**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №1

«Хранение пользователей в PostgreSQL»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-32Б |  | преподаватель каф. ИУ5 |
| Старкин Александр |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2024 г.

### Задание:

Разработать даталогическую модель на основе которой будет создана база данных в PostgreSQL, хранящая информацию о пользователях и их друзьях.

1.Сущность users должна содержать колонки:

* id (ПК) – int64
* nickname - string
* notify\_interval - string
* fillial\_title - string
* faculty\_title - string
* course\_title - string
* group\_title - string
* cathedra\_title - string
* uuid – string

2.Сущность friends должна содержать колонки:

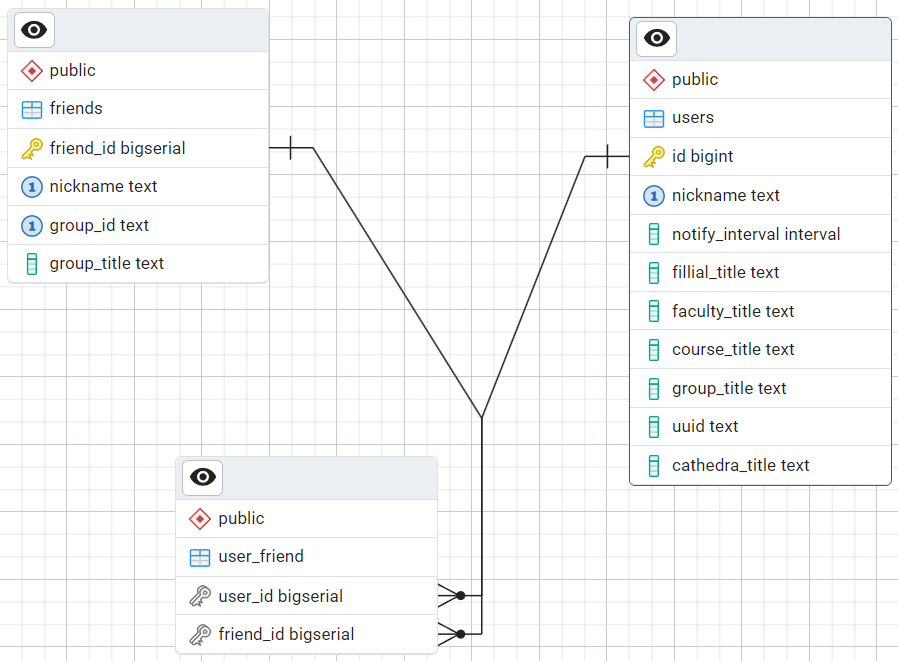
* friend\_id (ПК) – int64
* nickname - string
* group\_id - string
* group\_title - string

т.к. friends и users связаны отношением М:М создадим промежуточную таблицу связей user\_friend, содержащую уникальные id – friend\_id.

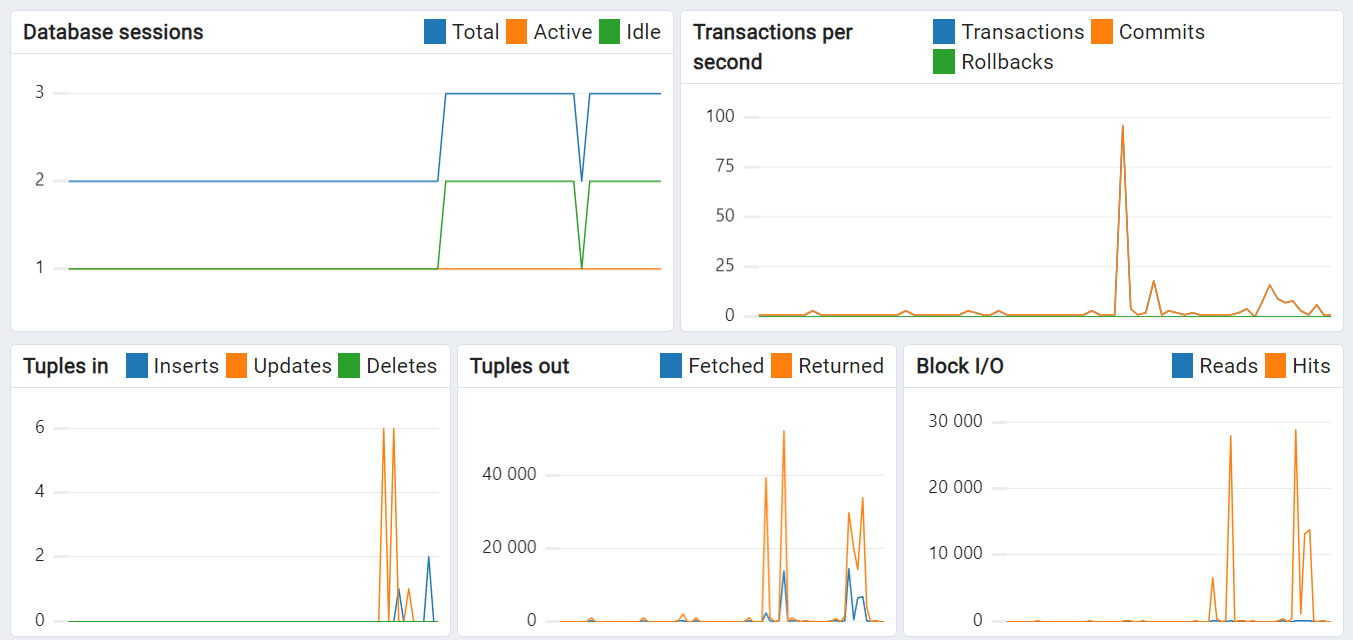
3.Сущность user\_friend должна содержать колонки:

* user\_id (ВК) – int64
* friend\_id (ВК) – int64

Даталогическая модель будет выглядеть так:



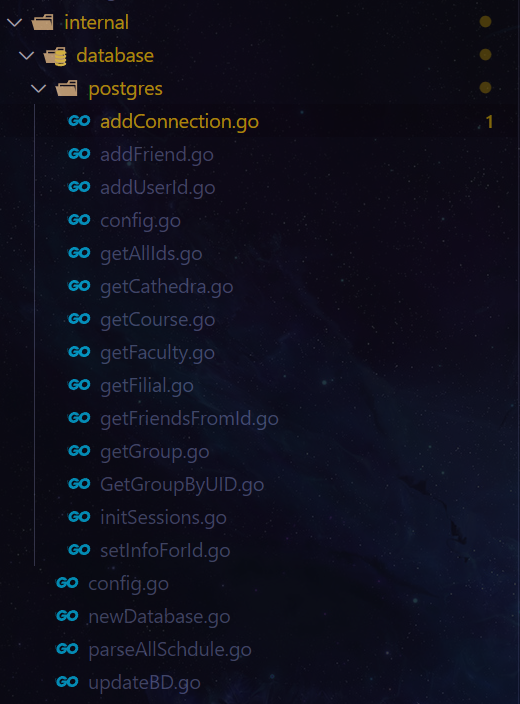
Проверка активности в бд.



**Реализуем работу с БД в go.**

Для этого создадим интерфейс IBd. Далее создадим структуру Postgres имплементирующую IBd, тем самым реализуем механизм полиморфизма в go (так называемый duck typing).

Структура пакета:



Текст программы

Пример добавления в бд.

package postgres

import (

    "context"

    "errors"

    "fmt"

    "os"

    errorsCustom "github.com/Robocotik/IT-s\_Friends/internal/models/errors"

    "github.com/Robocotik/IT-s\_Friends/internal/services/output"

    "github.com/jackc/pgx/v5/pgconn"

    "github.com/mymmrac/telego"

)

func (psql Postgres) AddConnection(ctx context.Context, bot \*telego.Bot, chatID int64, user\_id int64, friend\_id int64) error {

*// fmt.Printf("\n Я ДОБАВИЛ %s в %s : \n", user\_id, friend\_id)*

    \_, err := psql.Conn.Exec(

        ctx,

        "INSERT INTO user\_friend (user\_id, friend\_id) VALUES ($1, $2)",

        user\_id, friend\_id,

    )

*if* err != nil {

*if* pgErr, ok := err.(\*pgconn.PgError); ok {

*if* pgErr.Code == "23503" { *// Код ошибки для нарушения ограничения*

                fmt.Fprintf(os.Stderr, "Insert failed: constraint violation on name\_length: %v\n", pgErr.Message)

                output.WriteMessage(bot, chatID, errorsCustom.ErrFriendAlreadyAdded\_23503)

*return* errors.New(errorsCustom.ErrFriendAlreadyAdded\_23503)

            }

        }

        fmt.Fprintf(os.Stderr, "QueryRow failed in connection : %v\n", err)

        output.RiseError(bot, chatID, err)

*return* err

    }

*return* nil

}

Пример обновления записей бд.

package postgres

import (

    "context"

    "fmt"

    "os"

    "github.com/Robocotik/IT-s\_Friends/internal/models/structures"

    "github.com/Robocotik/IT-s\_Friends/internal/services/output"

    "github.com/mymmrac/telego"

)

func (psql Postgres) UpdateUser(bot \*telego.Bot, chatID int64, identity structures.Identity, id int64) error {

    \_, err := psql.Conn.Exec(context.Background(), `

    UPDATE users SET fillial\_title = $1,

     faculty\_title = $2,

     course\_title = $3,

     cathedra\_title = $4,

     group\_title = $5,

     uuid = $6

     WHERE id = $7;`,

        identity.Filial, identity.Faculty, identity.Course, identity.Cathedra, identity.Group, identity.Uuid, id)

*if* err != nil {

        fmt.Fprintf(os.Stderr, "Query failed on setting info for id: %v\n", err)

        output.RiseError(bot, chatID, err)

*return* err

    }

*return* nil

}

Пример получения данных с бд.

package postgres

import (

    "context"

)

func (psql Postgres) GetAllIds() ([]int64, error) {

    var res []int64

    rows, err := psql.Conn.Query(context.Background(),

        "SELECT id FROM users")

*if* err != nil {

*return* []int64{}, err

    }

*defer* rows.Close()

*for* rows.Next() {

        var id int64

*if* err := rows.Scan(&id); err != nil {

*return* nil, err

        }

        res = append(res, id)

    }

*if* err := rows.Err(); err != nil {

*return* nil, err

    }

*return* res, nil

}