1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт кибербезопасности и защиты информации

**ЛАБОРАТОРНАЯ РАБОТА № 3**

**Моделирование данных и разработка реляционных схем баз данных**

по дисциплине «Основы построения защищенных баз данных»

1. Выполнила
2. студентка гр. 4851003/90801 Кулеева А.Г.

1. Руководитель
2. ассистент Зубков Е.А.
3. Санкт-Петербург
4. 2023

# **Цель**

Получить навыков разработки ER модели данных и реализации схемы базы данных в (пост)реляционном сервере.

# Формулировка задания

Для работы могут использоваться схемы данных и запросы из работ 1, 3 или другая схема на выбор.

1. Разработать, используя методы инфологического проектирования, схему данных согласно варианту.
2. Перевести разработанную ER-диаграмму в реляционную схему базы данных\*.
3. Нормализовать схему данных.
4. Определить и настроить характеристики базы данных в реляционном сервере (согласно варианту).
5. Сформулировать ограничения целостности предметной области.
6. Реализовать схему базы данных при помощи скриптов на языке SQL, с учетом декларативных ограничений целостности.
7. Модифицировать скрипт создания схемы данных для автоматической перегенерации схемы при его запуске.
8. Составить матрицу доступа для пользователей данных (согласно варианту) и настроить права доступа пользователей.

\* Обратите внимание, чтобы при генерации схемы базы данных и ограничений были учтены все особенности данных, такие как обязательность значений, уникальность ключей, кардинальность и тип связей.

Наличие выделенных сущностей, таких как «врач», «пациент» или «работник» и их характеристик НЕ означает что у вас должны получиться именно такие отношения с такими полями. Структуры данных при проектировании могут существенно измениться, появиться или исчезнуть отдельные поля и отношения.

# Ход работы

## Разработка ER-схемы

Был получен вариант 8. На основе описания варианта была составлена ER-диаграмма (Рисунок 1).

Бронирование билетов

* Покупатель (персональные данные, включая ФИО и номер(а) документа(ов), дата регистрации, рейтинг, возраст, история покупок)
* Автобус (номер автобуса, число мест, техническое состояние /кодом/, год выпуска, дата последнего обслуживания, классы мест с числом мест каждого класса, базовая стоимость места)
* Журнал рейсов (код постоянного рейса, дата и время отправления, дата и время прибытия, автобус, водитель ФИО, число доступных билетов, список пассажиров, стоимость проданных билетов)
* Билеты (код постоянного рейса, дата и время отправления, дата и время прибытия, место, стоимость, история покупок, история возвратов)
* Место (автобус, номер, у окна или нет, категория стоимости, льготное или нет, детское или нет, для инвалидов или нет, класс комфорта)
* Наценки (код наценки, описание\*, процент от базовой стоимости)

\*Наценка идет за тип места и время до отправления рейса. Заранее дешевле, есть наценки, увеличивающие стоимость места, есть уменьшающие.

Пользователи системы:

* Кассир
* Покупатель
* Администратор

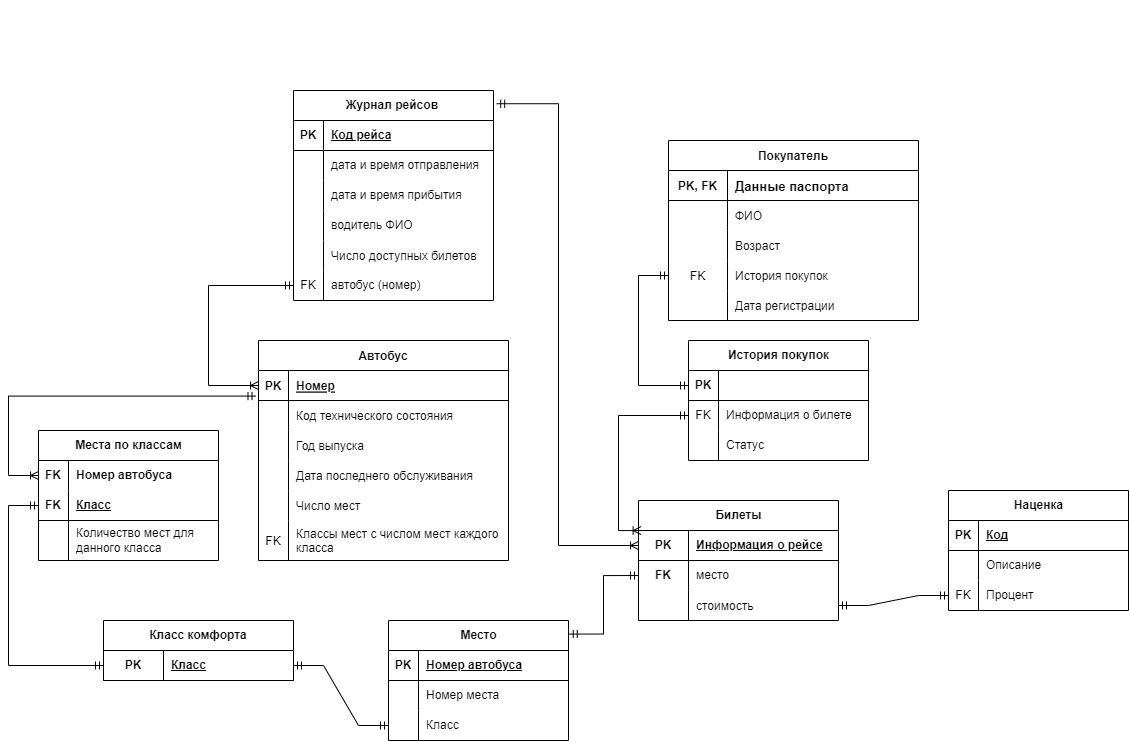


Рисунок 1 — ER-схема

В таблице «Автобус» указана информация об автобусах. Предполагаем, что на автовокзале есть несколько разных видов автобусов с различной вместимостью. Поле «классы мест» ссылается на отдельную таблицу. Количество классов и мест в каждом из них зависит от общего количества мест. Тогда в таблице будут записи следующего вида (Таблица 1):

Таблица 1 — Пример заполнения таблицы классификации мест

| Номер автобуса | Класс | Кол-во мест в классе | Стоимость |
| --- | --- | --- | --- |
| Номер1 | Класс1 | 10 | 300 |
| Номер1 | Класс2 | 15 | 200 |
| Номер1 | Класс3 | 20 | 100 |
| Номер2 | Класс1 | 5 | 200 |
| Номер2 | Класс2 | 5 | 150 |
| Номер2 | Класс3 | 10 | 100 |

Как видим, значения в таблице могут повторяться, поэтому в качестве первичного ключа используем два атрибута: номер и класс. Между таблицами «Автобус» и «Класс комфорта» используется связь агрегация, поскольку класс комфорта является частью от целого.

Место связно с классом комфорта ассоциацией. Для места первичным ключом будет номер автобуса. Такие значения, как «у окна», «для детей», «для льготников» были вынесены в таблицу классов и являются показателями класса комфорта.

Была создана таблица «Журнал рейсов» с информацией о рейсе, которая включает в себя номер рейса, номер автобуса, фио водителя, а также дату и время прибытия и отправления. Данная таблица связана с таблицей «Билет» связью 1 ко многим: один рейс (одна запись в журнале рейсов) может быть связан с несколькими билетами. Данная таблица также имеет внешний ключ «номер автобуса», который ассоциативно связан с таблицей «Автобус». Связь: обязательная ссылка, поскольку каждому рейсу соответствует какой-то автобус, а автобус в свою очередь может иметь несколько рейсов в день. В таблице «Журнал рейсов» уникальным значением является код рейса.

Таблица «Билеты» содержит информацию о рейсе (через FK), место, стоимость. Место связано с одноименной таблицей необязательной ссылкой: в любом билете указано место, но не любое место может иметь билет (место не продано).

Также реализована модель пользователя, которая содержит ПДн, дату регистрации на ресурсе и историю покупок, которая ссылается на одноименную таблицу. В этой таблице содержится список билетов (агрегация с таблицей «Билет») и статус заказа: выполнен, отменен, возврат, таким образом история возвратов была реализована через статус заказа в истории покупок.

## Перевод ER-диаграммы в реляционную схему бд

Схема базы данных — это ее представление в реляционном сервере. Схема представляет собой набор формальных описаний реляционных отношений и связей между ними. Разработаны правила перевода из модели сущность-связь в реляционную схему.

Сущности ER-диаграммы соответствуют таблицы схемы, кортежам, или экземплярам сущностей — записи в таблицах. Атрибуты, как уже упоминалось, представляют собой поля в записях, а домены — типы данных полей. Все связи на схеме данных делятся на два вида.

Идентифицирующие связи — атрибуты связанной таблицы мигрируют в первичный ключ таблицы. Обозначается сплошной линией.

Не идентифицирующие связи — атрибуты связанной таблицы мигрируют в тело таблицы. Обозначается пунктирной линией.

Каждая рассмотренная выше связь ER-диаграммы переводится в связь реляционной схемы логического уровня:

Наследование — идентифицирующая связь, в которой первичный ключ родителя полностью совпадает с внешним ключом наследника (также являющегося первичным для наследника).

Агрегация — идентифицирующая связь, в которой первичный ключ зависимой сущности представляет собой внешний ключ (на первичный ключ не зависимой сущности) и еще одно или более собственных полей.

Ассоциация — не идентифицирующая связь.

Таким образом, логическая схема бд будет выглядеть следующим образом (Рисунок 2).

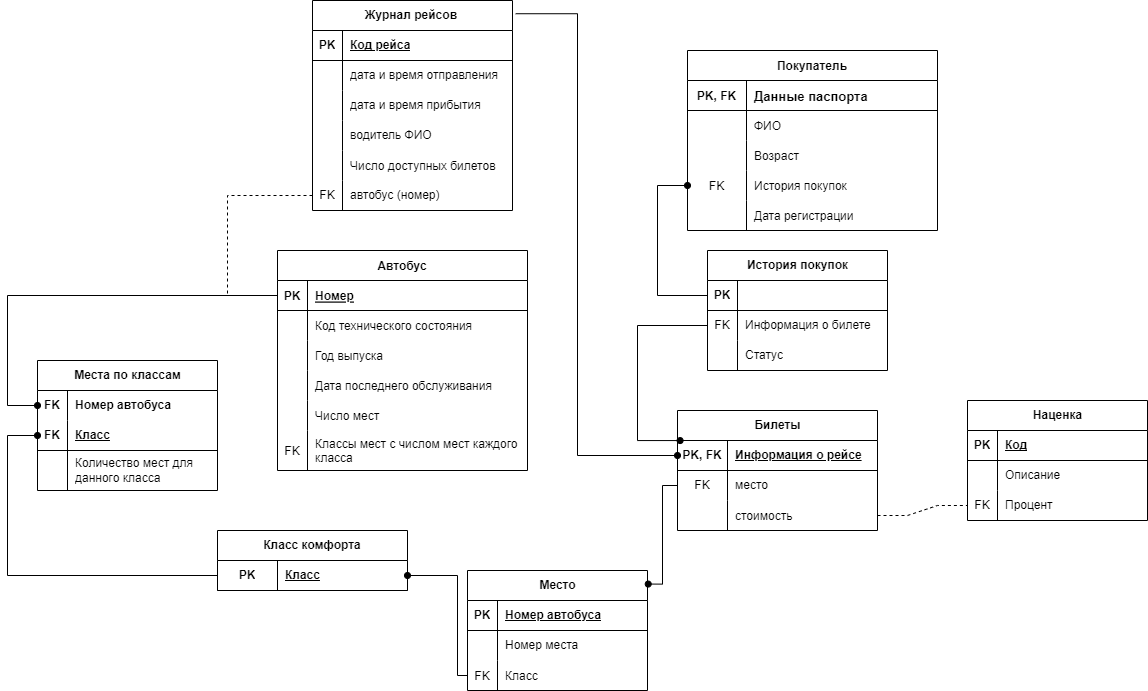


Рисунок 2 — Логическая схема

## Нормализация

### Первая нормальная форма

Переменная отношения находится в первой нормальной форме (1NF) тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов.

Данное отношение уже находится в 1НФ.

### Вторая нормальная форма

Отношение находится во 2НФ, если оно находится в 1НФ и каждый не ключевой атрибут неприводимо зависит от Первичного Ключа (ПК).

Неприводимость означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, от которого можно также вывести данную функциональную зависимость.

Данная схема находится в 2НФ. Все неключевые атрибуты зависят от ключевых. В таблице «Информация о билете» неключевых атрибутов нет.

### Третья нормальная форма

Отношение находится в 3НФ, когда находится во 2НФ и каждый не ключевой атрибут нетранзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы.

Отношение уже находится в 3НФ, данные особенности были учтены при проектировании схемы данных.

### Нормальная форма Бойса-Кодда (НФБК)

Переменная отношения находится в нормальной форме Бойса-Кодда (BCNF) тогда и только тогда, когда она находится в 3НФ и детерминанты всех ее функциональных зависимостей являются потенциальными ключами.

Иначе говоря, ключевые атрибуты составного ключа не должны зависеть от неключевых атрибутов.

Отсюда следует, что требования нормальной формы Бойса-Кодда предъявляются только к таблицам, у которых первичный ключ составной. Таблицы, у которых первичный ключ простой, и они находятся в третьей нормальной форме, автоматически находятся и в нормальной форме Бойса-Кодда.

Главное правило нормальной формы Бойса-Кодда (BCNF) звучит следующим образом: часть составного первичного ключа не должна зависеть от неключевого столбца.

В рассматриваемом отношении составной ПК есть только в таблице «Класс комфорта». Ключевые параметры не зависят от неключевых: модель автобуса (определяется номером) определяет общее количество мест, отсюда определяется количество в данном классе; класс определяет количество мест, а не наоборот. Соответственно, отношение уже находится в БКНФ.

### Дальнейшая нормализация

Рассматриваемое отношение уже находится в 4НФ, поскольку отсутствуют нетривиальные многозначные зависимости.

Отношение также находится в 5НФ. Пятая нормальная форма ориентирована на работу с зависимыми соединениями. Указанные зависимые соединения между тремя атрибутами встречаются очень редко. Зависимые соединения между четырьмя, пятью и более атрибутами указать практически невозможно.

## Проектирование бд

В ходе проектирования и написания кода для создания базы данных были задействованы следующие виды ограничений: ограничение атрибута для обеспечения целостности с помощью встроенных ключевых слов Postgres NOT NULL (для обязательных атрибутов) и UNIQUE (для первичных и потенциальных ключей), использована конструкция CHECK для установления допустимых значений стоимости билетов, возраста пассажиров, количества мест, наценок.

Ограничение целостности отношений с помощью ключевых слов Postgres REFERENCES и PRIMARY KEY.

Ограничение на уровне базы данных (ASSERTION) хоть и прописаны в стандарте SQL, но в Postgres не поддерживаются, могут быть в случае необходимости быть реализованным, например, с помощью ограничений на атрибуты или триггерами в случае сложных ограничений.

## Матрица доступа

В БД существуют 3 роли в соответствии с вариантом: Покупатель, Кассир и Администратор. У Администратора есть полных доступ ко всем таблицам БД, его действия регламентируются только встроенными ограничениями. У Кассира есть доступ на чтение к Классам комфорта, Местам по классам, Автобусам, Журналу рейсов, Билетам, доступ на запись в Журнал рейсов (изменять количество свободных мест), Билетам, Наценкам. У Покупателя есть доступ на чтение к Журналу рейсов, своей Истории покупок, Билетам, Месту, своим данным («Пользователь»).

Таблица 2 — Матрица доступа для пользователей БД

|  | Администратор | Кассир | Покупатель |
| --- | --- | --- | --- |
| Журнал рейсов | RW | RW | R |
| Пользователь | RW | - | R |
| Билеты | RW | RW | R |
| Наценка | RW | RW | - |
| История поездок | RW | - | R |
| Автобус | RW | R | - |
| Классы комфорта | RW | R | - |
| Места по классам | RW | R | - |
| Место | RW | - | R |

# Вывод

В ходе выполнения данной лабораторной работы были изучены основы разработки ER модели данных и реализации схемы базы данных в (пост)реляционном сервере. Была реализована собственная схема по варианту, приведена к логической, нормализована. На основе схемы была спроектирована база данный, на которую наложены ограничения целостности, прописана матрица доступа.

ПРИЛОЖЕНИЕ 1

DROP TABLE IF EXISTS voyage\_table CASCADE;

DROP TABLE IF EXISTS customer CASCADE;

DROP TABLE IF EXISTS ticket CASCADE;

DROP TABLE IF EXISTS travel\_history CASCADE;

DROP TABLE IF EXISTS discount CASCADE;

DROP TABLE IF EXISTS bus CASCADE;

DROP TABLE IF EXISTS sits\_per\_classes CASCADE;

DROP TABLE IF EXISTS classes CASCADE;

DROP TABLE IF EXISTS sits CASCADE;

CREATE TABLE IF NOT EXISTS bus (

number text NOT NULL PRIMARY KEY,

code\_health text NOT NULL,

year text NOT NULL,

date\_last\_sto text NOT NULL,

sits\_amount integer CHECK (sits\_amount < 60));

CREATE TABLE IF NOT EXISTS voyage\_table (

code text NOT NULL PRIMARY KEY,

departure text NOT NULL,

arrival text NOT NULL,

driver text NOT NULL,

bus\_num text NOT NULL REFERENCES bus(number) ,

tickets\_amount integer NOT NULL);

CREATE TABLE IF NOT EXISTS ticket (

sit\_number integer NOT NULL CHECK (sit\_number < 60),

ticket\_info text NOT NULL PRIMARY KEY REFERENCES voyage\_table(code),

cost integer NOT NULL UNIQUE);

CREATE TABLE IF NOT EXISTS travel\_history (

ticket\_info text NOT NULL PRIMARY KEY REFERENCES ticket(ticket\_info),

status text NOT NULL);

CREATE TABLE IF NOT EXISTS discount (

code text NOT NULL PRIMARY KEY,

description text NOT NULL,

percent integer NOT NULL REFERENCES ticket(cost));

CREATE TABLE IF NOT EXISTS customer (

passport\_number text NOT NULL UNIQUE PRIMARY KEY,

FIO text NOT NULL,

date\_of\_birth date NOT NULL,

registration\_date date NOT NULL,

travel\_history text REFERENCES travel\_history(ticket\_info));

CREATE TABLE IF NOT EXISTS classes (

class\_type text NOT NULL PRIMARY KEY);

CREATE TABLE IF NOT EXISTS sits\_per\_classes (

bus\_num text NOT NULL REFERENCES bus(number),

amount integer NOT NULL,

class\_type text NOT NULL REFERENCES classes(class\_type),

PRIMARY KEY(bus\_num, class\_type));

CREATE TABLE IF NOT EXISTS sits (

bus\_num text NOT NULL PRIMARY KEY REFERENCES bus(number),

sit\_number integer NOT NULL,

class\_type text NOT NULL REFERENCES classes(class\_type));

DROP ROLE IF EXISTS admin;

DROP ROLE IF EXISTS customer;

DROP ROLE IF EXISTS employee;

CREATE ROLE admin;

CREATE ROLE customer;

CREATE ROLE employee;

--CREATE USER pupkin\_a;

--CREATE USER pupkin\_c;

--CREATE USER pupkin\_e;

GRANT admin TO pupkin\_a;

GRANT employee TO pupkin\_e;

GRANT customer TO pupkin\_c;

GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO admin;

-- Роль employee имеет доступ на чтение к определенным таблицам и на запись в voyage\_table, ticket

GRANT SELECT ON TABLE bus TO employee;

GRANT SELECT ON TABLE classes TO employee;

GRANT SELECT ON TABLE voyage\_table TO employee;

GRANT SELECT ON TABLE ticket TO employee;

GRANT INSERT, UPDATE, DELETE ON TABLE voyage\_table TO employee;

GRANT INSERT, UPDATE, DELETE ON TABLE ticket TO employee;

-- Роль customer имеет доступ только на чтение к определенным таблицам

GRANT SELECT ON TABLE voyage\_table TO customer;

GRANT SELECT ON TABLE travel\_history TO customer;

GRANT SELECT ON TABLE ticket TO customer;

GRANT SELECT ON TABLE customer TO customer;

GRANT SELECT ON TABLE sits TO customer;