МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”
Кафедра Систем штучного інтелекту

РОЗРАХУНКОВА РОБОТА

з дисципліни “Організація баз даних та знань”
на тему “Календар подій” та “Фінансовий менеджер”

Виконав:
студент гр. КН-209
Гладун Ярослав

Балів	Дата

Викладач:
Мельникова Наталія Іванівна

Львів-2020

Зміст

***Ремарка**. У зв'язку з тим, що я брав участь у двох проектах, я вирішив написати про перший проект (календар подій), у який я вклав більше зусиль в напрямку баз даних.*

КАЛЕНДАР ПОДІЙ

1 Вступ

Мета даного проекту – це розробити веб сайт яким будуть користуватися викладачі кафедри та студенти. Даний веб сайт повинен містити такі основні сторінки, як:

- Реєстрація користувача;
- Логування користувача;
- Календар подій;
- Сторінка створення події.

Після того, як користувач зареєструється та увійде, від може створити подію. На цю подію він може призначити інших користувачів, вказавши їх емейли. Інші ж користувачі отримують сповіщення як і від самого календаря (сайту), так і на пошту, яку вказували під час реєстрації. Сповіщення, яке прийшло на пошту, містить у собі унікальне посилання, яке супроводжується повідомленням, що користувач може перейти по посиланню та прийняти подію, або ж відхилити її. Якщо користувач приймає подію, то сповіщення про це приходить тому, хто її створив та дана подія відображається у календарі користувача, який її прийняв. У інакшому ж випадку (коли користувач відхиляє подію), сповіщення про це також приходить людині, яка її створила, але дана подія відображатися у календарі користувача, який прийняв, не буде.

Після короткого опису частини функціоналу проекту, можна сказати декілька слів про **актуальність**.

По-перше, це може заощадити дуже багато часу. Наприклад потрібно поширити інформацію про конференцію, яку влаштовує певна IT-компанія для студентів кафедри. Щоб інформація дійшла до всіх студентів, окрім того, що треба про це оголосити на парах, потрібно й сформулювати повідомлення та написати до прикладу у телеграм. Можливо зробити розсилку. Усе це займає певний час. Використовуючи запропонований сервіс, викладач може створити подію, заповнивши просту форму (інформацію про подію), та вказавши групу людей (наприклад студенти другого курсу), яким ця подія буде призначена. Весь цей процес займає значно менше часу.

По-друге, усі користувачі точно отримають інформацію, що не завжди можливо у випадку використання методу, про який згадувалося раніше. Якщо наголосити студентам, що у цьому сервісі буде інформація про усі події протягом навчального року, то їм буде достатньо зайти на сайт, і вони побачать всі мероприємства, які будуть найближчим часом. У іншому ж випадку, їм потрібно перепитуватися у викладачів або одногрупників. Плюс якщо до інформації важко доступитися (на приклад про неї писали дуже давно у групі в телеграмі), то студент не дізнається про цю подію та не відвідає її, що у нашому випадку можна запобігти.

По-третє, *завжди* є фідбек від людей, які приймають подію. Повернемося до конференції від IT-компанії. Дуже часто організаторам потрібно знати кількість людей, які відвідають мероприємство, щоб розрахувати наприклад кількість місць, кількість їжі і т. д. Коли користувач приймає подію, він може вказати, чи точно відвідає, чи можливо її відвідає, чи не відвідає. Це є обов'язковий пункт щоб прийняти цю подію, та інформація, яка буде в результаті може досить сильно допомогти організаторам.

Ну і перед тим, як безпосередньо перейти до деталей реалізації, пов'язаних з базами даних, варто сказати декілька слів про сам проект, його організацію тощо. У першу чергу, ми не використовували ніяких готових API (наприклад Google calendar API).

Окрім того, що це потрібно було до вимог розрахункової роботи, ми вирішили зробити систему, яка буде максимально гнучкою. У випадку використання готових віджетів дуже важко щось змінити під себе (під потреби користувачів), а написання свого API з нуля дозволяє це зробити. Також у цьому розділі варто сказати, яку саме систему управління базами даних ми вирішили використовувати. Порадившись, вибір впав між MySQL та PostgreSQL. Познайомившись конкретніше з деякими відмінностями [1, 2], ми вирішили використовувати MySQL. Основних дві причини – це простота та наявність великої кількості різних джерел для вивчення.

2 Логічна схема бази даних проекту

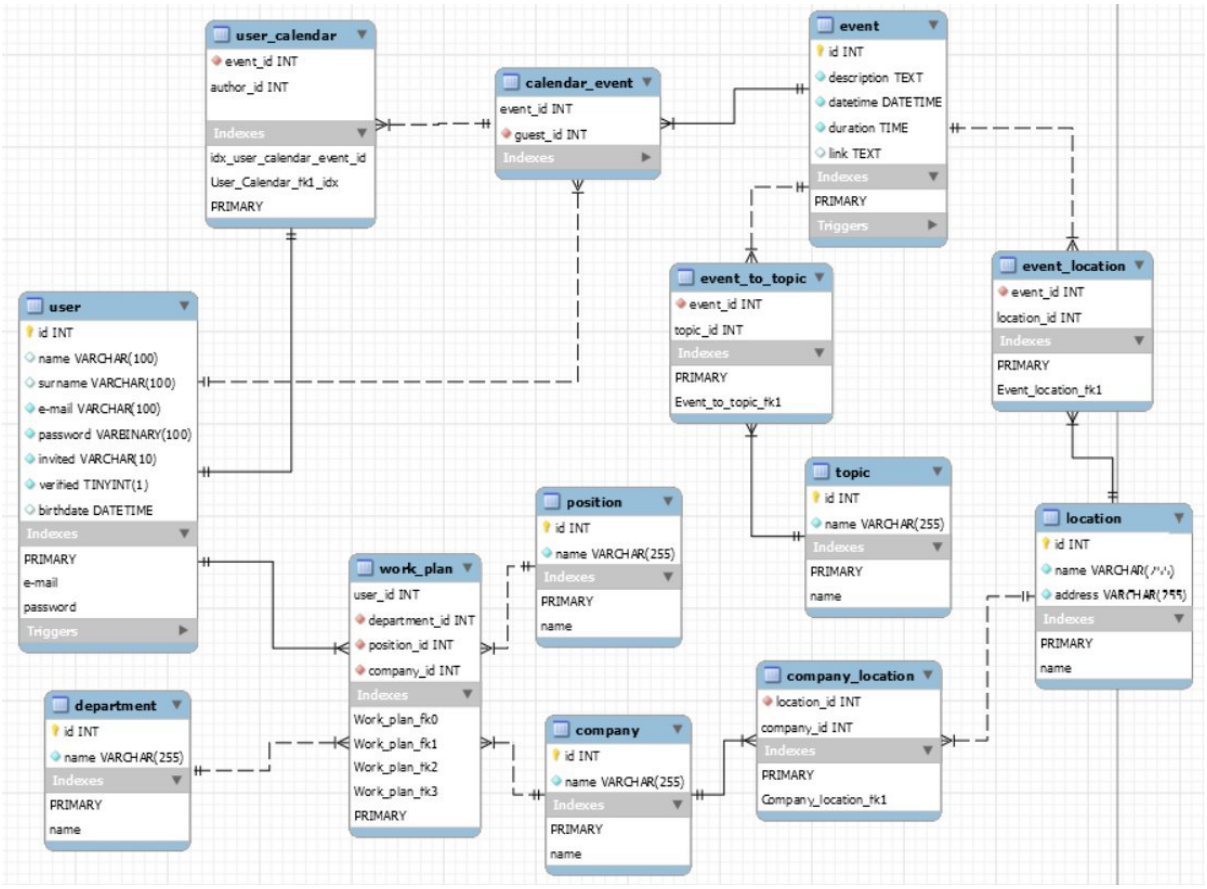


Рис 1. Схема бази даних (Нотация Баркера).

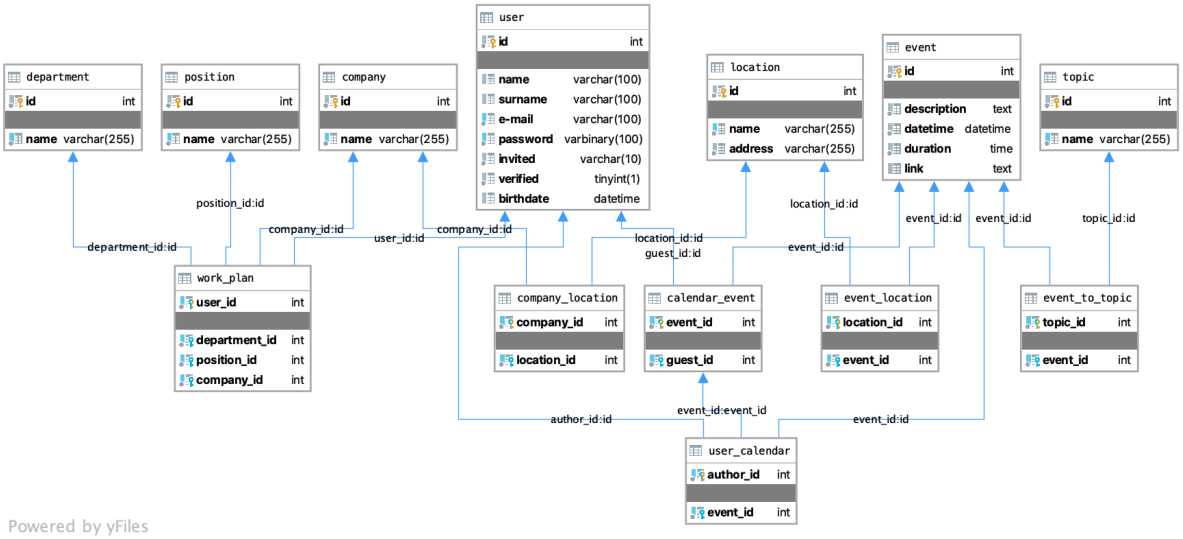


Рис 1. Схема бази даних.

На рисунку 1 можна побачити схему бази даних нашого проекту. Тут
- внутрішній ключ (primary key),
- зовнішній ключ (foreign key),
- поле, яке не може мути рівне NULL,

3 Опис структури БД

Отож коротко опишемо кожну з таблиць, які є у нашій базі даних.

1. Таблиця *location* містить такі три поля:
   - внутрішній ключ (*id*);
   - назва локації (*name*);
   - адреса локації (*address*).

Ця таблиця буде нам потрібна для того, щоб показати у якому саме місці буде відбуватися подія. Також буде потрібна для того, щоб вказати де знаходяться офіси компаній.

2. Таблиця *topic* містить два поля:
   - внутрішній ключ (*id*);
   - назва теми (*name*).

Ця таблиця стосується вже безпосередньо події: у ній записується тематика подій (наприклад: *computer science, summer internship, etc.*).

3. Таблиця *event* містить такі шість полів:
   - внутрішній ключ (*id*);
   - опис події (*description*);
   - дата проведення (*date*);
   - час проведення (*time*);
   - тривалість у годинах (*duration*);
   - посилання на запрошення (*link*).

Це одна із основних таблиць у нашій базі даних. Вона пов’язана із двома попередніми таблицями (*location, topic*) відношенням багато до багатьох. Тобто подія може мати багато тематик та проводитися у багатьох місцях одночасно.

4. Таблиця *department* містить такі два поля:
   - внутрішній ключ (*id*);
   - назва відділу (*name*).

Ця таблиця потрібна для того, щоб вказати у якому саме відділі певної структури (університет, компанія тощо) працює/вчиться користувач.

5. Таблиця *company* містить такі два поля:
   - внутрішній ключ (*id*);
   - назва відділу (*name*).

Ця таблиця вже показує безпосередньо структуру в якій працює користувач. Це може бути університет, компанія, різні державні структури і т. д.

6. Таблиця *position* містить такі два поля:
   - внутрішній ключ (*id*);
   - назва позиції (*name*).

Ця таблиця стосується того, хто саме людина у структурі, якій працює (наприклад *HR, developer, designer*).

7. Таблиця *work_plan* містить такі п’ять полів:
   - внутрішній ключ (*id*);
   - зовнішній ключ на користувача (*user_id*);
   - зовнішній ключ на *company* (*company_id*);

- зовнішній ключ на *department* (*department_id*);
- зовнішній ключ на *position* (*position_id*).

Ця таблиця стосується працевлаштування користувача. Тобто вона повністю описує де працює користувач (у якій компанії, на якій посаді і т. д.). На цю таблицю якраз і зсилається одна із основних – *user.*

8. Таблиця *user_calendar* містить такі два поля:
   - зв'язок із сутністю event (*event_id*);
   - зв'язок із сутністю user (*author_id*).

Дана таблиця дозволяє побачити події, які були створені користувачем. Тобто задає відповідність між користувачем та створеною ним подією.

9. Таблиця user містить такі вісім полів:
   - внутрішній ключ (*id*);
   - ім'я користувача (*name*);
   - прізвище користувача (*surname*);
   - пошта користувача (*email*);
   - пароль користувача (*password*);
   - поле, яке відповідає за те, чи підтвердив користувач пошту (*verified*);
   - дата народження (*birthdate*);

Ця таблиця персональну інформацію користувача. Вона має зв'язок багато до одного з таблицею *work_plan*, що також є персональною інформацією користувача про його працевлаштування. Також, ця таблиця має зв'язок один до одного з таблицею user_calendar, яка і задає відношення між користувачем та створеними ним подіями.

4 Фізична модель

```
-- -----------------------------------------------------
-- Table `calendardb`.`company`
-- -----------------------------------------------
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DA
TE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTI
ON';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
SHOW WARNINGS;
-- -----------------------------------------------------
-- Schema calendardb
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS `calendardb` ;

-- -----------------------------------------------------
-- Schema calendardb
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `calendardb` DEFAULT CHARACTER SET utf8mb4
COLLATE utf8mb4_0900_ai_ci ;
SHOW WARNINGS;
USE `calendardb` ;

---------
DROP TABLE IF EXISTS `calendardb`.`company` ;

SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS `calendardb`.`company` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `calendardb`.`location`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `calendardb`.`location` ;

SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS `calendardb`.`location` (
```

```
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NOT NULL,
  `address` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 4
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `calendardb`.`company_location`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `calendardb`.`company_location` ;

SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS `calendardb`.`company_location` (
  `location_id` INT NOT NULL,
  `company_id` INT NOT NULL,
  PRIMARY KEY (`company_id`),
  CONSTRAINT `Company_location_fk0`
    FOREIGN KEY (`location_id`)
    REFERENCES `calendardb`.`location` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `Company_location_fk1`
    FOREIGN KEY (`company_id`)
    REFERENCES `calendardb`.`company` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `calendardb`.`department`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `calendardb`.`department` ;

SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS `calendardb`.`department` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
```

```
SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `calendardb`.`event`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `calendardb`.`event` ;

SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS `calendardb`.`event` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `description` TEXT NOT NULL,
  `datetime` DATETIME NOT NULL,
  `duration` TIME NOT NULL,
  `link` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 22
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `calendardb`.`event_location`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `calendardb`.`event_location` ;

SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS `calendardb`.`event_location` (
  `event_id` INT NOT NULL,
  `location_id` INT NOT NULL,
  PRIMARY KEY (`location_id`),
  CONSTRAINT `Event_location_fk0`
    FOREIGN KEY (`event_id`)
    REFERENCES `calendardb`.`event` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `Event_location_fk1`
    FOREIGN KEY (`location_id`)
    REFERENCES `calendardb`.`location` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


SHOW WARNINGS;
```

```
-- -----------------------------------------------------
-- Table `calendardb`.`topic`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `calendardb`.`topic` ;

SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS `calendardb`.`topic` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 5
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `calendardb`.`event_to_topic`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `calendardb`.`event_to_topic` ;

SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS `calendardb`.`event_to_topic` (
  `event_id` INT NOT NULL AUTO_INCREMENT,
  `topic_id` INT NOT NULL,
  PRIMARY KEY (`topic_id`),
  CONSTRAINT `Event_to_topic_fk0`
    FOREIGN KEY (`event_id`)
    REFERENCES `calendardb`.`event` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `Event_to_topic_fk1`
    FOREIGN KEY (`topic_id`)
    REFERENCES `calendardb`.`topic` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `calendardb`.`position`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `calendardb`.`position` ;

SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS `calendardb`.`position` (
```

```
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `calendardb`.`user`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `calendardb`.`user` ;

SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS `calendardb`.`user` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(100) NULL DEFAULT NULL,
  `surname` VARCHAR(100) NULL DEFAULT NULL,
  `e-mail` VARCHAR(100) NOT NULL,
  `password` VARBINARY(100) NOT NULL,
  `invited` VARCHAR(10) NOT NULL,
  `verified` TINYINT(1) NOT NULL,
  `birthdate` DATETIME NULL DEFAULT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 10
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `calendardb`.`calendar_event`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `calendardb`.`calendar_event` ;

SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS `calendardb`.`calendar_event` (
  `event_id` INT NOT NULL,
  `guest_id` INT NOT NULL,
  PRIMARY KEY (`event_id`),
  CONSTRAINT `ToEvent_fk1`
    FOREIGN KEY (`event_id`)
    REFERENCES `calendardb`.`event` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `ToGuest_fk2`
    FOREIGN KEY (`guest_id`)
    REFERENCES `calendardb`.`user` (`id`)
```

```
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;

SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `calendardb`.`user_calendar`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `calendardb`.`user_calendar` ;

SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS `calendardb`.`user_calendar` (
  `event_id` INT NOT NULL,
  `author_id` INT NOT NULL,
  PRIMARY KEY (`author_id`),
  CONSTRAINT `get_author_fk1`
    FOREIGN KEY (`author_id`)
    REFERENCES `calendardb`.`user` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `get_event_fk2`
    FOREIGN KEY (`event_id`)
    REFERENCES `calendardb`.`calendar_event` (`event_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
AUTO_INCREMENT = 6
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `calendardb`.`work_plan`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `calendardb`.`work_plan` ;

SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS `calendardb`.`work_plan` (
  `user_id` INT NOT NULL,
  `department_id` INT NOT NULL,
  `position_id` INT NOT NULL,
  `company_id` INT NOT NULL,
  PRIMARY KEY (`user_id`),
  CONSTRAINT `Work_plan_fk0`
    FOREIGN KEY (`user_id`)
    REFERENCES `calendardb`.`user` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
```

```
  CONSTRAINT `Work_plan_fk1`
    FOREIGN KEY (`department_id`)
    REFERENCES `calendardb`.`department` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `Work_plan_fk2`
    FOREIGN KEY (`position_id`)
    REFERENCES `calendardb`.`position` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `Work_plan_fk3`
    FOREIGN KEY (`company_id`)
    REFERENCES `calendardb`.`company` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

SHOW WARNINGS;
USE `calendardb` ;
```

5 Ділова модель

Таблиця / Функції	Користувач	Події	Топік	Локація	Місце праці
Створення подій	*	*	*	*	
Запросити користувачів на подію	*	*			
Реєстрація (поглиблена)	*			*	*
Реєстрація (початкова)	*				
Перегляд місць проведення подій		*		*	
Перегляд запрошень	*	*			
Перегляд подій	*	*	*	*	

Таблиця 1. Ділова модель.

Операція “Створення події”

Для створення події ми дістаємо id користувача який відтепер є автором цієї події. Вносимо дані в таблицю події - заповнюємо обов'язкові поля “опис”, “Дата і час проведення”, “Тривалість” додатковим є вибір вставити посилання на подію.
З таблиці локації ми отримуємо id локації для подальшого приєднання події до назви та адреси локації. З таблиці топік ми використовуємо id топіка для подальшого приєднання його назви до події.

Операція “Запросити користувачів на подію”

Ми об’єднуємо таблиці ‘користувач’, ‘подія’ та дві допоміжні таблиці ‘user_calendar’ та ‘calendar_event’, що допомагають організувати зв’язок багато до багатьох, за id користувача-автора та id події. У таблицю ‘calendar_event’ записуються id користувачів, які є залоговані в системі та прийняли запрошення від автора на дану подію.

Операція “Реєстрація (поглиблена)”

Для поглибленої реєстрації користувач заповнює повністю таблицю користувача а також вказує належність до таблиць department, position та company у таблиці work_plan.

Операція “Реєстрація (початкова)”

Для початкової реєстрації користувач заповнює чотири необхідні поля такі як: пошту, пароль та йому потрібно підтвердити свою пошту за допомогою листа який він отримає.

Операція “Перегляд місць проведення подій”

З таблиці події ми витягаємо id події та з таблиці локації через допоміжну таблицю “ewent_location” ми витягаємо id локації.

Операція “Перегляд запрошень”

При логуванні користувача після переходу на сторінку підтвердження запрошення з таблиці користувача беремо id і виводимо ті id подій з допоміжної таблиці calendar_event, де наше id співпадає з id запрошеного.

Операція “Перегляд подій”

З таблиці користувача ми беремо id користувача та шукаємо його входження в об'єднання таблиць “user_calendar”, “calendar event” та “event”.

Операція “Перегляд користувачів запрошених на подію”

Ми виводимо входження id користувача (автора та гостя) в об’єднання таблиць ‘користувач’, ‘подія’ та допоміжну таблицю ‘calendar_event’. У таблицю ‘calendar_event’ записуються id користувачів, які є залоговані в системі та прийняли запрошення від автора на дану подію.

5 Запити до БД

- Заповнення тестовими даними даними

```
INSERT INTO user (`name`, `surname`, `e-mail`, `password`, `verified`, `birthdate`)
VALUES ('Dragutin', 'Violette', 'chiang@everyusb.org', 'dgsdgrefd', 1, '2001-03-03'),
    ('Hero', 'Predrag', 'robekaffomu-3474@yopmail.com', 'sdgedx8cv', 1, '2001-06-08'),
    ('Jathbiyya', 'Gudrun', 'owogenaw-4992@yopmail.com', 'fasrhbf8', 1, '2000-03-01'),
    ('Samara', 'Cajsa', 'ugessymmudd-0179@yopmail.com', 'gfdsv7f58', 1, '2003-05-08'),
    ('Filippo', 'Hossam', 'awasaxov-2999@yopmail.com', 'kjfdgkhdvc7', 1, '2002-02-01');

INSERT INTO company (`name`)
VALUES ('SoftServe'),
    ('NU LP'),
    ('GlobalLogic');

INSERT INTO position (`name`)
VALUES ('Student'),
    ('HR'),
    ('Developer'),
    ('Teacher');

INSERT INTO department (`name`)
VALUES ('Office 1'),
    ('Building 10');

SELECT *
FROM position;

INSERT INTO work_plan (user_id, department_id, position_id, company_id)
VALUES (10, 1, 1, 1),
    (12, 2, 2, 2),
    (13, 1, 3, 3),
    (14, 2, 4, 3);

INSERT INTO `event` (description, datetime, duration, link)
VALUES ('event related something 1', '2020-10-10', '10',
'http://1eventrelatedsomething.com'),
    ('event related something 2', '2020-9-9', '10', 'http://2eventrelatedsomething.com'),
    ('event related something 3', '2020-8-8', '10', 'http://3eventrelatedsomething.com'),
    ('event related something 4', '2020-7-7', '10', 'http://4eventrelatedsomething.com'),
    ('event related something 5', '2020-6-6', '10', 'http://5eventrelatedsomething.com');

INSERT INTO `topic` (`name`)
VALUES ('IT'),
    ('DB'),
    ('WEB'),
    ('ML');

SELECT * FROM event;
```

```
INSERT INTO event_to_topic (event_id, topic_id)
VALUES (22, 5), (23, 6), (24, 7), (25, 8);

INSERT INTO calendar_event (guest_id, event_id)
VALUES (10, 22),
       (11, 23),
       (12, 24),
       (13, 25),
       (12, 26);
```

- Запит, який повертає інформацію про працевлаштування всіх користувачів

```
SELECT user.name    name,
       user.surname surname,
       c.name       company_name,
       d.name       department,
       p.name       position
FROM user
         INNER JOIN work_plan wp on user.id = wp.user_id
         INNER JOIN company c on wp.company_id = c.id
         INNER JOIN department d on wp.department_id = d.id
         INNER JOIN position p on wp.position_id = p.id;
```

Результат виконання:

	name	surname	company_name	department	position
1	Jathbiyya	Gudrun	NU LP	Building 10	HR
2	Filippo	Hossam	GlobalLogic	Building 10	Teacher
3	Dragutin	Violette	SoftServe	Office 1	Student
4	Samara	Cajsa	GlobalLogic	Office 1	Developer

- Процедура, яка робить запит працевлаштування користувача

```
delimiter //
CREATE PROCEDURE get_work_info(IN id_ INT)
BEGIN
    SELECT user.name    name,
           user.surname surname,
           c.name       company_name,
           d.name       department,
           p.name       position
    FROM user
             INNER JOIN work_plan wp on user.id = wp.user_id
             INNER JOIN company c on wp.company_id = c.id
             INNER JOIN department d on wp.department_id = d.id
             INNER JOIN position p on wp.position_id = p.id
    WHERE user.id = id_;
```

```
END//
delimiter ;

CALL get_work_info(10);
```

Результат виконання:

	name	surname	company_name	department	position
1	Dragutin	Violette	SoftServe	Office 1	Student

- Запит, який повертає назву теми та кількість подій, пов'язану з нею

```
WITH topics_count AS (
   SELECT topic_id, COUNT(*) count
   FROM event_to_topic
   GROUP BY topic_id
)
SELECT name, count
FROM topics_count
       INNER JOIN topic ON topic_id = topic.id;
```

- Процедура, яка повертає назву теми та кількість подій, пов'язану з нею

```
delimiter //
CREATE PROCEDURE topics_event_count()
BEGIN
   WITH topics_count AS (
      SELECT topic_id, COUNT(*) count
      FROM event_to_topic
      GROUP BY topic_id
   )
   SELECT name, count
   FROM topics_count
          INNER JOIN topic ON topic_id = topic.id;
END//
delimiter ;

CALL topics_event_count();
```

Результат виконання обох вище наведених запитів:

	name	count
1	DB	1
2	IT	1
3	ML	1
4	WEB	1

- Запит, який повертає список подій, на які доданий користувач

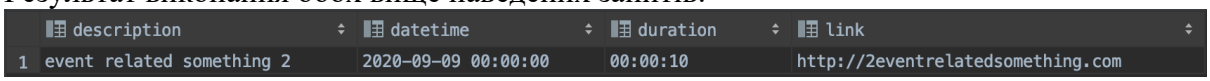
```
SELECT description, datetime, duration, link
FROM calendar_event
        INNER JOIN event e on calendar_event.event_id = e.id
WHERE event_id = 22;
```

- Процедура, яка повертає список подій, на які доданий користувач

```
delimiter //
CREATE PROCEDURE get_list_of_events(IN id_ INT)
BEGIN
   SELECT description, datetime, duration, link
   FROM calendar_event
          INNER JOIN event e on calendar_event.event_id = e.id
   WHERE event_id = id_;
END//
delimiter ;

CALL get_list_of_events(23);
```

Результат виконання обох вище наведених запитів:



	description	datetime	duration	link
1	event related something 2	2020-09-09 00:00:00	00:00:10	http://2eventrelatedsomething.com

6 Висновок

Взявши участь у даному проекті, я, окрім того, що досить сильно покращив навички стосовно баз даних, був активним спів розробником архітектури бази даних, яка буде використовуватися у глобальному проекті для нашої кафедри. Я покращив навички у створенні функцій та процедур, які сильно пришвидшують зв’язок між сервером та базою даних. Я відкоригував велику частину таблиць, зіткнувшись із певними перешкодами безпосередньо при розробці сайту. Я інтегрував деякий функціонал у сервер, використовуючи сформовані запити до бази даних. Я написав запит, який допоміг при розробці інтерфейсу користувача. Цей запит повертає інформацію про працевлаштування користувача. Також я навчився імпортувати базу даних із одного сервера на інший, що сильно заощаджує час при командній розробці проекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інтернет джерело: https://losst.ru/sravnenie-mysql-i-postgresql

2. Інтернет джерело: https://www.upguard.com/articles/postgresql-vs-mysql

3. Пасічник В.В., Резніченко В.А. Організація баз даних та знань - К.: Видавнича група BHV, 2006. — 384 с.: іл. — ISBN 966-552-156-X.

4. Coronel C., Morris S. Database Systems: Design, Implementation, and Management. 12th ed. – Cengage Learning, 2017. – 818 p.

5. Connolly T.M., Begg C.E. Database Systems: A Practical Approach to Design, Implementation and Management: Global Edition. – 6th Edition. – Pearson Education, 2015. – 1440 p.

6. Kroenke D.M., Auer D.J. Database Processing: Fundamentals, Design, and Implementation. 14th ed. – Pearson Education Ltd., 2016. – 638 p.

7. Elmasri R., Navathe S.B. Fundamentals of Database Systems. 7th ed. – Addison Wesley, 2016. – 1272 p.

8. Foster E.C., Godbole S. Database Systems: A Pragmatic Approach. Second Edition. – Apress, 2016. – 619 p.

9. Powell G. Beginning Database Design. – Wrox, 2006. – 500 p.

9. Bagui S., Earp R. Database Design Using Entity-Relationship Diagrams. 2nd ed. – CRC Press, 2011. – 362 p.
10. Hernandez M.J. Database Design for Mere Mortals. 3rd Edition. – Addison-Wesley Professional, 2013. – 672 p.

11. Dewson R. Beginning SQL Server for Developers. 4th ed. – Apress, 2015. – 670 p.

12. Петкович Душан. Microsoft SQL Server 2012. Руководство для начинающих. СПб.: БХВ-Петербург, 2013. — 816 с.

13. http://enisey.name/umk/teis/ch18s04s09.html

14. Нотация Баркера