МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра вычислительной техники



**Лабораторная работа №3**

по дисциплине: «WEB - ПРОГРАММИРОВАНИЕ»

на тему: **«Работа с базой данных MySQL»**

Вариант №6

Выполнил: Проверил:

Студент гр. АВТ-219: Гунько А. В.

Завёрткин М. А.

Новосибирск

2025

СОДЕРЖАНИЕ

Введение 3

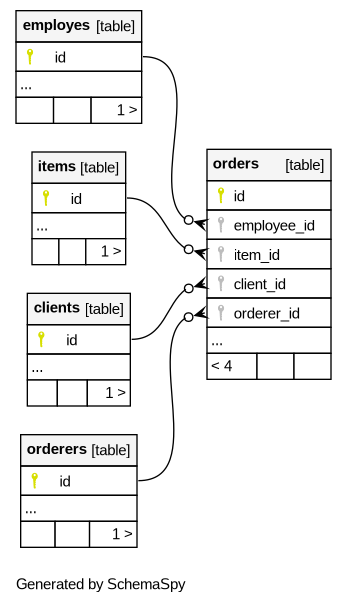
Ход работы 3

Вывод 6

# **Задание**

Задания различаются структурой баз данных, реализованных в соответствии с заданиями к лабораторной работе №1. Обязательна реализация связанных запросов на извлечение данных, запросов на поиск (по параметру), добавления, изменения (редактирования) и удаления данных в/из таблицы-связки.

**Структура базы данных**



**Веб-сервер на GO**

package main

import (

"database/sql"

"fmt"

\_ "github.com/lib/pq"

"io"

"log"

"net/http"

"strings"

)

var db \*sql.DB

const search\_form\_html = `

<form method="GET" action="">

<p>ID:<input type="number" step="1" min="1" name="id">

<input type="SUBMIT" value="Search"></p></form>

<a href="/index">Show all</a>

<a href="/insert">Insert</a>

`

const sql\_select = `

SELECT orders.id,employes.name,orders.price,items.name,clients.name,orderers.name

FROM orders

LEFT JOIN employes ON employee\_id = employes.id

LEFT JOIN items ON item\_id = items.id

LEFT JOIN clients ON client\_id = clients.id

LEFT JOIN orderers ON orderer\_id = orderers.id

ORDER BY orders.id

`

const table\_start = `

<table border="1"><tbody>

<tr><th><b>id</b></th><th><b>employee</b></th><th><b>price</b></th><th><b>item</b></th><th><b>client</b></th><th><b>orderer</b></th><th><b>Update</b></th><th><b>Delete</b></th></tr>

`

const table\_end = `

</tbody></table>

`

const table\_row\_fmt = `

<tr><td>%d</td><td>%s</td><td>%d</td><td>%s</td><td>%s</td><td>%s</td><td><a href="update?id=%d">Update</a></td><td><a href="delete?id=%d">Delete</a></td></tr>

`

func make\_row(w io.Writer, id int, employee string, price int, item string, client string, orderer string) {

fmt.Fprintf(w, table\_row\_fmt, id, employee, price, item, client, orderer, id, id)

}

func handler(w http.ResponseWriter, r \*http.Request) {

cur\_select := sql\_select

request\_id := r.URL.Query().Get("id")

if request\_id != "" {

cur\_select += fmt.Sprintf("\nWHERE orders.id = '%s'", request\_id)

}

fmt.Fprintf(w, "<h1>Just a db, bro</h1><div>")

fmt.Fprintf(w, "%s", cur\_select)

fmt.Fprintf(w, "%s", table\_start)

rows, err := db.Query(cur\_select)

if err != nil {

http.Error(w, "Internal DB error", http.StatusInternalServerError)

panic(err)

}

defer rows.Close()

for rows.Next() {

var id int

var employee string

var price int

var item string

var client string

var orderer string

err := rows.Scan(&id, &employee, &price, &item, &client, &orderer)

if err != nil {

fmt.Println(err)

continue

}

make\_row(w, id, employee, price, item, client, orderer)

}

fmt.Fprintf(w, "%s", table\_end)

fmt.Fprintf(w, search\_form\_html)

fmt.Fprintf(w, "</div>")

}

const sql\_insert\_tmpl = `

INSERT INTO orders (employee\_id, price, item\_id, client\_id, orderer\_id) VALUES

($1, $2, $3, $4, $5);

`

func handler\_insert(w http.ResponseWriter, r \*http.Request) {

if r.Method == http.MethodGet {

fmt.Fprintf(w, "<html><form method=\"POST\" action=\"/insert\"><p>")

fmt.Fprintf(w, "Employee<select name=\"employee\_id\">")

fmt.Fprintf(w, "%s", get\_options("employes"))

fmt.Fprintf(w, "</select>")

fmt.Fprintf(w, "Price<input type=\"number\" step=\"1\" min=\"1\" name=\"price\"/>")

fmt.Fprintf(w, "Item<select name=\"item\_id\">")

fmt.Fprintf(w, "%s", get\_options("items"))

fmt.Fprintf(w, "</select>")

fmt.Fprintf(w, "Client<select name=\"client\_id\">")

fmt.Fprintf(w, "%s", get\_options("clients"))

fmt.Fprintf(w, "</select>")

fmt.Fprintf(w, "Orderer<select name=\"orderer\_id\">")

fmt.Fprintf(w, "%s", get\_options("orderers"))

fmt.Fprintf(w, "</select>")

fmt.Fprintf(w, "<input type=\"SUBMIT\" value=\"Add\"></p></form></html>")

}

if r.Method == http.MethodPost {

\_, err := db.Exec(sql\_insert\_tmpl, r.FormValue("employee\_id"), r.FormValue("price"), r.FormValue("item\_id"), r.FormValue("client\_id"), r.FormValue("orderer\_id"))

if err != nil {

http.Error(w, "Error while inserting", http.StatusBadRequest)

return

}

http.Redirect(w, r, "/index", http.StatusSeeOther)

}

}

const sql\_update\_tmpl = `

UPDATE orders

SET employee\_id=$1, price=$2, item\_id=$3, client\_id=$4, orderer\_id=$5

WHERE id = $6

`

func handler\_update(w http.ResponseWriter, r \*http.Request) {

if r.Method == http.MethodGet {

request\_id := r.URL.Query().Get("id")

if request\_id == "" {

http.Error(w, "Id not present", http.StatusBadRequest)

return

}

var id int

var employee\_id int

var price int

var item\_id int

var client\_id int

var orderer\_id int

db.QueryRow(fmt.Sprintf("SELECT id,employee\_id,price,item\_id,client\_id,orderer\_id FROM orders WHERE id = %s", request\_id)).Scan(&id, &employee\_id, &price, &item\_id, &client\_id, &orderer\_id)

fmt.Fprintf(w, "<html><form method=\"POST\" action=\"/update\"><p>")

fmt.Fprintf(w, "Employee<select name=\"employee\_id\">")

fmt.Fprintf(w, "%s", get\_options\_selected("employes", employee\_id))

fmt.Fprintf(w, "</select>")

fmt.Fprintf(w, "Price<input type=\"number\" step=\"1\" min=\"1\" name=\"price\" value=\"%d\"/>", price)

fmt.Fprintf(w, "Item<select name=\"item\_id\">")

fmt.Fprintf(w, "%s", get\_options\_selected("items", item\_id))

fmt.Fprintf(w, "</select>")

fmt.Fprintf(w, "Client<select name=\"client\_id\">")

fmt.Fprintf(w, "%s", get\_options\_selected("clients", client\_id))

fmt.Fprintf(w, "</select>")

fmt.Fprintf(w, "Orderer<select name=\"orderer\_id\">")

fmt.Fprintf(w, "%s", get\_options\_selected("orderers", orderer\_id))

fmt.Fprintf(w, "</select>")

fmt.Fprintf(w, "<input type=\"hidden\" name=\"id\" value=\"%d\"/>", id)

fmt.Fprintf(w, "<input type=\"SUBMIT\" value=\"Update\"></p></form></html>")

}

if r.Method == http.MethodPost {

\_, err := db.Exec(sql\_update\_tmpl, r.FormValue("employee\_id"), r.FormValue("price"), r.FormValue("item\_id"), r.FormValue("client\_id"), r.FormValue("orderer\_id"), r.FormValue("id"))

if err != nil {

http.Error(w, "Unknown error while updating", http.StatusBadRequest)

return

}

http.Redirect(w, r, "/index", http.StatusSeeOther)

}

}

func get\_options(table string) string {

var builder strings.Builder

rows, err := db.Query(fmt.Sprintf("SELECT id,name FROM %s", table))

if err != nil {

panic(err)

}

defer rows.Close()

for rows.Next() {

var id int

var name string

err := rows.Scan(&id, &name)

if err != nil {

fmt.Println(err)

continue

}

builder.WriteString(fmt.Sprintf("<option value=\"%d\">%s</option>", id, name))

}

return builder.String()

}

func get\_options\_selected(table string, target\_id int) string {

var builder strings.Builder

rows, err := db.Query(fmt.Sprintf("SELECT id,name FROM %s", table))

if err != nil {

panic(err)

}

defer rows.Close()

for rows.Next() {

var id int

var name string

err := rows.Scan(&id, &name)

if err != nil {

fmt.Println(err)

continue

}

selected := ""

if id == target\_id {

selected = "selected"

}

builder.WriteString(fmt.Sprintf("<option value=\"%d\" %s>%s</option>", id, selected, name))

}

return builder.String()

}

func handler\_delete(w http.ResponseWriter, r \*http.Request) {

request\_id := r.URL.Query().Get("id")

if request\_id == "" {

http.Error(w, "Id not present", http.StatusBadRequest)

return

}

\_, err := db.Exec("DELETE FROM orders WHERE id = $1", request\_id)

if err != nil {

http.Error(w, "Invalid id", http.StatusBadRequest)

return

}

http.Redirect(w, r, "/index", http.StatusSeeOther)

}

func main() {

var err error

db, err = sql.Open("postgres", "user=postgres dbname=orders sslmode=disable")

if err != nil {

log.Fatal(err)

return

}

defer db.Close()

http.HandleFunc("/index", handler)

http.HandleFunc("/insert", handler\_insert)

http.HandleFunc("/update", handler\_update)

http.HandleFunc("/delete", handler\_delete)

fmt.Println("Start listening at http://localhost:8080/index")

log.Fatal(http.ListenAndServe(":8080", nil))

}