы пользователей. Каждый вариант использования обозначает набор действий, который может быть использован актёром для взаимодействии с системой.

2 Проектирование базы данных

Логическая схема базы данных представлена на рисунке 2.1.

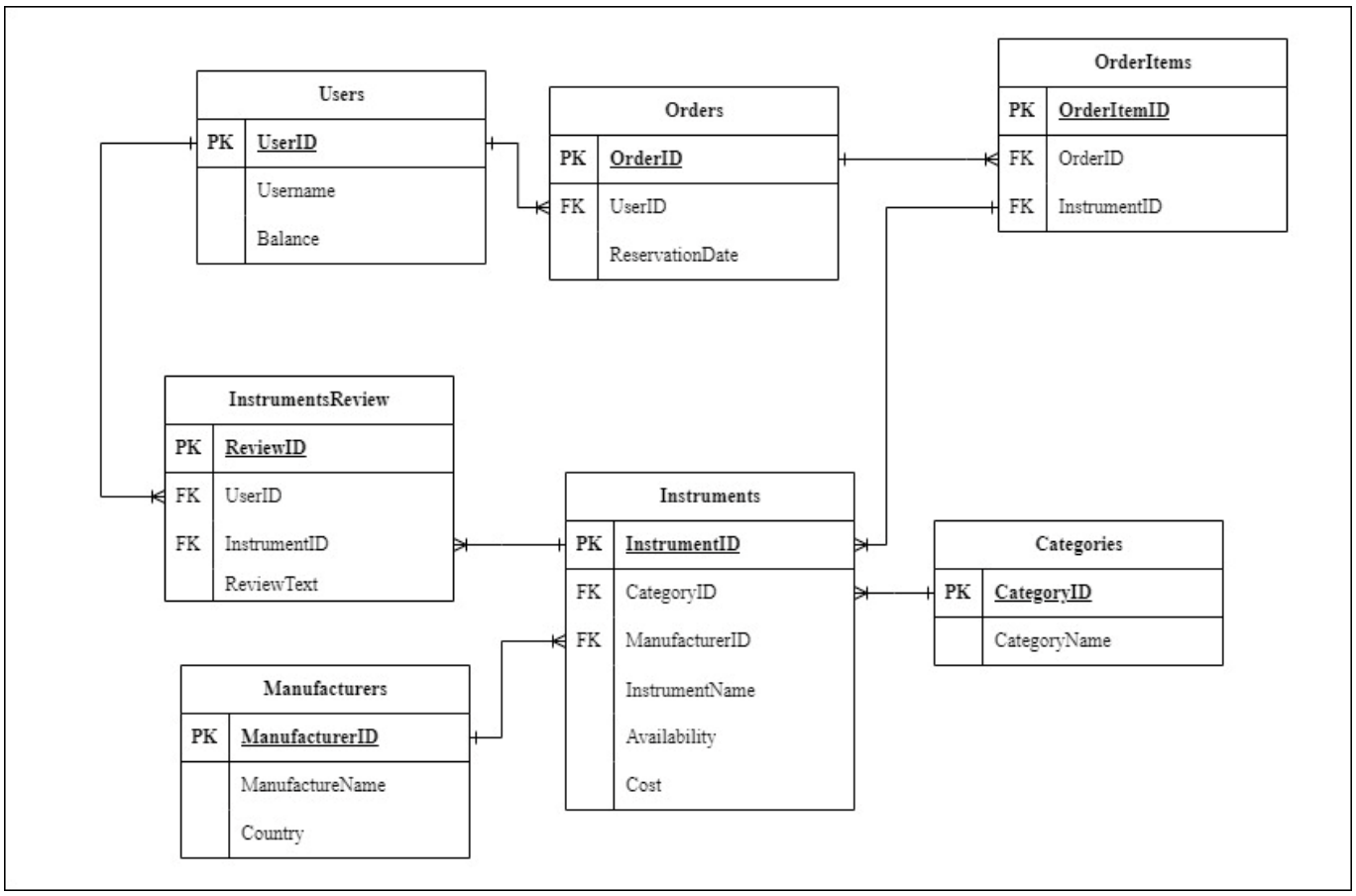

Рисунок 2.1 – Логическая схема базы данных

База данных содержит семь таблиц, хранящих информацию о пользователях, инструментах, производителях, заказах, отзывах и категориях инструментов. Листинг создания таблиц представлен в приложении А.

Таблица Users хранит информацию о пользователях. Описание её столбцов представлено в таблице 2.1.

Таблица 2.1 – Описание таблицы Users

Название столбца	Тип данных	Описание
userid	int	идентификатор пользователя, первичный ключ
username	nvarchar2(50)	имя пользователя
balance	number(10, 2)	баланс пользователя

Таблица Orders хранит информацию о заказах пользователей. Описание её столбцов представлено в таблице 2.2.

Таблица 2.2 – Описание таблицы Orders

Название столбца	Тип данных	Описание
orderid	int	идентификатор заказа, первичный ключ
userid	nvarchar2(50)	идентификатор пользователя, внешний ключ
reservationdate	date	дата заказа

Таблица OrderItems содержит информацию о конкретных элементах, которые были заказаны. Описание ее столбцов представлено в таблице 2.3.

Таблица 2.3 – Описание таблицы OrderItems

Название столбца	Тип данных	Описание
orderitemid	int	идентификатор элемента заказа, первичный ключ
orderid	int	идентификатор заказа, внешний ключ
instrumentid	int	идентификатор инструмента, внешний ключ

Таблица InstrumentsReview содержит информацию об отзывах. Описание ее столбцов представлено в таблице 2.4.

Таблица 2.4 – Описание таблицы InstrumentsReview

Название столбца	Тип данных	Описание
reviewid	int	идентификатор отзыва, первичный ключ
userid	int	идентификатор пользователя, внешний ключ
instrumentid	date	идентификатор инструмента, внешний ключ
reviewtext	nvarchar2(1000)	текст отзыва

Таблица Manufacturers содержит информацию о производителях. Описание ее столбцов представлено в таблице 2.5.

Таблица 2.5 – Описание таблицы Manufacturers

Название столбца	Тип данных	Описание
manufacturerid	int	идентификатор производителя, первичный ключ
manufacturername	nvarchar2(50)	имя производителя
country	nvarchar2(50)	страна производства

Таблица Instruments содержит информацию об инструментах. Описание ее столбцов представлено в таблице 2.6.

Таблица 2.6 – Описание таблицы Instruments

Название столбца	Тип данных	Описание
instrumentid	int	идентификатор инструмента, первичный ключ
categoryid	int	идентификатор категории, внешний ключ
manufacturerid	int	идентификатор производителя, внешний ключ

Продолжение таблицы 2.6

Название столбца	Тип данных	Описание
instrumentname	varchar(50)	название инструмента
availability	number(1,0)	наличие инструмента
cost	number(10,2)	цена инструмента

Таблица Instruments содержит информацию о категории инструментов. Описание ее столбцов представлено в таблице 2.7.

Таблица 2.7 – Описание таблицы Instruments

Название столбца	Тип данных	Описание
categoryid	int	идентификатор категории, первичный ключ
categoryname	nvarchar2(50)	имя категории

Таким образом, на данном этапе были созданы все таблицы и связи между ними.

3 Разработка объектов базы данных

3.1 Разработка таблиц

Все таблицы создаются в табличном пространстве администратора в базе данных MusicShop.

Каждая таблица имеет столбец идентификатора строки типа int. При добавлении строки в таблицу ей автоматически присваивается идентификатор при помощи автоинкремента. Для хранения строковых значений используется тип данных nvarchar2. Для хранения дат – date. Для хранения денежных значений – number(10, 2). Для хранения наличия инструмента используется тип number с допустимыми значениями 1 и 0. Описание столбцов таблиц приведено в разделе 2.

3.2 Разработка хранимых процедур

Описание используемых хранимых процедур представлено в таблице 3.1.

Таблица 3.1 – Описание используемых процедур

Название процедуры	Описание процедуры
InsertInstrument	Добавляет новый инструмент
UpdateInstrument	Обновляет информацию об инструменте
DeleteInstrument	Удаляет инструмент
CountSoldInstruments	Выводит количество проданных инструментов за период
GetSoldInstrumentsInfo	Выводит информацию о проданных инструментах
ReserveInstrument	Резервирует инструмент
CountAllInstruments	Выводит количество всех инструментов
ShowTopInstruments	Выводит популярные инструменты
ShowAllInstruments	Выводит всю информацию об инструментах
CreateReview	Создает отзыв на инструмент
ShowAllReviews	Выводит все отзывы
SearchInstruments	Позволяет найти инструменты с помощью технологии полнотекстового поиска

Листинг создания процедур представлен в приложении Б.

3.3 Создание пользователей

Для взаимодействия с базой данных были созданы два пользователя: MusicShop_ADMIN и MusicShop_USER. Им предоставлены права на вызов различных процедур.

Процедуры, которые разрешено вызывать пользователю:

– ReserveInstrument;

– ShowTopInstruments;

– ShowAllInstruments;

– CreateReview;
– ShowAllReviews;
– SearchInstruments.

Администратор имеет возможность вызвать любую из процедур.

Таким образом, в ходе данного раздела было выполнено создание объектов для базы данных магазина музыкальных инструментов, таких как таблицы и хранимые процедуры. Также были созданы два пользователя с правами на вызов различных процедур.

Листинг создания пользователей и выдачи им необходимых привилегий представлен в приложении В.

4 Описание процедур импорта и экспорта

Для генерации большого объема данных (более 100 000 строк), которые затем сохраняются в JSON-файл и импортируются в таблицу Instruments, был написан скрипт на Python с использованием библиотеки Faker. С ее помощью можно генерировать уникальные и осмысленные фейковые данные, такие как имена, адреса, электронные почты и т.д. Листинг скрипта-генератора JSON для таблицы Instruments представлен в листинге 4.1.

```
import json
from faker import Faker
import random

fake = Faker()

data = []
for _ in range(100_001):
    instrument = {
    "InstrumentID": _ + 1,
    "InstrumentName": fake.name() + ' ' + fake.word(),
    "CategoryID": random.randint(1, 7),
    "ManufacturerID": random.randint(1, 29),
    "Availability": random.choice([0, 1]),
    "Cost": round(random.uniform(1, 10000), 2)}
    data.append(instrument)

json_file_path = 'E:/BSTU/3_course/1term/Databases/Instruments.json'
with open(json_file_path, 'w') as json_file:
    json.dump(data, json_file, indent=2)
```

Листинг 4.1 – Генерация JSON-файла

После этого были созданы две процедуры для обработки данных: ImportInstrumentsFromJSON и ExportInstrumentsToJSON. Первая процедура используется для импорта данных, сгенерированных с помощью Python-скрипта, в таблицу Instruments. Вторая процедура позволяет экспортировать данные из таблицы Instruments их с сохранением в JSON-файл.

В данном разделе мы рассмотрели заполнение таблиц данными (импорте) и выгрузке данных из таблицы (экспорте), предложив решение на основе использования хранимых процедур и скрипта-генератора на Python.

Листинг процедур импорта и экспорта данных приведен в приложении Г.

5 Тестирование производительности

Для тестирования производительности заполним таблицу большим количеством данных (100 000 строк) из JSON-файла с помощью процедуры импорта.

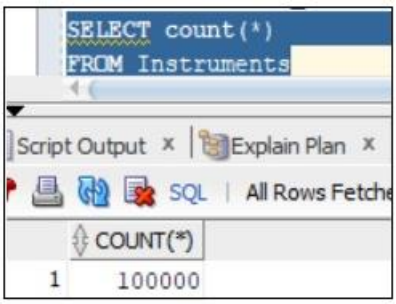

Рисунок 5.1 – Количество строк в таблице Instruments

Теперь проверим, какое количество времени занимает выполнение запроса для поиска инструмента по названию и цене.

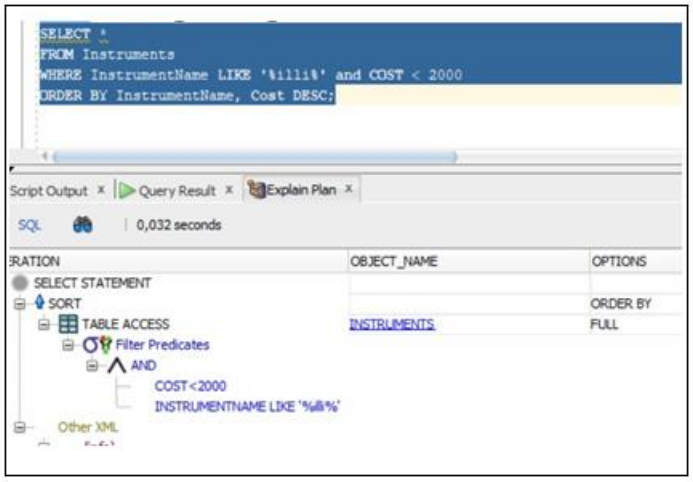

Рисунок 5.2 – Время поиска инструмента

Время ожидания составило 0.032 секунды, однако в реальности в базах данных хранятся миллионы записей, и поиск по ним может занимать значительное время. В данном случае мы просто смоделировали такую ситуацию.

Затем необходимо рассмотреть план выполнения этого запроса, как показано на рисунке 5.3.

Рисунок 5.3 – План запроса

Теперь попробуем оптимизировать данный запрос с помощью индекса.

```
CREATE INDEX idx_instrument_name ON Instruments(InstrumentName);
```

Листинг 5.4 – Создание индекса

Создание такого индекса обуславливаем тем, что этот индекс поможет ускорить выполнение запроса, индекс улучшит сортировку по столбцу InstrumentName.

Теперь проверим результат.

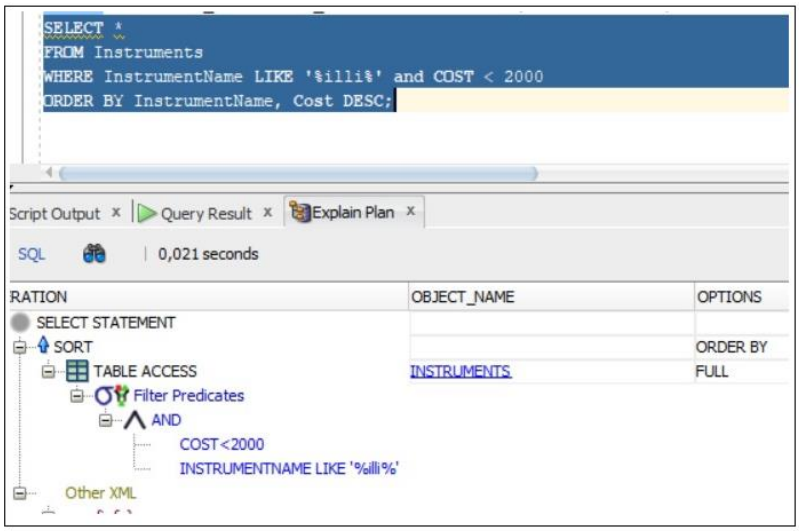

Рисунок 5.5 – Время выполнения запроса после создания индекса

После создания индекса время выполнения запроса уменьшилось до 0.021 секунд, что свидетельствует о том, что индекс является эффективным.

6 Описание технологии и её применения в базе данных

Полнотекстовый поиск – это поиск по всему содержимому документа или очень большого текста в базе данных, а не по определённым атрибутам, что позволяет заметно ускорять запросы к подобным данным и экономить вычислительные мощности.

В основе полнотекстовых поисков лежит индекс, куда загружаются все слова или словосочетания из текстовых документов или очень длинных строк. И при загрузке пользователем своего поискового запроса, поиск осуществляется по этому индексу.

Перед созданием данного индекса необходимо создать список слов и обработчик лексем, которые будут переданы в индекс и будут использоваться для взаимодействия с конструкциями языка. Код для создания приведен в листинге 6.1.

```
begin
     ctx_ddl.create_preference ('my_wordlist', 'BASIC_WORDLIST');
     ctx_ddl.create_preference ('my_lexer', 'AUTO_LEXER');
     ctx_ddl.set_attribute ('my_lexer', 'INDEX_STEMS', 'YES');
end;
```

Листинг 6.1 – создание лексера и списка слов

В листинге можно увидеть BASIC_WORDLIST: это тип набора слов. Подразумевается, что мы передаем стандартный набор лексических конструкций английского языка, так как именно на этом языке полнотекстовый поиск в Oracle работает максимально стабильно и корректно.

Следующий параметр AUTO_LEXER создает лексический анализатор для полнотекстового поиска, что позволяет искать слова не только по прямой комбинации букв, но использовать также формы слова в качестве результатов поиска.

Создание индекса представлено в листинге 6.2:

```
CREATE INDEX idx_instrument_name ON Instruments(InstrumentName)
INDEXTYPE IS CTXSYS.CONTEXT parameters ('LEXER my_lexer WORDLIST
my_wordlist');
```

Листинг 6.2 – создание индексов для полнотекстового поиска

Реализация процедуры полнотекстового поиска представлена в листинге 6.3.

```
CREATE OR REPLACE PROCEDURE SearchInstruments(p_SearchString
VARCHAR2) AS
  v_SearchString VARCHAR2(100);
  v_ResultCount NUMBER;
BEGIN
  IF LENGTH(p_SearchString) > 100 THEN
DBMS_OUTPUT.PUT_LINE('Error: Input string is too long.');
    RETURN;
  END IF;
```

Продолжение листинга 6.3

```
v_SearchString := '%' || p_SearchString || '%';
SELECT COUNT(*) INTO v_ResultCount FROM Instruments WHERE
CONTAINS(InstrumentName, v_SearchString) > 0;
  IF v_ResultCount > 0 THEN
    FOR instrument_rec IN (
      SELECT * FROM Instruments WHERE CONTAINS(InstrumentName,
v_SearchString) > 0
    ) LOOP
      DBMS_OUTPUT.PUT_LINE('InstrumentID: ' ||
instrument_rec.InstrumentID || ', InstrumentName: ' ||
instrument_rec.InstrumentName);
    END LOOP;
  ELSE
    DBMS_OUTPUT.PUT_LINE('No instruments found matching the search
criteria.');
  END IF;
END SearchInstruments;
BEGIN
  SearchInstruments('or');

END;
```

Листинг 6.3 – Создание процедуры полнотекстового поиска

Таким образом, в рамках реализуемой базы данных будет работать полнотекстовый поиск по названию музыкального инструмента.

7 Руководство пользователя

База данных поддерживает два пользователя. Для администратора и клиента созданы свои соединения к базе данных. У администратора – соединение ADMIN, у клиента – USER. Есть два сценария использования.

Первый – при входе от клиента. Есть возможность зарезервировать инструмент (ReserveInstrument), просмотреть популярные инструменты (ShowTopInstruments), просмотреть все инструменты (ShowAllInstruments), оставить отзыв (CreateReview), просмотреть все отзывы (ShowAllReviews), выполнить полнотекстовый поиск по ннструментам (SearchInstruments).

Второй – при входе от администратора: для вызова доступны все созданные и описанные ранее процедуры.

Заключение

При выполнении курсового проекта была создана база данных для магазина музыкальных инструментов. База данных была реализована в СУБД Oracle. Были реализованы все функциональные требования, а именно:

– управление инструментами (добавление, удаление, изменение);

– определение ролей (администратор, обычный пользователь);

– обеспечение резервирования инструментов;

– анализ продукции (количество проданных инструментов по периодам, популярные инструменты, общее количество инструментов).

Был реализован импорт и экспорт данных из базы данных в формат JSON. Была протестирована производительность базы данных при помощи таблицы, содержащей более 100000 строк, и создан индекс для улучшения производительности базы данных, а также реализована технология полнотекстового поиска.

Список используемых источников

1. Oracle Database Documentation [Электронный ресурс]. – Электронные данные. – Режим доступа: https://docs.oracle.com/en/database/oracle/oracle-database/index.html. Дата доступа: 30.11.2023.

2. Faker documentation [Электронный ресурс]. – Электронные данные. – Режим доступа: https://faker.readthedocs.io/en/master/. Дата доступа: 10.12.2023.

3. Python 3.12.1 documentation [Электронный ресурс]. – Электронные данные. – Режим доступа: 15.12.2023.

4. Нистюк О. А. Курс лекций по базам данных, [Электронный ресурс]. – Режим доступа: https://diskstation.belstu.by:5001/. – Дата доступа: 16.12.2023.

5. Full text search in Oracle [Электронный ресурс]. – Электронные данные. – Режим доступа: https://docs.oracle.com/en/database/oracle/oracle-database/21/adjsn/full-text-search-queries.html#GUID-58ADCDE5-7564-4DA0-BED7-B0DBFD5AE6FB. Дата доступа: 14.11.2023.

Приложение А

Листинг создания таблиц в базе данных

```
CREATE TABLE Categories (
    CategoryID INT PRIMARY KEY,
    CategoryName NVARCHAR2(50) NOT NULL UNIQUE
);
CREATE TABLE Manufacturers (
    ManufacturerID INT GENERATED BY DEFAULT AS IDENTITY (START WITH
1) PRIMARY KEY,
    ManufacturerName NVARCHAR2(50) NOT NULL UNIQUE,
    Country NVARCHAR2(50)
);
CREATE TABLE Instruments (
    InstrumentID INT GENERATED BY DEFAULT AS IDENTITY (START WITH 1)
PRIMARY KEY,
    InstrumentName VARCHAR(50) NOT NULL,
    CategoryID INT REFERENCES Categories(CategoryID) ON DELETE
CASCADE,
    ManufacturerID INT REFERENCES Manufacturers(ManufacturerID) ON
DELETE CASCADE,
    Availability NUMBER(1, 0) NOT NULL CHECK (Availability IN (0,
1)),
    Cost NUMBER(10, 2) NOT NULL CHECK (Cost > 0)
);
CREATE TABLE Users (
    UserID INT GENERATED BY DEFAULT AS IDENTITY (START WITH 1)
PRIMARY KEY,
    UserName NVARCHAR2(50) NOT NULL,
    Balance NUMBER(10, 2) NOT NULL CHECK (Balance >= 0)
);
CREATE TABLE Orders (
    OrderID INT GENERATED BY DEFAULT AS IDENTITY (START WITH 1)
PRIMARY KEY,
    UserID INT REFERENCES Users(UserID) ON DELETE CASCADE,
    ReservationDate DATE NOT NULL
);
CREATE TABLE OrderItems (
    OrderItemID INT GENERATED BY DEFAULT AS IDENTITY (START WITH 1)
PRIMARY KEY,
    OrderID INT REFERENCES Orders(OrderID) ON DELETE CASCADE,
    InstrumentID INT REFERENCES Instruments(InstrumentID) ON DELETE
CASCADE
);
CREATE TABLE InstrumentsReview (
    ReviewID INT GENERATED BY DEFAULT AS IDENTITY (START WITH 1)
PRIMARY KEY,
    UserID INT REFERENCES Users(UserID) ON DELETE CASCADE,
    InstrumentID INT REFERENCES Instruments(InstrumentID) ON DELETE
CASCADE,
    ReviewText NVARCHAR2(1000) NOT NULL
);
```

Приложение Б

Листинг создания процедур

```
CREATE OR REPLACE PROCEDURE CountSoldInstruments(p_StartDate
VARCHAR2, p_EndDate VARCHAR2) AS
  v_count NUMBER;
  v_StartDate DATE := TO_DATE(p_StartDate, 'YYYY-MM-DD');
  v_EndDate DATE := TO_DATE(p_EndDate, 'YYYY-MM-DD');
BEGIN
  SELECT COUNT(*) INTO v_count FROM Orders o
  JOIN OrderItems oi ON oi.OrderID = o.OrderID
  WHERE o.ReservationDate BETWEEN v_StartDate AND v_EndDate;

  IF v_count = 0 THEN
    DBMS_OUTPUT.PUT_LINE('No instruments sold in the specified
period.');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Amount of sold instruments in the specified
period: ' || TO_CHAR(v_count));
  END IF;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Error: No data available for counting in
the specified period.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END CountSoldInstruments;

EXEC CountSoldInstruments('2023-12-18', '2023-12-31');
drop procedure CountSoldInstruments


CREATE OR REPLACE PROCEDURE GetSoldInstrumentsInfo AS
BEGIN
  DECLARE
    v_sold_count NUMBER;
  BEGIN
    SELECT COUNT(DISTINCT oi.InstrumentID)
    INTO v_sold_count
    FROM OrderItems oi;

    IF v_sold_count = 0 THEN
      DBMS_OUTPUT.PUT_LINE('No instruments have been sold.');
      RETURN;
    END IF;
  END;

  FOR r IN (
    SELECT o.OrderID, u.UserName AS Buyer, i.InstrumentName,
COUNT(oi.InstrumentID) AS Quantity, o.ReservationDate
    FROM Orders o
    JOIN OrderItems oi ON o.OrderID = oi.OrderID
    JOIN Instruments i ON oi.InstrumentID = i.InstrumentID
```

```
    JOIN Users u ON o.UserID = u.UserID
    GROUP     BY     o.OrderID,     u.UserName,     i.InstrumentName,
o.ReservationDate
    ORDER BY o.ReservationDate
  ) LOOP
    DBMS_OUTPUT.PUT_LINE('OrderID: ' || r.OrderID ||
                         ', Buyer: ' || r.Buyer ||
                         ', Instrument: ' || r.InstrumentName ||
                         ', Quantity: ' || r.Quantity ||
                         ',  Date:  '  ||  TO_CHAR(r.ReservationDate,
'YYYY-MM-DD HH24:MI:SS'));
  END LOOP;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END GetSoldInstrumentsInfo;

EXEC GetSoldInstrumentsInfo
drop procedure GetSoldInstrumentsInfo


CREATE OR REPLACE TYPE INSTRUMENTS_TABLE AS TABLE OF INT;

CREATE   OR   REPLACE   PROCEDURE   ReserveInstrument(p_UserID   NUMBER,
p_InstrumentIDs INSTRUMENTS_TABLE, p_ReservationDate VARCHAR2) AS
  v_TotalInstruments NUMBER;
  v_Availability NUMBER;
  v_Date DATE;
  v_OrderID NUMBER;
BEGIN
  v_Date := TO_DATE(p_ReservationDate, 'YYYY-MM-DD');

  SELECT COUNT(*) INTO v_TotalInstruments FROM Instruments;

  IF v_TotalInstruments = 0 THEN
    DBMS_OUTPUT.PUT_LINE('Error:  total  instruments  amount  is  '  ||
v_TotalInstruments);
    RETURN;
  END IF;

  IF v_Date > SYSDATE THEN
    DBMS_OUTPUT.PUT_LINE('Error:  reservation  date  cannot  be  in  the
future.');
    RETURN;
  END IF;

  INSERT  INTO  Orders  (UserID,  ReservationDate)  VALUES  (p_UserID,
v_Date)
  RETURNING OrderID INTO v_OrderID;

  FOR i IN p_InstrumentIDs.FIRST .. p_InstrumentIDs.LAST LOOP
    SELECT  Availability  INTO  v_Availability  FROM  Instruments  WHERE
InstrumentID = p_InstrumentIDs(i);
```

```
    IF v_Availability = 1 THEN
      INSERT   INTO   OrderItems   (OrderID,   InstrumentID)   VALUES
(v_OrderID, p_InstrumentIDs(i));
      DBMS_OUTPUT.PUT_LINE('Instrument     with     ID     '     ||
p_InstrumentIDs(i) || ' successfully ordered.');
    ELSE
      DBMS_OUTPUT.PUT_LINE('Instrument     with     ID     '     ||
p_InstrumentIDs(i) || ' unavailable for order.');
    END IF;
  END LOOP;

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Error: no instrument with such ID');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END ReserveInstrument;


DECLARE
  instrument_ids INSTRUMENTS_TABLE;
BEGIN
  instrument_ids := INSTRUMENTS_TABLE(1, 2);
  ReserveInstrument(1, instrument_ids, '2023-12-18');
END;

select * from Orders
select * from OrderItems

delete from OrderItems

drop procedure ReserveInstrument


CREATE OR REPLACE PROCEDURE CountAllInstruments AS
  v_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_count FROM Instruments;

  IF v_count = 0 THEN
    DBMS_OUTPUT.PUT_LINE('No instruments found.');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Amount   of   all   instruments:   '   ||
TO_CHAR(v_count));
  END IF;

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Error: No data available for counting.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END CountAllInstruments;
```

```
EXEC CountAllInstruments;
drop procedure CountAllInstruments



CREATE OR REPLACE PROCEDURE ShowTopInstruments(p_TopCount NUMBER) AS
  v_count NUMBER := 0;
BEGIN
  FOR r IN (
    SELECT  i.InstrumentID,  i.InstrumentName,  COUNT(oi.InstrumentID)
AS OrderCount
    FROM Instruments i
    JOIN OrderItems oi ON i.InstrumentID = oi.InstrumentID
    GROUP BY i.InstrumentID, i.InstrumentName
    ORDER BY OrderCount DESC
    FETCH FIRST p_TopCount ROWS ONLY
  ) LOOP
    DBMS_OUTPUT.PUT_LINE('InstrumentID: ' || r.InstrumentID ||
                         ', InstrumentName: ' || r.InstrumentName ||
                         ', OrdersCount: ' || r.OrderCount);
    v_count := v_count + 1;
  END LOOP;

  IF v_count = 0 THEN
    DBMS_OUTPUT.PUT_LINE('Error: No popular instruments found.');
  END IF;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END ShowTopInstruments;


EXEC ShowTopInstruments(1);

drop procedure ShowTopInstruments


CREATE OR REPLACE PROCEDURE ShowAllInstruments AS
BEGIN
  FOR r IN (SELECT * FROM Instruments ORDER BY InstrumentID) LOOP
    DBMS_OUTPUT.PUT_LINE('InstrumentID: ' || r.InstrumentID ||
                         ', InstrumentName: ' || r.InstrumentName ||
                         ', CategoryID: ' || r.CategoryID ||
                         ', ManufacturerID: ' || r.ManufacturerID ||
                         ', Availability: ' || r.Availability ||
                         ', Cost: ' || r.Cost);
  END LOOP;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No instruments found.');
```

```
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END ShowAllInstruments;

EXEC ShowAllInstruments
drop procedure ShowAllInstruments


CREATE OR REPLACE PROCEDURE CreateReview(
    p_UserID NUMBER,
    p_InstrumentID NUMBER,
    p_ReviewText NVARCHAR2
) AS
    v_OrderID NUMBER;
BEGIN
    SELECT o.OrderID INTO v_OrderID
    FROM Orders o
    JOIN OrderItems oi ON o.OrderID = oi.OrderID
    WHERE o.UserID = p_UserID AND oi.InstrumentID = p_InstrumentID;

    IF v_OrderID IS NULL THEN
        DBMS_OUTPUT.PUT_LINE('Error: User did not order the specified
instrument.');
        RETURN;
    END IF;

    INSERT INTO InstrumentsReview (UserID, InstrumentID, ReviewText)
    VALUES (p_UserID, p_InstrumentID, p_ReviewText);

    DBMS_OUTPUT.PUT_LINE('Review successfully added.');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Error:   Order   not   found   for   the
specified user and instrument.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END CreateReview;

EXEC CreateReview(1, 2, 'fantastic!!!');


drop procedure CreateReview



CREATE OR REPLACE PROCEDURE ShowAllReviews AS
BEGIN
  FOR r IN (SELECT * FROM InstrumentsReview) LOOP
    DBMS_OUTPUT.PUT_LINE('ReviewID: ' || r.ReviewID ||
                         ', UserID: ' || r.UserID ||
                         ', InstrumentID: ' || r.InstrumentID ||
                         ', ReviewText: ' || r.ReviewText);
  END LOOP;
```

```
  IF SQL%NOTFOUND THEN
    DBMS_OUTPUT.PUT_LINE('No reviews found.');
  END IF;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END ShowAllReviews;

EXEC ShowAllReviews;
drop procedure ShowAllReviews



CREATE OR REPLACE PROCEDURE InsertInstrument(
    p_InstrumentName IN NVARCHAR2,
    p_CategoryID IN NUMBER,
    p_ManufacturerID IN NUMBER,
    p_Availability IN NUMBER,
    p_Cost in NVARCHAR2
)
IS
  v_category_count NUMBER;
  v_manufacturer_count NUMBER;
  v_cost NUMBER;
BEGIN
  SELECT  COUNT(*)  INTO  v_category_count  FROM  Categories  WHERE
CategoryID = p_CategoryID;

  IF v_category_count = 0 THEN
    DBMS_OUTPUT.PUT_LINE('Error: Category does not exist.');
    RETURN;
  END IF;

  SELECT COUNT(*) INTO v_manufacturer_count FROM Manufacturers WHERE
ManufacturerID = p_ManufacturerID;

  IF v_manufacturer_count = 0 THEN
    DBMS_OUTPUT.PUT_LINE('Error: Manufacturer does not exist.');
    RETURN;
  END IF;

  IF p_Availability NOT IN (0, 1) THEN
    DBMS_OUTPUT.PUT_LINE('Error:  Availability  must  be  either  0  or
1.');
    RETURN;
  END IF;

  v_cost := TO_NUMBER(p_Cost);

  IF v_cost <= 0 THEN
    DBMS_OUTPUT.PUT_LINE('Error: Cost must be greater than 0.');
    RETURN;
```

```
  END IF;

  INSERT INTO Instruments (InstrumentName, CategoryID,
ManufacturerID, Availability, Cost)
  VALUES (p_InstrumentName, p_CategoryID, p_ManufacturerID,
p_Availability, v_cost);
 DBMS_OUTPUT.PUT_LINE('Instrument ' || p_InstrumentName || ' was
successfully added');

  COMMIT;
EXCEPTION
  WHEN OTHERS THEN
    ROLLBACK;
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLCODE || ' - ' || SQLERRM);
    RAISE;
END InsertInstrument;


BEGIN
  InsertInstrument(
    'ExampleInstrument',  --  InstrumentName
    7,                     -- CategoryID
    8,                     -- ManufacturerID
    1,                     -- Availability
    '1000,45'                -- Cost
  );
END;

select * from instruments where InstrumentName = 'ExampleInstrument'
delete from instruments where InstrumentName = 'ExampleInstrument'

CREATE OR REPLACE PROCEDURE DeleteInstrument(
    p_InstrumentID IN NUMBER
)
IS
  v_instrument_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_instrument_count FROM Instruments WHERE
InstrumentID = p_InstrumentID;

  IF v_instrument_count = 0 THEN
    DBMS_OUTPUT.PUT_LINE('Error:  Instrument  with  ID  '  ||
p_InstrumentID || ' does not exist.');
    RETURN;
  END IF;

  DELETE FROM Instruments WHERE InstrumentID = p_InstrumentID;
  DBMS_OUTPUT.PUT_LINE('Instrument with ID ' || p_InstrumentID || '
successfully deleted');
  COMMIT;
EXCEPTION
  WHEN OTHERS THEN
    ROLLBACK;
```

```
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLCODE || ' - ' || SQLERRM);
    RAISE;
END DeleteInstrument;


BEGIN
  DeleteInstrument(100000);
END;

CREATE OR REPLACE PROCEDURE UpdateInstrument(
    p_InstrumentID IN NUMBER,
    p_InstrumentName IN NVARCHAR2,
    p_CategoryID IN NUMBER,
    p_ManufacturerID IN NUMBER,
    p_Availability IN NUMBER,
    p_Cost IN NVARCHAR2
)
IS
  v_category_count NUMBER;
  v_manufacturer_count NUMBER;
  v_cost NUMBER;
  v_instrument_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_instrument_count FROM Instruments WHERE
InstrumentID = p_InstrumentID;

  IF v_instrument_count = 0 THEN
    DBMS_OUTPUT.PUT_LINE('Error: Instrument with ID ' ||
p_InstrumentID || ' does not exist.');
    RETURN;
  END IF;

  SELECT COUNT(*) INTO v_category_count FROM Categories WHERE
CategoryID = p_CategoryID;

  IF v_category_count = 0 THEN
    DBMS_OUTPUT.PUT_LINE('Error: Category does not exist.');
    RETURN;
  END IF;

  SELECT COUNT(*) INTO v_manufacturer_count FROM Manufacturers WHERE
ManufacturerID = p_ManufacturerID;

  IF v_manufacturer_count = 0 THEN
    DBMS_OUTPUT.PUT_LINE('Error: Manufacturer does not exist.');
    RETURN;
  END IF;

  IF p_Availability NOT IN (0, 1) THEN
    DBMS_OUTPUT.PUT_LINE('Error: Availability must be either 0 or
1.');
    RETURN;
  END IF;
```

```
    v_cost := TO_NUMBER(p_Cost);

  IF v_cost <= 0 THEN
    DBMS_OUTPUT.PUT_LINE('Error: Cost must be greater than 0.');
    RETURN;
  END IF;

  UPDATE Instruments
  SET
    InstrumentName = p_InstrumentName,
    CategoryID = p_CategoryID,
    ManufacturerID = p_ManufacturerID,
    Availability = p_Availability,
    Cost = v_cost
  WHERE InstrumentID = p_InstrumentID;
  DBMS_OUTPUT.PUT_LINE('Instrument with ID ' || p_InstrumentID || '
was successfully updated');
  COMMIT;
EXCEPTION
  WHEN OTHERS THEN
    ROLLBACK;
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLCODE || ' - ' || SQLERRM);
    RAISE;
END UpdateInstrument;

BEGIN
  UpdateInstrument(
    1,                      -- InstrumentID,
    'UpdatedInstrument',    -- new InstrumentName
    7,                      -- new CategoryID
    8,                      -- new ManufacturerID
    1,                      -- new Availability
    '1500,99'                -- new Cost
  );
END;

select * from instruments where InstrumentID = 1
```

Приложение В

Листинг создания пользователей и выдачи им необходимых привилегий

```
alter session set container = MusicShop
grant CTXAPP to MusicShop_ADMIN;

CREATE TABLESPACE MusicShop_PDB
DATAFILE 'MusicShop_PDB.dbf'
SIZE 100M
AUTOEXTEND ON NEXT 5M
BLOCKSIZE 8192
EXTENT MANAGEMENT LOCAL;

CREATE TEMPORARY TABLESPACE TS_MusicShop_TEMP
TEMPFILE 'TS_MusicShop_TEMP.dbf'
SIZE 100M
AUTOEXTEND ON NEXT 5M
BLOCKSIZE 8192
EXTENT MANAGEMENT LOCAL;

CREATE PROFILE MAIN_ADMIN_PROFILE LIMIT
PASSWORD_LIFE_TIME 120
SESSIONS_PER_USER 15
FAILED_LOGIN_ATTEMPTS 10
PASSWORD_LOCK_TIME 1
PASSWORD_REUSE_TIME 10
PASSWORD_GRACE_TIME DEFAULT
CONNECT_TIME 180
IDLE_TIME 30;

create user MusicShop_ADMIN identified by admin
default tablespace MusicShop_PDB
quota unlimited on MusicShop_PDB
temporary tablespace TS_MusicShop_TEMP
profile MAIN_ADMIN_PROFILE;

grant all privileges to MusicShop_ADMIN;

CREATE TABLESPACE MusicShop_PDB_USER
DATAFILE 'MusicShop_PDB_USER.dbf'
SIZE 100M
AUTOEXTEND ON NEXT 5M
BLOCKSIZE 8192
EXTENT MANAGEMENT LOCAL;

CREATE TEMPORARY TABLESPACE TS_MusicShop_TEMP_USER
TEMPFILE 'TS_MusicShop_TEMP_USER.dbf'
SIZE 100M
AUTOEXTEND ON NEXT 5M
BLOCKSIZE 8192
EXTENT MANAGEMENT LOCAL;

CREATE PROFILE MAIN_USER_PROFILE LIMIT
```

```
PASSWORD_LIFE_TIME 120
SESSIONS_PER_USER 15
FAILED_LOGIN_ATTEMPTS 10
PASSWORD_LOCK_TIME 1
PASSWORD_REUSE_TIME 10
PASSWORD_GRACE_TIME DEFAULT
CONNECT_TIME 180
IDLE_TIME 30;

create user MusicShop_USER identified by 1234
default tablespace MusicShop_PDB_USER
quota unlimited on MusicShop_PDB_USER
temporary tablespace TS_MusicShop_TEMP_USER
profile MAIN_USER_PROFILE;

grant create session to MusicShop_USER;
GRANT     EXECUTE     ON     MusicShop_ADMIN.ShowAllInstruments     TO
MusicShop_USER;
GRANT EXECUTE ON MusicShop_ADMIN.ReserveInstrument TO MusicShop_USER;
GRANT     EXECUTE     ON     MusicShop_ADMIN.ShowTopInstruments     TO
MusicShop_USER;
GRANT EXECUTE ON MusicShop_ADMIN.CreateReview TO MusicShop_USER;
GRANT EXECUTE ON MusicShop_ADMIN.ShowAllReviews TO MusicShop_USER;
GRANT CREATE TYPE TO MusicShop_USER;
```

Приложение Г

Листинг процедур импорта-экспорта

```
CREATE OR REPLACE DIRECTORY UTL_DIR AS 'C:\MusicShop';
GRANT READ, WRITE ON DIRECTORY UTL_DIR TO public;

CREATE OR REPLACE PROCEDURE ImportInstrumentsFromJSON
IS
BEGIN
  FOR json_rec IN (SELECT InstrumentName, CategoryID, ManufacturerID,
Availability, Cost
                   FROM                JSON_TABLE(BFILENAME('UTL_DIR',
'Instruments.json'), '$[*]'
                                    COLUMNS (
                                      InstrumentName  VARCHAR2(50)  PATH
'$.InstrumentName',
                                      CategoryID       NUMBER       PATH
'$.CategoryID',
                                      ManufacturerID     NUMBER     PATH
'$.ManufacturerID',
                                      Availability      NUMBER      PATH
'$.Availability',
                                      Cost NUMBER PATH '$.Cost'
                                    ))
                  )
  LOOP
    BEGIN
      INSERT   INTO   Instruments   (InstrumentID,   InstrumentName,
CategoryID, ManufacturerID, Availability, Cost)
      VALUES (
        DEFAULT,
        json_rec.InstrumentName,
        json_rec.CategoryID,
        json_rec.ManufacturerID,
        json_rec.Availability,
        json_rec.Cost
      );
    EXCEPTION
      WHEN DUP_VAL_ON_INDEX THEN
        DBMS_OUTPUT.PUT_LINE('Instrument   with   the   ID   already
exists.');
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error   inserting   Instrument:   '   ||
SQLERRM);
        RAISE;
    END;
  END LOOP;
  COMMIT;
  DBMS_OUTPUT.PUT_LINE('Import completed successfully.');
END ImportInstrumentsFromJSON;

BEGIN
  ImportInstrumentsFromJSON;
```

```
END;


CREATE OR REPLACE PROCEDURE ExportInstrumentsToJSON
IS
  v_file UTL_FILE.FILE_TYPE;
BEGIN
  v_file := UTL_FILE.FOPEN('UTL_DIR', 'ExportInstruments.json', 'W');

  UTL_FILE.PUT_LINE(v_file, '[');

  FOR rec IN (SELECT InstrumentID, InstrumentName, CategoryID,
ManufacturerID, Availability, Cost
              FROM Instruments
              ORDER BY InstrumentID)
  LOOP
    UTL_FILE.PUT_LINE(v_file, JSON_OBJECT(
                                  'InstrumentID'                VALUE
rec.InstrumentID,
                                  'InstrumentName'              VALUE
rec.InstrumentName,
                                  'CategoryID' VALUE rec.CategoryID,
                                  'ManufacturerID'              VALUE
rec.ManufacturerID,
                                  'Availability'                VALUE
rec.Availability,
                                  'Cost' VALUE rec.Cost
                                ) || ',');

  END LOOP;
  UTL_FILE.PUT_LINE(v_file, ']');
  UTL_FILE.FCLOSE(v_file);

  DBMS_OUTPUT.PUT_LINE('Export completed successfully.');
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error exporting data: ' || SQLERRM);
    IF UTL_FILE.IS_OPEN(v_file) THEN
      UTL_FILE.FCLOSE(v_file);
    END IF;
END ExportInstrumentsToJSON;

BEGIN
  ExportInstrumentsToJSON;
END;
```