

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра МО ЭВМ**

**ОТЧЕТ**  
**по лабораторной работе №3**  
**по дисциплине «Базы данных»**  
**Тема: Реализация базы данных с использованием ORM**

Студентка гр. 1304

Чернякова В.А.

Преподаватель

Заславский М.М.

Санкт-Петербург

2023

## **Цель работы.**

Создание базы данных с использованием Object-Relational Mapping (ORM).

## **Задание.**

### Вариант 3(25).

В данной лабораторной работе рекомендуется использовать Sequelize (Node.js).

Вы можете использовать другой ORM по вашему выбору по согласованию с преподавателем, принимающим у вас практики.

Необходимо выполнить следующие задачи:

- Описать в виде моделей таблицы из 1-й лабораторной работы.
- Написать скрипт заполнения тестовыми данными: 5-10 строк на каждую таблицу, обязательно наличие связи между ними, данные приближены к реальности.
- Написать запросы к БД, отвечающие на вопросы из 1-й лабораторной работы с использованием **ORM**. Вывести результаты в консоль (или иной человеко-читаемый вывод).
- Запустить в репозиторий исходный код проекта, соблюсти. gitignore, убрать исходную базу из проекта (или иные нагенерированные данные бд если они есть).
- Описать процесс запуска: команды, зависимости.
- В отчете описать цель, текст задания в соответствии с вариантом, выбранную ORM, инструкцию по запуску, скриншоты (код) моделей ORM, скриншоты на каждый запрос (или группу запросов) на изменение/таблицы с выводом результатов (ответ), ссылку на PR в приложении, вывод.

## **Выполнение работы.**

### 1. Выбор ORM

Так как использование *Sequelize* является лишь рекомендацией, была выбрана другая *ORM* – *GORM*. *GORM* (*Go Object-Relational Mapper*) – это *ORM* библиотека для *Go*, которая предлагает простой и удобный способ взаимодействия с базами данных. *GORM* поддерживает различные базы данных *SQL*, включая *PostgreSQL*, *MySQL*, *SQLite* и *Microsoft SQL Server*.

## 2. Установка

В *IDE GoLand* была установлена библиотека *ORM* для *Golang* с помощью следующих команд:

```
go get -u gorm.io/gorm
go get -u gorm.io/driver/postgres
```

Вторая команда используется в *Go* для установки или обновления зависимости, в данном случае драйвера базы данных *postgresql*, используемого с *GORM*.

## 3. Подключение к базе данных

Подключение к базе данных *PostgreSQL*.

```
dsn      :=      "host=localhost      user=postgres      password=LeRa2003
dbname=postgres port=5432"
```

В этой строке определена строка подключения (*Data Source Name, DSN*) для базы данных *PostgreSQL*. *DSN* содержит информацию о том, как подключиться к базе данных, включая хост (*localhost*), имя пользователя (*postgres*), пароль (*LeRa2003*), имя базы данных (*postgres*) и порт (*5432*).

```
db, err := gorm.Open(postgres.Open(dsn), &gorm.Config{})
```

В этой строке выполняется попытка подключения к базе данных с использованием *GORM* и драйвера *PostgreSQL*. Функция *gorm.Open* принимает два аргумента. Первый аргумент *postgres.Open(dsn)* указывает *GORM* использовать драйвер *PostgreSQL* и передает *DSN* для подключения к базе данных. Второй аргумент *&gorm.Config{}* представляет конфигурацию *GORM* (в данном случае, конфигурация не определена, и используются значения по умолчанию). Результат этой операции, то есть подключенная база данных, сохраняется в переменной *db*, и любая ошибка сохраняется в переменной *err*.

## 4. Создание моделей

На основе структуры базы данных, спроектированной в лабораторной работе 1, были созданы соответствующие модели.

Модели – тоже самое что в *PostgreSQL* таблицы.

Модели представляют собой обычные структуры с базовыми типами *Go*, их указателями/псевдонимами или пользовательскими типами.

Основная структура модели:

- Название структуры – название модели.
- Столбцы содержат: название поля, тип данных, теги *GORM*.

Используемые типы данных в рамках лабораторной работы:

- *uint* – беззнаковое целое число.
- *string* – строка.
- *int* – целое число.

Используемые теги *GORM*:

- *primaryKey* – указывает столбец в качестве первичного ключа.
- *autoIncrement:false* или *autoIncrement:true* – запрещает или задает автоматический инкрементный столбец.

- *size* – масштаб столбца.
- *not null* – задает столбцу значение *NOT NULL*.
- *foreignKey* – указывает столбец в качестве внешнего ключа.
- *constraint:OnDelete:CASCADE* – установка ограничения.
- *default:null* – указывает значение столбца по умолчанию.

Ассоциации:

- Соединение «один ко многим».

Модель *Class*.

```
Students[]Student`gorm:"foreignKey:ClassNumber,ClassLetter;constraint:OnDelete:CASCADE" `
```

Данная модель будет связана с моделью *Student* по полям *ClassNumber*, *ClassLetter*.

```
Schedules[]Schedule`gorm:"foreignKey:ClassNumber,ClassLetter;constraint:OnDelete:CASCADE" `
```

Данная модель будет связана с моделью *Schedule* по полям *ClassNumber*, *ClassLetter*.

Модель *Classroom*.

```
Schedules[]Schedule`gorm:"foreignKey:ClassroomNumber;constraint:OnDelete:CASCADE" `
```

Данная модель будет связана с моделью *Schedule* по полю *ClassroomNumber*.

Модель *Student*.

```
Grades[]Grade`gorm:"foreignKey:StudentId;constraint:onDelete:CASCADE" `
```

Данная модель будет связана с моделью *Grades* по полю *StudentId*.

Модель *Subject*.

```
TeacherSubjects[]TeacherSubject`gorm:"foreignKey:SubjectId;constraint:onDelete:CASCADE" `
```

Данная модель будет связана с моделью *TeacherSubject* по полю *SubjectId*.

```
Grades[]Grade`gorm:"foreignKey:SubjectId;constraint:onDelete:CASCADE" `
```

Данная модель будет связана с моделью *Grade* по полю *SubjectId*.

```
Schedules[]Schedule`gorm:"foreignKey:SubjectId;constraint:onDelete:CASCADE" `
```

Данная модель будет связана с моделью *Schedule* по полю *SubjectId*.

Модель *Teacher*.

```
TeacherSubjects[]TeacherSubject`gorm:"foreignKey:TeacherId;constraint:onDelete:CASCADE" `
```

Данная модель будет связана с моделью *TeacherSubject* по полю *TeacherId*.

```
Schedules[]Schedule`gorm:"foreignKey:TeacherId;constraint:onDelete:CASCADE" `
```

Данная модель будет связана с моделью *Schedule* по полю *TeacherId*.

- Соединение «один к одному».

Модель *Classroom*.

```
TeacherClassroom[]TeacherClassroom`gorm:"foreignKey:ClassroomNumber;constraint:onDelete:CASCADE" `
```

Данная модель будет связана с моделью *TeacherClassroom* по полю *ClassroomNumber*.

#### Модель *Teacher*.

```
TeacherClassroom[]TeacherClassroom`gorm:"foreignKey:TeacherId;constraint:OnDelete:CASCADE"``
```

Данная модель будет связана с моделью *TeacherClassroom* по полю *TeacherId*.

На рисунках 1 – 9 представлены описания каждой из моделей.

```
package models

type Class struct { 2 usages
    ClassNumber uint `gorm:"primaryKey;autoIncrement:false"``
    ClassLetter string `gorm:"primaryKey;size:2;not null"``
    Students []Student `gorm:"foreignKey:ClassNumber,ClassLetter;constraint:OnDelete:CASCADE"``
    Schedules []Schedule `gorm:"foreignKey:ClassNumber,ClassLetter;constraint:OnDelete:CASCADE"``
}
```

Рисунок 1 – описание модели *class*.

```
package models

type Classroom struct { 2 usages
    ClassroomNumber uint `gorm:"primaryKey;autoIncrement:false"``
    TeacherClassroom []TeacherClassroom `gorm:"foreignKey:ClassroomNumber;constraint:OnDelete:CASCADE"``
    Schedules []Schedule `gorm:"foreignKey:ClassroomNumber;constraint:OnDelete:CASCADE"``
}
```

Рисунок 2 – описание модели *classroom*.

```
package models

type Grade struct { 4 usages
    GradeId uint `gorm:"primaryKey;autoIncrement:true"``
    StudentId uint `gorm:"not null"``
    QuaterNumber uint `gorm:"not null"``
    SubjectId uint `gorm:"not null"``
    Mark uint `gorm:"not null"``
}
```

Рисунок 3 – описание модели *grade*.

```
package models

type Schedule struct { 8 usages
    ScheduleId      uint    `gorm:"primaryKey;autoIncrement:true"`
    DayName          string  `gorm:"size:30;not null"`
    LessonNumber     uint    `gorm:"not null"`
    SubjectId        uint    `gorm:"not null"`
    TeacherId        uint    `gorm:"not null"`
    ClassroomNumber  uint    `gorm:"not null"`
    ClassNumber      uint    `gorm:"not null"`
    ClassLetter      string  `gorm:"size:2;not null"`
}
```

Рисунок 4 – описание модели *schedule*.

```
package models

type Student struct { 3 usages
    StudentId      uint    `gorm:"primaryKey;autoIncrement:true"`
    StudentName     string  `gorm:"size:50;not null"`
    StudentSurname  string  `gorm:"size:50;not null"`
    ClassNumber     int     `gorm:"not null"`
    ClassLetter     string  `gorm:"size:2;not null"`
    Grades          []Grade `gorm:"foreignKey:StudentId;constraint:OnDelete:CASCADE"`
}
```

Рисунок 5 – описание модели *student*.

```
package models

type Subject struct { 3 usages
    SubjectId      uint    `gorm:"primaryKey;autoIncrement:true"`
    SubjectName     string  `gorm:"size:50;not null"`
    TeacherSubjects []TeacherSubject `gorm:"foreignKey:SubjectId;constraint:OnDelete:CASCADE"`
    Grades          []Grade `gorm:"foreignKey:SubjectId;constraint:OnDelete:CASCADE"`
    Schedules       []Schedule `gorm:"foreignKey:SubjectId;constraint:OnDelete:CASCADE"`
}
```

Рисунок 6 – описание модели *subject*.

```
package models

type Teacher struct { 3 usages
    TeacherId      uint    `gorm:"primaryKey;autoIncrement:true"`
    TeacherName     string  `gorm:"size:50;not null"`
    TeacherSurname  string  `gorm:"size:50;not null"`
    TeacherPatronymic string `gorm:"size:50;default:null"`
    TeacherSubjects []TeacherSubject `gorm:"foreignKey:TeacherId;constraint:OnDelete:CASCADE"`
    TeacherClassroom []TeacherClassroom `gorm:"foreignKey:TeacherId;constraint:OnDelete:CASCADE"`
    Schedules       []Schedule `gorm:"foreignKey:TeacherId;constraint:OnDelete:CASCADE"`
}
```

Рисунок 7 – описание модели *teacher*.

```
package models

type TeacherClassroom struct { 4 usages
    TeacherId      uint `gorm:"primaryKey;autoIncrement:false"`
    ClassroomNumber uint `gorm:"primaryKey;autoIncrement:false"`
} =
```

Рисунок 8 – описание модели *teacherClassroom*.

```
package models

type TeacherSubject struct { 4 usages
    TeacherId uint `gorm:"primaryKey;autoIncrement:false"`
    SubjectId uint `gorm:"primaryKey;autoIncrement:false"`
} =
```

Рисунок 9 – описание модели *teacherSubject*.

## 5. Создание таблицы.

```
db.AutoMigrate(
    &models.Classroom{},
    &models.Teacher{},
    &models.Subject{},
    &models.TeacherClassroom{},
    &models.TeacherSubject{},
    &models.Class{},
    &models.Student{},
    &models.Grade{},
    &models.Schedule{},
)
```

В предоставленном коде используется функция *AutoMigrate* из библиотеки *GORM* для автоматического создания (или обновления) таблиц в базе данных, которые соответствуют структурам данных, перечисленным в качестве аргументов функции. Эта функция создает таблицы, если их еще нет, или обновляет их, если они уже существуют, чтобы они соответствовали описанным структурам данных. Функция *AutoMigrate* анализирует структуры данных и создает таблицы в базе данных с соответствующими полями и ограничениями, как они определены в структурах.



После запуска программы в *IDE DataGrip* можно отследить создание таблиц и соответствующих полей. На рисунках 10 – 18 представлены созданные таблицы.

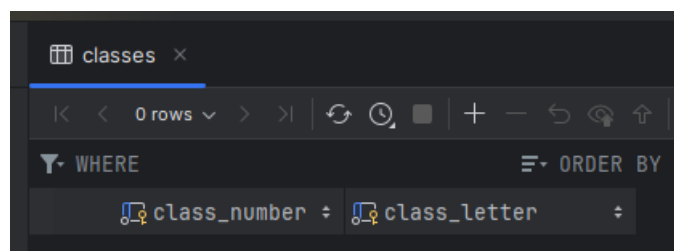


Рисунок 10 – таблица *classes*.

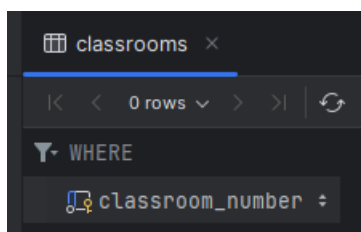


Рисунок 11 – таблица *classrooms*.

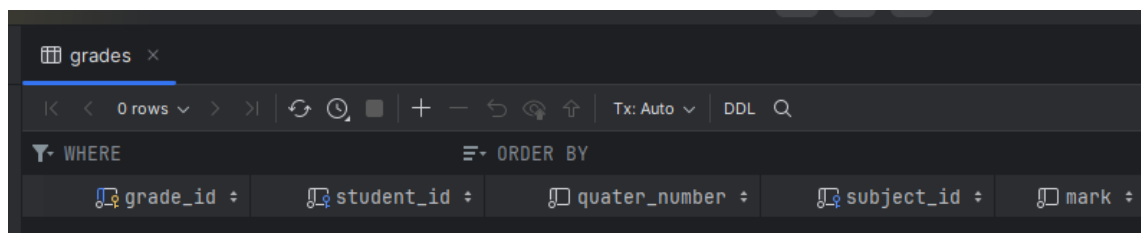


Рисунок 12 – таблица *grades*.

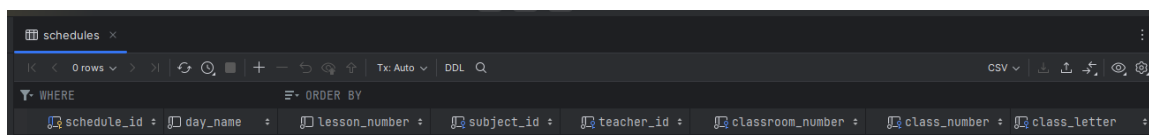


Рисунок 13 – таблица *schedules*.

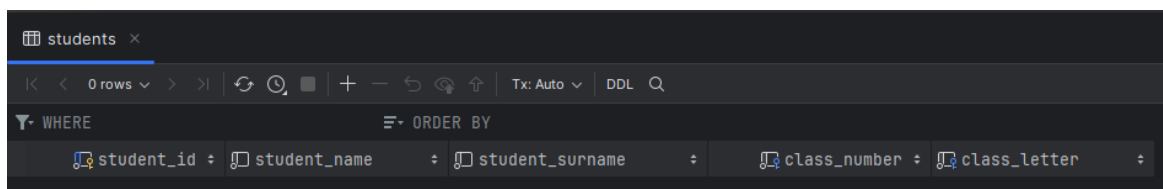


Рисунок 14 – таблица *students*.

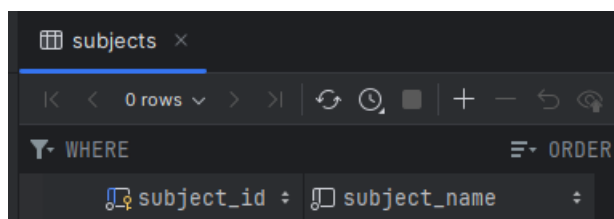


Рисунок 15 – таблица *subjects*.

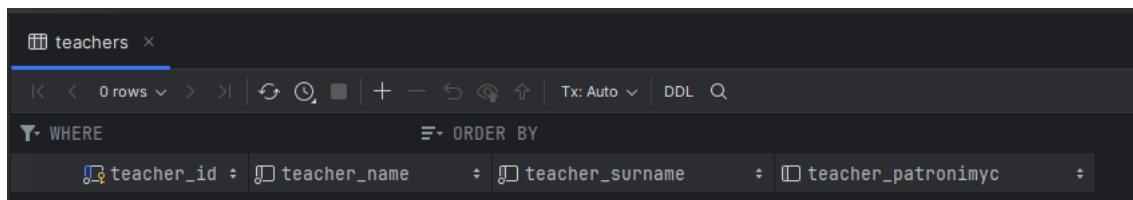


Рисунок 16 – таблица *teachers*.

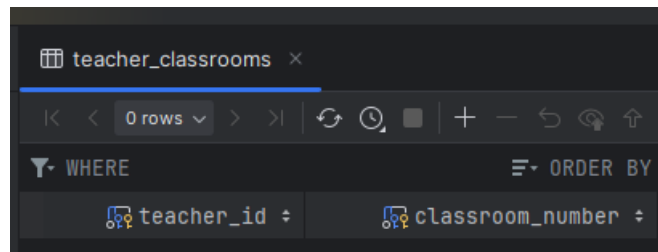


Рисунок 17 – таблица *teacher\_classrooms*.

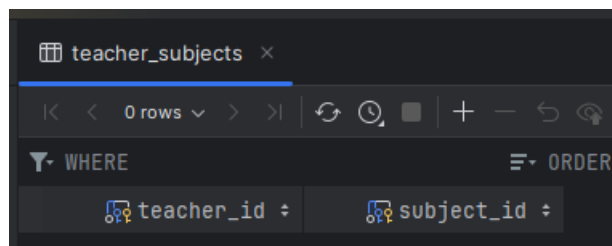


Рисунок 18 – таблица *teacher\_subjects*.

## 6. Добавление записей.

Для добавления записей в базу данных создавались переменные, являющиеся списком моделей и их конкретных полей со значениями, описанных ранее.

На рисунках 19 – 27 представлены такие переменные с тестовыми данными.

```
classes := []models.Class{
    {ClassNumber: 5, ClassLetter: "A"},
    {ClassNumber: 5, ClassLetter: "Б"},
    {ClassNumber: 6, ClassLetter: "Б"},
    {ClassNumber: 6, ClassLetter: "Г"},
    {ClassNumber: 7, ClassLetter: "В"},
    {ClassNumber: 7, ClassLetter: "Д"},
    {ClassNumber: 8, ClassLetter: "А"},
    {ClassNumber: 9, ClassLetter: "А"},
}
```

Рисунок 19 – тестовые данные для *Class*.

```

classrooms := []models.Classroom{
    {ClassroomNumber: 101},
    {ClassroomNumber: 205},
    {ClassroomNumber: 220},
    {ClassroomNumber: 135},
    {ClassroomNumber: 310},
    {ClassroomNumber: 222},
    {ClassroomNumber: 112},
    {ClassroomNumber: 300},
    {ClassroomNumber: 315},
}

```

Рисунок 20 – тестовые данные для *Classroom*.

```

subjects := []models.Subject{
    {SubjectName: "Математика"},
    {SubjectName: "Русский язык"},
    {SubjectName: "История"},
    {SubjectName: "Физика"},
    {SubjectName: "Биология"},
    {SubjectName: "Химия"},
    {SubjectName: "Иностранный язык"},
    {SubjectName: "Литература"},
    {SubjectName: "География"},
    {SubjectName: "Искусство"},
}

```

Рисунок 21 – тестовые данные для *Subject*.

```

teachers := []models.Teacher{
    {TeacherName: "Иванов", TeacherSurname: "Александр", TeacherPatronymic: "Петрович"},
    {TeacherName: "Смирнова", TeacherSurname: "Мария", TeacherPatronymic: "Игоревна"},
    {TeacherName: "Петров", TeacherSurname: "Андрей", TeacherPatronymic: "Сергеевич"},
    {TeacherName: "Козлов", TeacherSurname: "Елена", TeacherPatronymic: "Владимировна"},
    {TeacherName: "Соколов", TeacherSurname: "Игорь", TeacherPatronymic: "Анатолевич"},
    {TeacherName: "Волкова", TeacherSurname: "Наталья", TeacherPatronymic: "Александровна"},
    {TeacherName: "Михайлов", TeacherSurname: "Артем", TeacherPatronymic: "Дмитриевич"},
    {TeacherName: "Передерий", TeacherSurname: "Ольга"},
}

```

Рисунок 22 – тестовые данные для *Teacher*.



```

students := []models.Student{
    {StudentName: "Басалаев", StudentSurname: "Леонид", ClassNumber: 5, ClassLetter: "А"},
    {StudentName: "Кудашкина", StudentSurname: "Анастасия", ClassNumber: 5, ClassLetter: "А"},
    {StudentName: "Карасев", StudentSurname: "Алексей", ClassNumber: 5, ClassLetter: "Б"},
    {StudentName: "Максимилкина", StudentSurname: "Софья", ClassNumber: 5, ClassLetter: "Б"},
    {StudentName: "Мангутов", StudentSurname: "Тимур", ClassNumber: 6, ClassLetter: "Б"},
    {StudentName: "Аракчеева", StudentSurname: "Арина", ClassNumber: 6, ClassLetter: "Г"},
    {StudentName: "Чернякова", StudentSurname: "Валерия", ClassNumber: 7, ClassLetter: "В"},
    {StudentName: "Дубровин", StudentSurname: "Игорь", ClassNumber: 7, ClassLetter: "Д"},
    {StudentName: "Никитин", StudentSurname: "Никита", ClassNumber: 8, ClassLetter: "А"},
    {StudentName: "Бомштейн", StudentSurname: "Юлия", ClassNumber: 9, ClassLetter: "А"},
}

```

Рисунок 25 – тестовые данные для *Student*.

```

teachersClassrooms := []models.TeacherClassroom{
    {TeacherId: 7, ClassroomNumber: 101},
    {TeacherId: 2, ClassroomNumber: 205},
    {TeacherId: 5, ClassroomNumber: 220},
    {TeacherId: 8, ClassroomNumber: 135},
    {TeacherId: 1, ClassroomNumber: 310},
    {TeacherId: 3, ClassroomNumber: 112},
    {TeacherId: 6, ClassroomNumber: 300},
}

```

Рисунок 26 – тестовые данные для *TeacherClassroom*.

```

teachersSubjects := []models.TeacherSubject{
    {TeacherId: 2, SubjectId: 1},
    {TeacherId: 5, SubjectId: 1},
    {TeacherId: 7, SubjectId: 2},
    {TeacherId: 1, SubjectId: 2},
    {TeacherId: 3, SubjectId: 3},
    {TeacherId: 1, SubjectId: 4},
    {TeacherId: 4, SubjectId: 5},
    {TeacherId: 4, SubjectId: 6},
    {TeacherId: 7, SubjectId: 7},
    {TeacherId: 1, SubjectId: 8},
    {TeacherId: 8, SubjectId: 9},
    {TeacherId: 6, SubjectId: 10},
}

```

Рисунок 27 – тестовые данные для *TeacherSubject*.

```

createRecords := func(data interface{}) {
    result := db.Create(data)
    if result.Error != nil {
        fmt.Print("Error during adding tuple")
    }
}

```

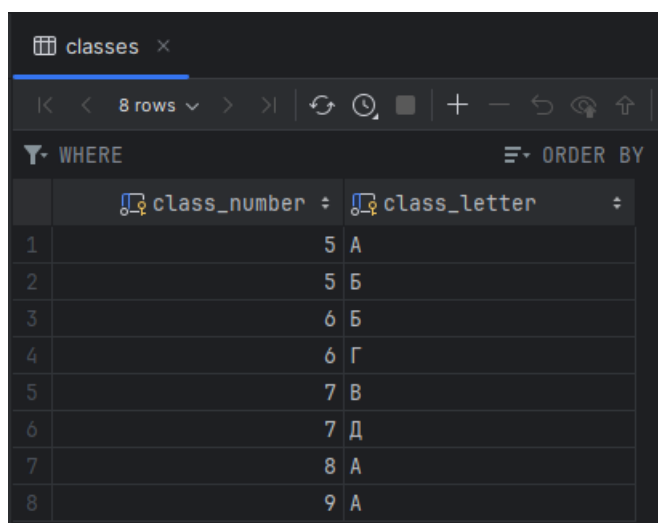
```

    }
}

```

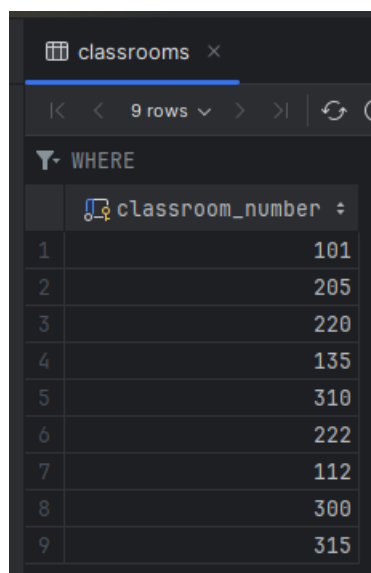
Данная функция принимает входные данные в виде интерфейса. Эта функция использует переданные данные для создания записей в базе данных с помощью метода *Create* объекта *db*, который является экземпляром *GORM* для взаимодействия с базой данных. Если при выполнении *Create* возникает ошибка, она выводит сообщение об ошибке. Далее эта функция вызывается от переменных, которые содержат необходимые данные для добавления.

На рисунках 28 – 36 изображены итоговые таблицы вместе с данными.



	class_number	class_letter
1	5	A
2	5	Б
3	6	Б
4	6	Г
5	7	В
6	7	Д
7	8	А
8	9	А

Рисунок 28 – таблица *classes* с данными.



	classroom_number
1	101
2	205
3	220
4	135
5	310
6	222
7	112
8	300
9	315

Рисунок 29 – таблица *classrooms* с данными.

	grade_id	student_id	quarter_number	subject_id	mark
1	1	1	1	2	4
2	2	2	1	10	5
3	3	3	1	5	3
4	4	4	1	3	3
5	5	5	1	1	4
6	6	6	1	9	4
7	7	7	1	6	5
8	8	8	1	4	5
9	9	9	1	7	5
10	10	10	1	8	3

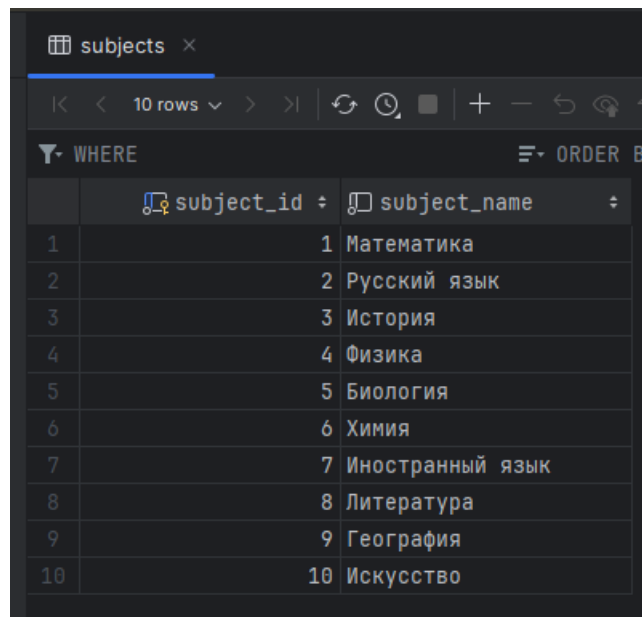
Рисунок 30 – таблица *grades* с данными.

	schedule_id	day_name	lesson_number	subject_id	teacher_id	classroom_number	class_number	class_letter
1	1	Понедельник	1	2	1	310	5	A
2	2	Понедельник	2	1	5	220	5	A
3	3	Понедельник	3	3	3	112	5	B
4	4	Понедельник	4	5	4	220	5	B
5	5	Вторник	1	7	7	101	6	B
6	6	Вторник	2	1	2	205	6	B
7	7	Вторник	5	9	8	135	6	Г
8	8	Вторник	6	8	1	310	6	Г
9	9	Среда	5	4	1	310	7	B
10	10	Среда	6	10	6	300	7	B
11	11	Среда	3	2	1	310	7	Д
12	12	Среда	4	5	4	315	7	Д
13	13	Четверг	1	6	4	135	8	A
14	14	Четверг	2	3	3	112	8	A
15	15	Пятница	1	9	8	135	9	A
16	16	Пятница	2	4	1	310	9	A

Рисунок 31 – таблица *schedules* с данными.

	student_id	student_name	student_surname	class_number	class_letter
1	1	Басалаев	Леонид	5	A
2	2	Кудашкина	Анастасия	5	A
3	3	Карасев	Алексей	5	B
4	4	Максимлчкина	Софья	5	B
5	5	Мангутов	Тимур	6	B
6	6	Аракчеева	Арина	6	Г
7	7	Чернякова	Валерия	7	B
8	8	Дубровин	Игорь	7	Д
9	9	Никитин	Никита	8	A
10	10	Бомштейн	Юлия	9	A

Рисунок 32 – таблица *students* с данными.

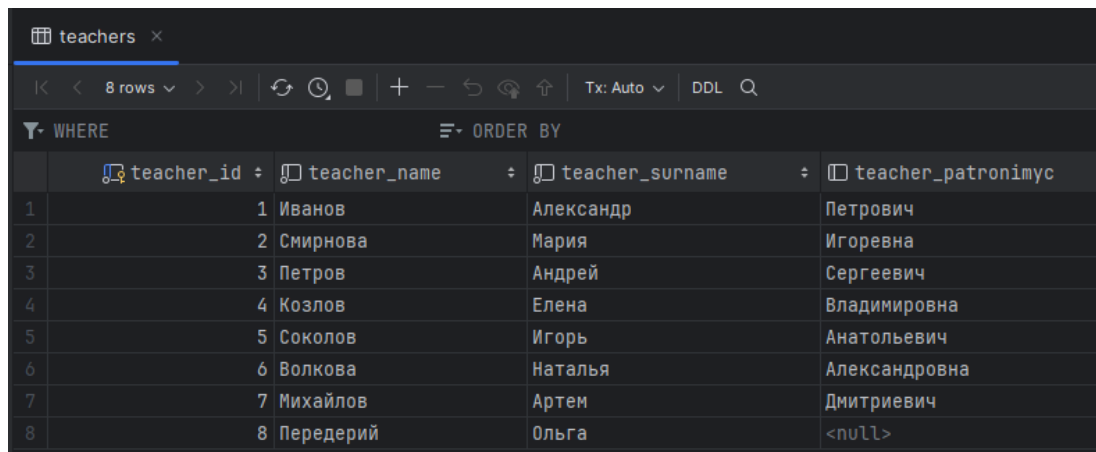


subjects

WHERE ORDER BY

	subject_id	subject_name
1	1	Математика
2	2	Русский язык
3	3	История
4	4	Физика
5	5	Биология
6	6	Химия
7	7	Иностранный язык
8	8	Литература
9	9	География
10	10	Искусство

Рисунок 33 – таблица *subjects* с данными.

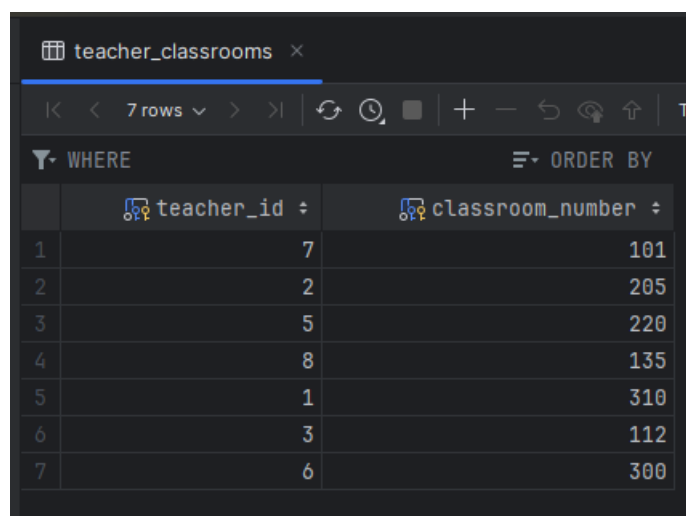


teachers

WHERE ORDER BY

	teacher_id	teacher_name	teacher_surname	teacher_patronym
1	1	Иванов	Александр	Петрович
2	2	Смирнова	Мария	Игоревна
3	3	Петров	Андрей	Сергеевич
4	4	Козлов	Елена	Владимировна
5	5	Соколов	Игорь	Анатольевич
6	6	Волкова	Наталья	Александровна
7	7	Михайлов	Артем	Дмитриевич
8	8	Передерий	Ольга	<null>

Рисунок 34 – таблица *teachers* с данными.



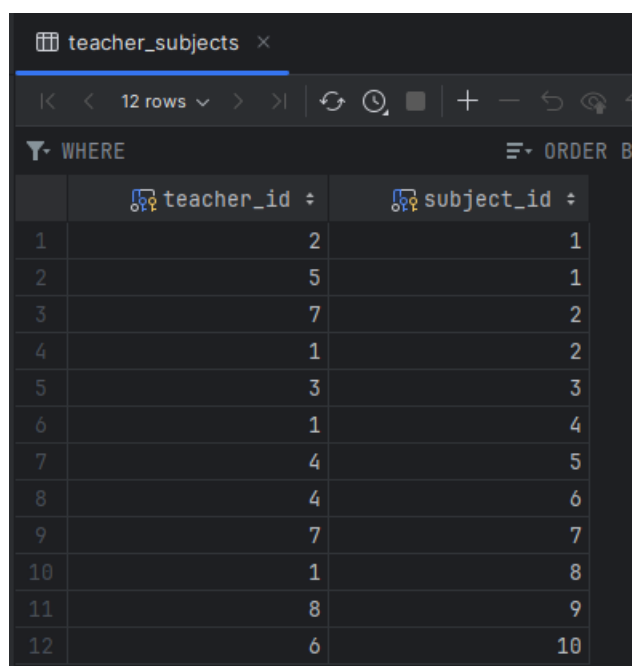
teacher\_classrooms

WHERE ORDER BY

	teacher_id	classroom_number
1	7	101
2	2	205
3	5	220
4	8	135
5	1	310
6	3	112
7	6	300

Рисунок 35 – таблица *teacher\_classrooms* с данными.





	teacher_id	subject_id
1	2	1
2	5	1
3	7	2
4	1	2
5	3	3
6	1	4
7	4	5
8	4	6
9	7	7
10	1	8
11	8	9
12	6	10

Рисунок 36 – таблица *teacher\_subjects* с данными.

7. Написание запросов к БД, отвечающих на вопросы из первой лабораторной работы.

Для каждого запроса была написана своя функция, которая принимает в качестве параметров указатель на объект *GORM*, который представляет собой соединение с базой данных и данные, по которым необходимо сделать выборку.

Методы *GORM*, используемые для реализации запросов:

- *db.Table("...")*: Этот метод *GORM* устанавливает таблицу как источник данных для запроса.
- *db.Select(...)*: Здесь указываются столбцы, которые должны быть выбраны в результате запроса.
- *db.Joins(«INNER JOIN таблица ON по каким полям»)*: Этот метод *GORM* выполняет объединение (*join*) таблиц с использованием *PostgreSQL INNER JOIN*. Он соединяет записи в обеих таблицах, где соотносятся указываемые значения полей.
- *db.Where(...)*: Этот метод *GORM* добавляет условия для выборки данных.
- *db.Count(...)*: Этот метод *GORM* выполняет запрос к базе данных и выполняет подсчет количества записей, удовлетворяющих условиям.

• `db.Find(&result)`: Этот метод *GORM* выполняет запрос к базе данных, который соответствует условиям и полученные записи сохраняются в переменную `result`, которая представляет собой срез (список) структур определенный в зависимости от функции.

Вопрос 1. Какой предмет будет в заданном классе, в заданный день недели на заданном уроке?

Функция с запросом представлена на рисунке 37.

```
func GetSubjectByClassDayNumber( 1 usage
    db *gorm.DB,
    classNumber int,
    classLetter string,
    lessonNumber int,
    dayName string) (result []models.Subject) {
    db.
        Joins(
            query: "INNER JOIN schedules ON schedules.subject_id = subjects.subject_id",
        ).
        Where(
            query: "class_number = ? AND class_letter = ? AND day_name = ? AND lesson_number = ?",
            classNumber, classLetter, dayName, lessonNumber,
        ).
        Find(&result)
    return
}
```

Рисунок 37 – функция с запросом по вопросу 1.

На рисунке 38 представлен результат запроса по вопросу 1.

```
182
183     fmt.Println(a... "Какой предмет будет в заданном классе, в заданный день недели на заданном уроке?")
184     for _, subject := range requests.GetSubjectByClassDayNumber(
185         db, classNumber: 9, classLetter: "A", lessonNumber: 2, dayName: "Пятница") {
186         fmt.Println(subject.SubjectName)
187     }
main()

Run go build lab3 x
Какой предмет будет в заданном классе, в заданный день недели на заданном уроке?
Физика
```

Рисунок 38 – результат запроса по вопросу 1.

Вопрос 2. Кто из учителей преподает в заданном классе?

Функция с запросом представлена на рисунке 39.

```
func GetClassTeachers(db *gorm.DB, classNumber int, classLetter string) (result []models.Teacher) {
    db.
        Joins(
            query: "INNER JOIN schedules ON schedules.teacher_id = teachers.teacher_id",
        ).
        Where(
            query: "class_number = ? AND class_letter = ?",
            classNumber, classLetter,
        ).
        Find(&result)
    return
}
```

Рисунок 39 – функция с запросом по вопросу 2.

На рисунке 40 представлен результат запроса по вопросу 2.

```
189     fmt.Println(a...: "Кто из учителей преподает в заданном классе?")
190     for _, teacher := range requests.GetClassTeachers(db, classNumber: 5, classLetter: "A") {
191         fmt.Println(teacher.TeacherName, teacher.TeacherSurname, teacher.TeacherPatronymic)
192     }
193
main()

Run go build lab3 x

Кто из учителей преподает в заданном классе?
Иванов Александр Петрович
Соколов Игорь Анатольевич
```

Рисунок 40 – результат запроса по вопросу 2.

Вопрос 3. В каком кабинете будет 5-й урок в среду у некоторого класса?

Функция с запросом представлена на рисунке 41.

```
func GetClassroom( 1 usage
    db *gorm.DB,
    lessonNumber int,
    dayName string,
    classNumber int,
    ClassLetter string) (result []models.Schedule) {
    db.
        Where(
            query: "lesson_number = ? AND day_name = ? AND class_number = ? AND class_letter = ?",
            lessonNumber, dayName, classNumber, ClassLetter,
        ).
        Find(&result)
    return
}
```

Рисунок 41 – функция с запросом по вопросу 3.

На рисунке 42 представлен результат запроса по вопросу 3.

```
194     fmt.Println(a... "В каком кабинете будет 5-й урок в среду у некоторого класса?")
195     for _, schedule := range requests.GetClassroom(db, lessonNumber: 5, dayName: "Среда", classNumber: 7, ClassLetter: "B") {
196         fmt.Println(schedule.ClassroomNumber)
197     }
198 }
199 main()
```

Run go build lab3 x

В каком кабинете будет 5-й урок в среду у некоторого класса?  
310

Рисунок 42 – результат запроса по вопросу 3.

Вопрос 4. В каких классах преподает заданный предмет заданный учитель?

Функция с запросом представлена на рисунке 43.

```
func GetTeacherClasses( 1 usage
db *gorm.DB,
teacherName string,
teacherSurname string,
teacherPatronymic string,
subject string) (result []models.Schedule) {
db.
    Joins(
        query: "INNER JOIN teachers ON schedules.teacher_id = teachers.teacher_id",
    ).
    Joins(
        query: "INNER JOIN subjects ON schedules.subject_id = subjects.subject_id",
    ).
    Where(
        query: "teacher_name = ? AND teacher_surname = ? AND teacher_patronymic = ? AND subject_name = ?",
        teacherName, teacherSurname, teacherPatronymic, subject,
    ).
    Find(&result)
return
}
```

Рисунок 43 – функция с запросом по вопросу 4.

На рисунке 44 представлен результат запроса по вопросу 4.

```
199     fmt.Println(a... "В каких классах преподает заданный предмет заданный учитель?")
200     for _, schedule := range requests.GetTeacherClasses(
201         db, teacherName: "Козлов", teacherSurname: "Елена", teacherPatronymic: "Владимировна", subject: "Биология") {
202         fmt.Println(schedule.ClassNumber, schedule.ClassLetter)
203     }
204 }
205 main()
```

Run go build lab3 x

В каких классах преподает заданный предмет заданный учитель?  
5 Б  
7 Д

Рисунок 44 – результат запроса по вопросу 4.

Вопрос 5. Расписание на заданный день недели для указанного класса?

Запрос выглядит следующим образом:

Функция с запросом представлена на рисунке 45.

```

type Question5 struct { 1 usage
    LessonNumber    int
    SubjectName      string
    TeacherName      string
    TeacherSurname   string
    TeacherPatronymic string
}

func GetSchedule(db *gorm.DB, classNumber int, classLetter string, dayName string) (result []Question5)
db.
    Table(name: "schedules").
    Select(query: "schedules.lesson_number, subjects.subject_name, teachers.teacher_name, "+
        "teachers.teacher_surname, teachers.teacher_patronymic").
    Joins(
        query: "INNER JOIN teachers ON schedules.teacher_id = teachers.teacher_id",
    ).
    Joins(
        query: "INNER JOIN subjects ON schedules.subject_id = subjects.subject_id",
    ).
    Where(
        query: "class_letter = ? AND class_number = ? and day_name = ?",
        classLetter, classNumber, dayName,
    ).
    Find(&result)
return
}

```

Рисунок 45 – функция с запросом по вопросу 5.

На рисунке 46 представлен результат запроса по вопросу 5.

```

205     fmt.Println(a... "Расписание на заданный день недели для указанного класса?")
206     for _, data := range requests.GetSchedule(db, classNumber: 6, classLetter: "Б", dayName: "Вторник") {
207         fmt.Println(
208             data.LessonNumber,
209             data.SubjectName,
210             data.TeacherName,
211             data.TeacherSurname,
212             data.TeacherPatronymic)
213     }
214
main()

```

Run go build lab3 x

Расписание на заданный день недели для указанного класса?

1 Иностранный язык Михайлов Артем Дмитриевич

2 Математика Смирнова Мария Игоревна

Рисунок 46 – результат запроса по вопросу 5.

**Вопрос 6.** Сколько учеников в указанном классе?

Функция с запросом представлена на рисунке 47.

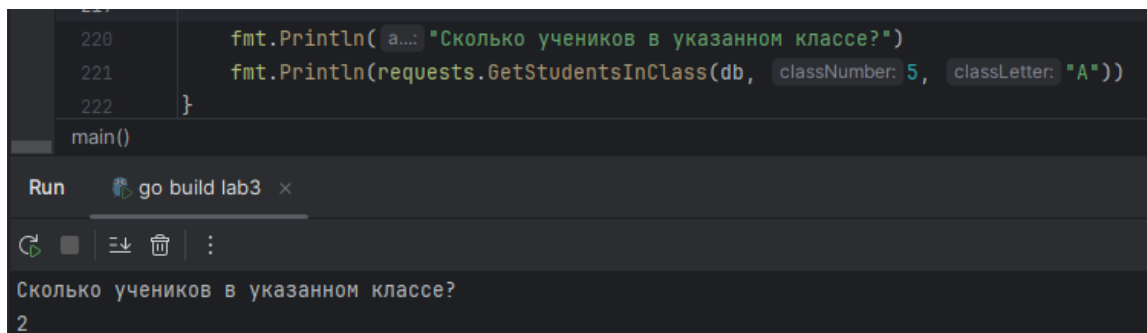
```

func GetStudentsInClass(db *gorm.DB, classNumber int, classLetter string) (result int64) { 1 usage
db.
    Table(name: "students").
    Joins(query: "INNER JOIN classes ON students.class_number = classes.class_number AND "+
        "students.class_letter = classes.class_letter").
    Where(query: "students.class_letter = ? AND students.class_number = ?", classLetter, classNumber).
    Count(&result)
return
}

```

Рисунок 47 – функция с запросом по вопросу 6.

На рисунке 48 представлен результат запроса по вопросу 6.



```
220     fmt.Println(a...: "Сколько учеников в указанном классе?")
221     fmt.Println(requests.GetStudentsInClass(db, classNumber: 5, classLetter: "A"))
222 }
main()

Run go build lab3 x

Сколько учеников в указанном классе?
2
```

Рисунок 48 – результат запроса по вопросу 6.

## 8. Запуск программы

Для запуска программы в командной строке ввести:

```
go run main.go
```

В приложении А предоставлена ссылка на PR.

## Выводы.

В данной лабораторной работе освоена работа с *ORM* для *Go* – *GORM*.

Описаны в виде моделей *GORM* таблицы из 1-й лабораторной работы.

Написана функция, заполняющая все таблицы тестовыми данными.

Написаны запросы к БД, отвечающие на вопросы из 1-й лабораторной работы с использованием *ORM*.

## **ПРИЛОЖЕНИЕ А**

### **ССЫЛКИ**

Ссылка на PR:

<https://github.com/moevm/sql-2023-1304/pull/52>