

Министерство образования и науки РФ
Санкт-Петербургский Политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа искусственного интеллекта
Направление 02.03.01 «Математика и компьютерные науки»

Отчёт

по прохождению практики
«Разработка модуля интеграции с базой данных MasterSCADA»

Студент:

Башарина Е.А., гр. 3530201/90101

Санкт-Петербург — 2022

Содержание

1	Цели работы	2
2	Описание базы данных	3
2.1	ER-диаграмма	3
2.2	Восстановление целостности базы данных	5
3	Задание 1. Подключение к БД с заданным периодом и считывание новых записей	6
3.1	Программная реализация	6
3.1.1	main.py	6
3.1.2	Job.py	7
3.1.3	Parser.py	8
3.2	Результат работы программы	9
4	Задание 2. Статистический анализ технических параметров системы	11
4.1	Программная реализация	11
4.1.1	main.py	11
4.1.2	Parser.py	12
4.2	Результат работы программы	13
	Заключение	16

1 Цели работы

В рамках прохождения практики требовалось разработать модуль интеграции с базой данных MasterSCADA (СУБД Firebird 2.5, ОС Windows, ЯП Python). Для этого было необходимо:

- изучить используемые инструменты и программное обеспечение, включая СУБД Firebird и утилиту администрирования баз данных IBEexpert;
- ознакомиться со структурой предоставленной базы данных;
- изучить структуру запросов на используемом диалекте SQL (SQL Dialect 3).

Поставленная цель подразделялась на следующие **задачи**:

1. Подключение к БД с заданным периодом и считывание новых записей;
2. Статистический анализ технических параметров системы и контроль выхода их за критические границы.

2 Описание базы данных

База данных MasterSCADA (далее - БД) является основным источником данных для программного комплекса «Централизованная диспетчерская платформа» аналитической подсистемы «Системы информационной безопасности корпоративной сети передачи данных ООО «Газпром газомоторное топливо»».

БД состоит из семи таблиц (рис. 1), которые содержат информацию о технологических параметрах системы (рис. 2).

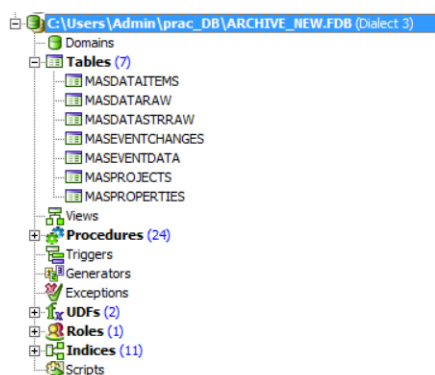


Рис. 1: Структура базы данных

№	Наименование параметра
БВК	
1.	Давление газа на входе
2.	Прямой расход газа на УУГ
3.	Обратный расход газа на УУГ
4.	Суммарный расход газа на УУГ
БКУ (в целом)	
5.	Давление на выходе БКУ, либо давление 4 ой ступени
6.	Наработка по каждому БКУ
БКУ (БАГ)	
7.	Давление в линии высокого давления
8.	Давление в линии среднего давления
9.	Давление в линии низкого давления
Газозаправочная колонка	
10.	Расход газа (по каждой ГЗК)
Аварийные сигналы	
11.	Контроль состояния "Аварийный останов"
12.	Контроль "Загазованность" в компрессорной
13.	Контроль "Состояние ПОС "
14.	Контроль "Пожар"
15.	Контроль "Охрана"

Рис. 2: Перечень передаваемых параметров

2.1 ER-диаграмма

Для более точного описания структуры базы данных необходимо было построить ER-диаграмму. Для этого был использован инструмент построения схем баз данных DBSchema. Однако, поскольку в базе данных отсутствуют внешние ключи (foreign keys), установление связей между таблицами не представилось возможным ни автоматическими средствами, ни вручную. На рис. 3 представлена схема таблиц базы данных без связей между ними.

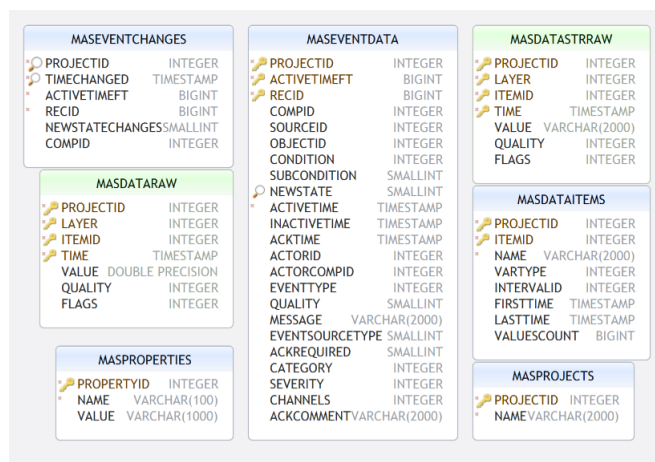
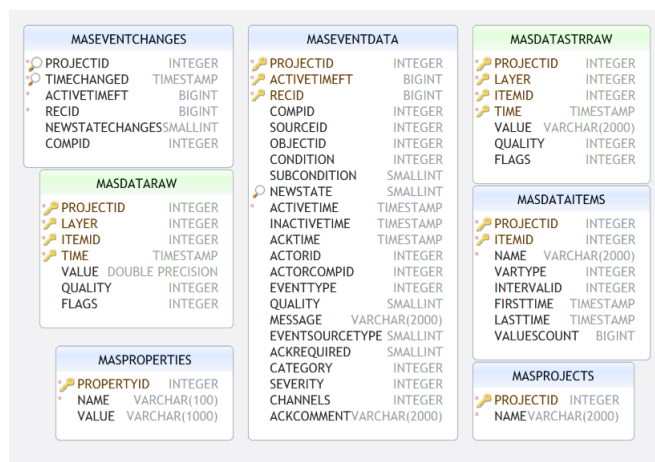
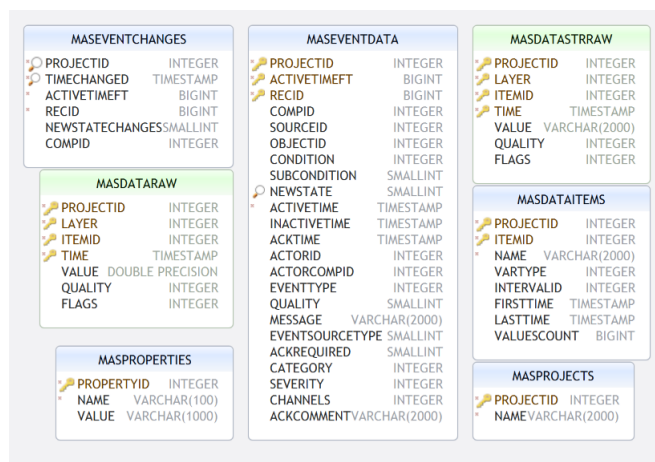
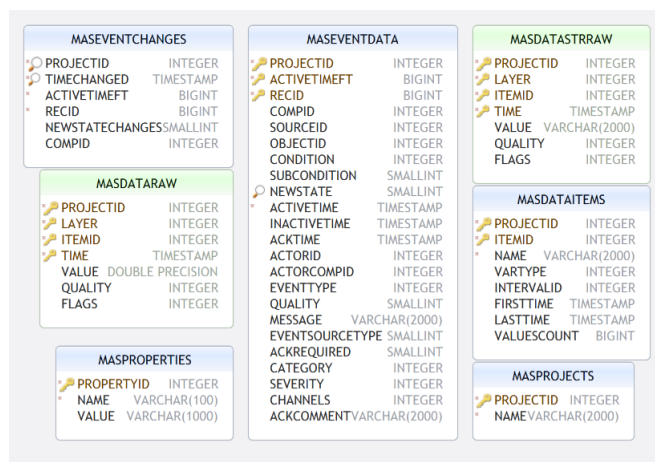
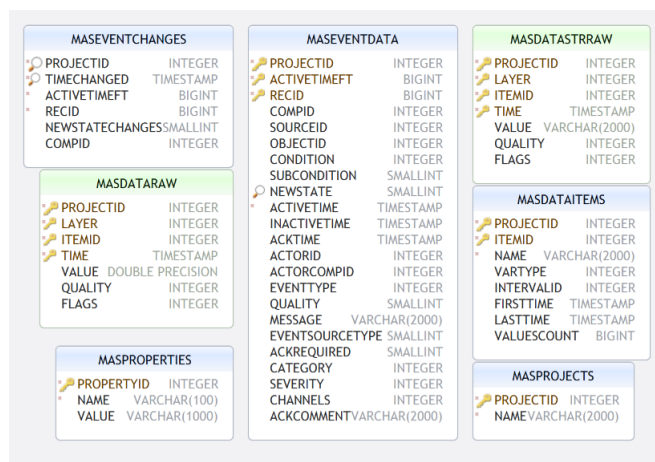
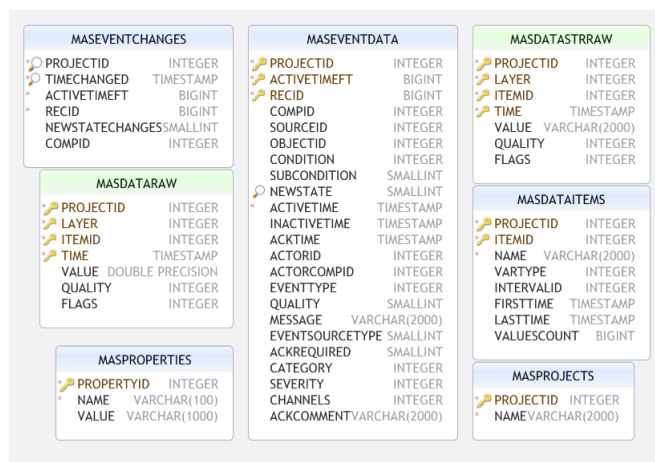


Рис. 3: Схема таблиц

Внешние ключи были определены вручную в соответствии с логикой организации базы данных, что позволило построить ER-диаграмму (рис. 4).

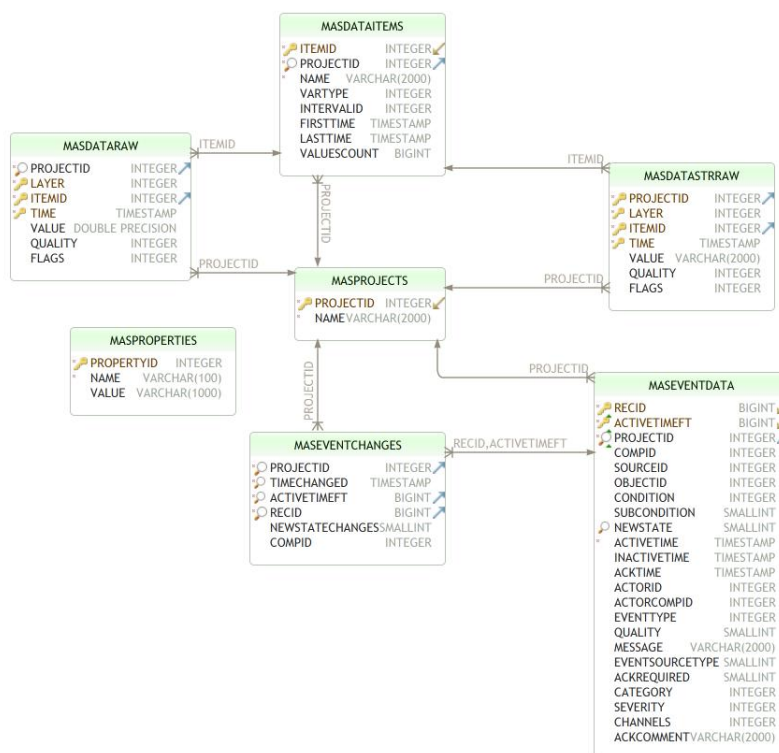


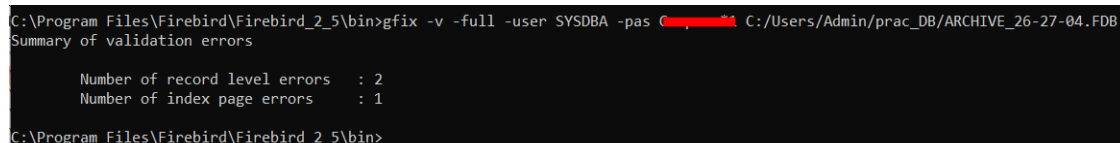
Рис. 4: ER-диаграмма

2.2 Восстановление целостности базы данных

В процессе ознакомления с базой было обнаружено, что целостность данных в одной из таблиц - MASDATAITEMS - нарушена. Проверка на наличие повреждений производилась при помощи следующей команды:

```
gfix -v -full -user SYSDBA -pas *** path_to_database
```

После проведения проверки был получен следующий результат (рис. 5), в связи с чем возникла необходимость в устранении повреждений.



```
C:\Program Files\Firebird\Firebird_2_5\bin>gfix -v -full -user SYSDBA -pas C:\Users\Admin\prac_DB\ARCHIVE_26-27-04.FDB C:/Users/Admin/prac_DB/ARCHIVE_26-27-04.FDB
Summary of validation errors

      Number of record level errors   : 2
      Number of index page errors     : 1

C:\Program Files\Firebird\Firebird_2_5\bin>
```

Рис. 5: Результат проверки целостности

Для этого информация была записана в ВАК-файл, а после восстановлена в новой базе данных посредством следующих команд:

```
gbak -b -v -ig -g -user SYSDBA -pas *** path_to_database backup.gbk
gbak -c -v -user SYSDBA -pas *** backup.gbk new_database.fdb
```

После проведения этой операции новая база данных была проверена на предмет нарушений целостности, однако никаких повреждений обнаружено не было.

3 Задание 1. Подключение к БД с заданным периодом и считывание новых записей

Первое задание заключалось в написании программы, которая позволила бы производить подключение к БД раз в заданный интервал времени и считывать новые записи, произошедшие за это время.

3.1 Программная реализация

Программа реализована на языке программирования Python, принимает на вход аргументы командной строки и возвращает файлы в формате csv, в которых содержатся новые записи, относящиеся к каждому периоду.

Программа состоит из трёх файлов:

- **main.py**
- **Job.py**
- **Parser.py**

Подробное описание этих файлов представлено ниже.

3.1.1 main.py

В файле main.py создается объект класса Job, а также задается период запуска программы считывания новых записей посредством модуля schedule.

```
import schedule
from Job import Job
from Parser import parse
from datetime import datetime
import pause
import re

args = parse()

job = Job(
    args.database,
    args.user,
    args.password
)

if re.fullmatch('([0-1][0-9]|2[0-3]):[0-5][0-9]:[0-5][0-9]', args.start):
    pause.until(
        datetime(datetime.today().year,
                  datetime.today().month,
                  datetime.today().day,
                  int(args.start[0:2]),
                  int(args.start[3:5]),
                  int(args.start[6:8]))
    )
else:
    print('Start format invalid, assuming current moment')
```

```
schedule.every(args.interval).minutes.do(job.work)
while True:
    if job.connected:
        schedule.run_pending()
    else:
        break
```

3.1.2 Job.py

Файл Job.py содержит описание одноимённого класса, который включает в себя следующие поля:

- con - объект класса Connection, подключенный к базе данных, переданной в параметрах конструктора;
- cur - объект класса Cursor, предназначенный для выполнения SQL-запросов;
- tables - список таблиц базы данных;
- last_time - строка, содержащая время последнего обращения к базе данных.

Кроме того, в классе содержится конструктор и метод work(). Последний посредством SQL-запроса реализует считывание новых записей из тех таблиц базы данных, в которых присутствует столбец с именем «TIME», а затем сохраняет полученные записи в файл формата csv. Название файла формируется из названия рассматриваемой таблицы, а также метки даты и времени его сохранения.

```
from datetime import datetime
from fdb import connect, fbcore

import pandas

class Job:
    def __init__(self, database, user, pwd):

        try:
            self.con = connect(
                database,
                user=user,
                password=pwd,
                charset='UTF8'
            )
            self.connected = True
        except fbcore.DatabaseError:
            print('Invalid credentials, cannot connect to the database')
            self.connected = False
        if self.connected:
            self.cur = self.con.cursor()
            self.cur.execute(
                'select a.RDB$RELATION_NAME from RDB$RELATIONS a where COALESCE(
RDB$SYSTEM_FLAG, 0) = 0 and RDB$RELATION_TYPE = 0'
            )
```



```

        self.tables = []
        for i in self.cur.fetchall():
            self.tables.append(i[0].strip())
        self.last_time = datetime.today().strftime('%Y-%m-%d %H:%M:%S')

    def work(self):
        new_time = datetime.today().strftime('%Y-%m-%d %H:%M:%S')
        for table in self.tables:
            self.cur.execute(
                "select rdb$field_name from rdb$relation_fields where rdb$relation_name = '"
+ table +
                "'"
            )
            columns = []
            for col in self.cur.fetchall():
                columns.append(col[0].strip())

            if 'TIME' in columns:
                self.cur.execute(
                    "select * from " +
                    table +
                    " where \"TIME\" between '" +
                    self.last_time +
                    "' and '" +
                    new_time +
                    "'"
                )

            df = pandas.DataFrame(self.cur.fetchall(), columns=columns)
            df.to_csv(
                table.lower() + '_' + self.last_time.replace(':', '.').
replace(' ', '_') + '.csv',
                index=False,
                encoding='utf-8'
            )
            self.last_time = new_time

```

3.1.3 Parser.py

Файл Parser.py содержит метод parse(), реализующий считывание следующих аргументов командной строки:

- database (--database) - путь к базе данных, с которой предстоит работать;
- user (--user) - имя пользователя для подключения к базе данных;
- password (--password) - пароль для подключения к базе данных;
- interval (--interval) - интервал времени в минутах, с которым необходимо осуществлять подключение к базе данных. Этот параметр опционален, по умолчанию равен 60 минутам;
- start time (--start) - время начала работы программы в формате «hh:mm:ss».

```

from argparse import ArgumentParser
from datetime import datetime

def parse():
    parser = ArgumentParser(description='Database connection')
    parser.add_argument(
        '--database',
        type=str,
        help='path to a FDB database'
    )
    parser.add_argument(
        '--user',
        type=str,
        help='username to access the database'
    )
    parser.add_argument(
        '--password',
        type=str,
        help='password to access the database'
    )
    parser.add_argument(
        '--interval',
        type=int,
        default=60,
        help='observation interval in minutes. Default value equals to 60 (one hour)'
    )
    parser.add_argument(
        '--start',
        type=str,
        default=datetime.today().strftime('%H:%M:%S'),
        help='time in format "hh:mm:ss" to start the program'
    )

    return parser.parse_args()

```

3.2 Результат работы программы

Программа генерирует файлы в формате csv с заданным интервалом. Название каждого файла состоит из двух частей:

- название таблицы, из которой выбраны данные;
- отметка даты и времени сохранения файла.

Пример сгенерированных файлов можно видеть на рис. 6, 7.

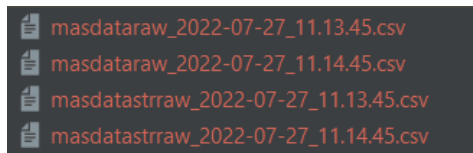


Рис. 6: Список файлов при генерации с интервалом в минуту

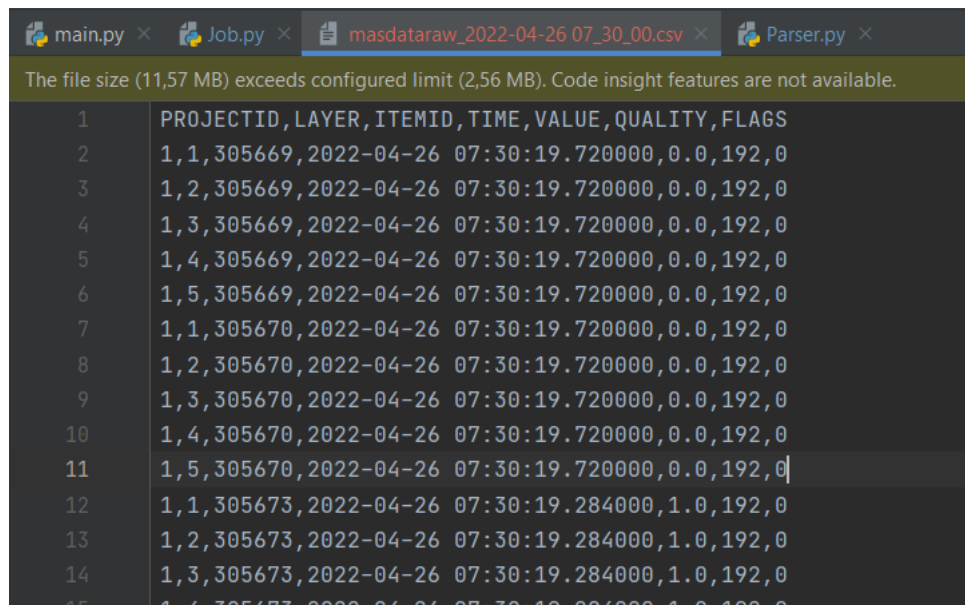


Рис. 7: Пример содержимого файла

4 Задание 2. Статистический анализ технических параметров системы

Второе задание заключалось в том, чтобы по одному или нескольким технологическим параметрам системы провести начальный статистический анализ с целью собрать описательную статистическую информацию: среднее и медианное значение по всем записям в БД, разброс значений, среднеквадратическое отклонение. Кроме того, необходимо было визуализировать эти данные.

4.1 Программная реализация

Программа реализована на языке программирования Python, принимает на вход аргументы командной строки и возвращает один файл в формате txt, содержащий описательную статистику, а также два графика в формате pdf.

Программа состоит из двух файлов:

- **main.py**
- **Parser.py**

Подробное описание этих файлов представлено ниже.

4.1.1 main.py

В файле main.py устанавливается соединение с БД, после которого выполняются следующие два SQL-запроса:

```
select "VALUE", "TIME" from MASDATARAW where ITEMID = itemid
select "NAME" from MASDATAITEMS where ITEMID = itemid
```

, где itemid - значение, переданное в аргументах командной строки. Полученные в первом запросе значения сохраняются в объект pandas.DataFrame, и в дальнейшем для столбца «VALUE» вызываются встроенные методы библиотеки pandas: min(), max(), mean(), median() и std(). Сводка описательной статистики, полученная посредством этих функций, сохраняется в формате txt.

Построение графиков также происходит при помощи методов библиотеки pandas: line() и boxplot(), позволяющие построить линейный график и боксплот соответственно. Таким образом, построенные графики позволяют отследить не только изменение значений во времени, но и выпадающие величины, наибольшие отклонения от медианы. Полученные изображения сохраняются в формате pdf.

```
import pandas
from fdb import connect, fbcore
from Parser import parse
# 308898, 308901

args = parse()

# connection to database
try:
```

```

con = connect(
    args.database,
    user=args.user,
    password=args.password,
    charset='UTF8'
)
except fbcore.DatabaseError:
    print('Invalid credentials, cannot connect to the database')
    con = None

if con:
    cur = con.cursor()
    cur.execute('select "VALUE", "TIME" from MASDATARAW where ITEMID = ' + args.itemid)
    df = pandas.DataFrame(cur.fetchall(), columns=['VALUE', 'TIME'])
    cur.execute('select "NAME" from MASDATAITEMS where ITEMID = ' + args.itemid)
    name = cur.fetchall()[0][0]
    result = ''
    result += 'Item name: ' + name + '\n'
    df.sort_values('TIME')

    # summary statistics
    result += 'Values vary from ' +
str(df['VALUE'].min()) + ' to ' + str(df['VALUE'].max()) + '\n'
    result += 'Mean: ' + str(df['VALUE'].mean()) + '\n'
    result += 'Median: ' + str(df['VALUE'].median()) + '\n'
    result += 'Standard Deviation: ' + str(df['VALUE'].std()) + '\n'
    with open(args.output + '/stat_' + args.itemid + '.txt', 'w') as file:
        file.write(result)

    # figures
    plt = df.boxplot()
    plt.set_title(name)
    plt.get_figure().savefig(args.output + '/fig_boxplot_' + args.itemid + '.pdf')

    plt = df.plot.line(x='TIME', y='VALUE')
    plt.set_xlabel('Value')
    plt.set_ylabel('Time')
    plt.set_title(name)
    plt.get_figure().savefig(args.output + '/fig_line_' + args.itemid + '.pdf')

```

4.1.2 Parser.py

Аналогично заданию 1, файл Parser.py содержит функцию parse(), предназначенную для считывания следующих аргументов командной строки:

- database (--database) - путь к базе данных, с которой предстоит работать;
- user (--user) - имя пользователя для подключения к базе данных;
- password (--password) - пароль для подключения к базе данных;
- item id (--itemid) - номер параметра, для которого требуется собрать информацию;
- output (--output) - путь к директории, предназначенной для сохранения сгенерированных файлов.

```

from argparse import ArgumentParser

def parse():
    parser = ArgumentParser(description='Database connection')
    parser.add_argument(
        '--database',
        type=str,
        help='path to a FDB database'
    )

    parser.add_argument(
        '--user',
        type=str,
        help='username to access the database'
    )

    parser.add_argument(
        '--password',
        type=str,
        help='password to access the database'
    )

    parser.add_argument(
        '--itemid',
        type=str,
        help='item id to check the values'
    )

    parser.add_argument(
        '--output',
        type=str,
        help='output path'
    )

    return parser.parse_args()

```

4.2 Результат работы программы

Программа генерирует три файла (рис. 8):

- текстовый документ с описательной статистикой (рис. 9);
- график типа «боксплот», или диаграмма размаха (рис. 10);
- линейный график (рис. 11).

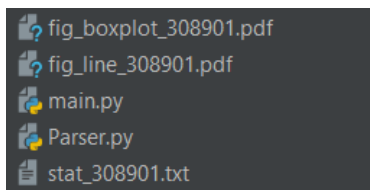


Рис. 8: Список сгенерированных файлов

```
main.py  stat_308901.txt  Parser.py
1 Item name: ССД.InSAT Modbus OPC Server DA.Node1.RS4M.T
2 Values vary from 277.6873474121094 to 286.4774475097656
3 Mean: 279.69395480483564
4 Median: 279.60333251953125
5 Standard Deviation: 1.4433749682186547
6 |
```

Рис. 9: Пример содержимого файла с описательной статистикой

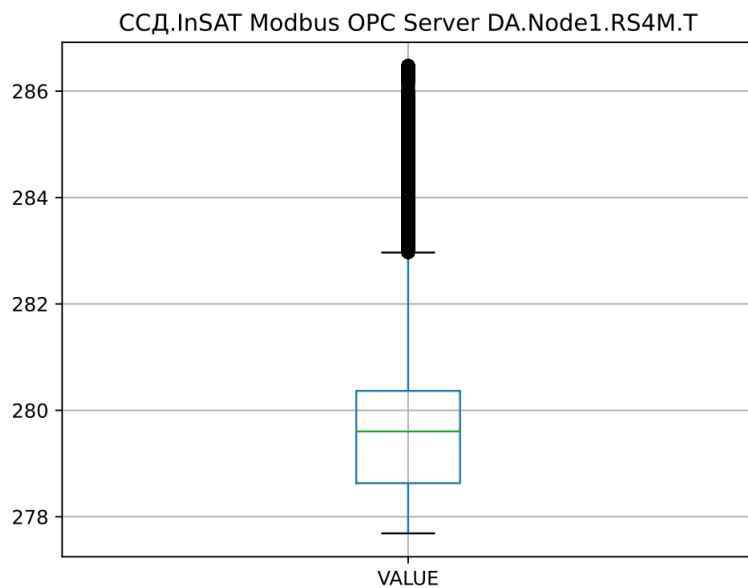


Рис. 10: Пример графика типа «боксплот»

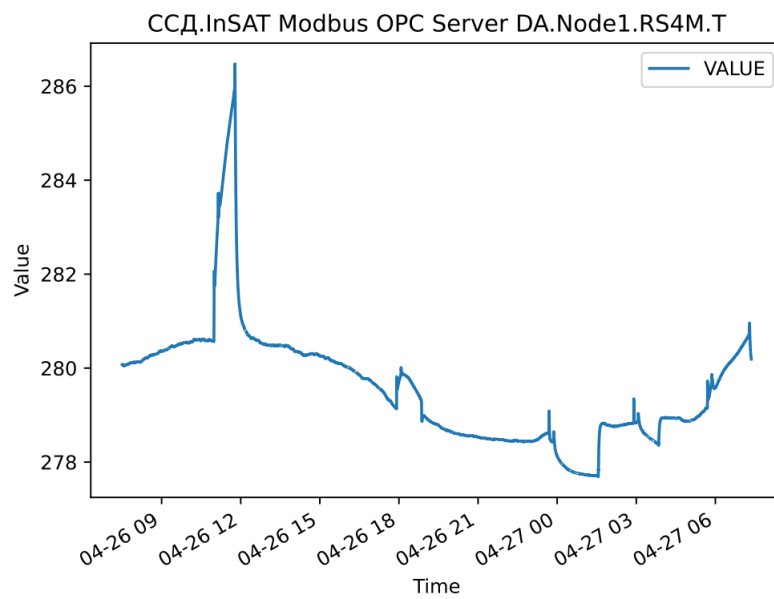


Рис. 11: Пример линейного графика

Заключение

В ходе прохождения практики были освоены такие инструменты, как СУБД Firebird, утилита администрирования баз данных IBEExpert, а также диалект SQL Dialect 3. Был получен опыт работы с базами данных посредством языка программирования Python и изучены специализированные модули, предназначенные для этого. Кроме того, был получен навык восстановления целостности поврежденной базы данных.

В результате прохождения практики были написаны два консольных приложения, реализующих работу с базой данных. Первая программа, принимающая на вход аргументы командной строки, подключается к базе данных с заданным периодом для отслеживания и считывания в формате csv новых записей, появившихся в ней. Вторая программа, также принимающая на вход аргументы командной строки, предназначена для сбора и визуализации описательной статистики по заданному технологическому параметру системы.