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

**ЛАБОРАТОРНАЯ РАБОТА № 4**

**Анализ схемы 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**

Получение навыков проактивной и реактивной защиты баз данных от логического вывода.

# Формулировка задания

Работа выполняется на основе данных из работы №3 по проектированию реляционной схемы.

1. Определите стороннего пользователя, имеющего доступ к открытым данным.

2. Определите в схеме данных конфиденциальные атрибуты (не мене 3- х) для различных пользователей.

3. Напишите скрипт, в автоматическом режиме определяющий функциональные зависимости предметной области на основе сбора данных о схеме из СУБД. Учтите все пути получения зависимостей (включая многозначные), виды ключей и связей.

4. Проанализируйте полученный список, при необходимости дополните его вручную.

5. Разработайте скрипт, построения полного замыкания функциональных зависимостей согласно схеме данных.

6. Разработайте скрипт, определяющий потенциально доступные пользователю конфиденциальные атрибуты и возможные каналы их вывода.

7. Оцените реалистичность каналов вывода с использованием внешних источников в отношении нового пользователя. Определите атрибуты, находящиеся под угрозой утечки.

# Ход работы

## Описание схемы данных

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

Данная схема уже нормализована, многозначные зависимости отсутствуют.

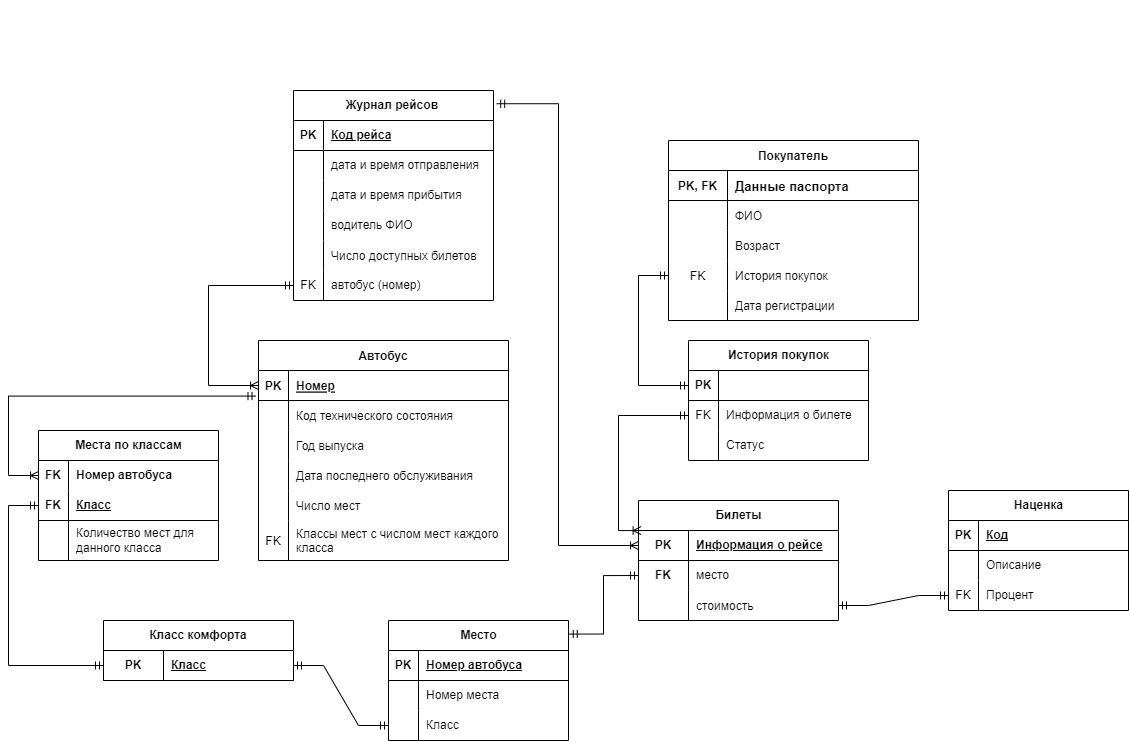


Рисунок 1 — ER-схема

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 1 — 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 |

## Определение в схеме данных конфиденциальных атрибутов для различных пользователей

В Таблице 2 все атрибуты отношения разделены на конфиденциальные и открытые.

Таблица 2 — Конфиденциальные и открытые атрибуты

|  |  |  |
| --- | --- | --- |
| Открытые для всех | Конфиденциальные для покупателей, открытые для сотрудников | Конфиденциальные для сотрудников, открытые для покупателей |
| Автобус (номер автобуса, число мест, техническое состояние, год выпуска, дата последнего обслуживания, классы мест с числом мест каждого класса) | Наценки (код наценки, описание\*, процент от базовой стоимости) | Покупатель (персональные данные, включая ФИО и номер(а) документа(ов), дата регистрации, возраст, история покупок) |
| Журнал рейсов (код постоянного рейса, дата и время отправления, дата и время прибытия, автобус, водитель ФИО, число доступных билетов) | Места по классам (номер автобуса, класс, количество мест для данного класса) | История покупок (информация о билете, статус) |
| Билеты (код постоянного рейса, место, стоимость) | Классы комфорта (класс) | Место (номер автобуса, номер места, класс комфорта) |

## Определение функциональных зависимостей

При проактивной защите деятельность по обеспечению безопасности включает на первом этапе получение набора функциональных зависимостей из схемы базы данных, к которой формируется запрос пользователя. Для восстановления функциональных зависимостей, на основании которых спроектирована схема, от сервера СУБД необходимо получить: список таблиц, список столбцов таблиц, информацию о первичных ключах таблиц и информацию о внешних ключах таблиц.

Этот процесс хорошо автоматизируется на основе запросов к системному каталогу сервера СУБД Postgres. Код для сбора функциональных зависимостей представлен в Приложении 1, результат выполнения функции представлен на Рисунке 2.

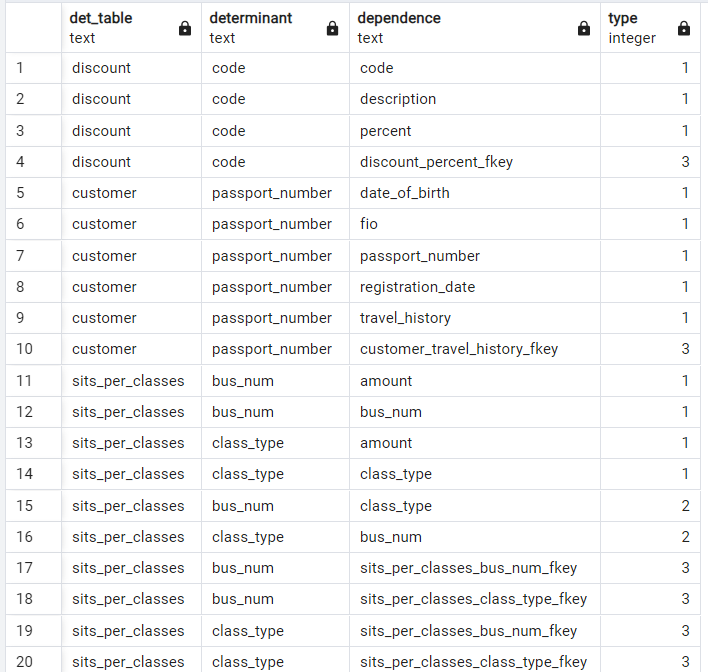


Рисунок 3 — Функциональные зависимости

На основании полученной информации можно сформировать множество функциональных зависимостей в схеме по следующим правилам:

1. Если в таблице Т множество атрибутов К составляет первичный ключ, а атрибут С является неключевым, то к множеству функциональных зависимостей добавляется зависимость К С.
2. Если в таблице Т атрибуты составляют первичный ключ, то к множеству функциональных зависимостей добавляются многозначные зависимости вида .

Были выявлены следующие функциональные зависимости в схеме (Таблица 3).

Таблица 3 — Функциональные зависимости

| Отношение | Ключевой атрибут | Неключевые атрибуты |
| --- | --- | --- |
| Для всех пользователей | | |
| Журнал рейсов | Код постоянного рейса | дата и время отправления, дата и время прибытия, автобус, водитель ФИО, число доступных билетов |
| Билеты | Код постоянного рейса | место, стоимость |
| Для пользователя с ролью Покупатель | | |
| Покупатель | Номер документа | ФИО, дата регистрации, дата рождения, история покупок |
| История покупок | Информация о билете | Статус покупки |
| Место | Номер автобуса | Класс, номер места |
| Для пользователя с ролью Сотрудник | | |
| Наценки | Код наценки | описание, процент от базовой стоимости |
| Места по классам | Номер автобуса | класс, количество мест для данного класса |
| Автобус | Номер автобуса | число мест, техническое состояние, год выпуска, дата последнего обслуживания, классы мест с числом мест каждого класса |
| Класс комфорта | Класс | - |

## Построение полного замыкания фз

Был разработан скрипт на языке SQL, который по известным атрибутам строит полное замыкание, «вытаскивая» неизвестные атрибуты. Пример работы представлен на Рисунке 4. Пусть известно по 2 атрибута из 2 таблиц (среди них есть ключевые). Скрипт выводит все атрибуты, которые зависят от детерминанта. При этом атрибуты из таблицы «Журнал рейсов» не были выведены, поскольку атрибут «Время прибытия» не является ключевым, то есть от него не зависят другие атрибуты.

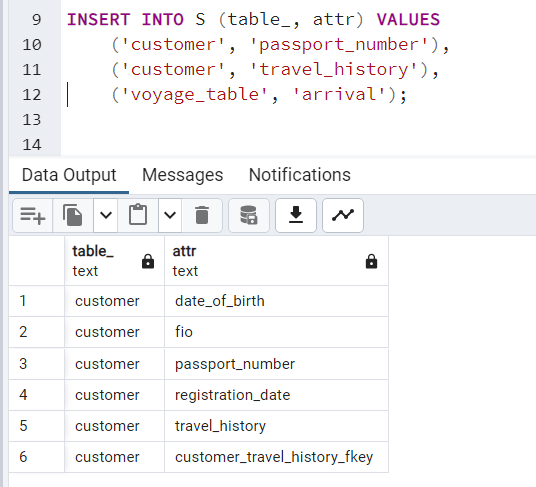


Рисунок 4 — Результат построения полного замыкания

## Возможности логического вывода

Администратор имеет полный доступ ко всем данным. Остальные пользователи могут провести атаку логического вывода только путем повышения привилегий, поскольку доступ к некоторым таблицам запрещен для определенных ролей.

У покупателя есть доступ к атрибуту «номер автобуса», который есть в нескольких отношения. Данный атрибут является ПК для таблицы «Автобусы», а также связан с таблицей «Классы». Если нарушитель повысит привилегии, то сможет получить информацию об автобусах и классах. Также таблица «Классы» связана с «Местом» как внешний ключ. Хотя таблица не содержит ничего, кроме самого класса, однако в случае её расширения также могут быть получены конфиденциальные атрибуты.

Кроме того, покупатель имеет доступ к таблице всех пользователей. Таким образом, зная данные паспорта другого человека, можно получить все информацию о нем, в том числе его историю покупок. Однако данная атака маловероятна, поскольку такие данные сложно получить.

У кассира нет доступа к ПДн покупателей, он может получить доступ только к информации об автобусах, рейсах, классах комфорта, наценках. Эти данные не несут в себе никакой ценности, поэтому атака логического вывода невозможна.

# Вывод

В ходе выполнения данной лабораторной работы были навыки проактивной и реактивной защиты баз данных от логического вывода. Были реализованы скрипты, которые находят в схеме ФЗ и строят по ним полное замыкание. Также была оценена возможность атаки логического вывода: вероятность атаки оценена, как низкая, ущерб средний.

ПРИЛОЖЕНИЕ 1

CREATE OR REPLACE FUNCTION table\_deps(name\_for\_deps text)

RETURNS void AS

$$

BEGIN

--для пк -> атрибут и многозначных зависимостей

INSERT INTO func\_deps (determinant, dependence)

SELECT DISTINCT t.column\_name, c.column\_name

FROM (

select ccu.column\_name from information\_schema.table\_constraints tc

join information\_schema.constraint\_column\_usage ccu

on ccu.constraint\_name = tc.constraint\_name

where constraint\_type = 'PRIMARY KEY'

and tc.table\_schema = 'public' and tc.table\_name = name\_for\_deps

) AS t

CROSS JOIN information\_schema.columns c

WHERE c.table\_schema = 'public' AND c.table\_name = name\_for\_deps;

--определение типа зависимости

UPDATE func\_deps

SET type = 1

WHERE type IS NULL and (dependence NOT IN (select ccu.column\_name from information\_schema.table\_constraints tc

join information\_schema.constraint\_column\_usage ccu

on ccu.constraint\_name = tc.constraint\_name

where constraint\_type = 'PRIMARY KEY'

and tc.table\_schema = 'public' and tc.table\_name = name\_for\_deps) or determinant = dependence);

UPDATE func\_deps

SET type = 2

WHERE type IS NULL;

--для внешнего ключа

INSERT INTO func\_deps (determinant, dependence, type)

SELECT DISTINCT t.column\_name, tc.constraint\_name, 3

FROM (

select ccu.column\_name from information\_schema.table\_constraints tc

join information\_schema.constraint\_column\_usage ccu

on ccu.constraint\_name = tc.constraint\_name

where constraint\_type = 'PRIMARY KEY'

and tc.table\_schema = 'public' and tc.table\_name = name\_for\_deps

) AS t

CROSS JOIN information\_schema.table\_constraints tc

join information\_schema.key\_column\_usage kcu

on kcu.constraint\_name = tc.constraint\_name

where constraint\_type = 'FOREIGN KEY'

and tc.table\_schema = 'public' and tc.table\_name = name\_for\_deps;

--добавление таблицы детерминанта

UPDATE func\_deps

SET det\_table = name\_for\_deps

WHERE det\_table IS NULL;

END;

$$

LANGUAGE plpgsql;

DROP TABLE IF EXISTS func\_deps CASCADE;

CREATE TABLE func\_deps (

det\_table text,

determinant text,

dependence text,

type INT

);

DO $$

DECLARE

text\_value text;

BEGIN

FOR text\_value IN (select table\_name from information\_schema.tables

where table\_schema = 'public' and table\_type = 'BASE TABLE')

LOOP

PERFORM table\_deps(text\_value);

END LOOP;

END $$;

select \* from func\_deps;

DROP TABLE IF EXISTS S CASCADE;

CREATE TABLE S (

table\_ text,

attr text

);

--добавление имеющихся атрибутов--

INSERT INTO S (table\_, attr) VALUES

--('customer', 'passport\_number'),

('voyage\_table', 'code'),

--('travel\_history', 'ticket\_info'),

('ticket', 'ticket\_info'),

--('sits', 'bus\_num');

('discount', 'code'),

('classes', 'class\_type'),

('sits\_per\_classes', 'bus\_num');

DROP TABLE IF EXISTS Z CASCADE;

CREATE TABLE Z (

table\_ text,

attr text

);

INSERT INTO Z (table\_, attr)

SELECT det\_table, dependence

FROM func\_deps

JOIN S AS fd

ON (func\_deps.det\_table, func\_deps.determinant) = (fd.table\_, fd.attr);

Select \* from Z;