МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА

(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій

(повне найменування інституту, назва факультету (відділення))

Кафедра математичного забезпечення комп'ютерних систем

(повна назва кафедри (предметної, циклової комісії))

Дипломна робота

на здобуття ступеня вищої освіти «бакалавр»

(освітньо-кваліфікаційний рівень)

на тему «Платформа дистанційного навчання вищої математики для IT спеціальностей. Модуль управління даними»
«Distance learning platform for IT specialists. Data management module»

Виконала: студентка денної форми навчання
спеціальності 123 Комп'ютерна інженерія
(шифр і назва напряму підготовки, спеціальності)
Ткаченко Аліна Михайлівна
(прізвище, ім'я, по-батькові)

Керівник доц. Петрушина Т.
(науковий ступінь, вчене звання, прізвище та ініціали, підпис)

Рецензент доц. Пенко В. Г.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:

Протокол засідання кафедри

№___ від «___»_____ 2021 р.

Завідувач кафедри

_____ _Є.В. Малахов_
(підпис) (прізвище, ініціали)

Захищено на засіданні ЕК № ___

протокол № __ від «___» ____ 2021 р.

Оцінка _____/_____/_____
(за національною шкалою, шкалою ECTS, бали)

Голова ЕК

_____ _Н.Ф. Казакова_
(підпис) (прізвище, ініціали)

Одеса – 2021

АНОТАЦІЯ

Дипломна робота виконана в рамках комплексної розробки навчальної платформи вищої математики для IT спеціальностей.

Мета роботи – розробка і реалізація модуля управління даними у вигляді єдиного інструменту організації та управління даними для всіх модулів.

Розроблено бази даних для модулів з використанням CASE-технологій: реляційної бази даних контенту, документо-орієнтованої бази даних для модуля інтелектуальної обробки текстів.

Для апробації запропонованого рішення реалізований сервіс інтелектуальної обробки тексту у вигляді моделі тематичного моделювання.

Для модуля управління даними використані PowerDesigner, реляційна СУБД PostgreSQL і нереляційна СУБД MongoDB. Для тематичного моделювання використовувалися мова програмування Python, бібліотека тематичного моделювання Gensim, платформа для відстеження параметрів моделі MLFlow.

Розроблені компоненти інтегровані з загальну архітектуру.

ABSTRACT

Diploma work was performed in the framework of the integrated development of the high mathematics training platform for IT specialties.

The purpose of the work is to develop and implement the data management module in the form of a single tool for organizing and managing data for all modules.

Databases for modules using Case-technologies are developed: relational content database, document-oriented database for intellectual text processing module.

To approbate the proposed solution, the system of intellectual processing of text in the form of a model of topic modeling is implemented.

For data management module, PowerDesigner is used, PostgreSQL relational DBMS and a non-relational DBMS MongoDB. For topic modeling, a Python programming language, a Gensim thematic modeling library, a platform for tracking the MlFlow model parameters are used.

Developed components integrated with general architecture.

АННОТАЦИЯ

Дипломная работа выполнена в рамках комплексной разработки учебной платформы высшей математики для IT специальностей.

Цель работы – разработка и реализация модуля управления данными в виде единого инструмента организации и управления данными для всех модулей.

Разработаны базы данных для модулей с использованием CASE-технологий: реляционной базы данных контента, документо-ориентированной базы данных для модуля интеллектуальной обработки текстов.

Для апробации предложенного решения реализован сервис интеллектуальной обработки текста в виде модели тематического моделирования.

Для модуля управления данными использованы PowerDesigner, реляционная СУБД PostgreSQL и нереляционная СУБД MongoDB. Для тематического моделирования использовались язык программирования Python, библиотека тематического моделирования Gensim, платформа для отслеживания параметров модели MLFlow.

Разработанные компоненты интегрированы с общую архитектуру.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

ACID – atomicity, consistency, isolation, durability. Це набір властивостей, що гарантують надійну роботу транзакцій бази даних: атомарність, узгодженість, ізольованість, довговічність.

API – application programming interface (прикладний програмний інтерфейс)

CAP – consistency, availability, partition tolerance (узгодженість даних, доступність та стійкість до розділення).

ETL – extract, transform, load (витяг, перетворення та завантаження) Процес, який використовується в базах даних для забезпечення їх роботи для підтримки прийняття рішень.

СКБД – система керування базами даних

ВСТУП

Робота виконана в рамках комплексного розроблення навчальної платформи для IT-спеціалістів. Практична цінність управління даними полягає в створенні базового рішення для подальшого розвитку: описаний єдиний інструмент для організації, внесення змін та ведення документації.

Актуальність роботи полягає в створенні гнучкої архітектури даних для мікросервісної архітектури додатку. Мікросервісна архітектура передбачає значне розширення — унаслідок чого є необхідність організації різноманітних бізнес-процесів за однаковими правилами організації даних.

Метою роботи є розроблення гнучкої структури даних платформи, що передбачає створення моделей даних, розробку внутрішніх стандартів використання інструментів для специфікації, підтримку внесення змін у структуру і формування актуальної документації, а також реалізація і апробація рішення для управління даними для модуля інтелектуальної обробки тексту.

Задачі дипломної роботи організації роботи з даними:

– формалізувати вимоги до архітектури даних;

– розробити моделі даних для кількох модулів;

– розробити правила ведення документації по структурам даних;

– спроектувати та реалізувати сервіс інтелектуальної обробки тексту;

– на прикладі розробки сервісу інтелектуальної обробки тексту продемонструвати описані методи управління даними.

1 ФОРМАЛІЗАЦІЯ ВИМОГ ДО ДАНИХ

1.1 Вимоги предметної області до даних

На етапі розроблення вимог до інтерфейсу були описані вимоги до предметної області, які представлені у форматі Use-Case діаграми и сценаріїв взаємодії користувача з системою.

Проаналізувавши діаграму та вимоги до створюваного інтерфейсу користувача були сформовані наступні вимоги до структури даних модуля:

- створення, зберігання та змінення структури курсів;
- зберігання додаткового матеріалу до курсу;
- гнучка організація кроків для забезпечення їхньої різноманітності, легкої перестановки та додавання нових.

Основні сутності предметної області та їх ключові атрибути:

- курс: назва, опис, автор;
- модуль: назва, до якого курсу належить, порядок модулю;
- урок: назва, опис, позиція уроку в модулі, чи є урок необхідним для проходження;
- частина уроку: позиція, тип (текст, відео, тест)

Результатом опису є сформована концептуальна модель даних. Детальнішій опис сутностей, їх атрибутів та зв’язків наведено у додатку А.

2 УПРАВЛІННЯ ДАНИМИ

Міжнародна асоціація управління даними (Data Management Association International) дала таке визначення поняттю «Управління даними» :

Управління даними (англ. data management) — це процес, пов'язаний з накопиченням, організацією, запам'ятовуванням, оновленням, зберіганням даних і пошуком інформації [1].

На рисунку 2.1 зображено колесо DAMA. Воно визначає області знань по управлінню даними. У центрі розташоване стратегічне керування даними, оскільки воно включає планування, спостереження й управління виконанням

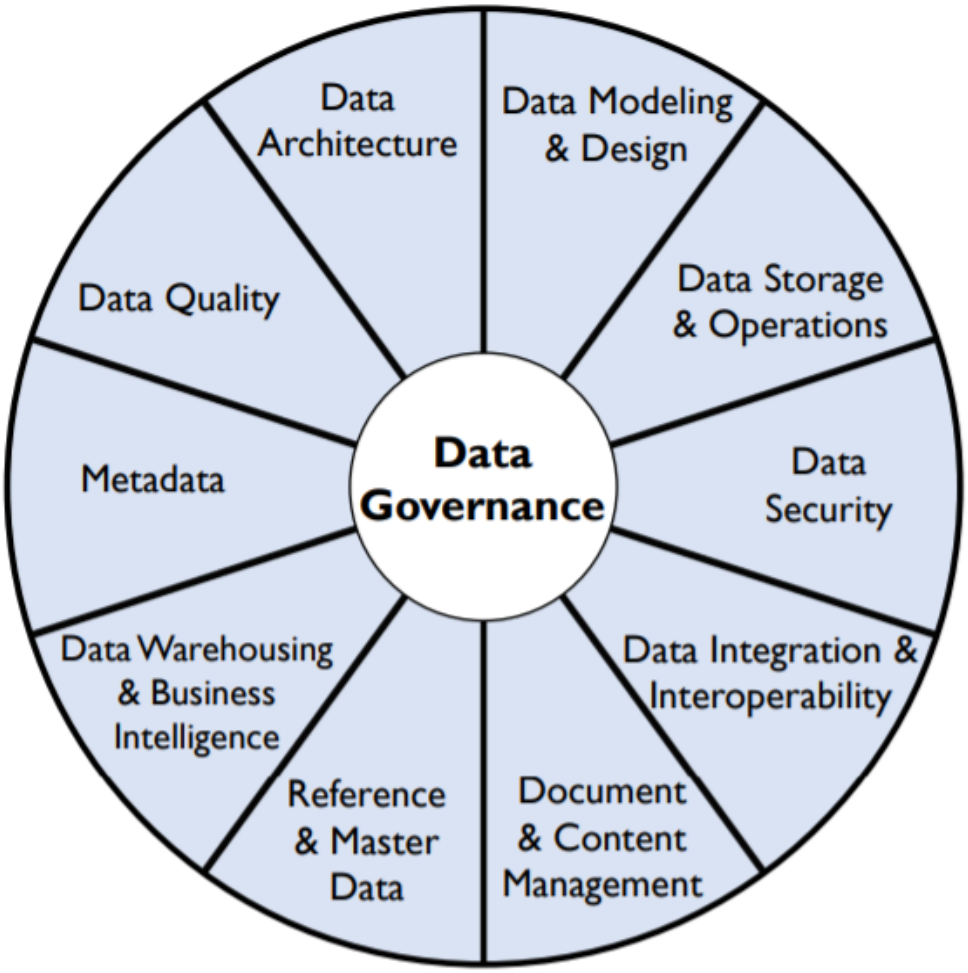

Рисунок 2.1 — Колесо DAMA

У рамках дипломної роботи розглядаються такі області управління даними:

- data architecture (архітектура даних) — методи організації й управління даними;
- data modeling & design (моделювання і дизайн даних) — структура даних у системі. До таких моделей належать концептуальна, логічна і фізична моделі;
- data security (безпека даних) — забезпечення конфіденційності даних, контроль доступу до даних;
- document & content management (управління документами та контентом) — опис системи, вимог до системи та даних;

2.1 Архітектура даних

Архітектура даних визначає план управління даними через узгодження зі стратегією організації встановлення стратегічних вимог до даних і проектів, цих вимог, що відповідають.

Архітектура даних [2]:

- дає уяву про основні поняття і процеси предметної області;
- дає змогу краще зрозуміти цілі розроблення;
- пропонує протоколи переміщення даних із джерела в місце їхнього аналізу і використання;
- забезпечує наявність системи для захисту даних;
- надає всім командам можливість приймати рішення на основі даних.

Є різні типи моделей даних: концептуальна, логічна і фізична модель. Моделі даних використовуються для представлення даних і того, як вони зберігаються в базі даних, а також для установки взаємозв’язку між елементами даних.

Концептуальна модель даних описує предметну область. Мета проектування полягає в тому, щоб організувати й описати бізнес-концепції і правила. Концептуальна модель розробляється незалежно від технічних характеристик устаткування, таких як об’єм сховища даних або специфікації програмного забезпечення (СКБД). На цьому рівні моделювання даних практично немає подробиць про фактичну структуру бази даних (наприклад, немає типів даних). Концептуальна модель даних описується трьома основними поняттями ER (Entity-Relationship) моделі [15]:

- сутність — річ із реального світу;
- атрибут — характеристики або властивості об’єкту;
- взаємозв’язок — залежність або зв’язок між двома об’єктами.

Логічна модель даних описує, як система має бути реалізована. Мета — розробити технічну карту правил.

Логічна модель даних використовуються для визначення структури елементів даних і встановити зв’язок між ними. Логічна модель даних додає інформацію до елементів концептуальної моделі даних. Перевага використання логічної моделі даних полягає в забезпеченні основи для формування бази для фізичної моделі, залишаючись загальною.

Фізична модель даних описує, як система буде реалізована з використанням конкретної СКБД. Метою є фактична реалізація бази даних. Фізична модель даних описує реалізацію бази даних средствами конкретного сервера конкретної моделі даних. Фізична модель даних містить стосунки між таблицями, які враховують кількість елементів і допустимість значень NULL для стосунків. Стовпці містять типи даних, обмеження і значення за умовчанням. Визначається первинний і зовнішній ключі, представлення, індекси, профілі доступу, авторизації та інші.

Розроблення архітектури даних передбачає врахування архітектури додатка. Мікросервісна архітектура є відокремленими незалежними сервісами, які спілкуються між собою виключно через єднальний сервіс. Ці особливості мають бути відбиті в архітектурі даних. Наприклад, у такій

архітектурі неможливе використання монолітної БД (наявний зв’язок між сервісами через дані) і необхідно забезпечити кожному модулю відокремлене бази даних: зберігається незалежність сервісів, забезпечується роздільність.

У такому разі за теоремою CAP — у якій говориться, що теоретично неможливо забезпечити одночасно узгодженість, доступність і роздільність — у нас залишається вибір між узгодженістю й доступністю.

Теорема CAP (також відома як теорема Брюера, на честь науковця Еріка Брюера) — твердження, що для будь-якої розподіленої комп'ютерної системи неможливо одночасно забезпечити виконання більше двох із перелічених трьох властивостей:

– узгодженість даних (усі вузли бачать однакові дані у будь-який момент часу);

– доступність (гарантія того, що кожен запит отримає коректну відповідь);

– стійкість до розділення (попри розділення на ізольовані секції або втрати зв'язку з частиною вузлів, система не втрачає стабільність і здатність коректно відповідати на запити).

З точки зору теореми, розподілені системи в залежності від пари забезпечених властивостей діляться на три класи:

– CA – розподілена система в якій забезпечена доступність та узгодженість даних не може забезпечувати стійкість до розділення. Прикладом такої системи є програмне забезпечення що підтримує ACID вимоги, наприклад реляційні бази даних;

– AP – розподілена система в якій не гарантується цілісність результату, але висока доступність і збереження працездатності при розділенні. Такі системи з'явилися значно раніше формулювання CAP теореми, наприклад DNS, але ріст популярності збігається з розповсюдженням даного принципу (зокрема деякі NoSQL системи не гарантують цілісність результату, посилаючись на дану теорему);

– CP – Система що забезпечує цілісність даних на всіх вузлах і здатність працювати при розділенні, але не гарантує доступність і може не відповідати на запити. Прикладами таких систем є розподілене програмне забезпечення фінансових систем, де узгодженість даних має найвищий пріоритет, наприклад, мережа банкоматів.

Якщо обрати доступність, то узгодженість можна отримати з допомогою мастер-даних.

Роботу сервісів із мастер-даними інших БД можна організувати в якості звернення до відповідного сервісу, що займає деякий час. Або копіювання мастер-данных у БД, але в якийсь момент часу дані не будуть погоджені. Розглянемо основні архітектурні підходи до рішення цієї задачі [3]:

– дублювання мастер-систем. Для цього кожен сервіс актуалізує власні мастер-дані, що дублює логіку. Для актуалізації потрібно час і можлива ситуація, коли частина сервісів працюють із застарілими даними;

– зберігання мастер-даних в одній БД і забезпечити . Усі сервіси працюють із єдиною версією даних і актуальність мастер-даних лежить тільки на одному сервісі. Але це означає, що на сервіс зростає навантаження, по потоку передаються великі об’єми даних. А якщо сервіс мастер-даних відключиться, то інші сервіси не зможуть отримати дані, потрібні для роботи;

– організація сховища даних. Дані зберігаються у відокремленому сховищі й усі сервіси звертаються виключно до нього. Сервіс, що відповідає за мастер-дані, актуалізує дані в сховищі. Але на оновлення даних потрібно час, на сховищі доводиться велике навантаження і якщо сховище буде недоступне, то сервіси також не матимуть доступу до даних. Можуть виникнути труднощі з додаванням нових джерел, оскільки необхідно підлаштовувати сховище під нове джерело даних, або всі нові джерела підлаштовувати під сховище;

– сповіщення про оновлення. Сервіс, що відповідає за мастер-дані, актуалізує дані, а інші сервіси зберігають у себе копію. Після зміні даних усі

сервіси отримують сповіщення на оновлення даних, а в разі збоїв у роботі сервісу мастер-даних сервіси можуть працювати зі своєю локальною версією даних.

2.2 Місце модулю управління даними в загальній архітектурі системи

На рисунку 3.2 зображена загальна архітектура додатку. Кольором виділено елементи системи, які відносяться до модулю управління даними.

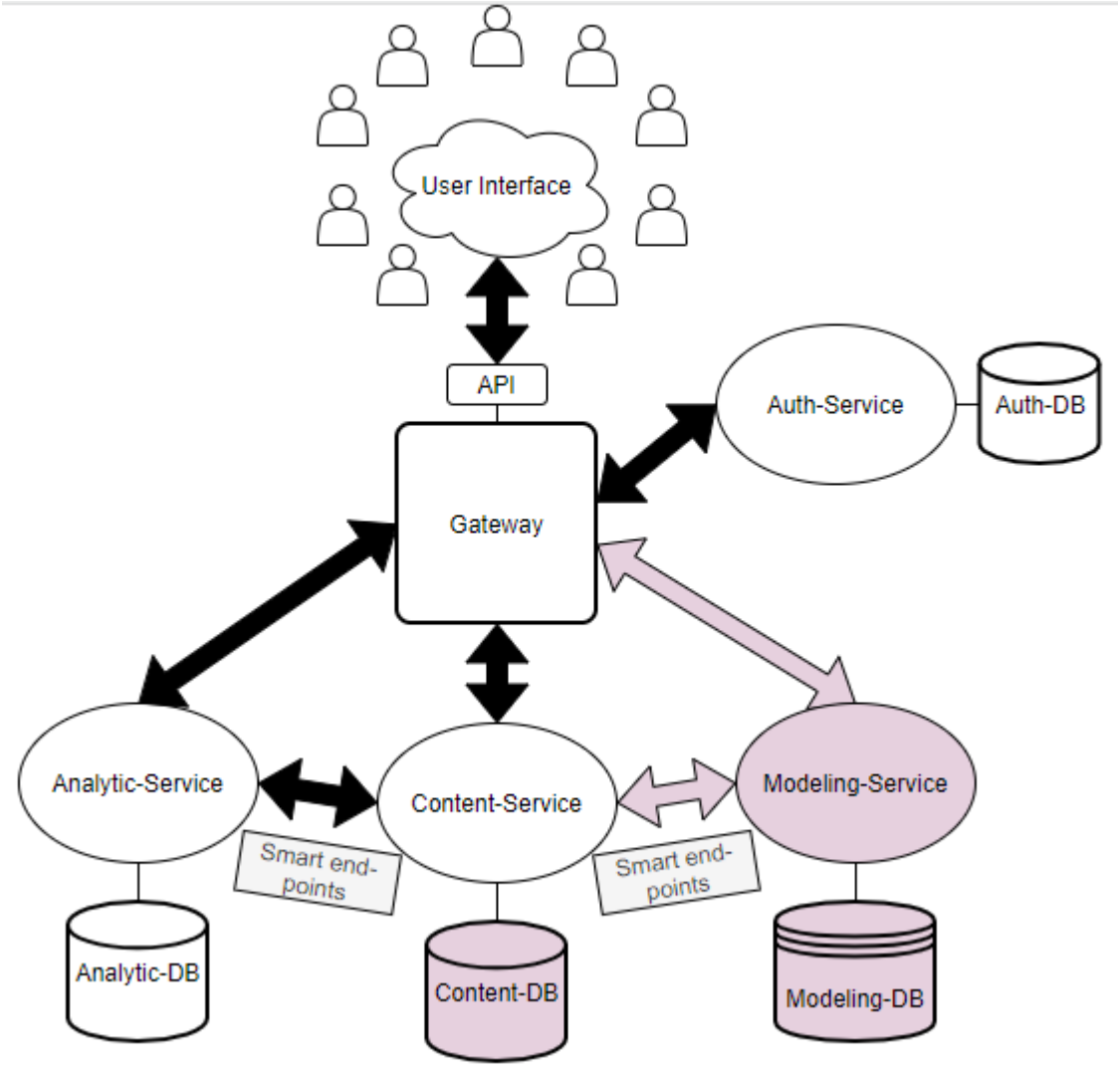

Рисунок 3.2 – Місце модулю управління даними

Можна побачити, що серверна частина платформи підрозділяється на сервіси для вирішення власних задач [16]. За комунікацію між сервісами відповідає компонент Gateway. Цей модуль є головним компонентом мікросервісної архітектури. Він являє собою API-шлюз, який надає інтерфейс, що використовується для доступу до усіх мікросервісів.

3 ПРОЕКТУВАННЯ БАЗ ДАНИХ

Проектування БД це важливий і довгий процес, який складається з багатьох етапів: опис вимог, формалізація вимог, проектування моделей даних, формування документації та інші. Над етапами можуть працювати різні спеціалісти, не обов'язково IT, а результат одного етапу використовується в інших. Тому є необхідність в загальному інструменті, яких допоможе пов'язати усі процеси та налагодити спілкування між спеціалістами: опис IT процесів та вимог бізнесу у зрозумілому для всіх форматі. Передовим інструментом для вирішення цієї задачі є PowerDesigner.

3.1 PowerDesigner

PowerDesigner – програмний засіб, розроблений для вирішення наступних завдань [12]:

а) Моделювання даних (Data Modelling): поєднує в собі набір унікальних технологій для моделювання даних – традиційні концептуальне, фізичне і логічне моделювання з унікальними моделями для бізнес-процесів і руху даних. PowerDesigner підтримує більше 60 СКБД різних виробників і версій.

б) Документування та управління архітектурою підприємства (Enterprise Architecture):

1) дає змогу моделювати бізнес, інформаційну та технологічну архітектури підприємства;

2) значно спрощує взаємодію бізнес та IT підрозділів, дозволяючи їм говорити на одній мові;

3) забезпечує можливість зберігати не тільки існуючу архітектуру ("As is"), але і цільову ("Target") і маршрутну карту змін.

4) Управління архітектурою Дає змогу більш швидко і ефективно реагувати на зміни ринку і стратегії бізнесу, за рахунок налагоджених

комунікацій між бізнесом та IT. Крім того, це дає можливість скоротити ресурсні, часові та фінансові інвестиції при впровадженні таких змін.

PowerDesigner не примушує використовувати розробників будь-які жорстко вбудовані методології або процеси. Кожна компанія може розробити і використовувати свої власні процеси, визначити ролі, відповідальності, порядок обробки моделей і документів тощо.

Ключовою перевагою PowerDesigner є підтримка всіх рівнів моделювання (бізнес, об'єктний, логічний, фізичний і ін.) і моделей даних, а так само можливість зберігати їх в єдиному репозиторії пов'язаними між собою з допомогою технології Link & Sync.

На рисунку 3.1.1 зображена схема взаємодії різних моделей даних в рамках проекту у PowerDesigner.

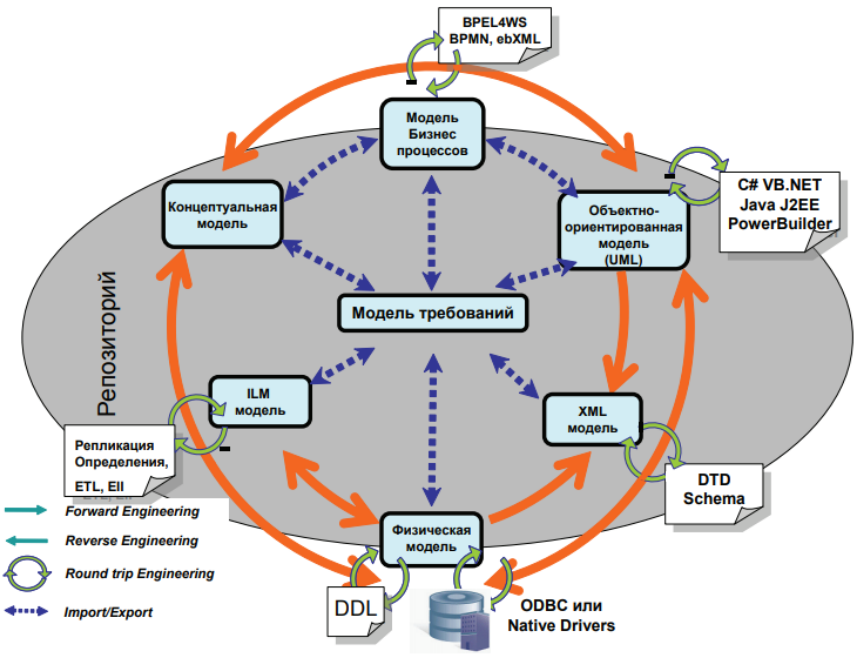

Рисунок 3.1 – Схема взаємодії моделей PowerDesigner

Велика кількість організацій по всьому світу користуються такими унікальними можливостями PowerDesigner як:

– можливість керувати часом, вартістю та ризиками при внесенні змін до додатка, архітектуру тощо;

– скорочення термінів виведення нових продуктів і послуг на ринок;

– підвищення віддачі від систем бізнес-аналізу (BI) завдяки можливості інтегрувати метадані і аналізувати рух даних від джерел до звітів;

– налагодження комунікацій між бізнесом та IT за рахунок використання єдиного середовища обміну інформацією;

– оптимізація використання IT ресурсів і перенаправлення їх на більш пріоритетні проекти;

– значне скорочення вартості, строків і ресурсів при інтеграції IT систем після злиття і поглинань;

На сьогоднішній день PowerDesigner є лідером численних рейтингів і має одну з найбільших часток ринку в світі серед засобів моделювання та управління архітектурою.

В рамках дипломного проекту інструменти PowerDesigner дозволяють працювати над сервісами незалежно, проектувати власні БД і вести уніфіковану документацію. Підтримка кількох нотацій Дає змогу одразу змінити нотацію. Інструмент генерації запиту на створення БД для різних СКБД Дає змогу розробникам впроваджувати нові СКБД для сервісів, не замислюючись над особливостями створення таблиць чи обмежень. Інструмент генерації запиту на зміну структури БД Дає змогу автоматизувати зазвичай об’ємну роботу на зміну типу, додавання нових атрибутів, тощо.

Також перевагою для проекту є те, що PowerDesigner Дає змогу сформувати запит, вказавши СКБД і у разі потреби перенести схему на іншу СКБД.

3.2 Концептуальна модель даних

Першим етапом розроблення архітектури є проектування концептуальної схеми. Схеми для різних модулів розділені на пакети. Це дає

змогу візуально розділити схеми між собою, але зберігає зв’язок між сутностями.

Наступна схема – модуль контенту (додаток Б). Це ключова модель проекту, яка зберігає структуру курсів, усі матеріали для відображення в інтерфейсі. При проектуванні схеми використовувались формалізовані вимоги предметної області до архітектури даних. А саме:

− Створення, зберігання та змінення структури курсів;

− Зберігання додаткового матеріалу до курсу;

− Гнучка організація кроків для забезпечення їхньої різноманітності, легкої перестановки та додавання нових.

Можливість гнучкої зміни структури проходження уроку, модулю і курсу реалізовано так: кожен елемент має атрибут «номер позиції», який легко змінити. Для зберігання додаткового матеріалу використовується зберігання шляху до файлу на сервері. Для забезпечення різноманіття кроків використовується тип кроку та поле data у якому можна зберігати як текст уроку, так і відеофайли.

3.3 Внесення змін до структури даних

. Перший етап – створення концептуальної моделі даних. Для цього у PowerDesigner треба доповнити діаграму необхідними сутностями (рис. 3.3).

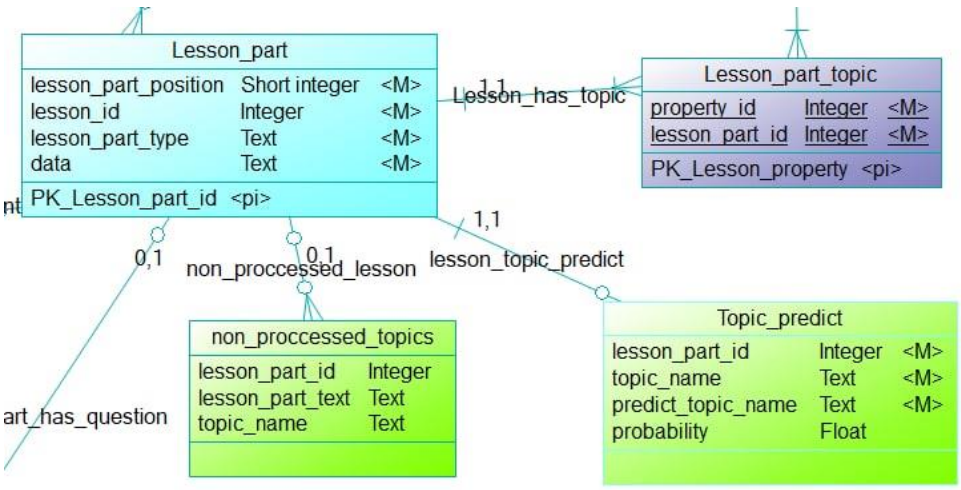

Рисунок 3.3 – Додавання сутностей модулю інтелектуальної обробки тексту

Поки модуль містить тільки дві сутності: сутність з результатами обробки (які закінчились успішно) та сутність з текстами, для яких модель не знайшла тему (ймовірність по темам розподілена рівномірно). Таблиці (колекції) у БД створюються за допомогою вставки значень у БД.

Після оновлення документації до структури отримаємо такий опис сутностей (табл. 3.1-3.3).

Таблиця 3.1 Атрибути сутності Topic_predict

Назва	Тип даних	Обов'язковість
lesson_part_id	Integer	X
predict_topic_name	Text	X
probability	Float	X
topic_name	Text	X

Таблиця 3.2 Зв'язки з сутностю Topic_predict

Назва	Сутність 2	Сутність1	Сутність 1 -> Сутність 2 кардинальність	Сутність 2 -> Сутність 1 кардинальність
lesson_topic_predict	Topic_predict	Lesson_part	0,1	1,1

Таблиця 3.3 Атрибути сутності Non_proccessed_topics

Назва	Тип даних	Обов'язковість
lesson_part_id	Integer	X
lesson_part_text	Text	X
topic_name	Text	X

Таблиця 3.4 Зв’язки з сутностю Non_proccessed_topics

Назва	Сутність 2	Сутність1	Сутність 1 -> Сутність 2 кардинальність	Сутність 2 -> Сутність 1 кардинальність
non_proccessed_lesson	Non_proccessed_topics	Lesson_part	0,n	0,1

3.4 Обрання СКБД

Сформована архітектура не обмежує у виборі СКБД для сервісів, але усі вони відносяться до двох типів систем: OLAP (Online Analytical Processing) чи OLTP (Online Transaction Processing).

OLTP це онлайн-транзакційна обробка, яка підтримує програми, орієнтовані на транзакції. OLTP в основному орієнтований на обробку запитів, підтримку цілісності даних в середовищах з множинним доступом, а також ефективність, яка вимірюється загальною кількістю транзакцій в секунду.

Нижче наведено важливі характеристики OLTP:

– OLTP використовує транзакції, які включають невеликі обсяги даних;

– індексовані дані в базі даних можуть бути легко доступні;

– OLTP має велику кількість користувачів;

– має швидкий час відгуку;

– БД безпосередньо доступні для кінцевих користувачів;

– OLTP використовує повністю нормалізовану схему для узгодженості бази даних;

– час відгуку системи OLTP короткий;

– строго виконує тільки існуючі операції з невеликою кількістю записів;

– OLTP зберігає записи за останні кілька днів або тиждень.

У якості OLTP в дипломному проекті обрана РСКБД PostgreSQL та NoSQL MongoDB.

PostgreSQL це популярна вільна об’єктно-реляційна система управління базами даних. PostgreSQL базується на мові SQL і підтримує численні можливості.

Переваги PostgreSQL:

− підтримка БД необмеженого розміру;

− потужні та надійні механізми транзакцій і реплікації;

− розширюється система вбудованих мов програмування;

− підтримка завантаження С-сумісних модулів;

− спадкування, легка розширюваність.

Поточні обмеження PostgreSQL:

− немає обмежень на максимальний розмір бази даних;

− немає обмежень на кількість записів у таблиці;

− немає обмежень на кількість індексів у таблиці;

− максимальний розмір таблиці — 32 Тбайт;

− максимальний розмір запису — 1,6 Тбайт.

Особливості PostgreSQL:

− функції в PostgreSQL є блоками коду, що виконуються на сервері. Хоча вони можуть писатися на чистому SQL, реалізація додаткової логіки, наприклад, умовних переходів і циклів, виходить за рамки SQL і вимагає використання деяких мовних розширень. Функції можуть писатися з використанням різних мов програмування;

− тригери в PostgreSQL визначаються як функції, що ініціюються, операціями;

− механізм правил в PostgreSQL є механізмом створення призначених для користувача обробників не лише DML- операцій, але й операції вибірки. Основна відмінність від механізму тригерів полягає в тому, що правила спрацьовують на етапі розбору запиту, до вибору оптимального плану

виконання й самого процесу виконання. Правила дають змогу перевизначати поведінку системи під час виконання SQL- операції до таблиці;

− індекси в PostgreSQL таких типів: B- дерево, хэш, R- дерево, GiST, GIN;

− можлива одночасна модифікація БД декількома користувачами з допомогою механізму Multiversion Concurrency Control (MVCC). Завдяки цьому дотримуються вимоги ACID, і практично не використовується блокування на читання;

− розширення PostgreSQL для власних потреб практично в будь-якому аспекті. Є можливість додавати власні перетворення типів, типи даних, домени (призначені для користувача типи зі спочатку накладеними обмеженнями), функції, індекси, оператори і процедурні мови;

Переваги OLTP:

– OLTP пропонує точний прогноз доходів і витрат;

– забезпечує міцну основу для стабільного бізнесу / організації завдяки своєчасному зміни всіх транзакцій;

– OLTP робить транзакції набагато простіше від імені клієнтів: це розширює клієнтську базу для організації, прискорюючи і спрощуючи окремі процеси;

– OLTP забезпечує підтримку великих баз даних;

поділ даних для маніпулювання даними легко;

– потрібна для використання завдань, які часто виконуються системою;

– завдання, які включають вставку, оновлення або видалення даних;

– використовується, коли вам потрібна узгодженість і паралелізм для виконання завдань, які забезпечують його більшу доступність;

MongoDB — документо-орієнтована система керування базами даних (СКБД) з відкритим вихідним кодом, яка не потребує опису схеми таблиць. MongoDB займає нішу між швидкими й масштабованими системами, що

оперують даними у форматі ключ/значення, і реляційними СКБД, функціональними і зручними у формуванні запитів.

Основні можливості MongoDB:

− документо-орієнтоване сховище (проста та потужна JSON-подібна схема даних);

− досить гнучка мова для формування запитів;

− динамічні запити;

− повна підтримка індексів;

− профілювання запитів;

− швидкі оновлення «на місці»;

− ефективне зберігання бінарних даних великих обсягів, наприклад, фото та відео;

− журналювання операцій, що модифікують дані в БД;

− підтримка відмовостійкості й масштабованості.

PostgreSQL використовується для модулів авторизації та контенту, для яких важлива цілісність даних та доступність. MongoDB використовується для модулю інтелектуальної обробки даних, який потребує розширення та можливих корективів структури залежно від модифікацій модулю.

Обираючи СКБД можна аналізувати потреби з точки зору CAP теореми.

Це означає, що при виборі СКБД треба оцінити, що саме є вагомим для системи. Наприклад, в проекті для БД авторизації необхідна в першу чергу узгодженість (безпека), потім доступність; для БД контенту необхідна доступність (робота усього додатку залежить від контенту), потім узгодженість; для БД інтелектуальної обробки тексту необхідно в першу чергу стійкість до розподілення (для незалежної роботи кількох моделей), потім узгодженість і доступність.

3.5 Фізична модель даних

Для того, щоб отримати запит на створення БД, необхідно обрати СКБД та автоматично створити фізичну модель даних. Для прикладу зробимо фізичну схему для концептуальної моделі авторизації. Для цього в меню інструменти обираємо «створити фізичну модель». Далі у діалоговому вікні (рис. 3.3) обираємо зі списку СКБД PostgreSQL 9.x та обираємо необхідні таблиці (рис. 3.4).

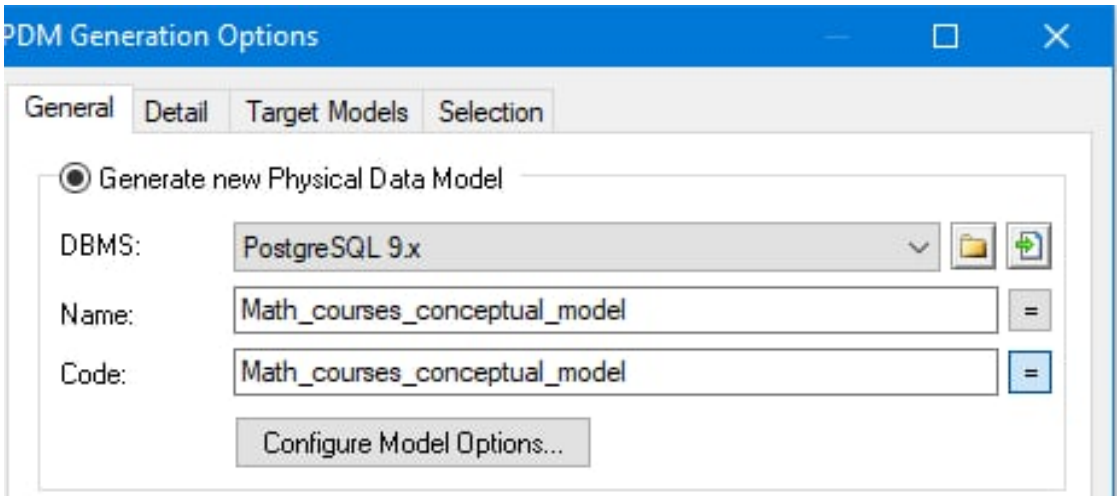

Рисунок 3.3 – Головне вікно створення фізичної моделі

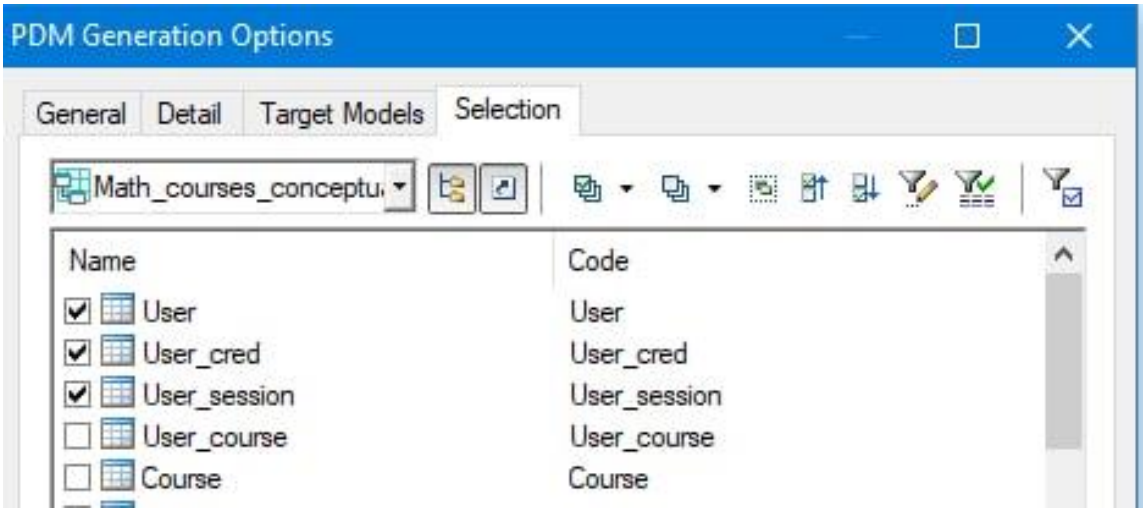

Рисунок 3.4 – Список таблиць для створення фізичної моделі

Отримаємо схему (додаток В), яка відповідає фізичній схемі обраних таблиць. Можна побачити, що змінились зв’язки, є позначки первинного та

вторинного ключів. Також, назви зв'язків змінились на ті, що генерує автоматично PostgreSQL.

Останнім пунктом є створення запиту на створення БД. Для цього обираємо необхідну фізичну модель даних, у меню Бази даних натискаємо створити БД, обираємо СКБД та налаштовуємо за бажанням (рис. 3.5)

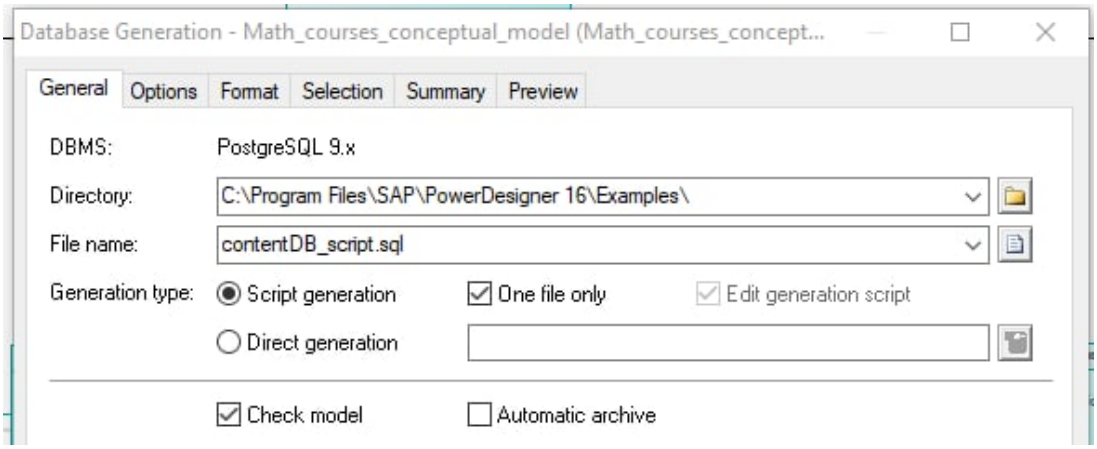

Рисунок 3.5 – Меню генерації запиту на створення БД контенту

В результаті отримаємо файл з запитом. Створення таблиці курсу виглядає наступним чином:

```
create table Course (
   course_id            SERIAL               not null,
   name                 TEXT                 not null,
   course_status        TEXT                 not null,
   video                TEXT                 null,
   description          TEXT                 not null,
   short_description    TEXT                 not null,
   language             TEXT                 not null,
   target_description   TEXT                 not null,
   requirement_description TEXT                 not null,
   work_load            INT4                 null,
   readiness            INT4                 not null default
0,
   constraint PK_COURSE primary key (course_id)
);
create unique index Course_PK on Course (
course_id
);
```

Повний текст запиту для модулю авторизації та контенту знаходиться у додатку Г.

4 ІНТЕЛЕКТУАЛЬНА ОБРОБКА ТЕКСТУ

Однією з проблем є складність оцінки розуміння матеріалу:

– мало ресурсів дозволяють робити тести з відкритими відповідями, де можна описати формулу, а єдиним варіантом оцінки є питання з правильною відповіддю. Але в цьому форматі є можливість "вгадати" правильну відповідь;

– дистанційно викладачу складніше оцінити причетність до уроку і розуміння лекції студентами: на відміну від занять віч-на-віч, в дистанційному навчанні студенти часто не вмикають камери і єдине, що може оцінити викладач - активність відповідей на поставлені викладачем питання.

В рамках роботи буде розглянуто друге питання, а саме оцінку тексту лекцій.

Це завдання актуальне, так як це дозволить зробити незалежну оцінку тексту і припустити, як саме сприймається текст тими, хто на момент прочитання тексту не має знань по темі. Сприйняття тексту можна оцінити виходячи з актуальності тексту: відповідність тексту темі, призначеної викладачем.

Необхідно враховувати специфіку математичного тексту, адже перед роботою з текстом потрібно чітко розуміти, які дані будуть оброблятися і, в залежності від цього, будувати процеси аналізу.

Модуль інтелектуальної обробки тексту використовуються для апробації розроблених методів управління даними.

Для оцінювання релевантности теми необхідно кожному тексту (уроку) присвоїти список тим, які повно описують тематику тексту. Іноді текст досить об’ємний і складно описати всі теми, описані в уроці. У такому разі в автора можуть виникнути труднощі з виділенням основних тем.

Для вирішення цієї проблеми вирішено використати програмну обробку тексту і формування результату в якості рекомендацій авторові. Також результат моделі має допомогти авторові в поліпшенні матеріалу: якщо тема,

важлива для уроку, розкрита менше інших, автор має змогу доповнити текст, спираючись на рекомендації.

Інтелектуальний аналіз тексту складається з декількох етапів:

1. очищення вхідних даних;
2. вибір даних для обробки;
3. застосування методів Data mining;
4. завантаження результатів у сховищі (БД);
5. пояснення результатів (оформлення звітів і графіків).

4.1 Математична мова

Для інтелектуальної обробки тексту використовуються тексти переважно наукового стилю, для якого характерно: [7]

- логічність викладу, точність;
- однозначність трактування сенсу;
- уникнення використання метафор;
- відсутність емоційного забарвлення;
- широке використання термінів і абстрактної лексики
- строга нормоване (відповідність нормам літературної мови).

Приклад тексту :

- сума цифр цілого числа, що ділиться на 3, ділиться на 3;
- сума кутів трикутника на площині дорівнює 180°;
- корені квадратного рівняння завжди різні.

4.2 Основні поняття інтелектуальної обробки тексту

В роботі вживаються наступні основні поняття аналізу тексту.

Так, мінімальною одиницею тексту є токен — послідовність символів, які вважаються одним об’єктом: абзац, речення, слова й розділові знаки. У цій роботі токеном вважатиметься останній випадок.

Документом у цій роботі вважатиметься список речень. Список документів із метаданими (ідентифікатор, частота) є корпусом.

Стоп-слова – список загальних термінів, які не несуть особливого смислового навантаження: приводи, союзи, займенники й так далі. Також корисно доповнити словами, які є стоп-словами в предметній області. Наприклад, під час роботі з математичними текстами можливе розширення списку стоп-слів символами грецького алфавіту, загальнонауковими термінами.

4.3 Предобробка тексту

Попередня обробка тексту є одним із ключових етапів побудови моделі інтелектуальної обробки тексту. Точність моделі безпосередньо залежить від якості й повноти даних. До цього етапу належать очищення даних і вибірка даних для подальшої обробки.

Очищення даних у тематичному моделюванні – послідовність із методів. Спочатку необхідно розбити документ на речення, а речення на слова – цей процес називають виділенням токенів. Вважатимемо токен мінімальною одиницею тексту. У результаті формується список речень, які розбиті на слова і знаки пунктуації.

Необхідно перевести усі токени в нижній регістр і видалити стоп-слова. Далі етап стеммизації. Це метод приведення слова до його кореня шляхом усунення суфіксів, приставок, закінчень тощо [6]. Стеммизація добре працює з англійським текстом, але не підходить для російської мови через мовні особливості: результатом методу частіше за все будуть слова, які не існують. Наприклад, «хочу» буде скорочено до «хоч», але цього слова немає у

словнику. Замість цього необхідно отримати слово «хотеть», для цього використовують лематизацію.

Лематизація – приведення слова в початкову форму (лему). Для іменників – називний відмінок, однина; для дієслів – інфінітивна форма; для прикметників – однина, називний відмінок, чоловічий рід. [4].

Наступним етапом буде виділення колокацій. Колокації (англ. collocations) – це комбінація двох або більше слів, які формують стійке словосполучення, що відіграє важливе значення для природного звучання іноземної мови. У колокаціях один компонент вибирається за змістом, а вибір другого залежить від вибору першого [5]. Наприклад, об’єднати «натуральне» і «множина» в поняття «натуральна множина», а «натуральний» і «стіл» в поняття «натуральній стіл». Оскільки корпус складається з математичних текстів, то біграма «натуральна множина» буде часто зустрічатись і потрапить до словнику, а біграма «натуральній стіл» має низьку ймовірність і буде проігноровано.

4.4 Числова модель тексту

Для аналізу структури та наповнення тексту необхідно перевести його у числовий формат, зручний для комп’ютерної обробки.

Простіший метод числового представлення тексту – сформувати «мішок слів». Це модель текстів, в якій кожен документ або текст виглядає як невпорядкований набір слів без відомостей про зв'язки між ними

Іншим способом представити текст у якості матриці – використати TF-ID – частота слова на зворотну частоту документу, де TF – частота слова або вірогідність знаходження слова в реченні [5]. Обчислюється за формулою 4.1

$$tf(t,d) = \frac{n_t}{\sum_t n_t} \tag{4.1}$$

де t – загальне число слів у документі;

d – число документів в корпусі;

n_t – число входження слова t в документ:

IDF – зворотна частота документа або унікальність токена в документі. Обчислюється за формулою 4.2

$$idf(t, D) = log\ log \frac{|D|}{|\{d_i \in D | t \in d_i\}|} \quad (4.2)$$

де t – загальне число слів у документі;

|D| – число документів у корпусі;

$|\{d_i \in D | t \in d_i\}|$ – число документів із колекції D, для яких є $n_t \neq 0$.

Це дає змогу коригувати вагу в слів, які часто трапляються й маловірогідно будуть корисні. Наприклад, знижуючи вагу слова «граф» у документі «Теорія графів».

4.5 Тематичне моделювання

Тематичне моделювання – створення моделі для вивчення, розпізнавання і витягання тим із набору документів і імовірнісного розподілу документів на безлічі отриманих тем. Результатом моделі є вірогідність відношення документу до кожної з тем [8].

4.6 Методи тематичного моделювання

Є багато методів тематичного моделювання, але на сьогодні найбільш популярними є LSA[9] , pLSA[10] і LDA[11].

Усі тематичні моделі ґрунтовані на припущенні, що кожен документ складається з декількох тим, і кожна тема складається з набору слів. Семантика документу фактично визначається деякими прихованими характеристиками.

Мета тематичного моделювання — виявити ці приховані змінні – теми, які визначають сенс документу.

Латентний семантичний аналіз (Latent semantic analysis – LSA) є одним з основних методів тематичного моделювання. Основна ідея полягає в тому, щоб взяти матрицю «документ – термін» і розкласти її на відокремлені матриці «документ-тема» й «термін – тема» [9].

Першим кроком є створення матриці «документ-термін». Позначимо кількість документів як m і кількість слів у словнику як n, тоді отримана матриця A матиме розмір m × n, у якій рядок – документ, а стовпець – слово. У простій версії LSA використовується простий підрахунок входження j слова і -му документі. Але такий підрахунок не враховує значень слів у документі. Натомість, у моделі використовують tf – idf оцінювання. У результаті слово має велику вагу, коли він часто трапляється в документі, але не часто в корпусі.

Після побудови матриці «документ-термін» можна вивчати приховані тематики в тексті. Але розмірність матриці може бути завеликою, а матриця розряджена й надмірна: досить декількох слів, які трапляються в малій кількості документів. У такому разі для оптимізації роботи можна виконати скорочення розмірності на A.

Для зменшення розмірності можна використати усічений SVD (Singular value decomposition). У лінійній алгебрі розкладання сингулярних значеннях – це метод, який розкладає будь-яку матрицю M у 3 матриці: M = U * S * V, де S є діагональною матрицею сингулярних значень із M.

Далі вибираються t тим, для яких значення сингулярності найвища.

У результаті усічений SVD вибирає t стовпців з U і V, де В цьому випадку $U \in \mathbb{R}\ (m \times t)$ – матриця «документ – тема», а $V \in \mathbb{R}\ (n \times t)$ – матриця «термін – тема». Як у U, так і в V стовпці відповідають одній з t тим.

Матриці «документ-тема» й «тема-термін» можуть використовуватися в методі косинусної відстані для оцінювання:

- схожість різних документів;
- схожість різних слів;

– схожість термінів і документів (для пошуку уривків, які найбільше підходять під запит).

Імовірнісний прихований семантичний аналіз використовує імовірнісний метод замість SVD для зменшення розмірності. Метод розраховує імовірнісну модель із прихованими темами в тексті й у результаті видає матрицю схожу з матрицею «документ-термін» [10].

У відмінності від LSA, pLSA додає вірогідність відношення теми до документу і слова до теми. Позначається як P (z | d) і P (w | z). Загальна вірогідність вираховується формулою 4.3

$$P(D,W) = \sum_z P(Z)P(D|Z)P(W|Z) \qquad (4.3)$$

де P – функція вірогідності;

D – документ;

W – слово;

Z – тема.

Обчислення розпочинається з теми P (z), а потім незалежно вважає вірогідність для документу – P (d | z) і для слова P (w | z). У результаті можна побачити відповідність матриць з LSA і вірогідності з pLSA (см рис. 4.1)

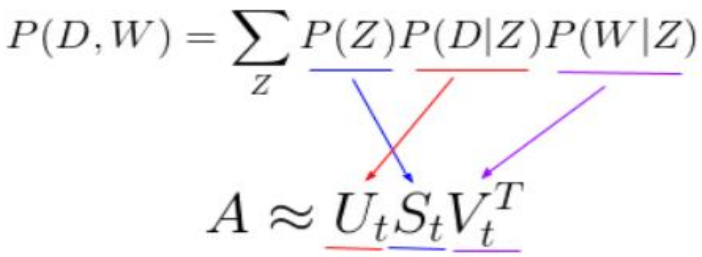

Рисунок 4.1 — Відповідність між LSA и pLSA

де вірогідність теми P (Z) відповідає діагональній матриці вірогідності сингулярної теми, вірогідність P (D | Z) відповідає матриці теми документу U, вірогідність слова з обліком тема P (W | Z) відповідає «термін-тема матриці» V.

Але модель має істотний недолік: кількість параметрів для pLSA лінійно росте зі збільшенням кількості документів, тому модель може перевчитися.

Отже, є два методи, які добре працюють на невеликій кількості даних, але погано справляються з великими масивами. Для другого випадку використовують модель LDA — розширення методу PLSA для роботи з великими корпусами.

Latent Dirichlet Allocation (LDA) — це метод латентного розміщення Діріхле, що використовує розподіл Діріхле. Такий розподіл дає змогу оцінювати ймовірності документів і термінів, не спираючись на текстові колекції [11]. На рис. 4.2 зображено графічне представлення моделі:

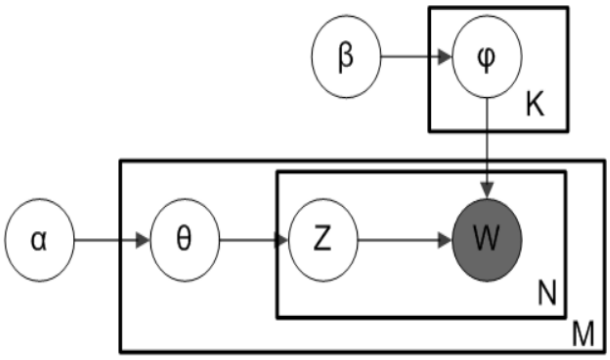

Рисунок 4.2 — графічна імовірнісна модель LDA

де w – слово (що спостерігається змінна); t – тема (прихована змінна); D – колекція документів; N – довжина документа в словах; K – кількість тем у колекції; θ – розподіл тем у документі; φ – розподіл слів у темі; α – апріорне розподіл Діріхле на параметри θ, β – апріорне розподіл Діріхле на параметри φ.

Перевага латентного розподілу Діріхле полягає в тому, що зі зростанням кількості документів не змінюється кількість параметрів і алгоритм зазвичай

працює краще на нових документах, ніж pLSA: корпус є навчальним набором для розподілу Діріхле й нові документи прогнозуються на підставі наявних.

5 РЕАЛИЗАЦИЯ МОДУЛЮ

5.1 Технологічний стек

− бібліотека NLTK [14] – пакет бібліотек і програм для символьної і статистичної обробки природної мови, написаних на мові програмування Python. Містить графічні представлення і приклади даних. Супроводжується великою документацією, а також книгою з поясненням основних концепцій, вартих за тими завданнями обробок природної мови, які можна виконувати з допомогою цього пакету. NLTK добре підходить для студентів, що вивчають комп'ютерну лінгвістику або близькі предмети, такі, як емпірична лінгвістика, штучний інтелект, інформаційний пошук і машинне навчання. NLTK з успіхом використовується у якості навчального посібника, інструменту індивідуального навчання і створення науково-дослідних систем. NLTK є вільним програмним забезпеченням. Проект очолює Стівен Берд;

− наступна бібліотека – Pymorphy2. Написана на мові Python (працює під 2.7 і 3.5) і може: наводити слово до нормальної форми (наприклад, «люди -> людина», або «гуляв -> гуляти»); ставити слово в потрібну форму. Наприклад, ставити слово в множину, міняти відмінок слова й так далі; повертати граматичну інформацію про слово (число, рід, відмінок, частина мови й так далі). Під час роботі використовується словник OpenCorpora; для незнайомих слів будуються гіпотези. Бібліотека досить швидка: зараз швидкість роботи — від декількох тыс слів/сік до > 100тыс слів/сік (залежно від виконуваної операції, інтерпретатора і встановлених пакетів); споживання пам'яті — 10.20Мб;

− CountVectorizer бібліотеки Scikit-learn використовується для перетворення колекції текстових документів у вектор кількості термінів / токенів. Він також дає змогу виконувати попередню обробку текстових даних перед створенням векторного представлення. Ця функціональність робить його дуже гнучким модулем представлення функцій для тексту;

- pyLDAvis – бібліотека для графічного представлення моделі. Кожен пухир на лівому графіку представляє тему. Чим більше пухиря, тим більше поширена ця тема. Хороша тематична модель матиме досить великі області, що не перетинаються, розкидані по всій діаграмі, а не згруповані в одному квадраті. Модель із занадто великою кількістю тем, як правило, має багато областей невеликого розміру, згруповані в одній області діаграми;
- Gensim – це библиотека для обробки природної мови, яка містить потужні інструменти для тематичного моделювання. Gensim надає такі алгоритми тематичного моделювання, як LDA і LSI, а також Word2Vec, Doc2Vec і FastText для роботи з векторними моделями слів. Крім того, ще однією важливою перевагою gensim є те, що він дає змогу обробляти великі текстові файли без необхідності завантажувати увесь файл у пам’ять.

5.2 Опис моделі

Для тематичного моделювання обрано аналіз вхідного тестування, яке складається з питань шкільної математики. Обрані такі теми для дослідження:

- лінійна і квадратична функції;
- функції та графіки;
- арифметична й геометрична прогресії;
- рівняння й нерівності.

Для створення моделі тематичного моделювання необхідно виконати такі етапи:

- отримання даних;
- преодобробка тексту;
- створення словнику;
- побудова моделі;
- Розроблення моделі поділяється на два этапи: побудова моделі й навчання; тестування.

5.3 Створення моделі

Першим етапом створення моделі є читання вхідних даних. Для роботи з файлами PDF використовується бібліотека Fitz:

```
doc = fitz.open(pdf_document)
```

В циклі йде читання документу сторінками. Текст приводиться до нижнього регістру та розбивається на речення завдяки функції sent_tokenize:

```
for current_page in range(4,len(doc)):
page = doc.loadPage(current_page)
page_text = page.getText(«text»)
pt = sent_tokenize(page_text.lower(), language=«russian»)
text.extend(pt)
pages.extend([current_page]*len(pt))
```

Далі речення розбирається на токени (слова та розділові знаки) та проводиться лемматизація завдяки бібліотеки pymorphy2 та функції parse(token):

```
def morph_tokens(tokens):
morph = pymorphy2.MorphAnalyzer()
return [morph.parse(token)[0].normal_form for token in toke
```

Метод MorphAnalyzer.parse () повертає один або декілька об’єктів типу Parse з інформацією про те, як слово розібране. Результатом розбору слова «прогрессия» має такий вигляд:

```
[Parse(word='прогрессия',tag=OpencorporaTag('NOUN,inan,femn
sing,nomn'),normal_form='прогрессия',score=1.0,methods_stack=((D
ictionaryAnalyzer(), 'прогрессия', 41, 0),))]
```

Далі йде фільтрування токенів від стоп-слів та розділових знаків:

```
    def filter_stop_words(text):
    text = text.replace('-\n', '').replace('\n', ' ')
    text = re.sub(r'[^\w\s]+|[\d]+', r'',text)
    tokens = word_tokenize(text, language=«russian»)
    stop_words = stopwords.words(«russian»)
    punctuation = [".",'«','»',',', «a»,";",'{','}', '(',')','x
','-','x',">","<"]
    return list(filter(lambda token: token not in stop_words an
d token not in punctuation and len(token)>1, tokens ))
```

Загальна функція предобробки тексту має вигляд:

```
def preprocessing(text):
tokens = filter_stop_words(text)
tokens = morph_tokens(tokens)
return ' '.join(tokens)
```

Для отримання корпусу необхідно кожне речення передати у функцію предобробки. Далі створюємо біграми та триграми на основі корпусу

```
from gensim.models import Phrases
bigram = Phrases(text_clean)
trigram = Phrases(bigram[text_clean
for idx in range(len(text_clean)):
    for token in bigram[text_clean[idx]]:
        if '_' in token: text_clean[idx].append(token)
    for token in trigram[text_clean[idx]]:
        if '_' in token:
text_clean[idx].append(token)
```

Та створюємо словник завдяки бібліотеки gensim.corpora.dictionary

```
from gensim.corpora.dictionary import Dictionary
from numpy import array
dictionary = Dictionary(text_clean)
dictionary.filter_extremes(no_below=10, no_above=0.1)
corpus = [dictionary.doc2bow(doc) for doc in text_clean]
```

Для створення моделі використовується бібліотека gensim.models.ldamulticore:

```
    from gensim.models.ldamulticore import LdaMulticore
    model=LdaMulticore(corpus=corpus,id2word=dictionary, num_to
pics=6
```

Як можна побачити, треба вказати кількість тем. Це можна оцінити самостійно по змісту тексту, але краще мати інструмент для програмного визначення кількості тем. Для цього використано поняття як когерентність тем.

```
    def compute_coherence_values(dictionary, corpus, texts,
limit, start=2, step=3):
        coherence_values = []

        for num_topics in range(start, limit, step):

model=LdaMulticore(corpus=corpus,id2word=dictionary,
num_topics=num_topics)
            model_list.append(model)
            coherencemodel = CoherenceModel(model=model,
texts=texts, dictionary=dictionary, coherence='c_v')

coherence_values.append(coherencemodel.get_coherence())
        return model_list, coherence_values
```

Когерентність – це оцінка узгодженості слів в темі. Для цього на кожній ітерації створюється LDA-модель з числом тем від 2 до 20, створюється модель оцінки когерентності CoherenceModel. Результатом будуть 2 списки – список отриманих моделей та значення когерентності. Для демонстрації результату можна побудувати графік залежності (рис. 5.1). Можна побачити, що найкраща узгодженість у моделі з 6 темами.

Таким чином, зберігаємо тільки найкращу модель у файл з розширенням lda для подальшої роботи. Зберігаємо отриманий словник у файл з розширенням dict, біграми та триграми у файли з розширенням phs та корпус у файл з розширенням lda-c.

```
    model=LdaMulticore(corpus=corpus,id2word=dictionary,
num_topics=6)
```

```
    model.save('best_model.lda')
    corpora.Dictionary.save(dictionary, "dictionary.dict")
    corpora.BleiCorpus.save_corpus(fname="corpus.lda-c",corpus=
corpus)
    bigram.save('bigram.phs';trigram.save('trigram.phs')
```

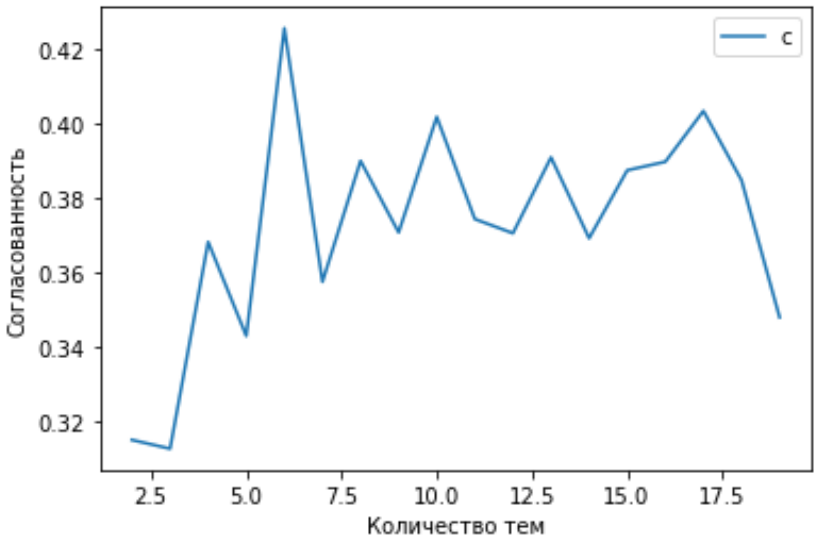

Рисунок 5.1 – Графік залежності узгодженості слів у темах від кількості тем

5.4 Відстежування параметрів моделі

Необхідно відстежувати параметри і метрики, які створюються. Mlflow — інструмент для створення відтворних і відповідальних проектів у галузі науки про дані. Він надає центральний сервер відстежування з простим призначеним для користувача інтерфейсом для перегляду експериментів і потужними інструментами для упаковки, управління й розгортання моделей. Для роботи з Mlflow необхідно описати параметри, які будуть відстежуватися (mlflow.log_param) та модель (mlflow.sklearn.log_model).

```
    mlflow.set_tracking_uri("http://localhost:5000")
    mlflow.set_experiment('test experiment')
    with mlflow.start_run():
        mlflow.log_param(
            'Topic count',6)
        mlflow.log_param('Coherence value',0.45907887922403967)
```

```
    mlflow.sklearn.log_model(
        model,
        artifact_path = '',
        registered_model_name='Test model ')
    mlflow.end_run()
```

На рисунку 5.2 зображений інтерфейс для слідкування за параметрами моделі. У цьому випадку проведено один експеримент, який повернув тип моделі, значення узгодженості та кількість обраних тем.

5.5 Створення класу моделі

Після створення самої моделі необхідно створити клас для аналізу тексту. Він містить вищеназвані методи, метод зчитування даних з БД контенту через gateway, прогнозування та загрузку даних у БД інтелектуальної обробки тексту.

Перше це отримання даних. Для цього робиться get запит до БД контенту:

```
def get_data(self):
 r = requests.get(url = self.URL)
 df = r.json()
 self.input_data =
pd.DataFrame(df['lesson_part_id'],
df['data'],df['topic_name'] )
```

Далі для кожного отриманого тексту рахується тема

```
def process_input_data(self):
self.input_data['probability']                                    =
self.input_data['text'].apply(lambda x: self.predict(x))
self.input_data[['predict_topic','probability']]                  =
pd.DataFrame(self.input_data['probability'].tolist(),
index=self.input_data.index)
```

Теми, для яких ймовірність дорівнює 1/кількість тем, зберігаються окремо і не відправляються до БД контенту як ті, що пройшли перевірку.

```
            self.non_processed_topics                                =
self.input_data[self.input_data.predict_topic                        <
1/len(self.lda.show_topics())]
            self.input_data                                          =
self.input_data[self.input_data.predict_topic                        >
1/len(self.lda.show_topics())]
```

Після завершення роботи алгоритму усі таблиці зберігаються у БД інтелектуальної обробки тексту як нові документи. І вкінці до БД контенту відправляється ідентифікатори тем, які успішно обробилися.

```
     process_input = [{  'datetime': datetime.now(),

'lesson_part_id':x.lesson_part_id,
                                 'text': x.text,
                                 'author_topic':x.topic_name,

'predict_topic':x.predict_topic_name,
                                 'probability':x.probability}
      for i,x in self.input_data.iterrows()]
     non_process_topic = [{
                                 'datetime': datetime.now(),

'lesson_part_id':x.lesson_part_id,
                                 'text': x.text,
                                 'author_topic':x.topic_name,
     'predict_topic':None,
     'probability':None  }
     for i,x in self.non_processed_topics.iterrows()]

     client = MongoClient(self.MongoConn)
     db=client.admin
     serverStatusResult=db.command("serverStatus")
     topic_modeling_db = client.topic_modeling_db
     topic_modeling_db.lesson_part_processed.insert_many(process
_input
topic_modeling_db.non_processed.insert_many(non_process_topic)
     self.post_data()
```

5.6 Результати тематичного моделювання

Після відпрацювання вищеназваних функцій можна перевірити коректність моделі. По-перше, можна подивитися, скільки створено тем, які слова належать до кожної з них, та яка вага слова в цій темі (його впливовість на сенс теми):

```
[(0,
  '0.038*"решение" + 0.026*"первый_член" + 0.025*"множество_решение" + 0.021*"член" + 0.019*"решить_система" + 0.018*"система" + 0.016*"график_функция" + 0.013*"первый" + 0.013*"который" + 0.013*"равный"'),
 (1,
  '0.036*"график_функция" + 0.025*"график" + 0.019*"система_уравнение" + 0.016*"построить_график" + 0.016*"последовательность" + 0.015*"точка_графика" + 0.013*"натуральный_число" + 0.013*"точка" + 0.012*"система" + 0.012*"промежуток"'),
 (2,
  '0.029*"квадратичный_функция" + 0.020*"являться" + 0.018*"решение" + 0.017*"промежуток" + 0.015*"каков_вероятность" + 0.013*"график" + 0.012*"область_определение" + 0.012*"первый" + 0.012*"кмч_кмч" + 0.011*"который"'),
 (3,
  '0.040*"первый_член" + 0.023*"геометрический_прогрессия" + 0.020*"первый" + 0.020*"член" + 0.019*"область_определение" + 0.016*"квадратичный_функция" + 0.015*"смотреть_смотреть" + 0.014*"смотреть" + 0.014*"сумма" + 0.012*"an"'),
 (4,
  '0.023*"значение_выражение" + 0.019*"получить" + 0.019*"верный_неравенство" + 0.015*"доказать" + 0.013*"привести_пример" + 0.012*"часть" + 0.012*"каков_вероятность" + 0.011*"сумма" + 0.011*"один" + 0.011*"день"'),
 (5,
  '0.054*"построить_график" + 0.045*"какой_значение" + 0.039*"арифметический_прогрессия" + 0.028*"график" + 0.027*"член" +
```

```
0.021*"какой" + 0.020*"построить" + 0.020*"иметь" +
0.018*"арифметический" + 0.015*"последовательность"')]
```

Результат моделі можна представити графічно (рис. 5.3):

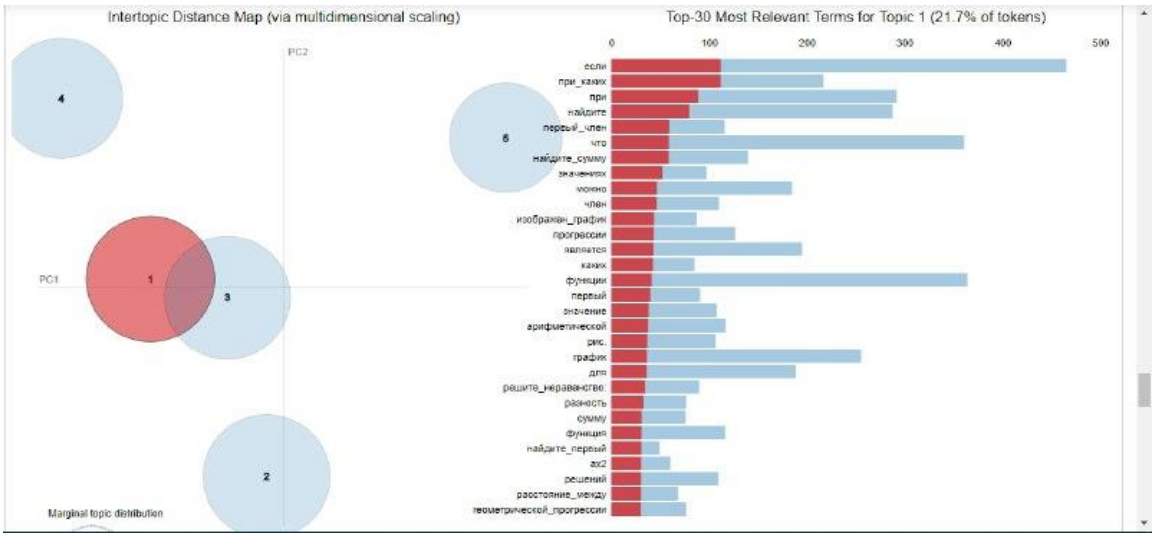

Рисунок 5.3 – Графічне представлення LDA моделі

Коло означає тему. Якщо теми розташовані поруч, то це свідчить про те, що теми в тексті близькі один до одного.

На рисунку 5.3 можна побачити, що 1 і 3 теми близькі між собою, тоді як 6 найбільш віддалена.

Графік справа показує розподілення 30 слів, які зустрічаються найчастіше в тексті. Синім кольором показана загальна кількість слова в корпусі, а червоним показана кількість слова в текстах по темі.

Вхідні дані (рис. 5.4) представленні таблицею: унікальний ідентифікатор кроку, текст та тема, яку призначив автор:

	lesson_part_id	text	topic_name
0	-1	Если второй член геометрической прогрессии мен...	геометрическая прогрессия
1	-2	Построить График функции пересекает координат...	геометрическая прогрессия
2	-3	Graph	геометрическая прогрессия

Рисунок 5.4 – Вхідні дані

У результаті роботи моделі отримані такі вихідні дані (рис. 5.5). Де перша частина – дані, для яких модель знайшла тему (знайдена тема зберігається в атрибуті predict_topic_name). Як можна побачити, замість ідентифікатору теми присвоєно слово, яке має найбільшу вагу в темі.

Успішно знайдені теми

```
lda.input_data
```

	lesson_part_id	text	topic_name	probability	predict_topic	predict_topic_name
0	-1	Если второй член геометрической прогрессии мен...	геометрическая прогрессия	0.913985	6	геометрический_прогрессия
1	-2	Построить График функции пересекает координат...	геометрическая прогрессия	0.665511	5	построить_график

Теми не знайдені

```
lda.non_processed_topics
```

	lesson_part_id	text	topic_name	probability	predict_topic
2	-3	Graph	геометрическая прогрессия	0.142857	0

Рисунок 5.4 – Вихідні дані моделі

В атрибуті probability зберігається ймовірність знайденої теми. Обирається найвища ймовірність. У майбутньому можна ігнорувати теми, для якай ймовірність нижча за порогову.

У другу таблицю попадають ті тексти, для яких модель не може спрогнозувати ймовірність (буде рівномірне розподілення ймовірності на всі теми).

У додатку Д продемонстровано успішне спілкування модулю з gateway. Спілкування реалізоване як генерація get запиту та ортримання даних з БД контенту. Далі йде обробка даних. Можна побачити жовте речення. Це означає, що отримано текст, для якої не визначено тематику. У даному випадку це текст на англійській мові з кроку 3. Результат роботи сервісу

зберігається в БД інтелектуальної обробки тектсу.

5.7 Оцінювання узгодженості тем

Оцінювання узгодженості (релевантності) оригінальної та прогнозованої тем тексту реалізована методом відстані Хелінгера (формула 5.8.1). Цей метод обраний замість косинусної відстані, так як відстань Хелінгера працює з ймовірними розподілами і краще вирішує задачу оцінки релевантності в рамках моделі [13].

$$H(P,Q) = 1/2 \int(\sqrt{dP/d\lambda} - \sqrt{dQ/d\lambda})d\lambda$$

5.8.1

де P і Q позначають дві ймовірні міри;

λ – третя міра ймовірності.

```
predict_topic = ["построить_график"]
original_topic = ["геометрическая_прогрессия"]
model = lda.lda
bow_predict = model.id2word.doc2bow(predict_topic)
bow_original = model.id2word.doc2bow(original_topic)
lda_bow_predict = model[bow_predict]
lda_bow_original = model[bow_original]

from gensim.matutils import hellinger
print("Релевантность темы:",hellinger(lda_bow_predict,
lda_bow_original))
```

В результаті отримаємо значення відстані між темами, чим менше значення, тим вища узгодженість. Для однакових тем відстань дорівнює 0.

```
Релевантность темы: 0.328793979451613
```

ВИСНОВКИ

Створений єдиний інструмент організації і управління даними, запропонований підхід до контрольованого внесення змін до структури даних, веденню актуальної документації по структурі цих усіх модулів, створений модуль інтелектуальної обробки даних, який реалізує метод тематичного моделювання для визначення релевантності теми.

Для модуля управління даними використані PowerDesigner, реляційна СУБД PostgreSQL і нереляційна СУБД MongoDB. Для створення моделі були використані такі технології: мова програмування Python, бібліотека для попередньої обробки й очищення даних NLTK, аналіз морфем російської мови pymorphy2, бібліотека тематичного моделювання Gensim, візуалізація результатів моделі pyLDAvis і відстежування результатів експериментів моделі з допомогою mlflow.

Для модуля управління даними подальший розвиток може включати створення сховища даних і OLAP обробка даних. Для сервісу інтелектуальної обробки даних – впровадження інструментів класифікації та кластеризації текстів.

Матеріали роботи доповідались на вісімнадцятій всеукраїнській конференції студентів і молодих науковців «Інформатика, інформаційні системи та технології» (Одеса, 23 квітня 2021 р.) та 77-ї звітній студентській науковій конференції Одеського національного університету імені І.І. Мечникова (26 – 27 квітня 2021 року) [17].

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The Data Management Body of Knowledge (DAMA-DMBOK2) Technics Publications, 2017. — 644 p. — ISBN 978-1634622349
2. Han, Jiawei. Data mining: concepts and techniques / Jiawei Han, Micheline Kamber, Jian Pei. – 3rd ed. p. cm. ISBN 978-0-12-381479-1
3. Управление мастер-данными в микросервисной архитектуре. [Електронний ресурс]. — Режим доступу: https://blog.byndyusoft.com/управление-мастер-данными-в-микросервисной-архитектуре-412fb6b7e6f8
4. Тематическое моделирование. [Електронний ресурс]. — Режим доступу: http://www.machinelearning.ru/wiki/index.php?title=Тематическое_моделирование
5. Бенгфорт Б., Билбро Р., Охеда Т. Прикладной анализ текстовых данных на Python. Машинное обучение и создание приложений обработки естественного языка. — СПб.: Питер, 2019. — 368 с.
6. Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008.
7. Yannis Haralambous. Text Mining Methods Applied to Mathematical Texts. CICM 2012 : Conferences on Intelligent Computer Mathematics, Jul 2012, Brême, Germany. ffhal-01864536]
8. Topic Modeling with LSA, PLSA, LDA & lda2Vec. [Електронний ресурс]. — Режим доступу: https://medium.com/nanonets/topic-modeling-with-lsa-psla-lda-and-lda2vec-555ff65b0b05
9. Landauer, T. K., Foltz, P. W., & Laham, D. (1998). Introduction to Latent Semantic Analysis. Discourse Processes, 25, 259-284.
10. Probabilistic Latent Semantic Analysis Thomas Hofmann EECS Department, Computer Science Division, University of California, Berkeley & International Computer Science Institute, Berkeley, CA

11. Latent Dirichlet Allocation. David M. Blei, Andrew Y. Ng and Michael I. Jordan. University of California, Berkeley. Berkeley, CA 94720
12. Что такое PowerDesigner. [Електронний ресурс]. — Режим доступу: https://www.sybase.ru/system/files/pdf/2009_what_pd_v10.pdf
13. Шемякин А. Е. Новый подход к построению объективных априорных распределений: информация Хеллингера // Прикладная эконометрика. 2012. №4 (28). [Електронний ресурс]. — Режим доступу: https://cyberleninka.ru/article/n/novyy-podhod-k-postroeniyu-obektivnyh-apriornyh-raspredeleniy-informatsiya.
14. Natural Language Processing with Python. [Електронний ресурс]. — Режим доступу: https://www.nltk.org/book/
15. Что такое ER-диаграмма и как ее создать? [Електронний ресурс]. — Режим доступу: https:// www.lucidchart.com/pages/ru/erd-диаграмма
16. Козлов М. С., Трубіна Н. Ф. Розробка архітектури навчальної платформи / Інформатика, інформаційні системи та технології: тези доповідей вісімнадцятої всеукраїнської конференції студентів і молодих науковців. Одеса, 23 квітня 2021 р. – Одеса, 2021. – с. 164-166
17. Петрушина Т. І., Ткаченко А. М. Тематичне моделювання текстів навчальної системи / Інформатика, інформаційні системи та технології: тези доповідей вісімнадцятої всеукраїнської конференції студентів і молодих науковців. Одеса, 23 квітня 2021 р. – Одеса, 2021. – с. 166-167

ДОДАТОК А

Документація до БД

Таблиця А.1 Атрибути сутності Answer

Назва	Тип даних	Обов'язковість
correct_answer	Text	X
id	Integer	X
question_id	Integer	

Таблиця А.2 Зв'язки з сутностю Answer

Назва	Сутність 2	Сутність1	Сутність 1 -> Сутність 2 кардинальність	Сутність 2 -> Сутність 1 кардинальність
question_has_answer	Answer	Question	0,n	0,1

Опис сутності Answer:

Варіанти відповіді на питання

Таблиця А.3 Зв'язки з сутностю Attached_material

Назва	Сутність 2	Сутність1	Сутність 1 -> Сутність 2 кардинальність	Сутність 2 -> Сутність 1 кардинальність
Lesson_has_attached	Attached_material	Lesson	0,n	0,1
Relationship_18	Attached_material (Shortcut)	Lesson (Shortcut)	0,n	0,1

Таблиця А.4. Атрибути сутності Attached_material

Назва	Тип даних	Обов’язковість
attached_id	Integer	X
path	Text	X

Таблиця А.5 Атрибути сутності Comment

Назва	Тип даних	Обов’язковість
comment_id	Serial	X
comment_type	Text	X
lesson_id	Integer	
lesson_part_id	Integer	
module_id	Integer	
parent_comment_id	Integer	
text	Text	X
time	Date & Time	X
uuid	Text	X

Опис сутності Comment: Комментаріхї до модулю, уроку чи кроку

Таблиця А.6 Зв’язки з сутностю Comment

Назва	Сутність 2	Сутність1	Сутність 1 -> Сутність 2 кардинальність	Сутність 2 -> Сутність 1 кардинальність
Lesson_has_module	Comment	Lesson	0,n	0,1
Lesson_part_has_comment	Comment	Lesson_part	0,n	0,1

User_post_comment	Comment	User	0,n	0,1

Сутність Course

Таблиця А.7 Атрибути сутності Course

Назва	Тип даних	Обов'язковість
course_id	Serial	X
course_status	Text	X
description	Text	X
language	Text	X
name	Text	X
readiness	Integer	X
requirement_description	Text	X
short_description	Text	X
target_description	Text	X
video	Text	
work_load	Integer	

Таблиця А.8 Зв'язки з сутностю Course

Назва	Сутність 2	Сутність1	Сутність 1 -> Сутність 2 кардинальність	Сутність 2 -> Сутність 1 кардинальність
Course_belongs_user	Course	User_course	1,1	0,n
Course_has_module	Module	Course	0,n	1,1

Сутність Lesson

Опис сутності Lesson:

Урок модулю

Таблиця А.9 Атрибути сутності Lesson

Назва	Тип даних	Обов’язковість
attached_id	Integer	
descripion	Text	X
lesson_id	Serial	X
lesson_position	Short integer	X
module_id	Integer	X
name	Text	X
requier	Boolean	X

Таблиця А.10 Зв’язки з сутностю Lesson

Назва	Сутність 2	Сутність1	Сутність 1 -> Сутність 2 кардинальність	Сутність 2 -> Сутність 1 кардинальність
Lesson_has_attached	Attached_material	Lesson	0,n	0,1
Lesson_has_lesson_part	Lesson_part	Lesson	0,n	0,1
Lesson_has_module	Comment	Lesson	0,n	0,1
Module_has_lesson	Lesson	Module	0,n	0,1
Relationship_18	Attached_material (Shortcut)	Lesson (Shortcut)	0,n	0,1

Сутність Lesson_part

Опис сутності Lesson_part:

Крок уроку, може бути тест, відео чи текст

Таблиця А.11 Атрибути сутності Lesson_part

Назва	Тип даних	Обов’язковість
data	Text	X
lesson_id	Integer	X
lesson_part_id	Serial	X
lesson_part_position	Short integer	X
lesson_part_type	Text	X

Таблиця А.12 Зв’язки з сутностю Lesson_part

Назва	Сутність 2	Сутність1	Сутність 1 -> Сутність 2 кардинальність	Сутність 2 -> Сутність 1 кардинальність
Action_on_lesson_part	Lesson_part_action	Lesson_part	0,n	0,1
Lesson_has_lesson_part	Lesson_part	Lesson	0,n	0,1
Lesson_has_topic	Lesson_part_topic	Lesson_part	1,n	1,1
Lesson_part_has_comment	Comment	Lesson_part	0,n	0,1
Lesson_part_has_question	Question	Lesson_part	0,n	0,1
Lesson_part_has_submission	Lesson_part_submission	Lesson_part	0,n	0,1

lesson_topic_predict	Topic_predict	Lesson_part	0,1	1,1
non_proccessed_lesson	Non_processed_topics	Lesson_part	0,n	0,1

Сутність Lesson_part_action

Таблиця А.13 Атрибути сутності Lesson_part_action

Назва	Тип даних	Обов’язковість
lesson_part_action_id	Serial	X
lesson_part_id	Integer	X
step_action_finish	Date & Time	
step_action_start	Date & Time	X
step_action_type	Text	X
uuid	Text	X

Таблиця А.14 Зв’язки з сутностю Lesson_part_action

Назва	Сутність 2	Сутність1	Сутність 1 -> Сутність 2 кардинальність	Сутність 2 -> Сутність 1 кардинальність
Action_on_lesson_part	Lesson_part_action	Lesson_part	0,n	0,1
User_action_on_lesson_part	User	Lesson_part_action	0,n	0,1

Сутність Lesson_part_submission

Опис сутності **Lesson_part_submission**

Відповіді на тест

Таблиця А.15 Атрибути сутності Lesson_part_submission

Назва	Тип даних	Обов’язковість
create_at	Date & Time	X
grade_at	Date & Time	
lesoon_part_submission_id	Serial	X
lesson_part_id	Integer	X
score	Integer	
status	Text	X
teacher_uuid	Text	
update_at	Date & Time	

Таблиця А.16 Зв’язки з сутностю Lesson_part_submission

Назва	Сутність 2	Сутність1	Сутність 1 -> Сутність 2 кардинальність	Сутність 2 -> Сутність 1 кардинальність
Lesson_part_has_submission	Lesson_part_submission	Lesson_part	0,n	0,1
User_post_submission	Lesson_part_submission	User	0,n	0,1

Сутність Lesson_part_topic

Таблиця А.17 Атрибути сутності Lesson_part_topic

Назва	Тип даних	Обов'язковість
lesson_part_id	Integer	X
property_id	Integer	X

Таблиця А.18 Зв'язки з сутностю Lesson_part_topic

Назва	Сутність 2	Сутність1	Сутність 1 -> Сутність 2 кардинальність	Сутність 2 -> Сутність 1 кардинальність
Lesson_has_topic	Lesson_part_topic	Lesson_part	1,n	1,1
Topic_refers_lesson_part	Lesson_part_topic	Topic	1,n	1,1

Сутність Module

Опис сутності **Module**

Модуль курсу

Таблиця А.19 Атрибути сутності Module

Назва	Тип даних	Обов'язковість
allow_comment	Boolean	X
course_id	Integer	X
description	Text	X
grading_policy	Text	X

is_available	Boolean	X
is_public	Boolean	X
module_id	Serial	X
module_position	Integer	X
name	Text	X

Таблиця А.20 Зв’язки з сутностю Module

Назва	Сутність 2	Сутність1	Сутність 1 -> Сутність 2 кардинальність	Сутність 2 -> Сутність 1 кардинальність
Course_has_module	Module	Course	0,n	1,1
Module_has_comment	Comment	Module	0,n	0,1
Module_has_lesson	Lesson	Module	0,n	0,1

Сутність Non_proccessed_topics

Таблиця А.21 Атрибути сутності Non_proccessed_topics

Назва	Тип даних	Обов’язковість
lesson_part_id	Integer	
lesson_part_text	Text	
topic_name	Text	

Опис сутності Non_proccessed_topics

Текст, для якого модель не знайшла тему.

Таблиця А.22 Зв’язки з сутностю Non_proccessed_topics

Назва	Сутність 2	Сутність1	Сутність 1 -> Сутність 2 кардинальність	Сутність 2 -> Сутність 1 кардинальність
non_proccessed_lesson	Non_proccessed_topics	Lesson_part	0,n	0,1

Сутність Question

Таблиця А.23 Атрибути сутності Question

Назва	Тип даних	Обов’язковість
id	Integer	X
lesson_part_id	Integer	X
name	Text	X
position	Text	

Сутність Topic

Таблиця А.25 Атрибути сутності Topic

Назва	Тип даних	Обов’язковість
property_id	Serial	X
topic_name	Text	X

Таблиця А.26 Зв’язки з сутностю Topic

Назва	Сутність 2	Сутність1	Сутність 1 -> Сутність 2 кардинальність	Сутність 2 -> Сутність 1 кардинальність
Topic_refers_lesson_part	Lesson_part_topic	Topic	1,n	1,1

Сутність Topic_predict

Таблиця А.27 Атрибути сутності Topic_predict

Назва	Тип даних	Обов’язковість
lesson_part_id	Integer	X
predict_topic_name	Text	X
probability	Float	
topic_name	Text	X

Опис сутності Topic_predict

БД для зберігання результатів моделі тематичного моделювання

Таблиця А.28 Зв’язки з сутностю Topic_predict

Назва	Сутність 2	Сутність1	Сутність 1 -> Сутність 2 кардинальність	Сутність 2 -> Сутність 1 кардинальність
lesson_topic_predict	Topic_predict	Lesson_part	0,1	1,1

ДОДАТОК Б

Концептуальна схема контент модулю

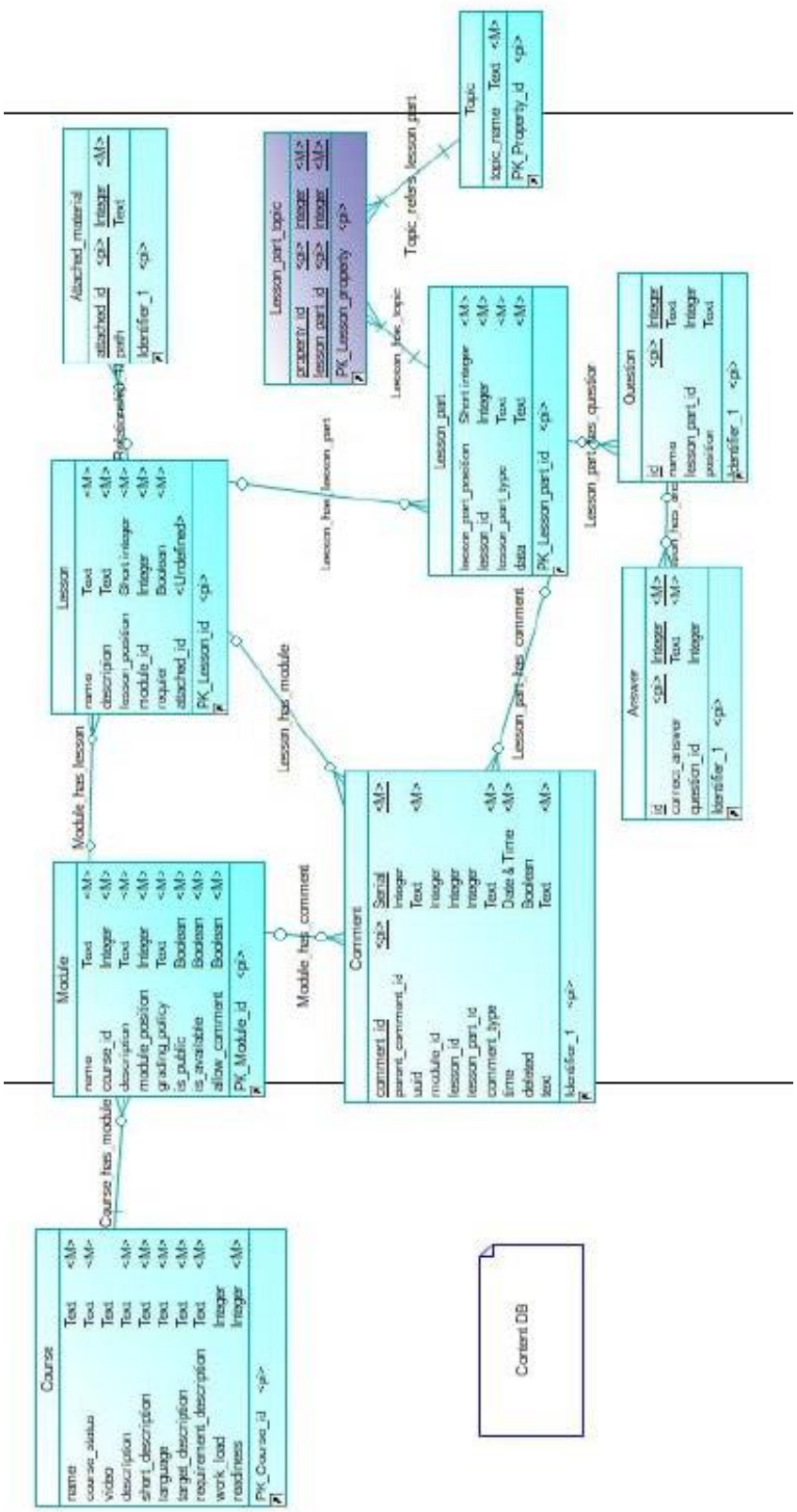

Рисунок Б.1 – Концептуальна схема контент

ДОДАТОК В

Фізична схема контент модулю

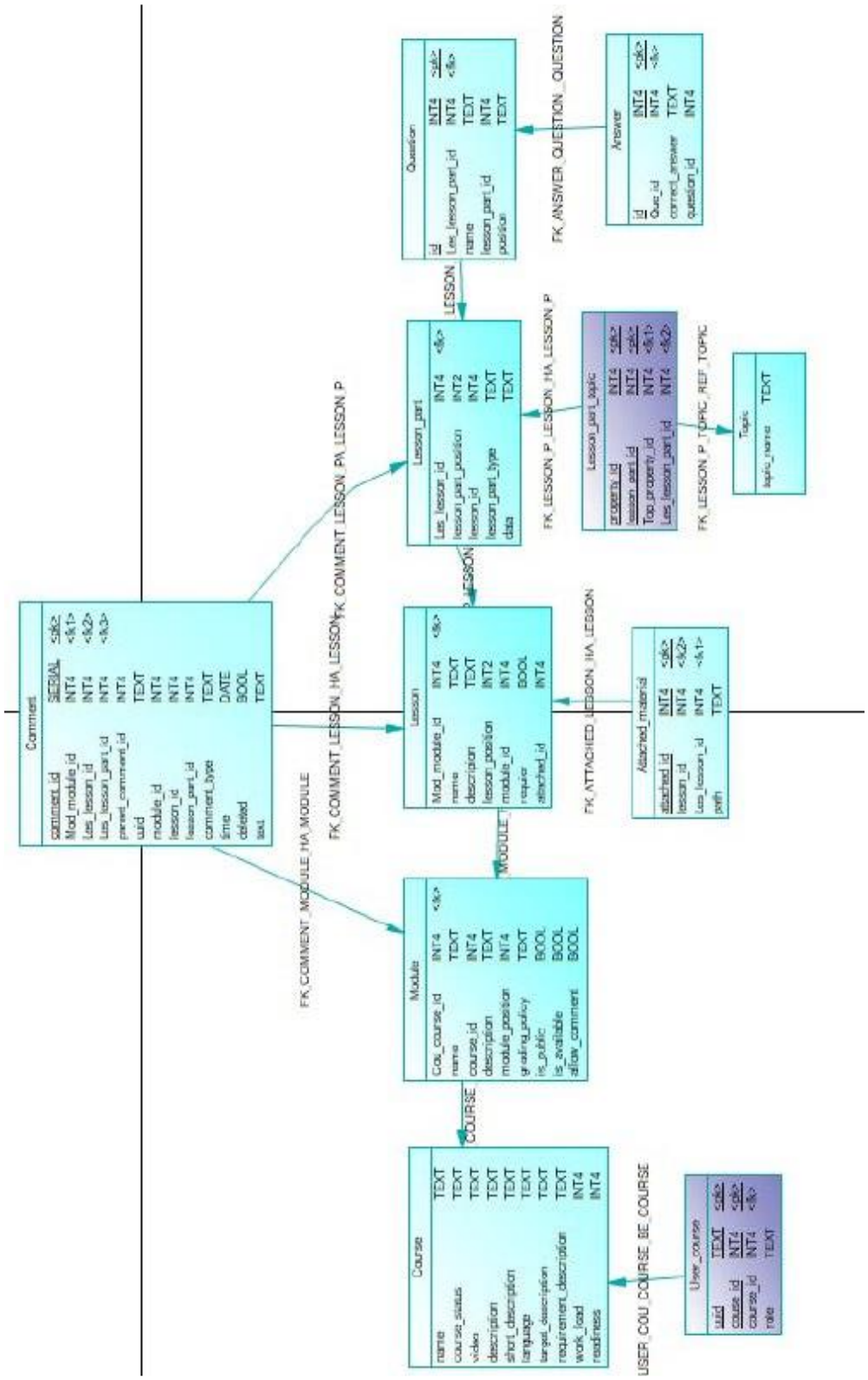

Рисунок В.1 – Фізична схема контент модулю

ДОДАТОК Г

Запити на створення БД

Запит на створення БД модулю авторизації

```
drop index User_action_on_lesson_part_FK;

drop index FK_User_has_Cred_FK;

drop index User_PK;

drop table "User";

drop index FK_User_has_Cred2_FK;

drop index User_cred_PK;

drop table User_cred;

drop index FK_User_has_session_FK;

drop index User_session_PK;

drop table User_session;

drop domain role;

/*==============================================================*/
/* Domain: role                                                 */
/*==============================================================*/
create domain role as TEXT;

/*==============================================================*/
/* Table: "User"                                                */
/*==============================================================*/
create table "User" (
   user_id              SERIAL               not null,
   uuid                 TEXT                 not null,
   email                TEXT                 not null,
   first_name           TEXT                 not null,
   last_name            TEXT                 not null,
   create_at            DATE                 not null,
   update_at            DATE                 null,
   constraint PK_USER primary key (user_id)
);

comment on column "User".user_id is
'Ідентифікатор запису у таблиці.';

/*==============================================================*/
/* Index: User_PK                                               */
/*==============================================================*/
create unique index User_PK on "User" (
```

```
user_id
);

/*==============================================================*/
/* Index: FK_User_has_Cred_FK                                   */
/*==============================================================*/
create  index FK_User_has_Cred_FK on "User" (

);

/*==============================================================*/
/* Index: User_action_on_lesson_part_FK                         */
/*==============================================================*/
create  index User_action_on_lesson_part_FK on "User" (

);

/*==============================================================*/
/* Table: User_cred                                             */
/*==============================================================*/
create table User_cred (
   user_cred_id         SERIAL               not null,
   user_id              INT4                 not null,
   encoded_pwd          TEXT                 not null,
   status               BOOL                 not null,
   uuid                 TEXT                 not null,
   update_pwd           DATE                 null,
   update_status        DATE                 null,
   constraint PK_USER_CRED primary key (user_cred_id)
);

comment on column User_cred.user_id is
'Ідентифікатор запису у таблиці.';

/*==============================================================*/
/* Index: User_cred_PK                                          */
/*==============================================================*/
create unique index User_cred_PK on User_cred (
user_cred_id
);

/*==============================================================*/
/* Index: FK_User_has_Cred2_FK                                  */
/*==============================================================*/
create  index FK_User_has_Cred2_FK on User_cred (
user_id
);

/*==============================================================*/
/* Table: User_session                                          */
/*==============================================================*/
create table User_session (
   user_session_id      SERIAL               not null,
   user_id              INT4                 not null,
   access_token         TEXT                 not null,
   refresh_token        TEXT                 not null,
   ip_address           TEXT                 not null,
```

```
   uuid                 TEXT                 not null,
   expires_in           DATE                 not null,
   created_at           DATE                 not null,
   updated_at           DATE                 null,
   constraint PK_USER_SESSION primary key (user_session_id)
);

comment on column User_session.user_id is
'Ідентифікатор запису у таблиці.';

/*==============================================================*/
/* Index: User_session_PK                                       */
/*==============================================================*/
create unique index User_session_PK on User_session (
user_session_id
);

/*==============================================================*/
/* Index: FK_User_has_session_FK                                */
/*==============================================================*/
create  index FK_User_has_session_FK on User_session (
user_id
);

alter table User_cred
   add constraint FK_USER_CRE_FK_USER_H_USER foreign key (user_id)
      references "User" (user_id)
      on delete restrict on update restrict;

alter table User_session
   add constraint FK_USER_SES_FK_USER_H_USER foreign key (user_id)
      references "User" (user_id)
      on delete restrict on update restrict;
```

Запит на створення БД модулю авторизації

```
drop index question_has_answer_FK;

drop index Answer_PK;

drop table Answer;

drop index Relationship_18_FK;

drop index Lesson_has_attached_FK;

drop index Attached_material_PK;

drop table Attached_material;

drop index Lesson_part_has_comment_FK;

drop index Lesson_has_module_FK;

drop index Module_has_comment_FK;

drop index Comment_PK;
```

```
drop table Comment;

drop index Course_PK;

drop table Course;

drop index Module_has_lesson_FK;

drop index Lesson_PK;

drop table Lesson;

drop index Lesson_has_lesson_part_FK;

drop index Lesson_part_PK;

drop table Lesson_part;

drop index Action_on_lesson_part_FK;

drop index Lesson_part_action_PK;

drop table Lesson_part_action;

drop index Lesson_part_has_submission_FK;

drop index Lesson_part_submission_PK;

drop table Lesson_part_submission;

drop index Lesson_has_topic_FK;

drop index Topic_refers_lesson_part_FK;

drop index Lesson_part_topic_PK;

drop table Lesson_part_topic;

drop index Course_has_module_FK;

drop index Module_PK;

drop table Module;

drop index Lesson_part_has_question_FK;

drop index Question_PK;

drop table Question;

drop index Topic_PK;

drop table Topic;

drop index Course_belongs_user_FK;
```

```
drop index User_course_PK;

drop table User_course;

drop domain role;

/*==============================================================*/
/* Domain: role                                                 */
/*==============================================================*/
create domain role as TEXT;

/*==============================================================*/
/* Table: Answer                                                */
/*==============================================================*/
create table Answer (
   id                   INT4                 not null,
   Que_id               INT4                 null,
   correct_answer       TEXT                 not null,
   question_id          INT4                 null,
   constraint PK_ANSWER primary key (id)
);

/*==============================================================*/
/* Index: Answer_PK                                             */
/*==============================================================*/
create unique index Answer_PK on Answer (
id
);

/*==============================================================*/
/* Index: question_has_answer_FK                                */
/*==============================================================*/
create  index question_has_answer_FK on Answer (
Que_id
);

/*==============================================================*/
/* Table: Attached_material                                     */
/*==============================================================*/
create table Attached_material (
   attached_id          INT4                 not null,
   lesson_id            INT4                 null,
   Les_lesson_id        INT4                 null,
   path                 TEXT                 null,
   constraint PK_ATTACHED_MATERIAL primary key (attached_id)
);

/*==============================================================*/
/* Index: Attached_material_PK                                  */
/*==============================================================*/
create unique index Attached_material_PK on Attached_material (
attached_id
);

/*==============================================================*/
/* Index: Lesson_has_attached_FK                                */
/*==============================================================*/
```

```
create  index Lesson_has_attached_FK on Attached_material (
Les_lesson_id
);

/*==============================================================*/
/* Index: Relationship_18_FK                                    */
/*==============================================================*/
create  index Relationship_18_FK on Attached_material (
lesson_id
);

/*==============================================================*/
/* Table: Comment                                               */
/*==============================================================*/
create table Comment (
   comment_id           SERIAL               not null,
   Mod_module_id        INT4                 null,
   Les_lesson_id        INT4                 null,
   Les_lesson_part_id   INT4                 null,
   parent_comment_id    INT4                 null,
   uuid                 TEXT                 not null,
   module_id            INT4                 null,
   lesson_id            INT4                 null,
   lesson_part_id       INT4                 null,
   comment_type         TEXT                 not null,
   "time"               DATE                 not null,
   deleted              BOOL                 null,
   text                 TEXT                 not null,
   constraint PK_COMMENT primary key (comment_id)
);

comment on column Comment.Mod_module_id is
'Ідентифікатор модулю';

/*==============================================================*/
/* Index: Comment_PK                                            */
/*==============================================================*/
create unique index Comment_PK on Comment (
comment_id
);

/*==============================================================*/
/* Index: Module_has_comment_FK                                 */
/*==============================================================*/
create  index Module_has_comment_FK on Comment (
Mod_module_id
);

/*==============================================================*/
/* Index: Lesson_has_module_FK                                  */
/*==============================================================*/
create  index Lesson_has_module_FK on Comment (
Les_lesson_id
);

/*==============================================================*/
/* Index: Lesson_part_has_comment_FK                            */
```

```
/*==============================================================*/
create  index Lesson_part_has_comment_FK on Comment (
Les_lesson_part_id
);

/*==============================================================*/
/* Table: Course                                                */
/*==============================================================*/
create table Course (
   course_id            SERIAL               not null,
   name                 TEXT                 not null,
   course_status        TEXT                 not null,
   video                TEXT                 null,
   description          TEXT                 not null,
   short_description    TEXT                 not null,
   language             TEXT                 not null,
   target_description   TEXT                 not null,
   requirement_description TEXT                 not null,
   work_load            INT4                 null,
   readiness            INT4                 not null default 0,
   constraint PK_COURSE primary key (course_id)
);

/*==============================================================*/
/* Index: Course_PK                                             */
/*==============================================================*/
create unique index Course_PK on Course (
course_id
);

/*==============================================================*/
/* Table: Lesson                                                */
/*==============================================================*/
create table Lesson (
   lesson_id            SERIAL               not null,
   Mod_module_id        INT4                 null,
   name                 TEXT                 not null,
   descripion           TEXT                 not null,
   lesson_position      INT2                 not null default 0,
   module_id            INT4                 not null,
   requier              BOOL                 not null default TRUE,
   attached_id          INT4                 null,
   constraint PK_LESSON primary key (lesson_id)
);

comment on column Lesson.Mod_module_id is
'Ідентифікатор модулю';

/*==============================================================*/
/* Index: Lesson_PK                                             */
/*==============================================================*/
create unique index Lesson_PK on Lesson (
lesson_id
);

/*==============================================================*/
/* Index: Module_has_lesson_FK                                  */
```

```
/*==============================================================*/
create  index Module_has_lesson_FK on Lesson (
Mod_module_id
);

/*==============================================================*/
/* Table: Lesson_part                                           */
/*==============================================================*/
create table Lesson_part (
   lesson_part_id       SERIAL               not null,
   Les_lesson_id        INT4                 null,
   lesson_part_position INT2                 not null,
   lesson_id            INT4                 not null,
   lesson_part_type     TEXT                 not null,
   data                 TEXT                 not null,
   constraint PK_LESSON_PART primary key (lesson_part_id)
);

/*==============================================================*/
/* Index: Lesson_part_PK                                        */
/*==============================================================*/
create unique index Lesson_part_PK on Lesson_part (
lesson_part_id
);

/*==============================================================*/
/* Index: Lesson_has_lesson_part_FK                             */
/*==============================================================*/
create  index Lesson_has_lesson_part_FK on Lesson_part (
Les_lesson_id
);

/*==============================================================*/
/* Table: Lesson_part_action                                    */
/*==============================================================*/
create table Lesson_part_action (
   lesson_part_action_id SERIAL               not null,
   Les_lesson_part_id   INT4                 null,
   lesson_part_id       INT4                 not null,
   uuid                 TEXT                 not null,
   step_action_type     TEXT                 not null,
   step_action_start    DATE                 not null,
   step_action_finish   DATE                 null,
   constraint PK_LESSON_PART_ACTION primary key (lesson_part_action_id)
);

/*==============================================================*/
/* Index: Lesson_part_action_PK                                 */
/*==============================================================*/
create unique index Lesson_part_action_PK on Lesson_part_action (
lesson_part_action_id
);

/*==============================================================*/
/* Index: Action_on_lesson_part_FK                              */
/*==============================================================*/
create  index Action_on_lesson_part_FK on Lesson_part_action (
```

```
Les_lesson_part_id
);

/*==============================================================*/
/* Table: Lesson_part_submission                                */
/*==============================================================*/
create table Lesson_part_submission (
   lesoon_part_submission_id SERIAL              not null,
   Les_lesson_part_id   INT4                 null,
   lesson_part_id       INT4                 not null,
   status               TEXT                 not null,
   score                INT4                 null,
   create_at            DATE                 not null,
   update_at            DATE                 null,
   grade_at             DATE                 null,
   teacher_uuid         TEXT                 null,
   constraint        PK_LESSON_PART_SUBMISSION       primary       key
(lesoon_part_submission_id)
);

/*==============================================================*/
/* Index: Lesson_part_submission_PK                             */
/*==============================================================*/
create unique index Lesson_part_submission_PK on Lesson_part_submission
(
lesoon_part_submission_id
);

/*==============================================================*/
/* Index: Lesson_part_has_submission_FK                         */
/*==============================================================*/
create  index Lesson_part_has_submission_FK on Lesson_part_submission (
Les_lesson_part_id
);

/*==============================================================*/
/* Table: Lesson_part_topic                                     */
/*==============================================================*/
create table Lesson_part_topic (
   property_id          INT4                 not null,
   lesson_part_id       INT4                 not null,
   Top_property_id      INT4                 not null,
   Les_lesson_part_id   INT4                 not null,
   constraint    PK_LESSON_PART_TOPIC    primary    key    (property_id,
lesson_part_id)
);

/*==============================================================*/
/* Index: Lesson_part_topic_PK                                  */
/*==============================================================*/
create unique index Lesson_part_topic_PK on Lesson_part_topic (
property_id,
lesson_part_id
);

/*==============================================================*/
/* Index: Topic_refers_lesson_part_FK                           */
```

```
/*==============================================================*/
create  index Topic_refers_lesson_part_FK on Lesson_part_topic (
Top_property_id
);

/*==============================================================*/
/* Index: Lesson_has_topic_FK                                   */
/*==============================================================*/
create  index Lesson_has_topic_FK on Lesson_part_topic (
Les_lesson_part_id
);

/*==============================================================*/
/* Table: Module                                                */
/*==============================================================*/
create table Module (
   module_id            SERIAL               not null,
   Cou_course_id        INT4                 not null,
   name                 TEXT                 not null,
   course_id            INT4                 not null,
   description          TEXT                 not null,
   module_position      INT4                 not null default 0,
   grading_policy       TEXT                 not null,
   is_public            BOOL                 not null default FALSE,
   is_available         BOOL                 not null default FALSE,
   allow_comment        BOOL                 not null default TRUE,
   constraint PK_MODULE primary key (module_id)
);

comment on column Module.module_id is
'Ідентифікатор модулю';

comment on column Module.name is
'Назва модулю';

/*==============================================================*/
/* Index: Module_PK                                             */
/*==============================================================*/
create unique index Module_PK on Module (
module_id
);

/*==============================================================*/
/* Index: Course_has_module_FK                                  */
/*==============================================================*/
create  index Course_has_module_FK on Module (
Cou_course_id
);

/*==============================================================*/
/* Table: Question                                              */
/*==============================================================*/
create table Question (
   id                   INT4                 not null,
   Les_lesson_part_id   INT4                 null,
   name                 TEXT                 not null,
   lesson_part_id       INT4                 not null,
```

```
   "position"           TEXT                 null,
   constraint PK_QUESTION primary key (id)
);

/*==============================================================*/
/* Index: Question_PK                                           */
/*==============================================================*/
create unique index Question_PK on Question (
id
);

/*==============================================================*/
/* Index: Lesson_part_has_question_FK                           */
/*==============================================================*/
create  index Lesson_part_has_question_FK on Question (
Les_lesson_part_id
);

/*==============================================================*/
/* Table: Topic                                                 */
/*==============================================================*/
create table Topic (
   property_id          SERIAL               not null,
   topic_name           TEXT                 not null,
   constraint PK_TOPIC primary key (property_id)
);

/*==============================================================*/
/* Index: Topic_PK                                              */
/*==============================================================*/
create unique index Topic_PK on Topic (
property_id
);

/*==============================================================*/
/* Table: User_course                                           */
/*==============================================================*/
create table User_course (
   uuid                 TEXT                 not null,
   couse_id             INT4                 not null,
   course_id            INT4                 not null,
   role                 role                 not null,
   constraint PK_USER_COURSE primary key (uuid, couse_id)
);

/*==============================================================*/
/* Index: User_course_PK                                        */
/*==============================================================*/
create unique index User_course_PK on User_course (
uuid,
couse_id
);

/*==============================================================*/
/* Index: Course_belongs_user_FK                                */
/*==============================================================*/
create  index Course_belongs_user_FK on User_course (
```

```
course_id
);

alter table Answer
   add constraint FK_ANSWER_QUESTION__QUESTION foreign key (Que_id)
      references Question (id)
      on delete restrict on update restrict;

alter table Attached_material
   add    constraint    FK_ATTACHED_LESSON_HA_LESSON    foreign    key
(Les_lesson_id)
      references Lesson (lesson_id)
      on delete restrict on update restrict;

alter table Attached_material
   add constraint FK_ATTACHED_RELATIONS_LESSON foreign key (lesson_id)
      references Lesson (lesson_id)
      on delete restrict on update restrict;

alter table Comment
   add    constraint    FK_COMMENT_LESSON_HA_LESSON    foreign    key
(Les_lesson_id)
      references Lesson (lesson_id)
      on delete restrict on update restrict;

alter table Comment
   add    constraint    FK_COMMENT_LESSON_PA_LESSON_P    foreign    key
(Les_lesson_part_id)
      references Lesson_part (lesson_part_id)
      on delete restrict on update restrict;

alter table Comment
   add    constraint    FK_COMMENT_MODULE_HA_MODULE    foreign    key
(Mod_module_id)
      references Module (module_id)
      on delete restrict on update restrict;

alter table Lesson
   add constraint FK_LESSON_MODULE_HA_MODULE foreign key (Mod_module_id)
      references Module (module_id)
      on delete restrict on update restrict;

alter table Lesson_part
   add    constraint    FK_LESSON_P_LESSON_HA_LESSON    foreign    key
(Les_lesson_id)
      references Lesson (lesson_id)
      on delete restrict on update restrict;

alter table Lesson_part_action
   add    constraint    FK_LESSON_P_ACTION_ON_LESSON_P    foreign    key
(Les_lesson_part_id)
      references Lesson_part (lesson_part_id)
      on delete restrict on update restrict;

alter table Lesson_part_submission
   add    constraint    FK_LESSON_P_LESSON_PA_LESSON_P    foreign    key
(Les_lesson_part_id)
```

```
      references Lesson_part (lesson_part_id)
      on delete restrict on update restrict;

alter table Lesson_part_topic
   add    constraint    FK_LESSON_P_LESSON_HA_LESSON_P    foreign    key
(Les_lesson_part_id)
      references Lesson_part (lesson_part_id)
      on delete restrict on update restrict;

alter table Lesson_part_topic
   add    constraint    FK_LESSON_P_TOPIC_REF_TOPIC    foreign    key
(Top_property_id)
      references Topic (property_id)
      on delete restrict on update restrict;

alter table Module
   add constraint FK_MODULE_COURSE_HA_COURSE foreign key (Cou_course_id)
      references Course (course_id)
      on delete restrict on update restrict;

alter table Question
   add    constraint    FK_QUESTION_LESSON_PA_LESSON_P    foreign    key
(Les_lesson_part_id)
      references Lesson_part (lesson_part_id)
      on delete restrict on update restrict;

alter table User_course
   add constraint FK_USER_COU_COURSE_BE_COURSE foreign key (course_id)
      references Course (course_id)
      on delete restrict on update restrict;
```

ДОДАТОК Д

Демонстрація спілкування з gateway

```
2021-06-20 14:24:35.600864 : connect to db
2021-06-20 14:24:36.464807 : select data
2021-06-20 14:24:36.465296 : get data error          1  \
   0
0 -1  Если второй член геометрической прогрессии мен...
1 -2  Построить График функции  пересекает координат...
2 -3                                              Graph

                2
0  геометрическая прогрессия
1  геометрическая прогрессия
2  геометрическая прогрессия
2021-06-20 14:24:36.539671 : start prediction
[(0, 0.014330527), (1, 0.014314425), (2, 0.014360135), (3, 0.014359763), (4, 0.014326867), (5, 0.014323241), (6, 0.9139851)]
[(0, 0.011035081), (1, 0.011037206), (2, 0.011046309), (3, 0.011026417), (4, 0.0110173235), (5, 0.6655112), (6, 0.2793265)]
[(0, 0.14285715), (1, 0.14285715), (2, 0.14285715), (3, 0.14285715), (4, 0.14285715), (5, 0.14285715), (6, 0.14285715)]
2021-06-20 14:24:37.461189 : end prediction
Warning: 1 topics are missing '
   lesson_part_id   text                topic_name  probability  \
2              -3  Graph  геометрическая прогрессия     0.142857

   predict_topic
                0
2
2021-06-20 14:24:37.466734 : save data. Topic_modeling_DB (MongoDB)
2021-06-20 14:24:37.686142 : success. Count of processed rows :2
2021-06-20 14:24:37.687830 : post data
```

Рисунок Д.1 – Демонстрація роботи модулю