**Лабораторная работа 2**

*Выполнил: Бурматов Александр, гр. КМБ-2015*

В ходе лабораторной работы разработано два приложения:

1. Сервис импорта данных, которое осуществляет прием данных (ненормализованных объектов), выполняет их нормализацию и сохраняет нормализованные объекты в базе данных.
2. Клиентское приложение, способное отправлять данные сервису импорта данных.

Все пересылаемые данные шифруются при помощи алгоритма TripleDES.

В качестве менеджера очередей сообщений был использован RabbitMQ.

Исходный код приложения размещен на GitHub:

<https://github.com/AlexBurmatov/DataImporter>

Описание работы:

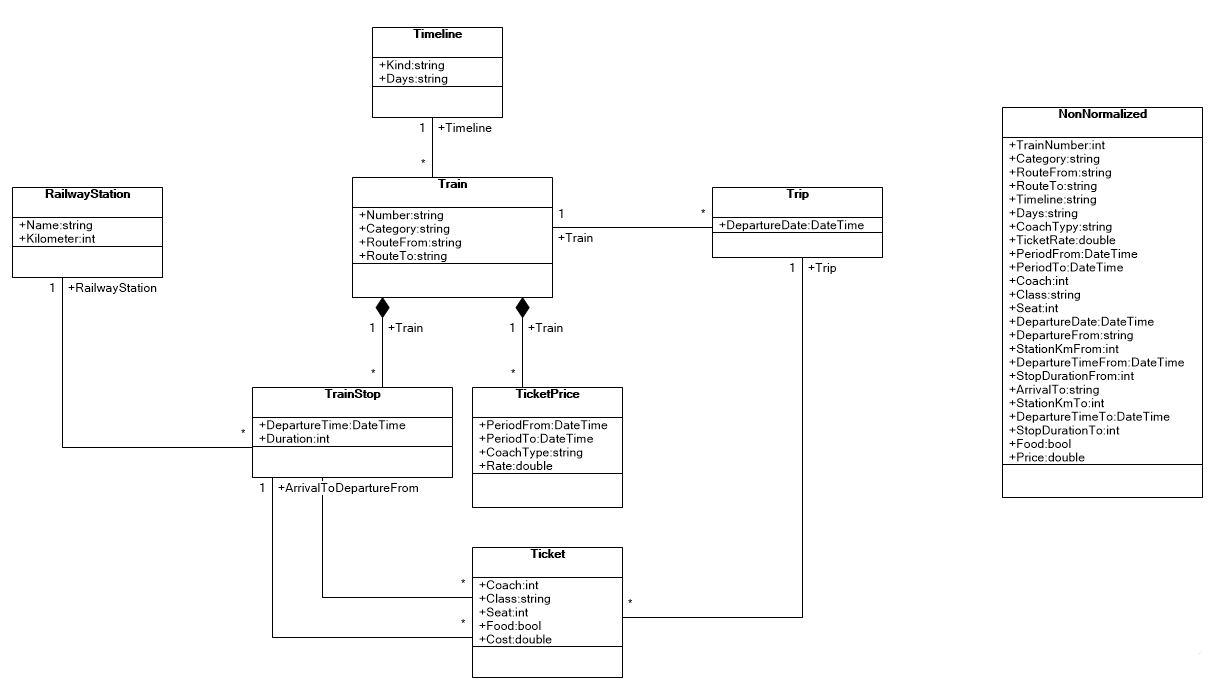
После успешного подключения клиентского приложения к сервису импорта данных (через сокеты) между ними происходит обмен ключами: сервис импорта данных отправляет открытый ключ (K1) шифрования RSA, а клиент, в свою очередь, отправляет сервису ключ (К2) симметричного шифрования 3DES, зашифрованный ключом К1.

Данные могут пересылаться двумя способами: через сокеты и с помощью очереди сообщений. В обоих случаях данные шифруются ключом К2.

Генерация нового ненормализованного объекта, т. е. заполнение его полей случайными значениями, и его отправка сервису осуществляется по нажатию клавиши: Enter – с помощью очереди сообщений, Пробел – через сокеты.

В сервисе импорта данных был использован алгоритм нормализации, реализованный в рамках входного контроля.

Предметная область:



Сценарии тестирования

1. Приложение позволяет выполнять прием и передачу данных из ненормализованной БД в нормализованную без модификации данных.

Следующий запрос демонстрирует, что данные при нормализации не были утеряны.

**select** train.**number**, train.category, train.routefrom, train.routeto,

timeline.kind, timeline.days, ticketprice.periodfrom, ticketprice.periodto, ticketprice.coachtype, ticketprice.rate,

ticket.coach, ticket."class", ticket.seat, trip.departuredate,

t1.departuretime departuretimefrom, r1."name" departurefrom, r1.kilometer stationkmfrom, t1.duration stopdurationfrom,

t2.departuretime departuretimeto, r2."name" arrivalto, r2.kilometer stationkmfrom, t2.duration stopdurationfrom,

ticket.food, ticket."cost"

**from** ticket

**join** trip **on** ticket.trip=trip.primarykey

**join** trainstop t1 **on** ticket.departurefrom=t1.primarykey

**join** trainstop t2 **on** ticket.arrivalto=t2.primarykey

**join** railwaystation r1 **on** t1.railwaystation=r1.primarykey

**join** railwaystation r2 **on** t2.railwaystation=r2.primarykey

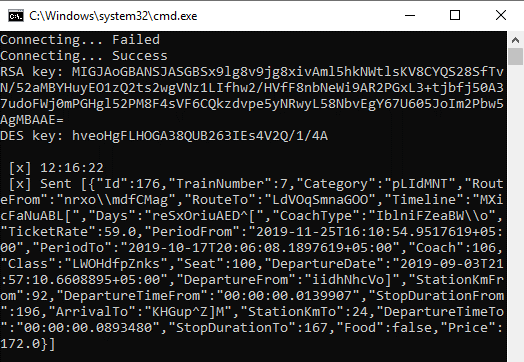
**join** train **on** t1.train=train.primarykey

**join** ticketprice **on** ticketprice.train=train.primarykey

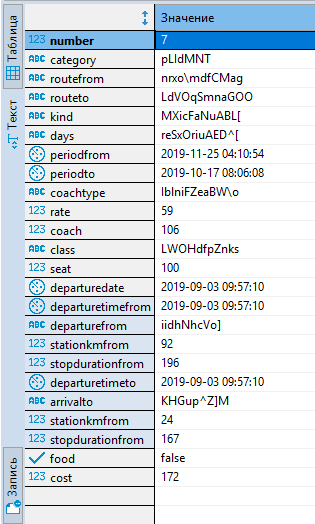
**join** timeline **on** train.timeline=timeline.primarykey

**where** ticket.seat='100';

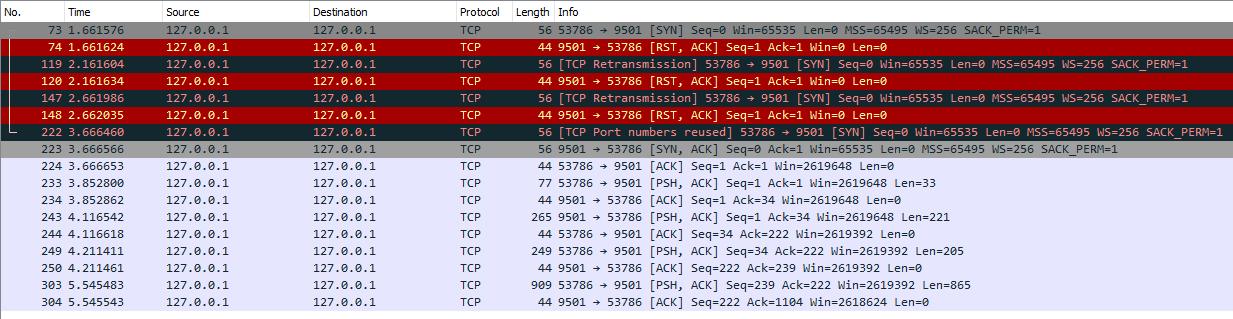
Сгенерированный объект:

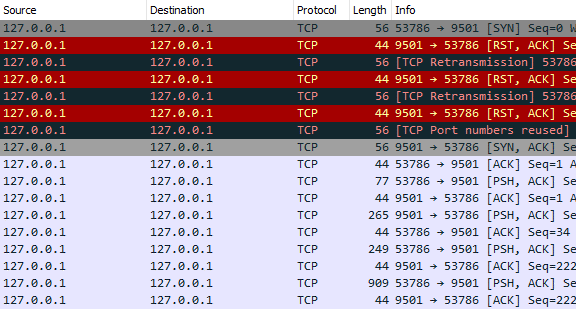


Результат запроса в нормализованной БД.

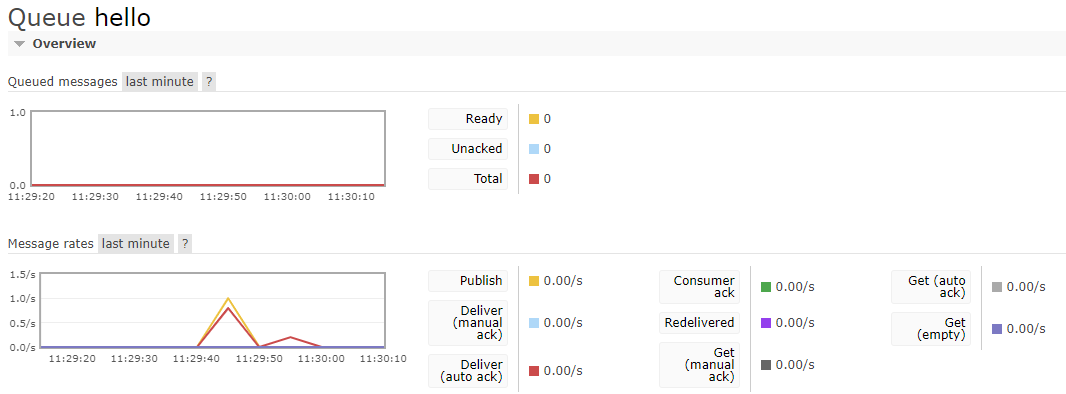


1. Сервис импорта данных открывает на локальной машине 9501 порт для подключения клиентов по сокетам. Запустив приложения, можем отследить в WireShark пакеты, которыми они обмениваются, что подтверждает их взаимодействие через сокеты.

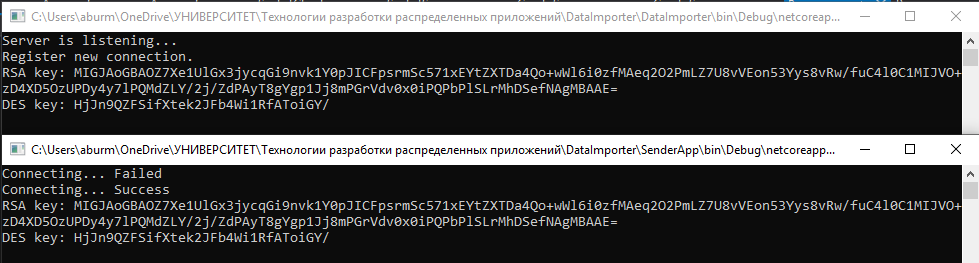




1. У сервиса импорта данных также есть возможность принимать сообщения, используя очередь сообщений. Отследить интенсивность потока сообщений можно с помощью веб-интерфейса менеджера сообщений RabbitMQ.

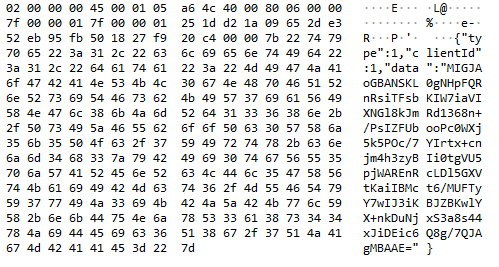


1. Шифрование осуществляет с использованием асимметричного и симметричного ключей. Корректность их передачи можно проверить, выполнив их вывод на экран консоли.



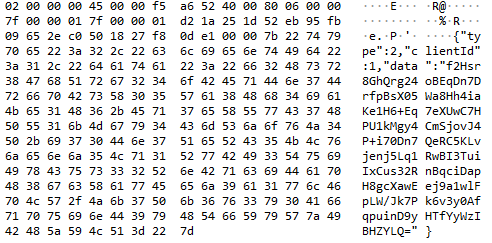
Ключ симметричного шифрования был расшифрован сервисом импорта данных корректно, т. к. они совпадают.

1. Отследив передаваемые по сети TCP-пакеты, можем убедиться, что ключ асимметричного шифрования передается в открытом (незашифрованном) виде.



Передаваемое значение ключа можно сравнить с ключом сервиса импорта данных из предыдущего пункта.

1. В следующем пакете передается ключ симметричного шифрования.



Ключ шифрования из предыдущих пунктов: Hj3n9QZFSifXtek2JFb4WilRfAToiGY/

Делаем вывод, что ключ действительно передается в зашифрованном виде.

1. Для корректной работы приложения по сети используется имя хоста «servicehost», на котором запускается сервис, и «apphost», где развернут менеджер очередей сообщений.

Для тестирования servicehost мы использовали динамически полученный IP-адрес. Поскольку RabbitMQ развернут в докере, то для него указан 127.0.0.1.

