Таблица 1 – База данных.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата\_Время | Предмет | ФИО\_преподавателя | Урок | Кол-во\_учеников |
| 2024-03-01 09:00 | Математика | Петров И.С. | М1У2 | 25 |
| 2024-03-01 10:00 | История | Сидорова А.П. | М1У3 | 28 |
| 2024-03-01 11:00 | Химия | Кузнецова О.В. | М1У4 | 22 |
| 2024-03-01 13:00 | Литература | Николаев Д.А. | М2У2 | 30 |

## Код программы:

package main

import (

    "database/sql"

    "fmt"

    "log"

    "math/rand"

    "time"

    \_ "modernc.org/sqlite"

)

type Lesson struct {

    DateTime     time.Time

    Subject      string

    TeacherName  string

    LessonNumber string

    StudentCount int

}

// DataBase база данных

type DataBase struct {

    conn \*sql.DB

}

func main() {

    // Подключение к БД

    conn, err := sql.Open("sqlite", "./lessons.db")

    if err != nil {

        log.Fatal(err)

    }

    defer conn.Close()

    // Создание  БД

    dataBase := DataBase{conn}

    // Создание таблицы по структуре

    dataBase.CreateTableIfNotExist(&Lesson{})

    for i := 0; i < 4+rand.Intn(3); i++ {

        // Пример вставки записи

        lesson := Lesson{

            DateTime:     time.Now().Add(time.Duration(rand.Intn(160)) \* time.Hour),

            Subject:      "Математика",

            TeacherName:  "Петров И.С.",

            LessonNumber: fmt.Sprintf("М%dУ%d", 1+rand.Intn(8), 1+rand.Intn(8)),

            StudentCount: 15 + rand.Intn(15),

        }

        // Создание строки

        dataBase.Create(&lesson)

    }

    // Обновление структуры, куда будет записываться урок

    lesson := Lesson{}

    // Пример получения записи по предмету

    err = dataBase.GetRowByColumn(&lesson, "subject", "Математика")

    if err != nil {

        panic(err)

    }

    fmt.Printf("Retrieved lesson: %+v\n", lesson)

    // Вывод всех записей

    fmt.Println("\nAll lessons:")

    lessonsInterface, err := dataBase.SelectAll(&lesson)

    if err != nil {

        panic(err)

    }

    lessons, ok := lessonsInterface.([]\*Lesson)

    if !ok {

        panic("Cannot convert lessonsInterface")

    }

    for \_, v := range lessons {

        fmt.Println(v)

    }

}

func (db \*DataBase) CreateTableIfNotExist(structure interface{}) (sql.Result, error) {

    // Определение запроса по типу

    switch v := structure.(type) {

    case \*Lesson:

        fmt.Printf("structure is an Lesson: %v\n", v)

        return db.createLessonTable()

    case nil:

        fmt.Println("structure is nil")

    default:

        fmt.Printf("structure is of an unknown type: %T\n", v)

    }

    return nil, fmt.Errorf("structure is of an unknown type: %T", structure)

}

func (db \*DataBase) createLessonTable() (sql.Result, error) {

    query := `

    CREATE TABLE IF NOT EXISTS lessons (

        date\_time DATETIME,

        subject TEXT,

        teacher\_name TEXT,

        lesson\_number TEXT,

        student\_count INTEGER

    );

    `

    res, err := db.conn.Exec(query)

    if err != nil {

        return nil, err

    }

    return res, nil

}

func (db \*DataBase) Create(structure interface{}) (sql.Result, error) {

    // Определение запроса по типу

    switch v := structure.(type) {

    case \*Lesson:

        fmt.Printf("structure is an Lesson: %v\n", v)

        return db.createLesson(\*v)

    case nil:

        fmt.Println("structure is nil")

    default:

        fmt.Printf("structure is of an unknown type: %T\n", v)

    }

    return nil, fmt.Errorf("structure is of an unknown type: %T", structure)

}

func (db \*DataBase) createLesson(lesson Lesson) (sql.Result, error) {

    query := `

    INSERT INTO lessons (date\_time, subject, teacher\_name, lesson\_number, student\_count)

    VALUES (?, ?, ?, ?, ?)

    `

    res, err := db.conn.Exec(query, lesson.DateTime, lesson.Subject, lesson.TeacherName, lesson.LessonNumber, lesson.StudentCount)

    if err != nil {

        return nil, err

    }

    return res, nil

}

// GetRowByColumn получает строку по столбцу и значению.

// structure - ссылка на структуру, куда будут записываться  данные

// column - столбец

// value - значение столбца

func (db \*DataBase) GetRowByColumn(structure interface{}, column string, value interface{}) error {

    // Определение запроса по типу

    switch v := structure.(type) {

    case \*Lesson:

        fmt.Printf("structure is an Lesson: %v\n", v)

        return db.getLessonByColumn(v, column, value)

    case nil:

        fmt.Println("structure is nil")

    default:

        fmt.Printf("structure is of an unknown type: %T\n", v)

    }

    return fmt.Errorf("structure is of an unknown type: %T", structure)

}

func (db \*DataBase) getLessonByColumn(lesson \*Lesson, column string, value interface{}) error {

    query := fmt.Sprintf("SELECT \* FROM lessons WHERE %s = ?", column)

    row := db.conn.QueryRow(query, value)

    err := row.Scan(&lesson.DateTime, &lesson.Subject, &lesson.TeacherName, &lesson.LessonNumber, &lesson.StudentCount)

    if err != nil {

        return err

    }

    return nil

}

func (db \*DataBase) SelectAll(structure interface{}) (interface{}, error) {

    // Определение запроса по типу

    switch v := structure.(type) {

    case \*Lesson:

        fmt.Printf("structure is an Lesson: %v\n", v)

        return db.selectAllLessons()

    case nil:

        fmt.Println("structure is nil")

    default:

        fmt.Printf("structure is of an unknown type: %T\n", v)

    }

    return nil, fmt.Errorf("structure is of an unknown type: %T", structure)

}

func (db \*DataBase) selectAllLessons() (interface{}, error) {

    query := "SELECT \* FROM lessons"

    rows, err := db.conn.Query(query)

    if err != nil {

        return nil, err

    }

    defer rows.Close()

    lessons := []\*Lesson{}

    for rows.Next() {

        var lesson Lesson

        err := rows.Scan(&lesson.DateTime, &lesson.Subject, &lesson.TeacherName, &lesson.LessonNumber, &lesson.StudentCount)

        if err != nil {

            return nil, err

        }

        lessons = append(lessons, &lesson)

    }

    return lessons, nil

}

## Результаты работы программы

Рисунок 1 - База данных до работы программы.

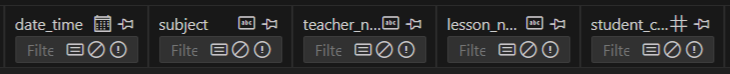


Рисунок 2 - База данных после работы программы.

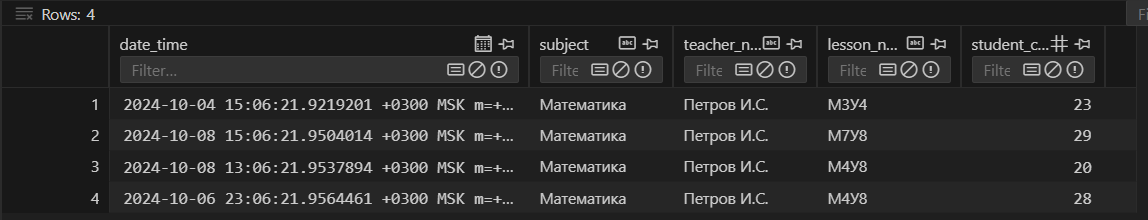


Рисунок 3 – Результат работы программы.

