МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ   
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-98 01 03 «Программное обеспечение информационной безопасности мобильных систем»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА**

по дисциплине «Базы данных»

Тема: база данных «Реализация базы данных для авиакомпании»

**Исполнитель**

студент(ка) 2 курса 8 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кривенчук М.И.

подпись, дата

**Руководитель**

ассистент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кантарович В.С.

должность, ученая степень, ученое звание подпись, дата

Допущен(а) к защите \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата, подпись

Курсовой проект защищен с оценкой

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кантарович В.С.

подпись дата инициалы и фам

**Содержание**

[Введение 5](#_Toc135306319)

[1. Постановка задачи 6](#_Toc135306320)

[2. Проектирование базы данных 7](#_Toc135306321)

[2.1 Схема базы данных 7](#_Toc135306322)

[2.2 Таблицы базы данных 8](#_Toc135306323)

[3. Разработка объектов базы данных 11](#_Toc135306325)

[3.1 Табличное пространство 11](#_Toc135306326)

[3.2 Профили безопасности, роли, пользователи 11](#_Toc135306327)

[3.3 Таблицы 11](#_Toc135306328)

[3.4 Триггеры 12](#_Toc135306328)

[3.5 Хранимые процедуры 12](#_Toc135306330)

[3.5.1 Процедуры клиента 12](#_Toc135306331)

[3.5.2 Процедуры менеджера 13](#_Toc135306332)

[3.5.3 Процедура вставки 100000 строк 13](#_Toc135306332)

[3.6 Функции 15](#_Toc135306329)

[4. Описание процедур экспорта и импорта 17](#_Toc135306334)

[4.1 Описание процедур экспорта 17](#_Toc135306335)

[4.2 Описание процедур импорта 18](#_Toc135306336)

[5. Тестирование производительности 20](#_Toc135306338)

[5.1 Оптимизирование запросов 20](#_Toc135306339)

[5.2 Проверка выполнения хранимых процедур 22](#_Toc135306340)

[6. Описание и применение технологии Full Text Search в базе данных 24](#_Toc135306342)

[6.1 Описание взаимодействия с технологией 26](#_Toc135306343)

[7. Руководство пользователя 27](#_Toc135306346)

[7.1 Сторона менеджера 27](#_Toc135306343)

[7.2 Сторона клиента 27](#_Toc135306343)

[Заключение 28](#_Toc135306347)

[Список используемых источников 29](#_Toc135306348)

# **Введение**

Современные информационные системы во многих областях нашей жизни не могут обойтись без баз данных. Они служат надежным инструментом для хранения, структурирования и обработки данных различного объема и сложности. Oracle Database, одна из наиболее мощных и распространенных систем управления базами данных, стала промышленным стандартом для обработки данных в крупных корпорациях и предприятиях. Эта система отличается высокой производительностью, масштабируемостью, надежностью и предлагает широкий спектр функций для обработки и анализа данных.

В авиатранспортной отрасли постоянно происходит обмен большим объемом данных. Важнейшим элементом управления информацией в авиакомпании является база данных. Она содержит данные о рейсах, пассажирах, билетах, сотрудниках и других ключевых аспектах авиаперевозок. Правильное проектирование и внедрение базы данных упрощает взаимодействие с системой как для руководства, так и для пользователей. Однако даже незначительные ошибки или недочеты могут привести к серьезным сбоям, нарушениям или замедлению работы системы.

Исходя из этого, очевидно, что создание базы данных для авиакомпании - это сложный и трудоемкий процесс, требующий тщательного анализа и глубоких знаний в этой области.

Целью данного проекта является создание реляционной базы данных для авиакомпании. Эта база данных предназначена для хранения информации о рейсах, клиентах и билетах, а также для обеспечения быстрого доступа к хранимой информации.

# **Постановка задачи**

В рамках курсового проекта необходимо спроектировать инфраструктуру базы данных авиакомпании. Для этого необходимо проанализировать сферу, и определить, какими должны быть следующие объекты и их содержание в нашей базе данных – табличные пространства и их файлы, таблицы и связи между ними, а также ограничения целостности для них, профили безопасности, пользователи, триггеры, хранимые процедуры, функции, индексы.

Далее необходимо разработать эти объекты в базе данных СУБД Oracle 19c посредством написания скриптов на языке SQL. Необходимо заполнить таблицы данными посредством импорта из JSON файлов. Применить технологию Full Text Search для эффективного поиска и анализа текстовых данных в базах данных или других хранилищах информации.

Обязательно должна присутствовать реализация следующих функций: управление авиакомпанией (управление маршрутами, рейсами, самолётами, билетами; проверка наличия свободных мест на рейс, добавление новых узлов (пунктов назначения)), операции над билетами (бронирование, возврат) и т.д.). Данные операции должны быть реализованы с помощью хранимых процедур.

Необходимо протестировать производительность базы данных на таблице, содержащей не менее 100 000 строк, и внести изменения в структуру в случае необходимости.

# **Проектирование базы данных**

* 1. Схема базы данных

Для начала определим, какие нам нужны таблицы. Вообще, в базе данных авиакомпании могут насчитываться десятки таблиц, если это крупная компания и она оперирует большим количеством данных, предоставляет широкий спектр услуг (пассажирская перевозка, грузовая перевозка, перевозка багажа, предоставление бронирования через компаний-партнёров и т.д). Однако, мы ограничимся минимальным необходимым числом таблиц и включим туда какой-нибудь сервис (в данном случае, будет перевозка багажа). Предварительная диаграмма таблиц и их отношения представлены на рисунке 2.1.

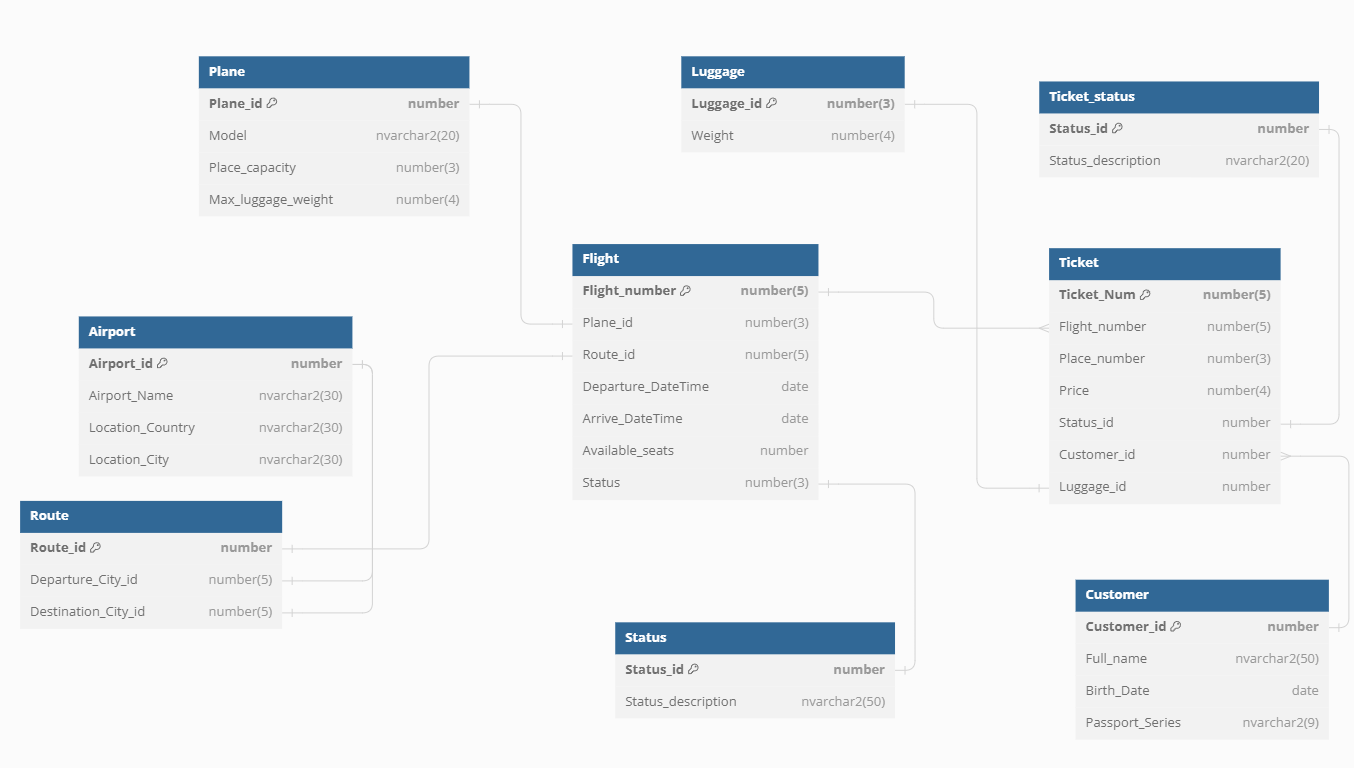


Рисунок 2.1 – UML-диаграмма базы данных

Также в этих таблицах указаны предварительные типы данных для столбцов, которые могут измениться при необходимости.

* 1. Таблицы базы данных

Теперь опишем таблицы базы данных: названия, названия столбцов, типы данных столбцов, описание содержания столбцов. Описание представлено в таблицах 2.1 – 2.9.

Таблица 2.1 – Содержание таблицы Airport (Аэропорт)

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| airport\_id | NUMBER | Уникальный идентификатор аэропорта |
| airport\_name | VARCHAR2(60) | Название аэропорта |
| location\_country | VARCHAR2(40) | Название страны расположения |
| location\_city | VARCHAR2(40) | Название города расположения |

Эта таблица хранит в себе данные об аэропортах.

Таблица 2.2 – Содержание таблицы Plane (Самолёт)

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| plane\_id | NUMBER | Уникальный идентификатор самолёта |
| plane\_model | NUMBER | Модель самолёта |
| places\_capacity | NUMBER | Количество посадочных мест |
| max\_luggage\_weigth | NUMBER | Максимальный вес багажа для перевозки |

В этой таблице содержатся данные о самолётах, используемых компанией для авиаперевозок.

Таблица 2.3 – Содержание таблицы Route (Маршрут)

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| route\_id | NUMBER | Уникальный идентификатор маршрута |
| departure\_airport\_id | NUMBER | Уникальный идентификатор аэропорта отправления |
| destination\_airport\_id | NUMBER | Уникальный идентификатор аэропорта прибытия |

В этой таблице содержатся данные о маршрутах. Таблица содержит ограничение внешнего ключа для столбцов departure\_airport\_id и destination\_airport\_id.

Таблица 2.4 – Содержание таблицы Status (Статус полёта)

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| status\_id | NUMBER | Уникальный идентификатор статуса полёта |
| status\_description | VARCHAR2(30) | Описание статуса полёта |

Эта таблица хранит в себе варианты статусов полёта, которые могут быть установлены для рейса.

Следующая таблица 2.5 содержит описание таблицы базы данных Flight, в которой будут храниться данные о рейсах. Таблица содержит ограничение внешнего ключа для столбцов plane\_id, route\_id и status.

Таблица 2.5 – Содержание таблицы Flight (Рейс)

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| flight\_number | NUMBER | Уникальный идентификатор рейса |
| plane\_id | NUMBER | Уникальный идентификатор самолёта |
| route\_id | NUMBER | Уникальный идентификатор маршрута |
| departure\_datetime | DATE | Время отправления |
| arrive\_datetime | DATE | Время прибытия |
| available\_seats | NUMBER(3) | Текущее количество свободных мест |
| status | NUMBER | Уникальный идентификатор статуса |

Таблица 2.6 – Содержание таблицы Luggage (Багаж)

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| luggage\_id | NUMBER | Уникальный идентификатор багажа |
| weigth | NUMBER(3) | Вес багажа |

Эта таблица хранит в себе строки с данными о весе багажа и соответствующего ему идентификатора. Данные из этой таблицы найдут применение в таблице Ticket.

Таблица 2.7 – Содержание таблицы Ticket (Билет)

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ticket\_num | NUMBER | Уникальный идентификатор билета |
| flight\_number | NUMBER | Уникальный идентификатор рейса |
| place\_number | NUMBER (3) | Номер места |
| price | NUMBER (7,2) | Стоимость билета |
| status\_num | NUMBER | Уникальный идентификатор статуса |
| customer\_id | NUMBER | Уникальный идентификатор клиента |
| luggage\_id | NUMBER | Уникальный идентификатор багажа |

Эта таблица хранит в себе данные о билетах, заказанных клиентами. Таблица содержит ограничение внешнего ключа для столбцов flight\_number, customer\_id, luggage\_id и status\_num.

Таблица 2.8 – Содержание таблицы Ticket\_status (Статус билета)

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| Status\_num | NUMBER | Уникальный идентификатор статуса |
| status\_description | VARCHAR2(30) | Описание статуса |

Предпоследняя таблица содержит идентификатор и соответствующий ему статус, который может применяться к статусу билета.

Таблица 2.9 – Содержание таблицы Customer (Клиенты)

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| customer\_id | NUMBER | Уникальный идентификатор клиента |
| full\_name | VARCHAR2(100) | ФИО клиента |
| date\_of\_birth | DATE | Дата рождения клиента |
| passport\_series | VARCHAR2(10) | Серия паспорта клиента |

Последняя таблица содержит данные о клиентах.

Представленная структура базы данных обеспечивает эффективное и надежное хранение информации, необходимой для функционирования системы управления авиакомпанией. Предусмотренные ограничения целостности и связи между таблицами гарантируют целостность данных и согласованность информации в системе.

# **Разработка объектов базы данных**

* 1. **Табличное пространство**

После создания подключаемой базы данных для нашего проекта и до создания таблиц и заполнения её данными нам необходимо создать табличное пространство – листинг 3.1

create tablespace TS\_AIR

datafile 'D:\labs\oracle\ts\_air.dbf'

size 20 m

autoextend on next 10 m

maxsize 200M

extent management local;

commit;

Листинг 3.1 – Создание табличного пространства

После создания пользователей необходимо сделать это табличное пространство для них табличным пространством по умолчанию и выделить квоту.

* 1. **Профили безопасности, роли, пользователи**

В нашей базе данных будет 2 пользователя – клиент и менеджер. Отличаться у них будут профили безопасности и параметры: число сессий для пользователя, «время жизни» пароля, количество неверных попыток ввода пароля, время сессии и простоя. У них будут также свои роли, отличающиеся привилегиями – пользователь сможет только выполнять хранимые процедуры, определенные для него и создавать подключение, менеджер же – просматривать все таблицы, создавать подключение, выполнять хранимые процедуры на добавление/удаление данных из таблиц. На данном этапе выдадим этим пользователям лишь привилегию на создание подключения.

* 1. **Таблицы**

Перейдём к написанию скриптов для создания таблиц. Из общего можно отметить, что в каждой таблице есть поле, представляющее ID, которое является первичным ключом и заполняется с помощью Identity. Все числовые типы данных представлены типом number, в таблице Flight присутствует тип date для реализации времени отправления и прибытия, а также даты рождения клиентов. Символьные данные представлены типом varchar2. Почти на всех столбцах стоит ограничение целостности «not null» кроме тех, где это предусмотрено. На столбце с серий паспорта таблицы Customer стоит ограничение целостности «unique».

После создания таблиц они были заполнены небольшим количеством тестовых данных для проверки работоспособности и разработки объектов следующих пунктов. База данных данного курсового проекта содержит 9 таблиц, которые описаны в главе 2.

* 1. **Триггеры**

Для данной реализации у нас будет 3 триггера: update\_available\_seats, increase\_available\_seats и lug\_weight\_check. Первый – BEFORE INSERT на таблицу Ticket для обновления ячейки available\_seats для конкретных рейсов таблицы Flight, второй – AFTER UPDATE на случай отмены билета, который также будет регулировать количество свободных мест на рейс (при отмене билета повышает кол-во свободных мест). И последний BEFORE INSERT на таблицу Ticket, он будет проверять, не превышает ли текущий вес багажа максимальное возможное значение.

* 1. Хранимые процедуры

Процедуры позволяют ограничивать доступ к данным, предоставляя пользователям возможность выполнять только определенные операции. Также Процедуры обычно выполняются быстрее, чем отдельные SQL-запросы, поскольку они компилируются один раз и сохраняются в памяти базы данных.

* + 1. Процедуры пользователя

Для пользователя будет реализовано 5 процедур:

* SEARCH\_FLIGHTS. Процедура поиска рейса;
* BOOKING\_TICKET. Процедура бронирования билета;
* CANCEL\_TICKET. Процедура возврата билета;
* GET\_TICKETS\_BY\_CUSTOMER. Процедура просмотра всех билетов пользователя;
* GET\_AVAILABLE\_SEATS\_COUNT. Процедура просмотра числа свободных мест на рейс;
  + 1. Процедуры менеджера

Для менеджера определим следующие процедуры:

* ADD\_FLIGHT. Процедура добавления рейса;
* CANCEL\_FLIGHT. Процедура отмены рейса;
* UPDATE\_FLIGHT. Процедура редактирования рейса;
* DELETE\_FLIGHT. Процедура удаления рейса;
* ADD\_ROUTE. Процедура добавления маршрута;
* UPDATE\_ROUTE. Процедура редактирования маршрута;
* DELETE\_ROUTE. Процедура удаления маршрута;
* ADD\_PLANE. Процедура добавления самолета;
* UPDATE\_PLANE. Процедура редактирования самолета;
* DELETE\_PLANE. Процедура удаления самолета;
* ADD\_TICKET. Процедура добавления билета;
* BOOKED\_TICKET. Процедура бронирования билета клиенту;
* UPDATE\_TICKET. Процедура редактирования билета;
* DELETE\_TICKET. Процедура удаления билета;
* ADD\_AIRPORT. Процедура добавления аэропорта;
* ADD\_AIRPORT. Процедура добавления аэропорта;
* UPDATE\_AIRPORT. Процедура редактирования аэропорта;
* DELETE\_AIRPORT. Процедура удаления аэропорта;
* MONITOR\_USERS. Процедура мониторинга пользователей;
* FLIGHT\_HISTORY. Процедура просмотра истории полетов;
* CHECK\_FLIGHTS. Процедура проверки статуса рейса;
* MONITOR\_PLANES. Процедура мониторинга авиапарка;
* SHOW\_BOOKED\_AND\_CANCELLED\_TICKETS. Процедура просмотра статистики купленных билетов;
* SHOW\_AIRPORTS. Процедура просмотра аэропортов;
* SHOW\_ROUTES. Процедура просмотра маршрутов;
  + 1. Процедура вставки 100000 строк

Для тестирования производительности нам нужно заполнить какую-нибудь таблицу 100000 строками. Пусть это будет таблица Flight. Создадим следующую хранимую процедуру: мы будем проходить по каждому аэропорту и создавать для него случайный рейс, дату и время будем делать случайную, значение  вычисляется путем добавления случайного числа дней (в диапазоне от 0 до 365) к дате '01.01.2024 00:00:00' и случайного числа минут (в диапазоне от 180 до 960), самолёт также выбираем случайно.Код представлен в листинге 3.5.

CREATE OR REPLACE PROCEDURE populate\_flight\_table AS

v\_plane\_id NUMBER;

v\_route\_id NUMBER;

v\_dep\_datetime DATE;

v\_arr\_datetime DATE;

v\_available\_seats NUMBER;

v\_status\_id NUMBER := 1;

BEGIN

FOR i IN 1..100000 LOOP

SELECT plane\_id, places\_capacity INTO v\_plane\_id, v\_available\_seats

FROM (

SELECT plane\_id, places\_capacity,

ROW\_NUMBER() OVER (ORDER BY DBMS\_RANDOM.VALUE) AS rn

FROM Plane

)

WHERE rn = 1;

SELECT route\_id INTO v\_route\_id

FROM (

SELECT route\_id

FROM Route

ORDER BY DBMS\_RANDOM.VALUE

)

WHERE ROWNUM = 1;

v\_dep\_datetime := TO\_DATE('01.01.2024 00:00:00', 'DD.MM.YYYY HH24:MI:SS') +

INTERVAL '1' DAY \* DBMS\_RANDOM.VALUE(0, 365);

v\_arr\_datetime := v\_dep\_datetime + INTERVAL '1' MINUTE \* DBMS\_RANDOM.VALUE(180, 960);

INSERT INTO Flight (plane\_id, route\_id, departure\_datetime, arrive\_datetime, available\_seats, status)

VALUES (v\_plane\_id, v\_route\_id,

TO\_DATE(TO\_CHAR(v\_dep\_datetime, 'DD.MM.YYYY HH24:MI:SS'), 'DD.MM.YYYY HH24:MI:SS'),

TO\_DATE(TO\_CHAR(v\_arr\_datetime, 'DD.MM.YYYY HH24:MI:SS'), 'DD.MM.YYYY HH24:MI:SS'),

v\_available\_seats, v\_status\_id);

END LOOP;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Table Flight populated successfully.');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

ROLLBACK;

END;

Листинг 3.5 – Процедура для заполнения таблицы Flight

Таким образом, процедура populate\_flight\_table генерирует случайные значения для идентификаторов самолета и маршрута, а также случайные даты и время отправления и прибытия, и вставляет эти значения в таблицу Flight.

* 1. Функции

Для вывода самого популярного рейса за день, месяц и год была разработана функция Popular\_Flight которая возвращает рейс и число купленных билетов. Рассмотрим детально процедуру поиска авиабилетов – Листинг 3.6.

CREATE OR REPLACE FUNCTION Popular\_Flight(p\_date IN DATE, p\_period IN VARCHAR2)

RETURN SYS\_REFCURSOR AS

result\_cursor SYS\_REFCURSOR;

l\_period VARCHAR2(10);

BEGIN

l\_period := UPPER(p\_period);

IF l\_period NOT IN ('DAY', 'MONTH', 'YEAR') THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Invalid period. Must be DAY, MONTH, or YEAR.');

END IF;

OPEN result\_cursor FOR

SELECT

f.flight\_number,

a1.location\_city AS departure\_city,

a2.location\_city AS arrival\_city,

COUNT(\*) AS total\_tickets

FROM Ticket t

JOIN Flight f ON t.flight\_number = f.flight\_number

JOIN Route r ON f.route\_id = r.route\_id

JOIN Airport a1 ON r.departure\_city\_id = a1.airport\_id

JOIN Airport a2 ON r.destination\_city\_id = a2.airport\_id

WHERE

CASE

WHEN l\_period = 'DAY' THEN TO\_CHAR(f.departure\_datetime, 'DD.MM.YYYY')

WHEN l\_period = 'MONTH' THEN TO\_CHAR(f.departure\_datetime, 'MM.YYYY')

WHEN l\_period = 'YEAR' THEN TO\_CHAR(f.departure\_datetime, 'YYYY')

END =

CASE

WHEN l\_period = 'DAY' THEN TO\_CHAR(p\_date, 'DD.MM.YYYY')

WHEN l\_period = 'MONTH' THEN TO\_CHAR(p\_date, 'MM.YYYY')

WHEN l\_period = 'YEAR' THEN TO\_CHAR(p\_date, 'YYYY')

END

GROUP BY f.flight\_number, a1.location\_city, a2.location\_city

ORDER BY total\_tickets DESC

FETCH FIRST ROW ONLY;

RETURN result\_cursor;

END;

Листинг 3.6 – Функция для поиска самого популярного рейса

Наша функция возвращает курсор. Этот курсор содержит информацию о самом популярном рейсе за указанный период времени (в данном случае за месяц). Курсор возвращает следующие значения: номер рейса, город вылета, город прилета и общее количество проданных билетов на этот рейс. Пример вызова функции представлен на рисунке 3.1.



Рисунок 3.1 – Результат выполнения функции

# **Описание процедур экспорта и импорта**

* 1. Описание процедур экспорта

База данных авиакомпании может включать в себя большое количество информации. Здесь и возникает вопрос о том, каким образом можно эффективно экспортировать и импортировать данные. Для хранения данных был выбран JSON-формат. Теперь предстоит разработать соответствующие хранимые процедуры для импорта/экспорта данных. Разберём структуру процедур на примере таблицы Airport – листинг 4.1. Предварительно создаём объект Directory, где прописываем путь к папке для хранения данных.

Для каждого аэропорта в таблице Airport: создается JSON-объект, содержащий данные об аэропорте, такие как идентификатор, название, страна и город. Далее JSON-объект преобразуется в строку и строка с JSON-объектом записывается в файл "export.json".

CREATE OR REPLACE PROCEDURE EXPORT\_AIRPORT\_JSON(return\_out OUT INT)

IS

output\_file UTL\_FILE.FILE\_TYPE;

json\_data CLOB;

CURSOR airports\_cursor IS

SELECT JSON\_OBJECT(

'airport\_id' VALUE airport\_id,

'airport\_name' VALUE airport\_name,

'location\_country' VALUE location\_country,

'location\_city' VALUE location\_city

) AS json\_data

FROM Airport;

BEGIN

output\_file := UTL\_FILE.FOPEN('DIR', 'export.json', 'W');

FOR airport\_rec IN airports\_cursor LOOP

json\_data := airport\_rec.json\_data;

UTL\_FILE.PUT\_LINE(output\_file, json\_data);

END LOOP;

UTL\_FILE.FCLOSE(output\_file);

return\_out := 1;

END;

Листинг 4.1 – Процедура для экспорта данных из таблицы Airport

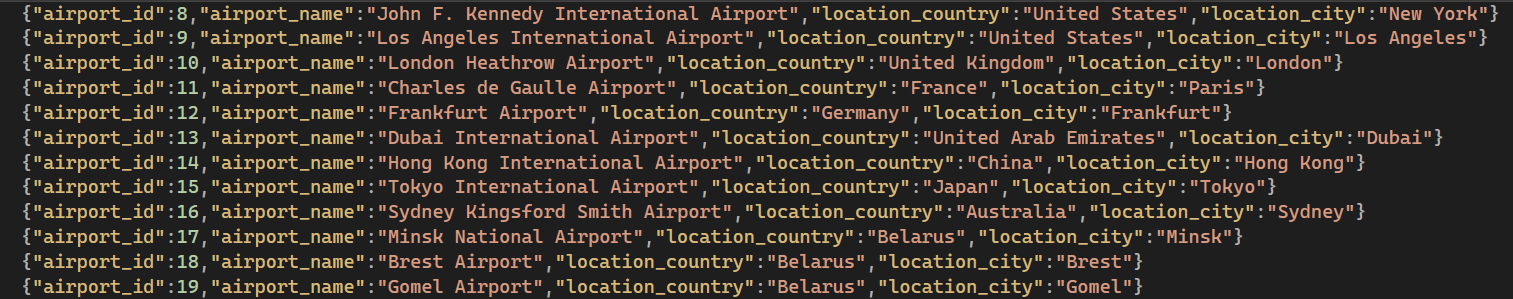


Рисунок 4.1 – Структура экспортированных данных

* 1. Описание процедур импорта

В процедуре IMPORT\_AIRPORT\_JSON импортирует данные об аэропортах из файла "import.json" в таблицу Airport. Она использует функцию JSON\_TABLE, чтобы разбить JSON-структуру на колонки и строки, которые можно вставить в таблицу.

Данные из файла "import.json" извлекаются и вставляются в таблицу Airport в столбцы airport\_name, location\_country и location\_city.

По завершении импорта значение 1 присваивается переменной return\_out, чтобы указать успешное выполнение процедуры.

CREATE OR REPLACE PROCEDURE IMPORT\_AIRPORT\_JSON(RETURN\_OUT OUT NUMBER)

AS

BEGIN

INSERT INTO Airport (airport\_name, location\_country, location\_city)

SELECT jt.airport\_name, jt.location\_country, jt.location\_city

FROM JSON\_TABLE(

BFILENAME('DIR', 'import.json'), '$.airports[\*]'

COLUMNS (

airport\_name VARCHAR2(100) PATH '$.airport\_name',

location\_country VARCHAR2(100) PATH '$.location\_country',

location\_city VARCHAR2(100) PATH '$.location\_city'

)

) jt;

RETURN\_OUT := 1;

END;

Листинг 4.2 – Процедура для импорта данных в таблицу Airport



Рисунок 4.2 – Структура импортированных данных

# **Тестирование производительности**

* 1. Оптимизирование запросов

У нас уже созданы необходимые объекты базы данных, а также таблицы заполнены необходимым количеством данных – рисунок 5.1. Теперь необходимо протестировать, что всё работает, как нужно и как задумано.

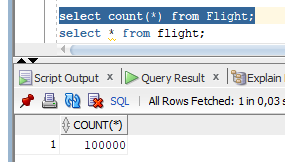


Рисунок 5.1 – Количество строк в таблицe Flight

С помощью select-запроса найдём все отмененные рейсы конкретного маршрута – рисунок 5.2. Теперь просмотрим план выполнения данного запроса – рисунок 5.3.

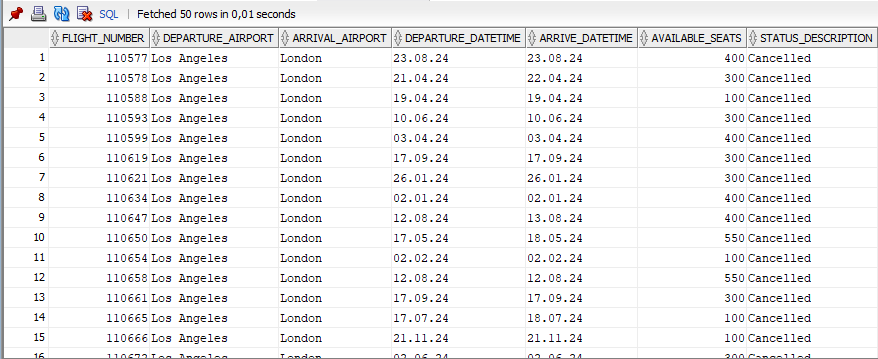


Рисунок 5.2 – Результат SELECT-запроса

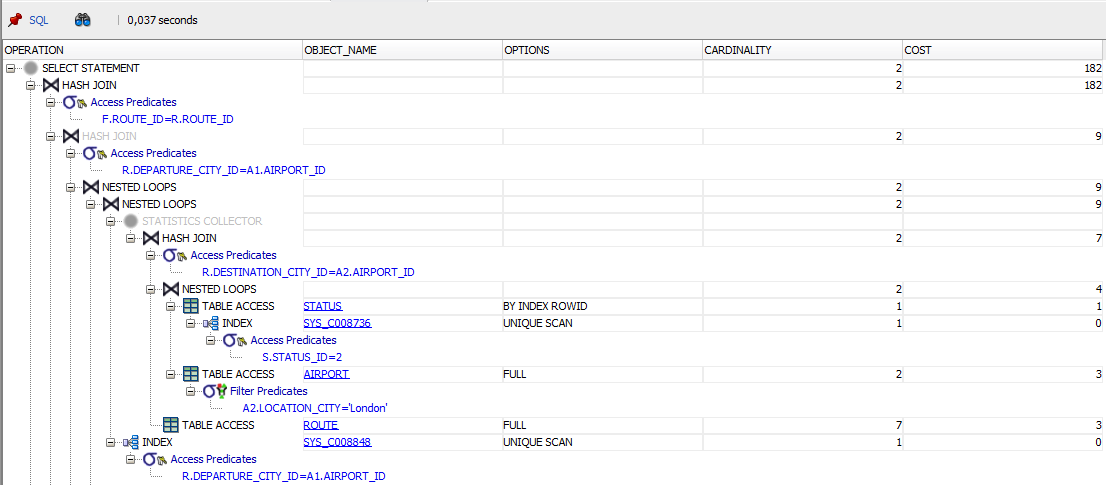


Рисунок 5.3 – План запроса

Теперь попробуем оптимизировать данный запрос с помощью индекса – Листинг 5.1.

create index idx\_flight\_status on Flight(status);

Листинг 5.1 – Создание индекса

Создание такого индекса обуславливаем тем, что при выборке у нас есть условие по этому столбцу.

Теперь проверим выполнение поиска рейсов – рисунок 5.4.

Как мы видим, ситуация изменилась: время запроса сократилось и стоимость выполнения уменьшилась. Следовательно делаем вывод, что создание данного индекса в текущих обстоятельствах является необходимым.

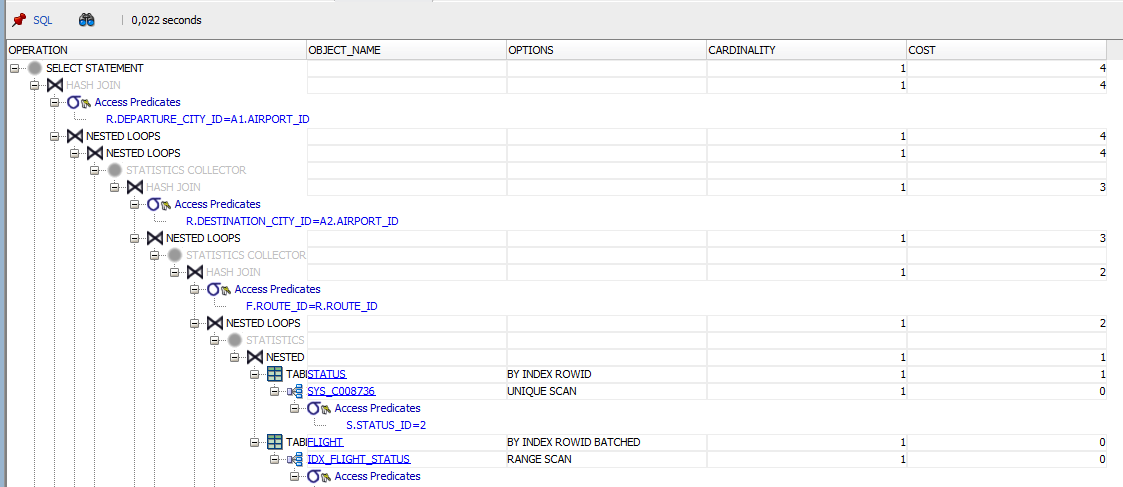


Рисунок 5.4 – План после создания индекса

* 1. Проверка выполнения хранимых процедур

Теперь протестируем основные хранимые процедуры заказа билета и багажа, отмены билета, отмены рейса.

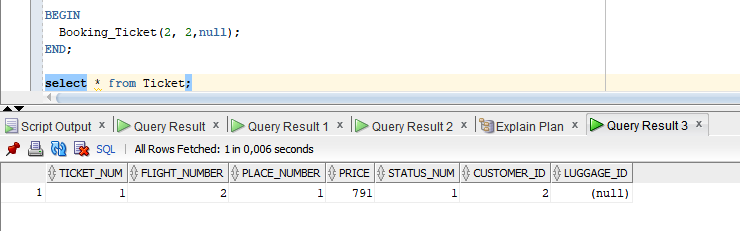
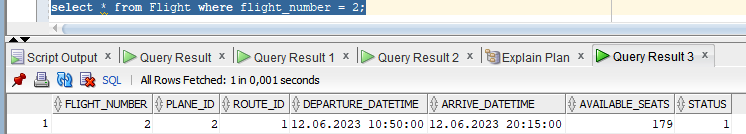
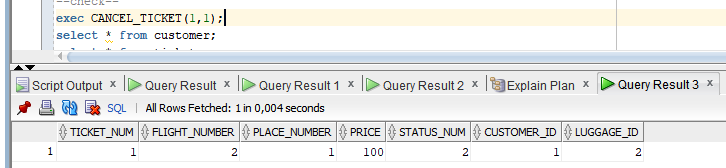


Рисунок 5.5 – Бронирование билета

Отмена билета с помощью процедуры CANCEL\_TICKET, количество свободных мест до и после отмены билета – результат на рисунке 5.6.





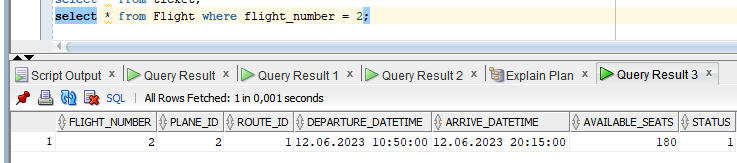


Рисунок 5.6 – Отмена билета

Демонстрация билетной истории – на рисунке 5.8.

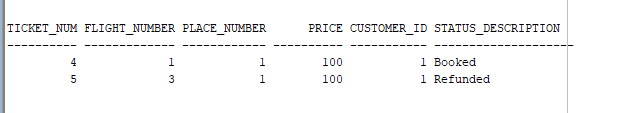


Рисунок 5.7 – Билетная история пользователя

Все операции вставки строк в таблицы за менеджера выполняются без проблем, процедуры выполняются быстро.

# **Описание и применение технологии Full Text Search в базе данных**

* 1. Описание взаимодействия с технологией

Oracle Text — это компонента, входящая в состав Oracle Database, предназначенная для создания приложений с использованием полнотекстового поиска. В предыдущих версиях Oracle имела также названия ConText и interMedia Text. Начиная с 11-ой версии входит во все редакции СУБД.

Для использования полнотекстового поиска нужно выдать привилегии CTX\_DDL для использования технологии (Рис. 6.1):



Рис. 6.1 – предоставление привилегий

Далее нам нужно определить предварительные настройки. Для этого мы создаем Lexer – «коэффициентом соответствия». Есть несколько видов Lexer, но мы будем использовать AUTO\_LEXER – Идентифицирует индексируемый язык путем изучения содержимого и применяет подходящие параметры (включая выделение корней) для этого языка. Лучше всего работает, когда каждый документ содержит один язык и имеет по крайней мере пару абзацев текста для облегчения идентификации. И установим параметр BASIC\_WORDLIST – Используйте параметр списка слов, чтобы включить такие параметры запроса, как выделение корней и нечеткое сопоставление для вашего языка. Вы также можете использовать параметр «Список слов», чтобы включить индексирование подстрок и префиксов, что повышает производительность для запросов с подстановочными знаками. Лучше всего работает, когда каждый документ содержит один язык и имеет по крайней мере пару абзацев текста для облегчения идентификации. Далее устанавливаем атрибуты. Настройка представлена на Рис. 6.2

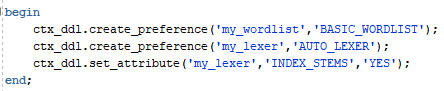


Рис.6.2 – предварительная настройка для индекса

Далее нам нужно создать индекс для осуществления поиска они бывают 4 типов:

* CONTEXT – основной тип индекса. Применяется для индексирования содержимого одной или нескольких колонок таблицы;
* CTXCAT – применяется для индексации небольших текстовых фрагментов, не требует периодической переиндексации;
* CTXRULE – применяется при создании приложений для классификации документов;
* CTXXPATH – предназначен для ускорения работы с полями типа XMLType.

Создаем индекс CTXCAT для таблицы airport и укажем имя LEXER и WORDLIST Рис. 6.3 

Рис. 6.3 – создание индекса для таблицы airport и столбца airport\_name

Далее с помощью конструкции constrains мы выполним поиск отрывка слова в столбце airport\_name таблицы airport.

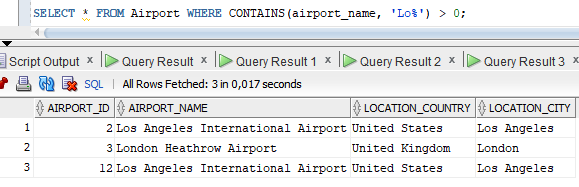


Рис. 6.4 – результат запроса

Оператор and. Он может используется чтобы найти в строке оба слова. Пример выполнения оператора and на рис. 6.5

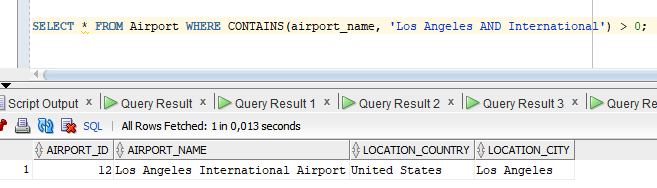


Рис. 6.5 – результат выполнения запроса с оператором and

Так же можно использовать логические операторы, например выбор из нескольких значений (| или or или ,).

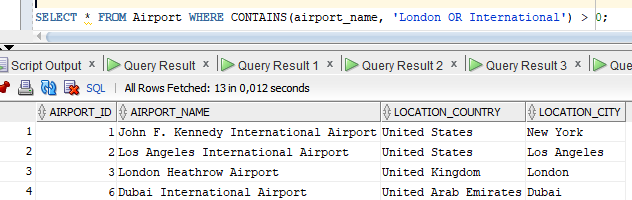


Рис. 6.6 – результат выполнения запроса с оператором or

Оператор not (~) Используется для поиска документов, содержащих одно условие запроса, а не другое. В примере на Рис. 6.7 мы выбираем что Los Angeles не International.

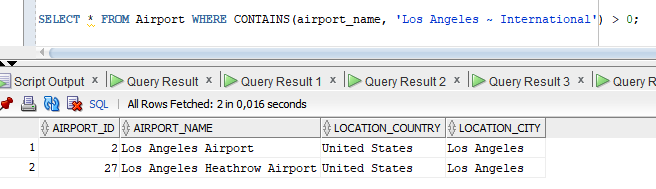


Рис. 6.7. – результат выполнения запроса с использованием оператора not

Оператор near предназначен для поиска на определенном расстоянии друг от друга слов. На рисунке поиск слов Charles и Airport которые находятся на расстоянии 2 слов друг от друга (Рис. 6.8).

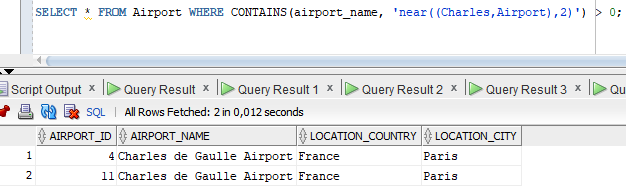


Рис. 6.8 – результат выполнения запроса используя оператор near

Полнотекстовый поиск предоставляет гибкие и точные возможности поиска по текстовым данным, учитывая контекст и особенности языка. Это делает его ценным инструментом для поиска и анализа текстовой информации.

1. **Руководство пользователя**

Так как база данных, поддерживает 2 типа пользователей, то есть и 2 сценария использования. Первый – при входе от менеджера, второй – от клиента.

* 1. Сторона менеджера

Для начала работы в роли менеджера необходимо авторизоваться под учетной записью manager. После успешной авторизации станут доступны функциональности администрирования базы данных авиакомпании.

Менеджер обладает возможностью управления данными таблиц Plane, Airport, Route, Flight и Ticket. Он может добавлять туда данные или изменять их. В случае, если данные устаревают или становятся неактуальными, менеджер может удалить их.

Важной задачей менеджера является мониторинг пользователей, истории полетов, авиапарка. Также менеджер может просматривать статистику купленных билетов. Все процедуры которые может проводить менеджер описаны в главе 3.

* 1. Сторона клиента

Клиент, начиная работу в системе, должен сначала осуществить вход под своей учетной записью. После успешной авторизации становятся доступными функции, предназначенные для конечного пользователя.

Клиенту доступны операции бронирования и возврата билета на рейс.

Одним из важных функционалов для клиента является возможность просмотра своих собственных бронирований, поиска рейсов и просмотра числа свободных мест рейса. Все процедуры которые может проводить клиент описаны в главе 3.

# **Заключение**

В рамках курсового проекта была разработана и внедрена структура базы данных для авиакомпании, используя систему управления базами данных Oracle 19c. Были определены и созданы ключевые объекты базы данных, включая табличные пространства, таблицы, ограничения целостности, профили безопасности, пользователей, триггеры и хранимые процедуры. Кроме того, был осуществлен импорт данных из JSON-файлов для заполнения таблиц.

Основным элементом проекта стало использование технологии Full Text Search для эффективного поиска и анализа текстовых данных в базах данных или других репозиториях информации. Были изучены и применены полезные функции, предлагаемые этой технологией.

Особое внимание было уделено производительности базы данных. Было проведено тестирование на таблице, содержащей более 100 000 строк, и проведен анализ необходимых изменений в структуре базы данных для достижения оптимальной производительности.

В итоге успешного выполнения курсового проекта была создана база данных авиакомпании с широким функционалом, обеспечивающим эффективное управление авиаперевозками, операциями с билетами и рейсами. Проект демонстрирует преимущества использования СУБД Oracle 19c и технологии Full Text Search для создания мощной и гибкой системы управления авиакомпанией.

# **Список используемых источников**

1. Блинова Е.А. Курс лекций по базам данных, [Электронный ресурс]. – Режим доступа: https://diskstation.belstu.by:5001/. – Дата доступа: 10.04.2023.
2. Работа с файлами в Oracle, [Электронный ресурс]. – Режим доступа: https://docs.oracle.com/cd/F49540\_01/DOC/server.815/a68001/utl\_file.htm. – Дата доступа: 01.05.2024.

3 Пользовательское руководство по Oracle Text. Создание подключения, [Электронный ресурс]. – Режим доступа: https://docs.oracle.com/cd/B13789\_01/text.101/b10730.pdf – Дата доступа: 02.05.2024.

4 Набор разработчика Oracle для Visual Studio, [Электронный ресурс]. – Режим доступа:https://marketplace.visualstudio.com/items?itemName=OracleCorporation.OracleDeveloperToolsForVisualStudio2022&ssr=false#overview – Дата доступа: 25.04.2024.