Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Дисциплина «Методы сбора, хранения, обработки и анализа данных»

Студент: Самсоник А.И.

ФИТ 3 курс 1 группа

Преподаватель: Блинова

Содержание

[**Лабораторная работа № 1 3**](#_Toc144645880)

[**Лабораторная работа № 2 4**](#_Toc144645881)

# **Лабораторная работа № 1**

Задание, состоящий из: UML диаграммы; логической схемы БД; физических схем БД для СУБД SQL Server и Oracle.

|  |  |
| --- | --- |
| № | База данных |
| 13 | Туристическая фирма |

База данных в туристической компании может использоваться для решения различных бизнес-задач. Вот список типичных задач, для которых она может быть полезной:

1. Управление клиентской информацией:

- Хранение данных о клиентах, включая контактную информацию, историю бронирований и предпочтения.

2. Бронирование и заказ туров:

- Регистрация и управление заказами туров и бронированиями гостиниц, включая даты, места и стоимость.

3. Управление инвентарем:

- Отслеживание доступности мест на турах, перелетах и гостиницах.

- Учет и управление инвентарем, включая номера гостиниц, билеты и транспортные средства.

4. Финансовый учет:

- Учет финансовых операций, связанных с бронированием и оплатой туров.

- Формирование счетов и отчетов для клиентов и бухгалтерии.

5. Анализ и отчетность:

- Создание отчетов о продажах, прибыли, заполненности туров и других ключевых метриках.

- Анализ данных для выявления популярных направлений и сезонных трендов.

6. Маркетинг и CRM:

- Управление маркетинговыми кампаниями и акциями.

- Сегментация клиентской базы для более эффективной коммуникации и персонализации предложений.

7. Служба поддержки клиентов:

- Отслеживание запросов и обращений клиентов.

- Предоставление актуальной информации и помощь клиентам.

8. Системы безопасности и доступа:

- Обеспечение безопасности данных и ограничение доступа к конфиденциальной информации шифрованием данных.

9. Расширение и развитие бизнеса:

- Анализ новых рынков и возможностей для расширения предложения туров.

- Управление ростом и масштабированием бизнеса.

На основе типичных бизнес-задач была построена логическая схема базы данных (Рисунок 1).

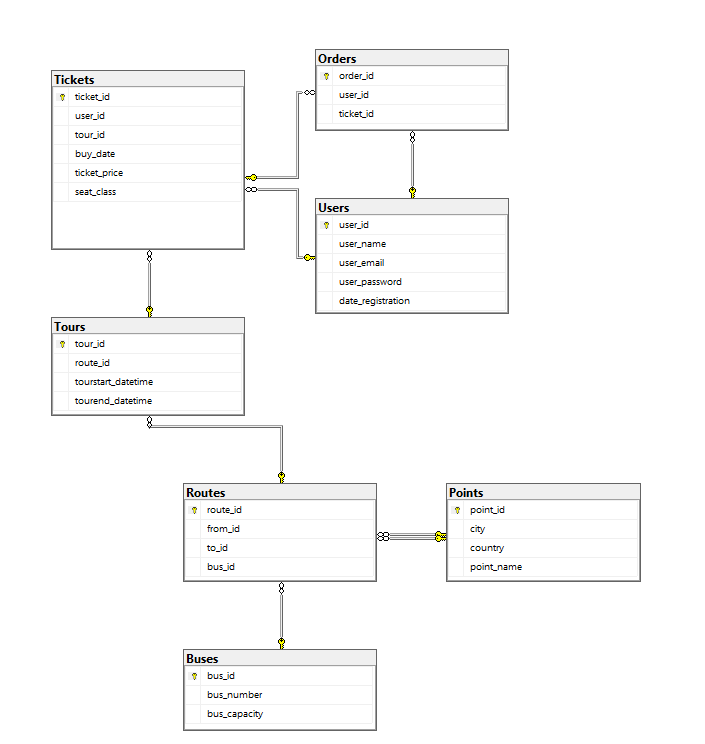


Рисунок 1 – Логическая схема базы данных

Диаграмма таблиц базы данных (Database Table Diagram) - это визуальное представление структуры базы данных и отношений между таблицами, которые хранятся в этой базе данных. Диаграмма базы данных будет представлена на рисунке 2.

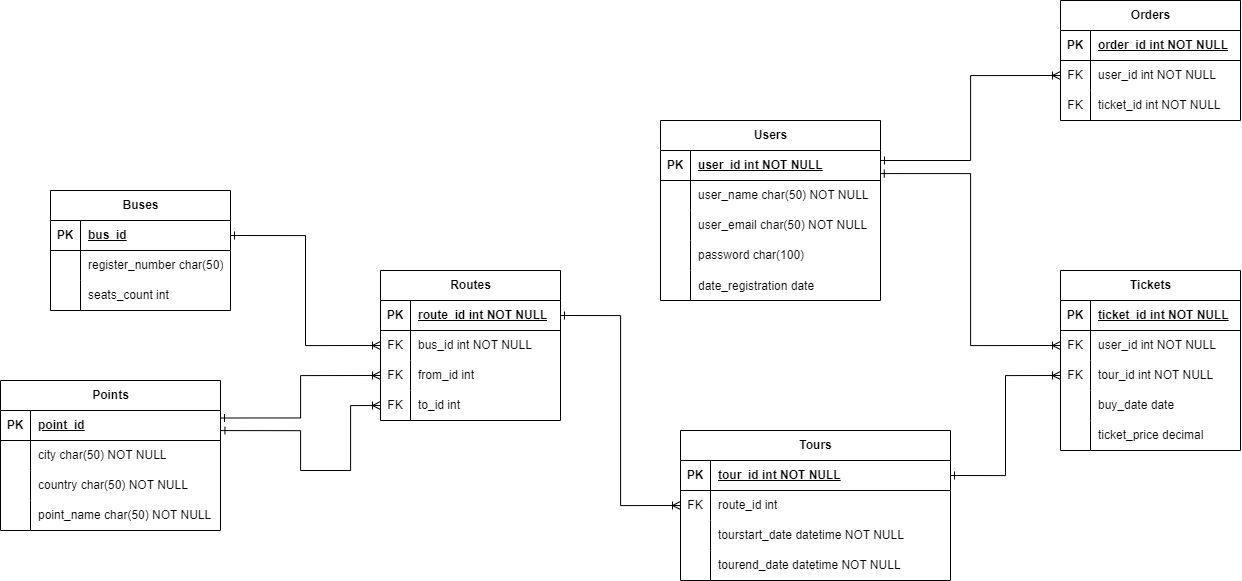


Рисунок 2 – Физическая схема базы данных Oracle

Диаграмма показывает связи между таблицами и полями, а также отношения между ними, такие как связи "один-ко-многим", "многие-ко-многим" и "один-к-одному". Например, таблица Users связана с таблицами Tickets и Orders через внешние ключи user\_id.

Для реализации базы данных было разработано 7 таблиц. В структуру схемы базы данных для проекта входят следующие таблицы: Users, Orders, Tickets, Routes, Points, Buses, Tours.

Эти таблицы предназначены для управления данными авиакомпании. Они могут использоваться для отслеживания заказов и билетов, которые пользователи могут покупать для полетов на самолетах компании, а также для отслеживания расписания полетов и данных самолетов и аэропортов. Ниже будет описание про каждую из них более подробно.

В зависимости от роли пользователя, он может иметь доступ к различным функциям системы. Роли пользователей будут следующими:

* Гость.
* Пользователь.
* Сотрудник турфирмы.

На основе предоставленного списка ролей построена диаграмма вариантов использования (Рисунок 3).

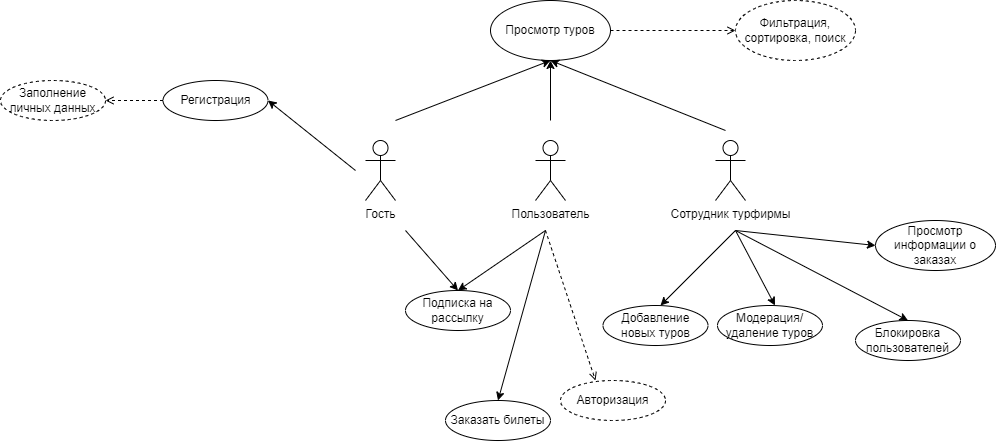


Рисунок 3 – UML-диаграмма вариантов использования

В начале работы с приложением пользователь является гостем. Ему будет доступен только просмотр расписания туров и информации о ценах на билеты без возможности покупки. После регистрации пользователь становится зарегистрированным пользователем.

Роль Пользователь получает возможность просматривать расписание рейсов, выбирать места и покупать билеты, просматривать свою историю полетов и управлять своим профилем. Кроме того, Пользователь может подписаться на рассылки об акциях и скидках, получать информацию об изменениях в расписании туров и задавать вопросы в службу поддержки.

Роль Сотрудник турфирмы включает в себя управление туристической компанией и мониторинг действий пользователей. Сотрудник турфирмы имеет доступ к просмотру информации о всех зарегистрированных пользователях, в том числе их истории полетов и покупок. При необходимости менеджер может редактировать информацию о турах и пользователях, например, изменять расписание туров и цены на билеты, а также блокировать пользователей за нарушение правил сервиса.

# **Лабораторная работа № 2**

Отчет, состоящий из: списка таблиц с указанием атрибутов (название, тип данных, ограничения, назначение), списка прочих объектов (тип объекта, наименование, назначение, зависимые объекты) отдельно для СУБД SQL Server и Oracle.

1.Таблица "Users" содержит информацию о пользователях. Эта таблица включает следующие поля:

user\_id - уникальный идентификатор пользователя (int, первичный ключ);

user\_name - имя пользователя (char(50), обязательное поле);

user\_email - адрес электронной почты пользователя (char(50), уникальный);

Password - пароль пользователя (char(100));

Date\_registration - дата регистрации пользователя (date).

2.Таблица "Orders" используется для отслеживания заказов пользователей. Она содержит следующие поля:

order\_id - уникальный идентификатор заказа (int, первичный ключ);

user\_id - идентификатор пользователя, который разместил заказ (int, внешний ключ, ссылается на user\_id таблицы Users);

ticket\_id - идентификатор билета, связанного с заказом (int).

3.Таблица "Tickets" используется для отслеживания билетов, которые пользователи могут приобретать. Она содержит следующие поля:

ticket\_id - уникальный идентификатор билета (int, первичный ключ);

user\_id - идентификатор пользователя, который приобрел билет (int, внешний ключ, ссылается на user\_id таблицы Users);

tour\_id - идентификатор рейса, связанного с билетом (int, внешний ключ, ссылается на tour\_id таблицы Tours);

Buy\_date - дата покупки билета (date);

Ticket\_price - стоимость билета (decimal);

Class - класс билета (char(50)).

4.Таблица "Tours" используется для отслеживания расписания туров. Она содержит следующие поля:

Tour\_id - уникальный идентификатор рейса (int, первичный ключ);

route\_id - идентификатор маршрута, связанного с туром (int, внешний ключ, ссылается на route\_id таблицы Routes);

Tourstart\_datetime – дата и время начала тура (date, обязательное поле);

Tourend\_datetime – дата и время рейса (time, обязательное поле).

5.Таблица "Routes" используется для отслеживания маршрутов, по которым летают рейсы. Она содержит следующие поля:

route\_id - уникальный идентификатор маршрута (int, первичный ключ);

from\_id - идентификатор точки отправления.

to\_id - идентификатор точки прибытия.

6.Таблица "Point" содержит информацию о точках назначения и отправления. Имеющиеся поля:

point\_id (идентификатор аэропорта, типа INTEGER, первичный ключ);

City (название города, в котором расположена точка, тип VARCHAR(50));

Country (название страны, в которой расположена точка, тип VARCHAR(50));

Point\_name (название точки, тип VARCHAR(50)).

7.Таблица "Buses" содержит информацию об автобусах, используемых авиакомпанией. Имеющиеся поля:

bus\_id (идентификатор автобуса, типа INTEGER, первичный ключ);

register\_number (регистрационный номер автобуса, тип VARCHAR(50));

Capacity (вместимость автобуса, тип INTEGER).

Для базы данных турфирмы было разработано представление tour\_info\_view (см. Листинг 2.1), которое содержит основную информацию о турах, а также количество проданных на каждый тур билетов и вместимость автобуса (т.е. общее количество билетов)

|  |
| --- |
| CREATE VIEW tour\_info\_view AS  SELECT s.tour\_id, r.from\_id, r.to\_id, s.tourstart\_datetime, COUNT(t.ticket\_id) AS tickets\_sold, b.bus\_capacity  FROM Tours s  JOIN Routes r ON s.route\_id = r.route\_id  JOIN Buses b ON r.bus\_id = b.bus\_id  LEFT JOIN Tickets t ON s.tour\_id = t.tour\_id  GROUP BY s.tour\_id, r.from\_id, r.to\_id, s.tourstart\_datetime, b.bus\_capacity  order by s.tour\_id; |

Листинг 2.1 – Представление tour\_info\_view

Также было создано представление ordered\_tickets\_view (см. Листинг 2.1), которое содержит информацию о проданных билетах на тур, отсортированных по дате покупки.

|  |
| --- |
| CREATE OR REPLACE VIEW ordered\_tickets\_view AS  SELECT \*  FROM Tickets  ORDER BY buy\_date; |

Листинг 2.2 – Представление ordered\_tickets\_view

Для оптимизации поиска были созданы индексы для столбцов, которые часто используются в представлениях, триггерах и функциях. Их создание представлено на Листинге 2.3.

|  |
| --- |
| ---Индекс для столбца tour\_id в таблице Tickets  CREATE INDEX ix\_Tickets\_tour\_id ON Tickets(tour\_id) TABLESPACE COMPANY\_SAIQDATA;  ---Индекс для столбца ticket\_id в таблице Orders  CREATE INDEX ix\_Orders\_ticket\_id ON Orders(ticket\_id) TABLESPACE COMPANY\_SAIQDATA;  ---Индекс для столбца rout\_id в таблице Tours  CREATE INDEX ix\_Tours\_route\_id ON Tours(route\_id) TABLESPACE COMPANY\_SAIQDATA;  ---Индекс для столбца user\_id в таблице Tickets  CREATE INDEX ix\_Tickets\_user\_id ON Tickets(user\_id) TABLESPACE COMPANY\_SAIQDATA; |

Листинг 2.3 – Создание индексов базы данных

В целом, использование индексов позволяет существенно ускорить операции поиска, сортировки и фильтрации данных в базе данных, особенно в случае большого объема данных. Однако создание индексов может занять дополнительное время при добавлении или изменении данных в таблицах, поэтому необходимо сбалансировать количество и тип индексов для оптимальной производительности базы данных.

Триггеры используются для обеспечения целостности данных и контроля доступа к данным, а также для автоматической обработки данных при выполнении определенных операций в таблице.

Триггер ticket\_capacity\_trigger установлен перед добавлением строки в таблицу Tickets. Он проверяет количество уже проданных билетов на тур, и если оно совпадает с общим количеством билетов, то не даёт добавить новый билет на этот тур и выбрасывает пользовательское исключение. Создание данного триггера представлено на Листинге 2.4.

|  |
| --- |
| CREATE OR REPLACE TRIGGER ticket\_capacity\_trigger  BEFORE INSERT ON Tickets  FOR EACH ROW  DECLARE  v\_capacity NUMBER;  v\_tickets\_count NUMBER;  BEGIN  SELECT bus\_capacity INTO v\_capacity  FROM Buses  WHERE bus\_id = (  SELECT bus\_id  FROM Routes  WHERE route\_id = (  SELECT route\_id  FROM Tours  WHERE tour\_id = :new.tour\_id  )  );    SELECT COUNT(\*) INTO v\_tickets\_count  FROM Tickets  WHERE tour\_id = :new.tour\_id;    IF v\_tickets\_count = v\_capacity THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Cannot add ticket, flight is full.');  END IF;  END; |

Листинг 2.4 – Создание триггера ticket\_capacity\_trigger

Триггер add\_order установлен после добавления строки в таблицу Tickets для автоматического добавления этого билета в таблицу Заказы. Создание триггера представлено на Листинге 2.5.

|  |
| --- |
| CREATE OR REPLACE TRIGGER add\_order  AFTER INSERT ON Tickets  FOR EACH ROW  BEGIN  INSERT INTO Orders (user\_id, ticket\_id)  SELECT u.user\_id, :new.ticket\_id  FROM Users\_ u  WHERE u.user\_id = :new.user\_id;  END; |

Листинг 2.5 – Создание триггера add\_order

Таким образом, триггеры обеспечивают сохранение целостности и структурированности данных.

В ходе разработки базы данных были созданы процедуры, позволяющие удалять, изменять и добавлять данные в таблицы. К примеру, процедура добавляющая нового пользователя представлена на Листинге 2.6.

|  |
| --- |
| CREATE OR REPLACE PROCEDURE add\_user(  user\_name IN CHAR,  user\_email IN CHAR,  user\_password IN CHAR,  date\_registration IN DATE  ) AS  BEGIN  INSERT INTO Users\_ (user\_name, user\_email, user\_password, date\_registration)  VALUES (user\_name, user\_email, user\_password, date\_registration);  END;  / |

Листинг 2.6 – Создание процедуры добавления нового пользователя

Были созданы процедуры удаления данных, принимающие в качестве входного параметра уникальный идентификатор, которые очищают информацию о заданном значении не только в основной, но и во всех связных таблицах. В качестве примера приведена процедура удаления пользователя на Листинге 2.7.

|  |
| --- |
| CREATE OR REPLACE PROCEDURE delete\_user(p\_user\_id IN NUMBER)  AS  BEGIN  -- удаление заказов пользователя  DELETE FROM orders WHERE user\_id = p\_user\_id;    -- удаление билетов пользователя  DELETE FROM tickets WHERE user\_id = p\_user\_id;    -- удаление пользователя  DELETE FROM users\_ WHERE user\_id = p\_user\_id;    COMMIT;  END;  / |

Листинг 2.7 – Создание процедуры удаления пользователя

