Державний вищий навчальний заклад
«Прикарпатський національний університет імені Василя Стефаника»

Кафедра інформаційних технологій

# КУРСОВИЙ ПРОЕКТ

з дисципліни “Бази даних”
на тему: Проектування та розробка бази даних “Зберігання службових документів”

Студента 3 курсу, групи ІПЗ-3
напряму підготовки (спеціальності)
«Інженерія програмного забезпечення»
**Щербія Н.Б**
Керівник старший викладач
**Козич О.В.**

Національна шкала: _____
Університетська шкала:_____
Оцінка ECTS: _____

Члени комісії: _____ _____
(підпис)(прізвище та ініціали)
_____ _____
(підпис)(прізвище та ініціали)
_____ _____
(підпис)(прізвище та ініціали)

м. Івано-Франківськ – 2019 рік

Державний вищий навчальний заклад
„Прикарпатський національний університет імені Василя Стефаника”

# ЗАВДАННЯ НА КУРСОВИЙ ПРОЕКТ

_____
(прізвище, ім’я, по батькові студента)
Кафедра _____ Дисципліна _____
Спеціальність _____ Курс ____ Група _____ Семестр _____
1. Тема проекту_____
_____
_____
2. Рекомендована література_____
_____
_____
_____
_____
_____
_____
_____
3. Перелік питань, які підлягають розробці _____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
4. Дата видачі завдання _____
Термін подачі до захисту_____
5. Студент _____ Керівник_____

# КАЛЕНДАРНИЙ ПЛАН

| Пор. № | Назва етапів підготовки курсового проекту | Термін виконання | Форма контролю |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# РЕФЕРАТ

Пояснювальна записка: 30 сторінок, 9 рисунків, 4 джерела.

Ключові слова: БД, SQL, MYSQL

Об`єктом дослідження є використання системи керування реляційними базами даних MYSQL.

Мета роботи – створити базу даних для зберігання службових документів.

Стислий опис тексту пояснювальної записки:

У даному курсовому проекті описано основні етапи проектування та розробки бази даних “Зберігання службових документів”.

## ABSTRACT

Explanatory note: 30 pages, 9 figures, 4 sources.

Keywords: DB, SQL, MYSQL.

The object of the study is to use the MYSQL relational database management system.

The purpose of the work is to create a database for storing business documents.

Brief description of the text of the explanatory note:

This course project describes the basic stages of designing and developing a database for storing business documents.

**ЗМІСТ**

# ВСТУП

Під час інформаційної діяльності людина збирає і накопичує відомості про все, що її оточує. До появи обчислювальної техніки вся інформація зберігалася у письмовому або друкованому вигляді. Однак зі збільшенням обсягів інформації загострювалося питання її зберігання та обробки. Щоб користувач легко міг знаходити потрібну інформацію, вона має бути організована певним чином. Це стосується не лише інформації у комп'ютері, а й будь-якої інформації про об'єкти реального світу. Постійна зміна параметрів суспільних явищ, вдосконалення технологій зумовлює значне збільшення обсягів використовуваних даних. Для впорядкування їх об'єднують в певні групи за класифікаційними ознаками, зокрема, тематикою, сферою застосування тощо. Це можуть бути архіви, описи майна і матеріалів, бухгалтерські документи, особисті справи відділів кадрів, кримінальні справи на злочинців тощо. І всюди для користування інформацією необхідні засоби для її систематизації і швидкого пошуку.

Саме накопичення і неможливість обробки великих даних без спеціального для цього інструменту зумовило створення бази даних.

# 1 ТЕОРІЯ БАЗ ДАНИХ

Для створення бази даних мною була вибрана мова програмування MySQL.

## 1.1 Бази даних

База даних (БД) - упорядкований набір логічно взаємопов'язаних даних, що використовується спільно, та призначений для задоволення інформаційних потреб користувачів. В загальному випадку база даних містить схеми, таблиці, подання, збережені процедури та інші об'єкти. Дані у базі організовують відповідно до моделі організації даних. Головним завданням БД є гарантоване збереження значних обсягів інформації та надання доступу до неї користувачеві або ж прикладній програмі. Таким чином БД складається з двох частин: збереженої інформації та системи управління нею. З метою забезпечення ефективності доступу записи даних організовують як множину фактів (елемент даних). Таким чином, сучасна база даних, крім саме даних, містить їх опис та може містити засоби для їх обробки.

### 1.1.1 Класифікація баз даних

Існує величезна кількість різновидів баз даних, що відрізняються за критеріями.

Класифікація БД за моделлю даних:

- ієрархічні,
- мережеві,
- реляційні,
- об'єктні,
- об'єктно-орієнтовані,
- об'єктно-реляційні.

Класифікація БД за вмістом:

- географічні.
- історичні.
- наукові.
- мультимедійні.

Класифікація БД за ступенем розподіленості:

- централізовані (зосереджені);
- розподілені.

### 1.1.2 Сфери використання баз даних

Первинним призначенням бази даних є зберігання масивів даних. Але їх широко використовують і для збереження адміністративної інформації та спеціалізованих даних, наприклад, для інженерних даних чи для економічних моделей. Прикладами використання баз даних можуть бути:

- автоматизовані системи обліку ;
- реєстри та каталоги;
- геоінформаційні системи;
- лінгвістичні бази даних, тобто машинні словники різного типу і призначення;
- бази даних транспортних систем;
- системи керування вмістом Інтернет-сайтів, які зберігають у базах даних інформацію про web-сторінки сайту (прикладом можуть бути широко вживані системи керування вмістом Joomla! та WordPress).

У сучасних інформаційних системах для забезпечення роботи з базами даних використовують системи керування базами даних (СКБД).

## 1.2 Системи керування базами даних

**Система управління базами даних** (*СКБД*) — набір взаємопов'язаних даних (база даних) і програм для доступу до цих даних. Надає можливості створення, збереження, оновлення та пошуку інформації в базах даних з контролем доступу до даних.

Сучасні СКБД забезпечують:

- набір засобів для підтримки таблиць і співвідношень між зв’язаними таблицями;
- введення, модифікацію інформації, пошук і подання інформації у текстовому або графічному вигляді;
- засоби програмування, за допомогою яких ви можете створювати власні додатки.

### 1.2.1 Функції СКБД

До функцій СКБД прийнято відносити такі:
Безпосереднє управління даними у зовнішній пам'яті.

Ця функція включає забезпечення необхідних структур зовнішньої пам'яті як для зберігання даних, що безпосередньо входять у БД, так і для службових цілей, наприклад, для прискорення доступу до даних в деяких випадках (звичайно для цього використовуються індекси). У деяких реалізаціях СКБД активно використовуються можливості існуючих файлових систем, в інших робота проводиться аж до рівня пристроїв зовнішньої пам'яті. Але користувачі у будь-якому випадку не зобов'язані знати, чи використовує СКБД файлову систему і якщо використовує, то як організовані файли. Зокрема, СКБД підтримує власну систему іменування об'єктів БД.

Управління буферами оперативної пам'яті.

СКБД, як правило, працюють з БД значного розміру; принаймні цей розмір звичайно істотно більший за доступну ємність оперативної пам'яті. Зрозуміло, що якщо при зверненні до будь-якого елемента даних проводитиметься обмін із зовнішньою пам'яттю, то вся система працюватиме із швидкістю пристрою зовнішньої пам'яті. Практично єдиним способом реального збільшення цієї швидкості є буферизація даних в оперативній пам'яті.

Управління транзакціями.

Транзакція - це послідовність операцій над БД, що розглядаються СКБД як єдине ціле. Або транзакція успішно виконується, і СКБД фіксує зміни БД, проведені цією транзакцією, в зовнішній пам'яті, або жодна з цих змін ніяк не відображається на стані БД. Поняття транзакції необхідне для підтримки логічної цілісності БД.

Журналізація.

Однією з основних вимог до СКБД є надійність зберігання даних у зовнішній пам'яті. Під надійністю зберігання розуміють те, що СКБД повинна бути у змозі відновити останній узгоджений стан БД після будь-якого апаратного або програмного збою. Звичайно розглядаються два можливі види апаратних збоїв: так звані м'які збої, які можна трактувати як раптовий зупин роботи комп'ютера (наприклад, аварійне виключення живлення), і жорсткі збої, що характеризуються втратою інформації на носіях зовнішньої пам'яті.

У будь-якому випадку для відновлення БД потрібно мати у своєму розпорядженні деяку додаткову інформацію. Іншими словами, підтримка надійності зберігання даних у БД вимагає надмірності зберігання даних, причому та частина даних, яка використовується для відновлення, повинна зберігатися особливо надійно. Найбільш поширеним методом підтримки такої надмірної інформації є ведення журналу змін БД.

Журнал - це особлива частина БД, недоступна користувачам СКБД і підтримувана з особливою ретельністю (іноді підтримуються дві копії журналу,

що розміщуються на різних фізичних дисках), до якої надходять записи про всі зміни основної частини БД. У різних СКБД зміни БД журналізуються на різних рівнях: іноді запис у журналі відповідає деякій логічній операції зміни БД (наприклад, операції видалення рядка з таблиці реляційної БД), іноді - мінімальної внутрішньої операції модифікації сторінки зовнішньої пам'яті; у деяких системах одночасно використовуються обидва підходи.

У всіх випадках дотримуються стратегії "попереднього" запису в журнал (так званого протоколу Write Ahead Log - WAL). Ця стратегія полягає у тому, що запис про зміну будь-якого об'єкта БД повинен потрапити у зовнішню пам'ять журналу раніше, ніж змінений об'єкт потрапить у зовнішню пам'ять основної частини БД. Відомо, що якщо в СКБД коректно дотримується протокол WAL, то за допомогою журналу можна вирішити усі проблеми відновлення БД після будь-якого збою.

Найпростіша ситуація відновлення - індивідуальний відкат транзакції. Строго кажучи, для цього не потрібен загальносистемний журнал змін БД.

Для відновлення БД після жорсткого збою використовують журнал і архівну копію БД.

Архівна копія - це повна копія БД до моменту початку заповнення журналу.

Підтримка мов БД. Для роботи з базами даних використовуються спеціальні мови, у цілому звані мовами баз даних.

У сучасних СКБД, як правило, підтримується єдина інтегрована мова, що містить усі необхідні засоби для роботи з БД, починаючи від її створення, і забезпечує базовий призначений для користувача інтерфейс з базами даних. Стандартною мовою найбільш поширених у даний час реляційних СКБД є мова SQL (Structured Query Language).

### 1.3 Різновиди баз даних

#### 1.3.1 Ієрархічні бази даних

Ієрархічні бази даних підтримують деревоподібну організацію інформації. Зв'язки між записами виражаються у вигляді відносин предок/нащадок, а в кожного запису є тільки один батьківський запис. Це допомагає підтримувати цілісність посилань. Коли запис видаляється з дерева, всі його нащадки також повинні бути вилучені.

Ієрархічні бази даних мають централізовану структуру, тобто безпеку даних легко контролювати. На жаль, певні знання про фізичний порядок зберігання записів все-таки необхідні, тому що відносини предок/нащадок реалізуються у вигляді фізичних покажчиків з одного запису на інші.

Це означає, що пошук запису здійснюється методом прямого обходу дерева. Записи, розміщені в одній половині дерева, шукаються швидше, ніж в іншій. Звідси виникає необхідність правильно впорядковувати записи, щоб час їх пошуку був мінімальним. Це важко, тому що не всі відношення, які існують у реальному світі, можна виразити в ієрархічній базі даних. Відношення «один до багатьох» є природними, але практично неможливо описати відношення «багато хто до багатьох» або ситуації, коли запис має кілька предків. Доти, поки у додатках будуть кодуватися відомості про фізичну структуру даних, будь-які зміни цієї структури будуть призводити до перекомпіляції.

#### 1.3.2 Мережні бази даних.

Мережна модель розширює ієрархічну модель, дозволяючи групувати зв'язки між записами у множини. З логічної точки зору зв'язок - це не сам запис. Зв'язки лише виражають відношення між записами. Як і в ієрархічній моделі,

зв'язки ведуть від батьківського запису до дочірнього, але цього разу підтримується множинне успадковування.

Відповідно до специфікації CODASYL мережна модель підтримує DDL (Data Definition Language - мова визначення даних) і DML (Data Manipulation Language - мова обробки даних). Це спеціальні мови, призначені для визначення структури бази даних і складання запитів. Незважаючи на їх наявність, програміст, як і раніше, повинен знати структуру бази даних.

У мережній моделі допускаються відношення «багато хто до багатьох», а записи не залежать один від одного. При видаленні запису віддаляються й всі його зв'язки, але не самі зв'язані записи.

У мережній моделі потрібно, щоб зв'язки встановлювалися між існуючими записами, для того щоб уникнути дублювання й перекручування цілісності. Дані можна ізолювати у відповідних таблицях і зв'язати із записами в інших таблицях.

Програмістові не потрібно піклуватися про те, як організується фізичне зберігання даних на диску. Це послабляє залежність додатків і даних. Але в мережній моделі потрібно, щоб програміст пам'ятав структуру даних при формуванні запитів.

Оптимальну структуру бази даних складно сформувати, а готову структуру важко змінювати. Якщо вид таблиці зазнає змін, усі відношення з іншими таблицями повинні бути встановлені заново, щоб не порушилася цілісність даних. Складність подібного завдання приводить до того, що програмісти найчастіше скасовують деякі обмеження цілісності заради спрощення додатків.

### 1.3.3 Реляційні бази даних.

У реляційній моделі база даних являє собою централізоване сховище таблиць, що забезпечує безпечний одночасний доступ до інформації з боку

багатьох користувачів. У рядках таблиць частина полів містить дані, що стосуються безпосередньо запису, а частина - посилання на записи інших таблиць. Таким чином, зв'язки між записами є невід'ємною частиною реляційної моделі.

Кожен запис таблиці має однакову структуру. Наприклад, у таблиці, що містить опис автомобілів, у всіх записів буде той самий набір полів: виробник, модель, рік випуску, пробіг і т.д. Такі таблиці легко зображувати у графічному вигляді.

У реляційній моделі досягається інформаційна й структурна незалежність. Записи не зв'язані між собою настільки, щоб зміна одного з них стосувалася інших, а зміна структури бази даних не обов'язково приводить до перекомпіляції працюючих з нею додатків.

У реляційних СКБД застосовується мова SQL, що дозволяє формулювати довільні, нерегламентовані запити. Це мова четвертого покоління, тому будь-який користувач може швидко навчитися робити запити. До того ж, існує безліч додатків, що дозволяють будувати логічні схеми запитів у графічному вигляді. Все це відбувається за рахунок жорсткості вимог до продуктивності комп'ютерів. На щастя, сучасні обчислювальні потужності більш ніж адекватні.

У реляційних базах даних можуть бути розходження у реалізації мови SQL, хоча це й не проблема реляційної моделі. Кожна реляційна СКБД реалізує якусь підмножину стандарту SQL плюс набір унікальних команд, що ускладнює завдання програмістам, які намагаються перейти від однієї СКБД до іншої. Доводиться робити нелегкий вибір між максимальною переносимістю й максимальною продуктивністю. У першому випадку потрібно дотримуватися мінімального загального набору команд, підтримуваних у кожній СКБД. У другому випадку програміст просто зосереджується на роботі в даній конкретній СКБД, використовуючи переваги її унікальних команд і функцій.

### 1.3.4 Об’єктно-орієнтована база даних (ООБД)

Об’єктно-орієнтована база даних дозволяє програмістам, які працюють із мовами третього покоління, інтерпретувати всі свої інформаційні сутності як об'єкти, що зберігаються в оперативній пам'яті. Додатковий інтерфейсний рівень абстракції забезпечує перехоплення запитів, що стосуються тих частин бази даних, які перебувають у постійному сховищі на диску. Зміни, внесені в об'єкти, оптимальним чином переносяться з пам'яті на диск.

Перевагою об’єктно-орієнтованих баз даних є спрощений код. Додатки одержують можливість інтерпретувати дані в контексті тієї мови програмування, на якій вони написані. Реляційна база даних повертає значення всіх полів у текстовому вигляді, а потім вони зводяться до локальних типів даних. В об’єктно-орієнтованих базах даних цей етап ліквідований. Методи маніпулювання даними завжди залишаються однаковими незалежно від того, перебувають дані на диску або в пам'яті.

Дані в об’єктно-орієнтованих базах даних здатні на вигляд будь-якої структури, яку можна виразити використовуваною мовою програмування. Відношення між сутностями також можуть бути досить складними. Об’єктно-орієнтована база даних управляє кеш-буфером об'єктів, переміщаючи об'єкти між буфером і дисковим сховищем у міру необхідності.

За допомогою об’єктно-орієнтованих баз даних вирішуються дві проблеми. По-перше, складні інформаційні структури виражаються в них краще, ніж у реляційних базах даних, а по-друге, усувається необхідність транслювати дані з того формату, що підтримується в СКБД. Наприклад, у реляційній СКБД розмірність цілих чисел може становити 11 цифр, а у використовуваній мові програмування - 16. Програмістові потрібно враховувати цю ситуацію.

Об’єктно-орієнтовані СКБД виконують багато додаткових функцій. Це окупається сповна, якщо відношення між даними дуже складні. У такому разі

продуктивність об’єктно-орієнтованих баз даних виявляється вищою, ніж у реляційних СКБД. Якщо ж дані менш складні, додаткові функції виявляються надлишковими. В об'єктній моделі даних підтримуються нерегламентовані запити, але мовою їх складання необов'язково є SQL. Логічне подання даних може не відповідати реляційній моделі, тому застосування мови SQL стане безглуздим. Найчастіше зручніше обробляти об'єкти в пам'яті, виконуючи відповідні види пошуку.

Великим недоліком об’єктно-орієнтованих баз даних є їх тісні зв'язки із застосовуваною мовою програмування. До даних, що зберігаються в реляційній СКБД, можуть належати будь-які додатки, тоді як, приміром, Java-об'єкт, поміщений в об’єктно-орієнтовану базу даних, буде становити інтерес лише для додатків, написаних на Java.

### 1.3.5 Об’єктно-реляційні бази даних.

Об’єктно-реляційні СКБД поєднують у собі риси реляційної й об'єктної моделей. Їх виникнення породжене тим, що реляційні бази даних добре працюють із убудованими типами даних і набагато гірше - з користувацькими, нестандартними. Коли з'являється новий важливий тип даних, доводиться або включати його підтримку в СКБД, або змушувати програміста самостійно управляти даними в додатку.

Не всяку інформацію має сенс інтерпретувати у вигляді ланцюжків символів або цифр. Уявимо собі музичну базу даних. Пісню, закодовану у вигляді аудіофайлу, можна помістити в текстове поле великого розміру, але як у такому випадку буде здійснюватися текстовий пошук?

Перебудова СКБД із метою включення в неї підтримки нового типу даних - не кращий вихід з положення. Замість цього об’єктно-реляційна СКБД дозволяє завантажувати код, призначений для обробки "нетипових" даних.

Таким чином, база даних зберігає свою табличну структуру, але спосіб обробки деяких полів таблиць визначається ззовні, тобто програмістом.

### 1.4 SQL

SQL (Structured query language — мова структурованих запитів) — декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД, створення схеми бази даних і її модифікація, система контролю за доступом до бази даних.. Сам по собі SQL не є ні системою керування базами даних, ні окремим програмним продуктом.

SQL – це діалогова мова програмування для здійснення запиту і внесення змін до бази даних, а також управління базами даних. Багато баз даних підтримує SQL з розширеннями до стандартної мови. Ядро SQL формує командна мова, яка дозволяє здійснювати пошук, вставку, обновлення, і вилучення даних, використовуючи систему управління і адміністративні функції. SQL також включає CLI (Call Level Interface) для доступу і управління базами даних дистанційно.

Перша версія SQL була розроблена на початку 1970-х років у IBM. Ця версія носила назву SEQUEL і була призначена для обробки і пошуку даних, що містилися в реляційній базі даних IBM, System R . Мова SQL пізніше була стандартизована Американськими Держстандартами (ANSI) в 1986. Спочатку SQL розроблялась як мова запитів і управління даними, пізніші модифікації SQL створено продавцями системи управління базами даних, які додали процедурні конструкції, control-of-flow команд і розширення мов. З випуском стандарту SQL:1999 такі розширення були формально запозичені як частина мови SQL через Persistent Stored Modules (SQL/PSM).

### 1.5 MySQL

MySQL — вільна система керування реляційними базами даних.

MySQL був розроблений компанією «TcX» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL — одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

MySQL виникла як спроба застосувати mSQL до власних розробок компанії: таблиць, для яких використовувалися ISAM — підпрограми низького рівня для індексного доступу до даних. У результаті був вироблений новий SQL-інтерфейс, але API-інтерфейс залишився в спадок від mSQL. Звідки походить назва «MySQL» — достеменно не відомо. Розробники дають два варіанти: або тому, що практично всі напрацювання компанії починалися з префікса My, або на честь дівчинки на ім'я My, дочки Майкла Монті Віденіуса, одного з розробників системи.

MySQL — компактний багатопотоковий сервер баз даних. Характеризується високою швидкістю, стійкістю і простотою використання. MySQL вважається гарним рішенням для малих і середніх застосувань. Сирцеві коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатопоточності, що підвищує продуктивність системи в цілому. Можливості сервера MySQL:

- простота у встановленні та використанні;
- підтримується необмежена кількість користувачів, що одночасно працюють із БД;
- кількість рядків у таблицях може досягати 50 млн;
- висока швидкість виконання команд;
- наявність простої і ефективної системи безпеки.

# 2 ПРОЕКТУВАНЯ БАЗИ ДАНИХ

## 2.1 Постановка задачі

Темою даної курсової роботи є база даних «Зберігання службових документів».

## 2.2 Опис сутностей

Зберігання і ведення службових документів є досить клопіткою роботою. Завдяки цій базі даних, робота з ведення документації може бути спрощеною.

Для цього ми створили 5 таблиць.

Перша таблиця “Documents” містить інформацію про документи.

Ця таблиця містить в собі наступні поля:

- id – ідентифікаційний номер документа;
- name – назва документа;
- creation_date – дата створення документа;
- author_name – ім’я особи, що створила документ;
- document_type – вид документу;
- description – являє собою короткий опис документа.

Друга таблиця “Documents_in_Process” містить інформацію про документи на певних стадіях документообігу.

Вона містить в собі наступні поля:

- document_id
- register_id
- staff
- current_state

Третя таблиця “Registers” містить в собі всі документи які були зареєстровані на певному факультеті.

Вона містить в собі наступні поля:

- register_id – ідентифікаційний номер журналу кафедри;

- document_id – ідентифікаційний номер документа;
- document_author_id – автор документа;
- registration_date – дата реєстрації документа в журналі;
- document_state – стадія документообігу;
- faculty_id - ідентифікаційний номер факультету.

Четверта таблиця “Documents_Types” містить в собі види документів і містить в собі наступні поля.

- type_id - ідентифікаційний номер виду документа;
- type_name – назва виду документа.

П’ята таблиця “Authors” містить в собі авторів документів:

- author_id - ідентифікаційний номер автора;
- author_name – ім’я автора.

Шоста таблиця “Faculties” містить в собі інформація про факультети:

- faculty_id - ідентифікаційний номер факультету;
- faculty_name – назва факультету.

Сьома таблиця “Categories” містить в собі дані про журнали різних кафедр:

- categoty_id - ідентифікаційний номер категорії журнала;
- category_name – назва категорії журнала кафедри;
- end_period – дата закінчення терміна дії журнала.

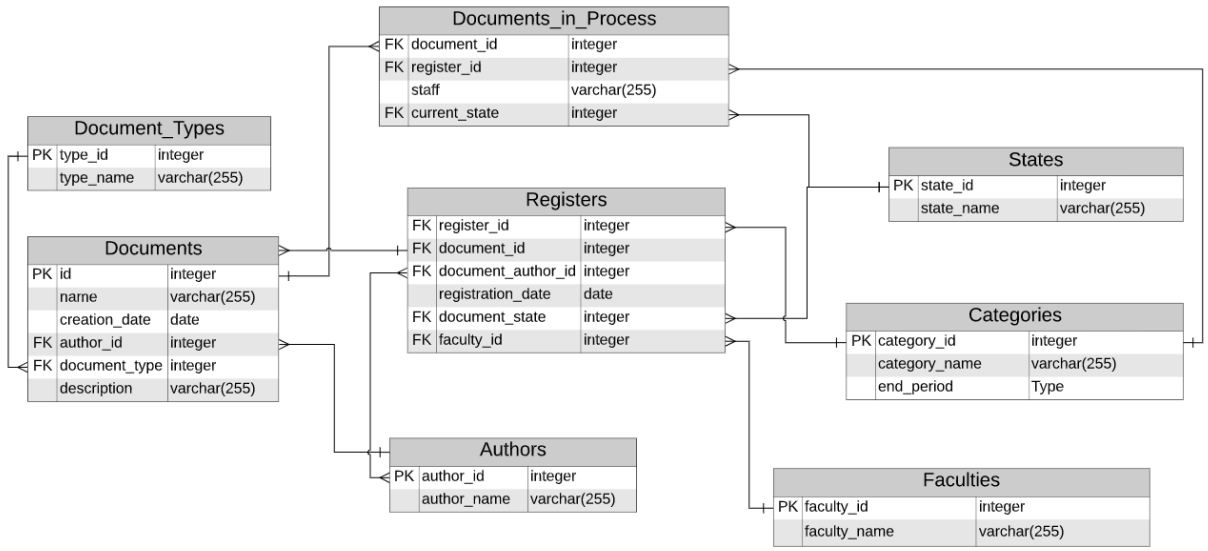
Восьма таблиця “States” містить в собі стани документів на певних етапах:

- state_id - ідентифікаційний номер стану;
- state_name – назва стану.

## 2.3 Відношення сутностей:

- Documents -- (N..1) -- Document_Types
- Documents -- (N..1) -- Authors
- Documents -- (N..1) -- Registers
- Register -- (N..N) -- Authors
- Register -- (N..1) -- Faculties
- Register -- (N..1) -- Categories
- Register -- (N..1) -- States
- Documents_in_Process -- (N..1) -- Documents
- Documents_in_Process -- (N..1) -- Categories
- Documents_in_Process -- (N..N) -- State

## 2.4 ER діаграма



# 3 СТВОРЕННЯ БАЗИ ДАНИХ

## 3.1 Створення таблиць

Створення таблиці “Documents”.

```
CREATE TABLE `Documents` (
      `id` integer NOT NULL UNIQUE AUTO_INCREMENT,
       `name` varchar(255),
       `creation_date` date,
       `author_id` integer,
       `document_type` integer,
       `description` varchar(255),
  PRIMARY KEY (`id`),
  FOREIGN KEY (`author_id`) REFERENCES Authors(`author_id`) ON
UPDATE CASCADE ON DELETE SET NULL,
 FOREIGN KEY (`document_type`) REFERENCES
Document_Types(`type_id`) ON UPDATE CASCADE ON DELETE SET NULL
 );
```

Створення таблиці “Registers”.

```
CREATE TABLE `Registers` (
       `register_id` integer,
       `document_id` integer,
       `document_author_id` integer,
       `registration_date` date,
       `document_state` integer,
       `faculty_id` integer,
  FOREIGN KEY (`register_id`) REFERENCES
Categories(`category_id`) ON UPDATE CASCADE ON DELETE SET NULL
,
  FOREIGN KEY (`document_id`) REFERENCES  Documents(`id`)ON
UPDATE CASCADE ON DELETE CASCADE,
  FOREIGN KEY (`document_author_id`) REFERENCES
Authors(`author_id`)ON UPDATE CASCADE ON DELETE SET NULL,
  FOREIGN KEY (`document_state`) REFERENCES States(`state_id`)
ON UPDATE CASCADE,
    FOREIGN KEY (`faculty_id`) REFERENCES
Faculties(`faculty_id`) ON UPDATE CASCADE
);
```

Створення таблиці “Documents_in_Process”.

```
CREATE TABLE `Documents_in_Process` (
       `document_id` integer NOT NULL AUTO_INCREMENT,
       `register_id` integer,
       `staff` varchar(255),
       `current_state` integer,
  FOREIGN KEY (`document_id`) REFERENCES Documents(`id`)ON
UPDATE CASCADE ON DELETE CASCADE,
  FOREIGN KEY (`register_id`) REFERENCES
Categories(`category_id`)ON UPDATE CASCADE,
  FOREIGN KEY (`current_state`) REFERENCES States(`state_id`)
ON UPDATE CASCADE
```

```
);
```

Створення таблиці “Document_Types”.

```
CREATE TABLE `Document_Types` (
      `type_id` integer NOT NULL UNIQUE AUTO_INCREMENT,
      `type_name` varchar(255),
  PRIMARY KEY (`type_id`)
);
```

Створення таблиці “Categories”.

```
CREATE TABLE `Categories` (
     `category_id` integer NOT NULL UNIQUE AUTO_INCREMENT,
      `category_name` varchar(255),
      `end_period` date,
  PRIMARY KEY (`category_id`)
);
```

Створення таблиці “States”.

```
CREATE TABLE `States` (
       `state_id` integer NOT NULL UNIQUE AUTO_INCREMENT,
       `state_name` varchar(255),
  PRIMARY KEY (`state_id`)
);
```

Створення таблиці “Faculties”.

```
CREATE TABLE `Faculties` (
       `faculty_id` integer NOT NULL UNIQUE AUTO_INCREMENT,
       `faculty_name` varchar(255),
  PRIMARY KEY (`faculty_id`)
);
```

Створення таблиці “Authors”.

```
CREATE TABLE `Authors` (
       `author_id` integer NOT NULL UNIQUE AUTO_INCREMENT,
       `author_name` varchar(255) UNIQUE,
  PRIMARY KEY (`author_id`)
);
```

### 3.2 Наповнення таблиць даними.

Наповнення даними “Documents”.

```
INSERT INTO documents
(name,creation_date,author_id,document_type,description)
VALUES ('Заява на отримання соціальної стипендії','2019-09-
20',4,1,'група ПМ-3');

INSERT INTO documents
(name,creation_date,author_id,document_type,description)
```

```
VALUES ('Заява на видачу дубліката залікової книжки','2019-09-
20',6,1,'група ІПЗ-3');

INSERT INTO documents
(name,creation_date,author_id,document_type,description)
VALUES ('Заява на відрахування із числа студентів за власним
бажанням','2019-12-17',5,1,'група ІПЗ-3');

INSERT INTO documents
(name,creation_date,author_id,document_type,description)
VALUES ('Доповідна записка про відрядження','2019-12-
18',8,4,'-');

INSERT INTO documents
(name,creation_date,author_id,document_type,description)
VALUES ('Пояснювальна записка','2019-12-18',1,2,'група А-13');
```

Наповнення даними “Registers”.

```
INSERT INTO
registers(registers.register_id,registers.document_id,register
s.document_author_id,registers.registration_date,registers.doc
ument_state,registers.faculty_id)
VALUES(2,1,4,'2019-09-21',3,1);

INSERT INTO
registers(registers.register_id,registers.document_id,register
s.document_author_id,registers.registration_date,registers.doc
ument_state,registers.faculty_id)
VALUES(1,2,6,'2019-09-21',3,1);

INSERT INTO
registers(registers.register_id,registers.document_id,register
s.document_author_id,registers.registration_date,registers.doc
ument_state,registers.faculty_id)
VALUES(1,3,5,'2019-12-18',2,1);

INSERT INTO
registers(registers.register_id,registers.document_id,register
s.document_author_id,registers.registration_date,registers.doc
ument_state,registers.faculty_id)
VALUES(4,4,8,'2019-12-20',1,2);

INSERT INTO
registers(registers.register_id,registers.document_id,register
s.document_author_id,registers.registration_date,registers.doc
ument_state,registers.faculty_id)
VALUES(5,5,1,'2019-12-20',3,2);
```

Наповнення даними “Documents_in_Process”.

```
INSERT INTO
documents_in_process(documents_in_process.document_id,document
```

```
s_in_process.register_id,documents_in_process.staff,documents_
in_process.current_state)
VALUES (1,2,'Пилипів Володимир Михайлович',3);


INSERT INTO
documents_in_process(documents_in_process.document_id,document
s_in_process.register_id,documents_in_process.staff,documents_
in_process.current_state)
VALUES (2,1,'Пилипів Володимир Михайлович',3);


INSERT INTO
documents_in_process(documents_in_process.document_id,document
s_in_process.register_id,documents_in_process.staff,documents_
in_process.current_state)
VALUES (3,1,'Соломко Андрій Васильович',2);


INSERT INTO
documents_in_process(documents_in_process.document_id,document
s_in_process.register_id,documents_in_process.staff,documents_
in_process.current_state)
VALUES (4,5,'Венгринович Андрій Антонович',1);

INSERT INTO
documents_in_process(documents_in_process.document_id,document
s_in_process.register_id,documents_in_process.staff,documents_
in_process.current_state)
VALUES (5,5,'Яцків Наталія Яремівна',3);
```

Наповнення даними “Document_Types”.

```
INSERT INTO document_types(type_name)
VALUES ('Заява');

INSERT INTO document_types(type_name)
VALUES ('Пояснювальна записка');

INSERT INTO document_types(type_name)
VALUES ('Службова записка');

INSERT INTO document_types(type_name)
VALUES ('Доповідна записка');
```

Наповнення даними “Categories”.

```
INSERT INTO
categories(categories.category_name,categories.end_period)
VALUES ('Журнал для заяв студентів 3-го курсу кафедри
інформаційних технологій','2019-12-29');

INSERT INTO
categories(categories.category_name,categories.end_period)
```

```
VALUES ('Журнал для заяв студентів 3-го курсу кафедри
диференціальних рівнянь і прикладної математики','2019-12-
29');

INSERT INTO
categories(categories.category_name,categories.end_period)
VALUES ('Журнал для заяв студентів 3-го курсу кафедри
англійської філології','2019-12-29');

INSERT INTO
categories(categories.category_name,categories.end_period)
VALUES ('Журнал для доповідних записок студентів 3-го курсу
кафедри англійської філології','2019-12-29');

INSERT INTO
categories(categories.category_name,categories.end_period)
VALUES ('Журнал для пояснювальних записок студентів 3-го курсу
кафедри англійської філології','2019-12-29');
```

Наповнення даними “States”.

```
INSERT INTO states(states.state_name)
VALUES ('Надходження');

INSERT INTO states(states.state_name)
VALUES ('Розгляд');

INSERT INTO states(states.state_name)
VALUES ('Підтверджено');

INSERT INTO states(states.state_name)
VALUES ('Відхилено');
```

Наповнення даними “ Faculties”.

```
INSERT INTO faculties(faculty_name)
VALUES ('Факультет математики та інформатики');


INSERT INTO faculties(faculties.faculty_name)
VALUES('Факультет іноземних мов');
```

Наповнення даними “ Authors”.

```
INSERT INTO authors(author_name)
VALUES ('Артюхов Роман Вячеславович');

INSERT INTO authors(author_name)
VALUES ('Білан Максим Олександрович');

INSERT INTO authors(author_name)
VALUES ('Грінченко Анна Максимівна');
```

```
INSERT INTO authors(author_name)
VALUES ('Дєєв Сергій Сергійович');

INSERT INTO authors(author_name)
VALUES ('Злосчастьєв Данило Костянтинович');

INSERT INTO authors(author_name)
VALUES ('Зюбіна Анна Володимирівна');

INSERT INTO authors(author_name)
VALUES ('Огійчук Артем Олександрович');

INSERT INTO authors(author_name)
VALUES ('Поліщук Орест Володимирович');
```

### 3.3 Написання запитів до бази даних.

Обрати всі документи які були погоджені(розглянуті).

```
SELECT
documents.name ,authors.author_name,
registers.register_id,states.state_name
FROM registers
JOIN documents ON documents.id = registers.document_id
JOIN authors ON authors.author_id = documents.author_id
JOIN states ON registers.document_state = states.state_id
WHERE registers.document_state = 3;
```

| name | author_name | register_id | state_name |
|---|---|---|---|
| Заява на отримання соціальної стипендії | Дєєв Сергій Сергійович | 2 | Підтверджено |
| Заява на видачу дубліката залікової книжки | Зюбіна Анна Володимирівна | 1 | Підтверджено |
| Пояснювальна записка | Артюхов Роман Вячеславович | 5 | Підтверджено |

Обрати всі документи, вид яких - заява.

```
SELECT
documents.name , documents.creation_date ,
document_types.type_name FROM documents
JOIN document_types ON documents.document_type =
document_types.type_id
WHERE document_types.type_name = 'Заява';
```

| name | creation_date | type_name |
|---|---|---|
| Заява на отримання соціальної стипендії | 2019-09-20 | Заява |
| Заява на видачу дубліката залікової книжки | 2019-09-20 | Заява |
| Заява на відрахування із числа студентів за власни... | 2019-12-17 | Заява |

Вивести назви журналів і документів, які належать до факультету математики та інформатики.

```
SELECT
categories.category_name,documents.name,faculties.faculty_name
FROM registers
JOIN categories
ON registers.register_id = categories.category_id
JOIN documents ON registers.document_id = documents.id
JOIN faculties ON registers.faculty_id = faculties.faculty_id
WHERE faculties.faculty_id = 1;
```

| category_name | name | faculty_name |
|---|---|---|
| Журнал для заяв студентів 3-го курсу кафедри інфор... | Заява на видачу дубліката залікової книжки | Факультет математики та інформатики |
| Журнал для заяв студентів 3-го курсу кафедри інфор... | Заява на відрахування із числа студентів за власни... | Факультет математики та інформатики |
| Журнал для заяв студентів 3-го курсу кафедри дифер... | Заява на отримання соціальної стипендії | Факультет математики та інформатики |

Вивести документи, дата створення яких пізніше 01 грудня 2019 року.

```
SELECT * FROM documents
WHERE documents.creation_date > '2019-12-01';
```

| id | name | creation_date | author_id | document_type | description |
|---|---|---|---|---|---|
| 3 | Заява на відрахування із числа студентів за власни... | 2019-12-17 | 5 | 1 | група ІПЗ-3 |
| 4 | Доповідна записка про відрядження | 2019-12-18 | 8 | 4 | - |
| 5 | Пояснювальна записка | 2019-12-18 | 1 | 2 | група А-13 |

## 3.4 Створення тригеру

```
DELIMITER |
CREATE TRIGGER trigger_update_register BEFORE UPDATE ON
documents_in_process
  FOR EACH ROW
BEGIN
    UPDATE registers SET registers.document_state =
NEW.current_state WHERE NEW.document_id =
registers.register_id;
  END
|
DELIMITER ;
```

Даний тригер буде реагувати на обновлення в таблиці "Documents_in_Process" і оновлювати стан документа з одинаковим ідентифікаційним номером в таблиці "Registers".

Приклад використання:

Таблиця “Documents_in_Process”:

| document_id | register_id | staff | current_state |
|---|---|---|---|
| 1 | 2 | Пилипів Володимир Михайлович | 3 |
| 2 | 1 | Пилипів Володимир Михайлович | 3 |
| 3 | 1 | Соломко Андрій Васильович | 2 |
| 4 | 5 | Венгринович Андрій Антонович | 1 |
| 5 | 5 | Яцків Наталія Яремівна | 3 |

Таблиця “Registers”:

| register_id | document_id | document_author_id | registration_date | document_state | faculty_id |
|---|---|---|---|---|---|
| 2 | 1 | 4 | 2019-09-21 | 3 | 1 |
| 1 | 2 | 6 | 2019-09-21 | 3 | 1 |
| 1 | 3 | 5 | 2019-12-18 | 2 | 1 |
| 4 | 4 | 8 | 2019-12-20 | 1 | 2 |
| 5 | 5 | 1 | 2019-12-20 | 3 | 2 |

Тепер зробимо UPDATE таблиці “Documents_in_Process”

```
UPDATE documents_in_process
SET documents_in_process.current_state = 2
WHERE document_id = 4;
```

В результаті обидві таблиці обновили стан одного і того ж документу.

Таблиця “Documents_in_Process”:

| document_id | register_id | staff | current_state |
|---|---|---|---|
| 1 | 2 | Пилипів Володимир Михайлович | 3 |
| 2 | 1 | Пилипів Володимир Михайлович | 3 |
| 3 | 1 | Соломко Андрій Васильович | 2 |
| 4 | 5 | Венгринович Андрій Антонович | 2 |
| 5 | 5 | Яцків Наталія Яремівна | 3 |

Таблиця “Registers”:

| register_id | document_id | document_author_id | registration_date | document_state | faculty_id |
|---|---|---|---|---|---|
| 2 | 1 | 4 | 2019-09-21 | 3 | 1 |
| 1 | 2 | 6 | 2019-09-21 | 3 | 1 |
| 1 | 3 | 5 | 2019-12-18 | 2 | 1 |
| 4 | 4 | 8 | 2019-12-20 | 2 | 2 |
| 5 | 5 | 1 | 2019-12-20 | 3 | 2 |

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. https://dl.sumdu.edu.ua/textbooks/55148/306150/index.html
2. https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85
3. http://lib.mdpu.org.ua/e-book/vstup/L5.htm
4. https://uk.wikipedia.org/wiki/MySQL