

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

ФАКУЛЬТЕТ УПРАВЛІННЯ ФІНАНСАМИ ТА БІЗНЕСУ

Кафедра цифрової економіки та бізнес-аналітики

КУРСОВА РОБОТА

з навчальної дисципліни „Проектування та адміністрування
БД і СД”

Тема:

Інформаційна система курсів з програмування

Галузь знань: _____ 05 «Соціальні та поведінкові науки» _____

Спеціальність: _____ 051 «Економіка» _____

Спеціалізація: _____ «Інформаційні технології в бізнесі» _____

Освітній ступінь: _____ бакалавр _____

Науковий керівник:

к.е.н., Старух А.І.

(науковий ступінь, посада, прізвище, ініціали)

_____ “___” травня 2021 р.

(підпис)

Виконавець:

студент(ка) групи УФЕ-21 с

Прийма А. В.

_____ “___” травня 2021 р.

(підпис)

Загальна кількість балів _____

(підпис, ПП членів комісії)

ЛЬВІВ 2021

Зміст

ВСТУП.....	3
РОЗДІЛ І. АНАЛІЗ ВИМОГ	5
1.1 Постановка завдання.....	5
1.2 Розробка моделі варіантів використання веб-сайту;	6
2.1 Опис моделі даних.....	7
2.1.1 Системи управління базами даних	10
2.1.2 Проектування бази даних	13
2.2 Нормалізація відношень	15
2.2.1 Нормалізація плюси та мінуси	18
2.3 Визначення типів даних.....	19
2.3.1 Перелік таблиць бази даних	20
2.3.2 Перелік полів таблиць бази даних	20
2.3.3 Склад таблиць бази даних	22
2.4 Реалізація SQL-скрипту.....	24
ВИСНОВКИ.....	27
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	28
ДОДАТКИ.....	29

ВСТУП

Актуальність теми дослідження. На сьогоднішній день тема курсів програмування стало дуже популярною серед усіх вікових груп людей. Багато людей хочуть покращити свої навички в користування комп'ютера або освоїти нову професію. Тому що не залежно від сфери діяльності, всюди впроваджуються інформаційні системи та нові технології.

Багато хто зараз мріє про те, щоб отримати професію в сфері програмування, так як: ІТ-професії актуальні, перспективні, високооплачувані. Хоча зараз кількість інформації у відкритому доступі необмежена але через певний час люди неодмінно стикається з труднощами, з'являється невпевненість у своїх діях, початковий запал зникає та зона комфорту пропадає.

Кожен отримує з курсів щось своє. Для когось це можливість нарешті вивчити нову професію, що не виходило зробити самостійно. Хтось знаходить однодумців. Хтось збагачує своє портфоліо та дізнається нюанси напрямку. Так як зацікавленість у навчанні дуже важлива, а викладач курсу завжди готовий з цим допомогти, це стає явною перевагою перед самонавчанням.

Мета і завдання дослідження. *Метою курсової роботи є дослідження та аналіз актуальності інформаційної системи курсів з програмування.*

Для досягнення мети в роботі поставлено й вирішено такі теоретичні та практичні завдання:

- визначити особливості створення бази даних для інформаційної системи курсів з програмування ;
- розробити інформаційну систему (базу даних);
- перевірити програмний продукт на наявність дефектів та знайти шляхи покращення.

Об'єктом дослідження виступає конкретний website курсів програмування, для якого створюється інформаційна система та процес розробки відповідного програмного забезпечення.

Предметом є теоретичні, методичні та практичні аспекти розробки програмного забезпечення мовою запитів MySQL для створення бази даних курсів з програмування.

Практичне значення отриманих результатів. База даних може використовуватись для потенційною повноцінної інформаційної системи комп'ютерних курсів.

Використане програмне забезпечення. Для створення програмного продукту використовувалось середовище (PhpMyAdmin) та MySQL Workbench.

Структура роботи. Курсова робота складається з двох розділів («Аналіз вимог», «Розробка бази даних»), висновків, списку використаних джерел та додатків. Загальний обсяг роботи – 41 сторінок.

РОЗДІЛ I. АНАЛІЗ ВИМОГ

1.1 Постановка завдання

Діяльність інформаційної системи курсів з програмування – комплексний процес, що містить в собі безліч складових, які повинні бути реалізовані для комфортного навчання студентів. Таким чином, при розробці інформаційної бази для системи, потрібно було передбачати всі можливі процеси та задачі які могли виникнути у процесі навчання. Тому було прийняте рішення створити єдину базу даних, яка буде використовуватися різноманітними компонентами системи, створеними як на цьому етапі, так і в майбутньому. Це дозволить значно спростити пошук та редагування інформації, а також скоротить час наповнення системи потрібними даними. Для повноцінної роботи початкового функціоналу система повинна мати доступ до наступної інформації:

- перелік усіх необхідних даних по предметам та курсах, що викладаються;
- дані про контент який буде включений у курс;
- дані про усіх користувачів(викладачів та студентів)

Також, система повинна бути побудована таким чином, щоб у майбутньому її можна було розширити додатковою інформацією та функціоналом.

1.2 Розробка моделі варіантів використання веб-сайту;

Однією з цілей даної курсової роботи є розробка бази даних, яка буде забезпечувати стабільну та коректну роботу серверної частини інформаційної системи курсів програмування шляхом надання актуальної інформації для генерації відповідних даних. Як вже було зазначено, з того факту, що більшість даних використовують однорідні дані, було зроблено висновок про доцільність створення єдиної бази даних, яка б забезпечила своїм наповненням всю необхідну інформацію для генерації повного спектру контенту, що використовуються на сайті. Таке рішення допоможе запобігти появі помилок при наповненні БД, а також спростить задачу створення відповідного інтерфейсу користувача для редагування наявних даних. Таким чином, на даному етапі постає задача вибору типу бази даних, який буде більш доцільним для поставленої задачі, а також відповідного інструментарію для її розробки, який будуть задовольняти наступним вимогам:

- простота у використанні;
- безкоштовність;
- наявність достатньої кількості матеріалів з підтримкою у відкритому доступі.

Наступним етапом є проектування та реалізація БД, яка буде використовуватися серверною частиною системи для зберігання даних. При цьому мають бути враховані усі спільні та відмінні риси документів, що будуть генеруватися, а також повинен бути закладений потенціал для подальшого розширення структури БД, при появі необхідності доповнення системи новим функціоналом.

РОЗДІЛ II. РОЗРОБКА БАЗИ ДАНИХ

2.1 Опис моделі даних

База даних – сукупність даних, що організовані у відповідності до концептуальної структури, яка описує характеристики цих даних та відношення між ними, при чому така сукупність даних, яка підтримує одну або більше областей використання. У джерелах наводяться класифікації за декількома ознаками.

За ступенем розподіленості існують централізовані та розподілені БД. Централізовані – повністю підтримуються на одній ЕОМ. Розподіленими є ті, компоненти яких розміщуються в різних вузлах комп'ютерної мережі у відповідності до певного критерію. Очевидно, що розподілені БД є більш надійними, оскільки при відмові одного з вузлів у мережі інші залишаються дієздатними, а при умові, що кожний вузол буде мати резервні копії даних з інших вузлів, система буде мати високий рівень відмовостійкості. Проте, такі БД потребують значно більших ресурсів для підтримки роботи, а в контексті діяльності кафедри, потрібно враховувати обмеженість ресурсів, як людських, так і матеріальних. Також при взаємодії компонентів через мережу можна очікувати меншу швидкодію, порівняно з централізованою системою. Саме тому для вирішення даної задачі можна зупинитися на централізованому типі БД, що значно спростить підтримку та використання. Така БД зможе бути використана або на сервері кафедри, або безпосередньо встановлена на ЕОМ людини, відповідальної за укладання документів, що генеруються автоматизованою системою, яка розробляється. А надійність бази даних можна забезпечити регулярним створенням резервних копій – дамтів.

За середовищем постійного збереження виокремлюють БД, що зберігаються у вторинній, тобто постійній пам'яті комп'ютера (як правило жорсткий диск), в оперативній пам'яті, та третинній – змінних носіях інформації. В даному випадку, оскільки для підтримки документації важливим фактором є надійність збереження даних, оптимальним варіантом слід

вважати традиційну модель зберігання – у вторинній пам'яті. Вона програє у швидкодії оперативній пам'яті, проте менш вибаглива до характеристик ЕОМ, на якій буде працювати БД.

Ще однією принциповою класифікацією баз даних є класифікація за моделлю даних. Модель даних визначає логічну структуру БД та визначає основні принципи відносно того, як дані можуть зберігатися, організовуватись, та яким чином з ними можна працювати.

У ієрархічній моделі БД дані подаються у вигляді деревовидної структури. Вони зберігаються як записи, що поєднуються між собою за допомогою посилань (рис. 1.1). Запис – це набір полів, кожне з яких містить одне значення. Тип сутності запису визначає, які поля запис має.

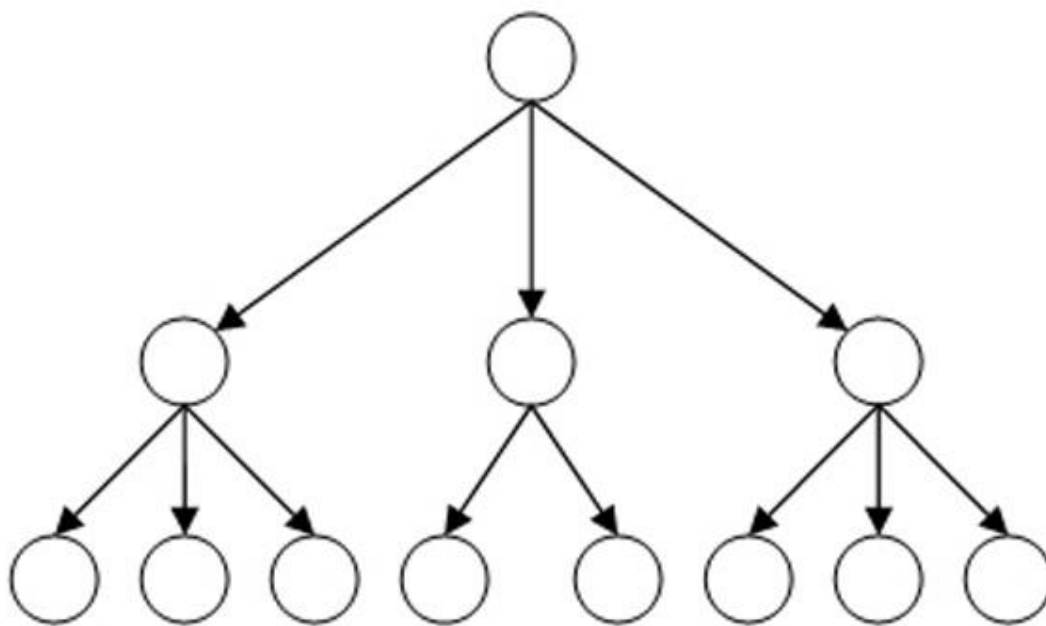


рис. 2.1

Запис у ієрархічній моделі БД відповідає рядку у реляційній моделі, а тип сутності відповідає таблиці.

Ієрархічна модель строго визначає, що кожний запис-нащадок, має тільки один запис-предок, в той час як один предок може мати декілька нащадків. Щоб отримати дані з ієрархічної моделі усе дерево необхідно обійти

починаючи від кореневого вузла. Ця модель визнана першою моделлю БД та створена IBM ще у 1960-х, тож багато спеціалістів вважає її морально застарілою.

У реляційній моделі використовується підхід оперування даними використовуючи структуру та мову, сумісну з логікою предикатів першого порядку, вперше описаною Едгаром Коддом, в якій усі дані представлені в термінах кортежів, згрупованих у відношення. БД, що організовані в рамках даної моделі, називаються реляційними базами даних (рис. 1.2).

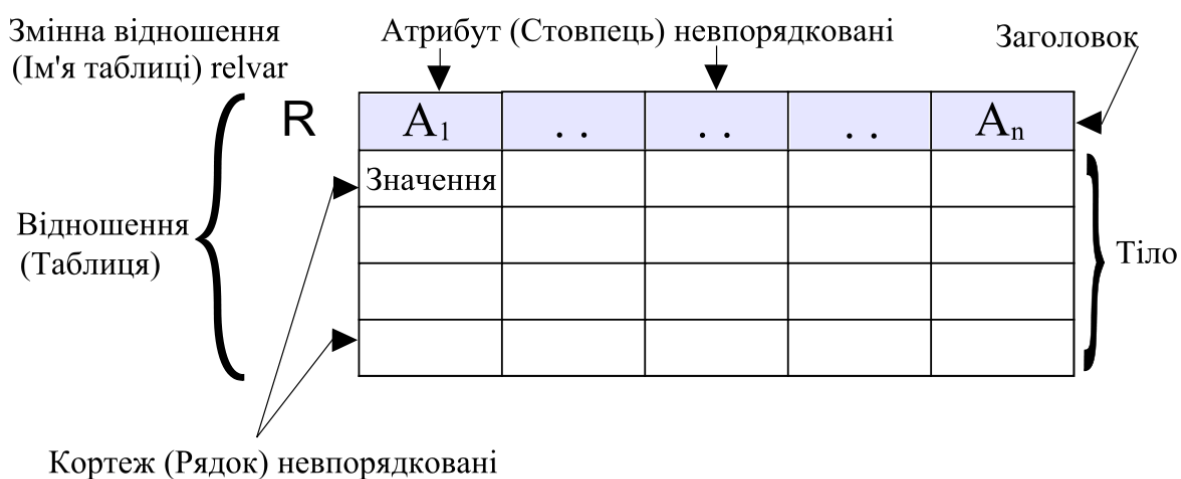


рис. 2.2

Метою реляційної моделі є надання декларативного методу визначення даних та запитів: користувачі чітко задають, яку інформацію містить БД, та яку вони хочуть отримати, залишаючи при цьому усі деталі реалізації на програмне забезпечення СУБД.

Більшість баз даних використовують мову SQL. Вона є стандартизованою, але різні реалізації в рамках певних СУБД можуть дещо відхилятися від цього стандарту з метою надання більших можливостей чи поліпшення швидкодії. Слід зазначити, що дана модель організації даних у БД є наразі найбільш поширеною, та має безліч реалізацій, як з відкритим програмним кодом, так і комерційних, коштовних систем.

У об'єктно-орієнтованій моделі інформація представляється у вигляді об'єктів, так само, як і в об'єктно-орієнтованих мовах програмування. Це дозволяє значно спростити налагодження взаємодії між БД та ОО-мовою програмування, оскільки не потрібно використовувати ORM-фреймворки, чи самостійно писати об'єкти для доступу до даних БД. Для використання в рамках даної роботи було прийнято рішення обрати реляційну базу даних. В даний час ця модель є фактичним стандартом, на який орієнтуються практично всі сучасні комерційні СУБД, оскільки даний формат має велику розповсюдженість, відповідно наявна велика кількість джерел інформації стосовно використання таких БД, а також наявна велика кількість СУБД під цей тип, тобто є можливість обрати оптимальний варіант для використання в системі, що розробляється.

Отже, для більш чіткого обґрунтування вибору типу бази даних, що буде використовуватись, наводиться перелік переваг реляційної бази даних :

- простота і доступність для розуміння користувачем. Єдиною використовуваною інформаційною конструкцією є «таблиця»;

- строгі правила проектування, які базуються на математичному апараті;

- повна незалежність даних. Зміни в прикладній програмі при зміні реляційної БД мінімальні;

- для організації запитів і написання прикладного ПЗ немає необхідності знати конкретну організацію БД у зовнішній пам'яті.

2.1.1 Системи управління базами даних

На сьогоднішній день існує досить багато систем управління базами даних.

PostgreSQL. Безкоштовна об'єктно-реляційна система управління базами даних. Вона є однією з найбільш просунутих СУБД, що в першу чергу орієнтується на повну відповідність стандартам та можливості розширення, тобто повністю намагається відповідати SQL-стандартам ANSI/ISO. З-поміж інших цю СУБД також виокремлює той факт, що вона має об'єктно

орієнтований функціонал та підтримує відповідні концепти. Система PostgreSQL заснована на ядрі, створеному безліччю розробників. У подібних випадках розумно зосередитися на оснащенні системи новими можливостями, але не займатися оптимальним їх втіленням, оскільки у випадку виникнення необхідності завжди можна буде повернутися до оптимізації відповідних ділянок коду.

Також варто зазначити що ця система не достатньо оптимізована для вирішення повсякденних не дуже важких задач. Її використання більше передбачається у випадках, коли важливими виявляються підвищена надійність та підтримки об'єктних підходів до БД. У випадку, коли більш важливим є виконання простих операцій зчитування-запису, PostgreSQL показує не найкращі результати.

Oracle Database або Oracle RDBMS. Об'єктно-реляційна система управління базами даних компанії Oracle Corporation. Ця СУБД забезпечує ефективне, надійне і безпечне управління даними таких критично важливих для бізнесу додатків, як онлайнові середовища, виконує масштабну обробку транзакцій (OLTP), сховища даних з високою інтенсивністю потоку запитів, а також ресурсоємні інтернет-додатки. Редакція Oracle Database Enterprise Edition надає інструментальні засоби і функції, що забезпечують відповідність вимогам сучасних корпоративних додатків в області доступності та масштабованості. Ця редакція містить всі компоненти Oracle Database, а також допускає розширення за допомогою придбання додаткових модулів та програм.

Система Oracle Database дозволяє звертатися до даних з будь-якого додатку, розробленого із застосуванням технологій Microsoft. NET, Visual Studio та веб-додатків. Основною умовою є лише наявність справних бібліотек, що дають змогу підключатися до серверу бази даних Oracle. Це комерційна СУБД, але є безкоштовна версія, яку можна без проблем скачати прямо з офіційного сайту компанії. Втім, варто зазначити, що безкоштовна версія має зменшений функціонал.

Microsoft SQL Server. Система Microsoft SQL Server відштовхується від концепції платформи даних Майкрософт: вона спрощує управління будь-якими даними в будь-якому місці і в будь-який момент часу. Вона дозволяє зберігати в базах даних інформацію, отриману з структурованих, напівструктурованих і неструктурованих джерел, таких як зображення та музика. У SQL Server є великий набір інтегрованих служб, які розширюють можливості використання даних: можна складати запити, виконувати пошук, провести синхронізацію, робити звіти, аналізувати дані. Всі дані зберігаються на основних серверах, що входять до складу центру обробки даних. До них здійснюється доступ з настільних комп'ютерів і мобільних пристроїв. Таким чином, можна повністю контролювати дані незалежно від того, де вони збережені.

Система MS SQL Server дозволяє звертатися до даних з будь-якого додатку, розробленого із застосуванням технологій Microsoft .NET та Visual Studio, а також в межах сервісно-орієнтованої архітектури і бізнес-процесів – через Microsoft BizTalk Server. SQL Server дозволяє створити надійну, продуктивну, інтелектуальну платформу, що відповідає всім вимогам по роботі з даними. Ця система є комерційною, тобто потребує значних витрат на встановлення та налагодження роботи.

Система MySQL. Безкоштовна система управління базами даних. MySQL є власністю компанії Oracle Corporation, що отримала її разом з поглиненою Sun Microsystems, що здійснює розробку і підтримку програми. Розповсюджується під GNU General Public License або під власною комерційною ліцензією. Крім цього розробники створюють функціональність за замовленням ліцензійних користувачів.

MySQL є рішенням для малих і середніх додатків. Входить до складу серверів WAMP, LAMP і в портативні збірки серверів Denver, XAMPP. Зазвичай MySQL використовується як сервер, до якого звертаються локальні або віддалені клієнти, проте в дистрибутиві входить бібліотека внутрішнього сервера, що дозволяє включати MySQL в автономні програми. СУБД, що обиралася, повинна задовольняти вимогам щодо простоти використання,

безкоштовності та наявності достатньої кількості довідкового матеріалу. Тому, вибір було зроблено на користь MySQL. По-перше, дана СУБД є безкоштовним open-source продуктом, тобто не викличе жодного фінансового навантаження при використанні. По-друге, супутній продукт MySQL Workbench дозволяє просто взаємодіяти зі встановленою БД, надаючи можливість через зручний інтерфейс користувача виконувати будь-які маніпуляції з даними. І нарешті, ця СУБД є однією з найбільш розповсюджених, займаючи друге місце за популярністю у світі, поступаючись лише корпоративній СУБД від Oracle . Саме тому в мережі наявна велика кількість матеріалів, як довідкових, включно з офіційним сайтом, так і навчальних, де розглядається широкий спектр проблем та шляхи їх вирішення.

Отже, для більш чіткого обґрунтування вибору СУБД, що буде використовуватись, наводиться перелік переваг MySQL :

Відкритий вихідний код та безкоштовність.

Продумана та швидкодіюча.

Займає небагато дискового простору.

Може бути легко встановлена на багатьох операційних системах, таких як Windows, Unix-like і інших.

Враховуючи її розповсюдження в Мережі можна знайти безліч матеріалів, в яких розглядається вирішення певних питань та проблем, що виникають при роботі з нею.

Добре підходить для невеликих застосунків.

2.1.2 Проектування бази даних

Проектування бази даних. Перед тим як створювати таблиці, форми та інші об'єкти, потрібно задати структуру бази даних. Добра структура бази даних є основою для створення адекватної вимогам, ефективної бази даних. Сам процес проектування бази даних являє собою складний процес проектування відображення опису предметної області у схему внутрішньої

моделі даних. Перебіг цього процесу є послідовністю більш простих процесів проектування менш складних відображень. Ця послідовність у процесі проектування весь час уточнюється, вдосконалюється таким чином, щоб були визначені об'єкти, їх властивості та зв'язки, які будуть потрібні майбутнім користувачам системи.

Проектування бази даних складається з декількох етапів і починається з попередньої структуризації предметної області. Насамперед, необхідно виділити всі об'єкти, які будуть використовуватись в базі даних, вказати їхні властивості (характеристики) і встановити зв'язки між ними. Цей етап називають концептуальним проектуванням бази даних. В процесі проектування структури бази даних потрібно створити діаграму концептуальної моделі даних. На основі визначених елементів і зв'язків створити ER – діаграму.

Діаграма концептуальної моделі даних представлена на рисунку 2.1

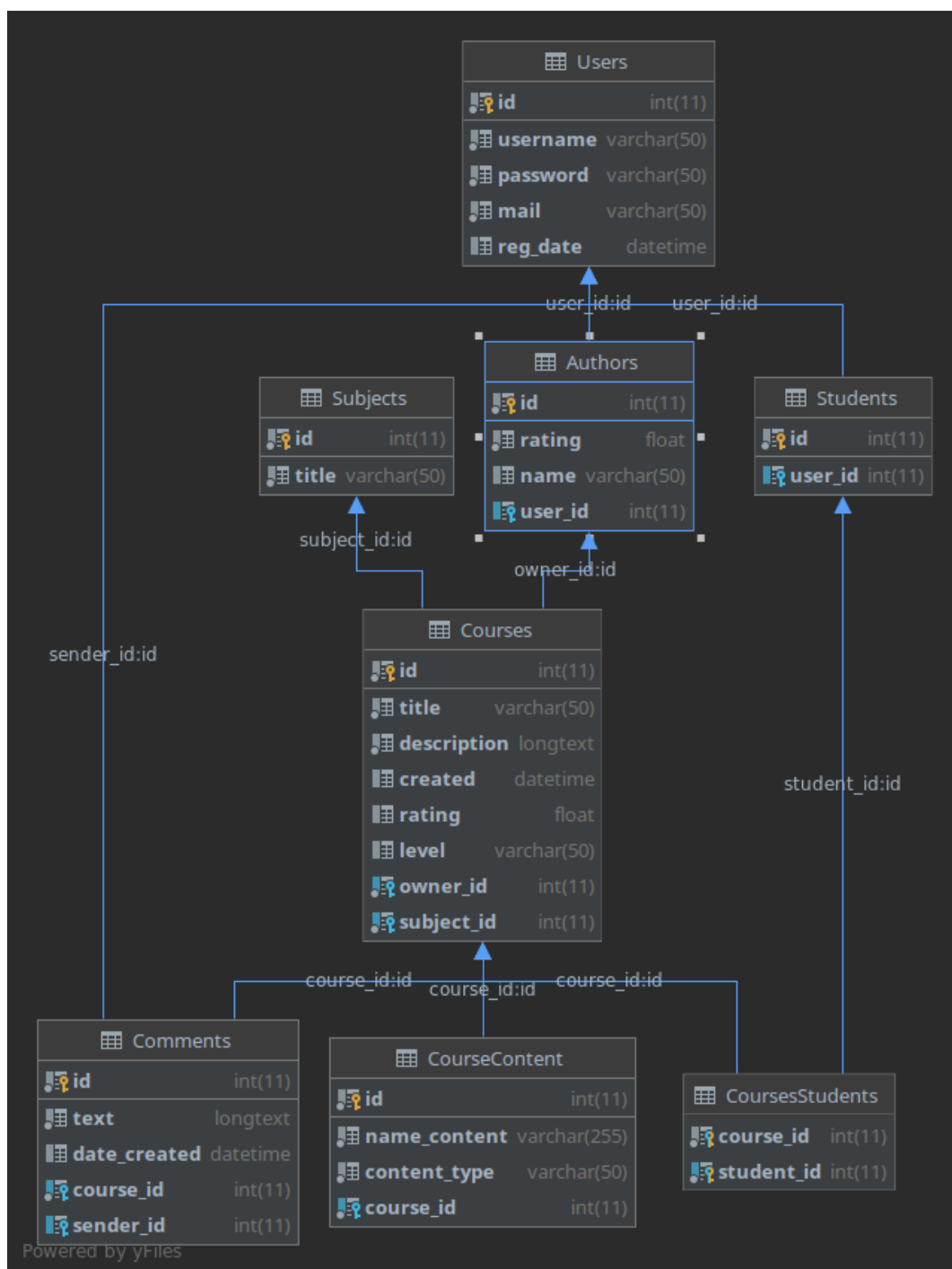


рис. 2.1 Концептуальна модель даних

2.2 Нормалізація відношень

При проектуванні баз даних в першу чергу робиться наголос на вірогідність та несуперечливість зберігаємих даних, до того ж, ці властивості не повинні втручатися в процесі роботи з даними, тобто після багаточисельних

змін, знищень та доповнень даних по відношенню до першопочаткового стану бази даних.

Для підтримки бази даних в стабільному стані використовується ряд механізмів, які отримали узагальнену назву засобів підтримки цілісності. Ці механізми використовуються статично (на етапі проектування бази даних), так і динамічно (в процесі роботи з базою даних).

База даних повинна володіти обмеженнями, які будуть задовольняти її в процесі створення, незалежно від її наповнення даними. Доведення структури бази даних до відповідності цим обмеженням – це нормалізація. В цілому суть цих обмежень дуже проста: кожен факт, який зберігається в базі даних, повинен зберігатися один єдиний раз, так як дублювання може призвести до незгоди між копіями однакової інформації. Треба уникати невизначеностей, а також надмірності зберігаємої інформації.

Отже, наша мета – доведення відношень до НФ. Треба відмітити, що в процесі нормалізації постійно зустрічається ситуація, коли відношення необхідно розкласти на декілька інших відношень. Тому коректно було б сказати про нормалізацію не окремих відношень, а всієї їх сукупності в базі даних.

Нормалізація може не викликати проблему, якщо база даних проектується одразу ж за визначеними правилами. Іншими словами, можна спочатку створити базу даних як-небудь, а потім нормалізувати її, або ж з самого початку будувати її за правилами для того, щоб в подальшому не треба було б її переробляти.

Нормалізація відношень забезпечує ефективність структур даних в реляційній БД. Цей процес зменшує надмірність даних (зберігання однакових даних в декількох місцях). В результаті більш раціонально використовується зовнішня пам'ять, зменшується ймовірність порушення узгодженості даних.

Нормалізація - це послідовне перетворення початкової бази даних до НФ, при цьому кожна наступна НФ обов'язково містить у собі попередню.

В реляційній теорії нараховують 6 НФ:

1 НФ;

2 НФ;

3 НФ;

НФ Бойса-Кодда (НФБК);

4 НФ;

5 НФ.

На практиці обмежуються 3 НФ, її достатньо для створення надійної схеми бази даних. НФ більш високих порядків представляють тільки академічну зацікавленість із-за надмірної складності. Крім того, при реалізації абстрактної схеми бази даних у вигляді реальної бази іноді розробники змушені зробити крок назад – провести денормалізацію з метою підвищення ефективності, так як ідеальна з точки зору теорії структура може статися занадто складною на практиці.

Основні властивості нормальних форм:

Кожна наступна нормальна форма покращує властивості попередньої нормальної форми.

При переході до наступної нормальної форми властивості попередніх нормальних форм зберігаються.

Принцип нормалізації ER-діаграм. Нормальні форми ER-діаграм мають багато спільного з нормалізацією реляційних відносин.

Для першої нормальної форми ER-схеми характерно відсутність повторюваних атрибутів. Проводиться виявлення так

званих неявних сутностей і створення на їх основі нових сутностей.

Для другої нормальної форми характерна відсутність атрибутів, залежних від частини унікального ідентифікатора суті. На основі цієї частини унікального ідентифікатора виділяється окрема сутність.

Для забезпечення третьої нормальної форми необхідно усунути атрибути, які залежать від атрибутів, що не входять в унікальний ідентифікатор. На основі цих атрибутів також будуються нові сутності.

Нормальні форми більш високих порядків в ER-моделях не знаходять практичного застосування.

ER-діаграма на рисунку 2.1 перебуває в 3NF, тому що сутності не мають властивостей, що залежать від не ключових властивостей

2.2.1 Нормалізація плюси та мінуси

Метою нормалізації відносин БД є усунення надлишкової інформації. Нормалізовані відносини БД містять тільки один елемент надлишкових даних - це атрибути зв'язку, присутні одночасно в батьківському і дочірньому відносинах. Оскільки надлишкові дані в стосунках не зберігаються, то економиться місце на носіях інформації. Однак в нормалізованій БД є і недоліки, перш за все, практичного характеру.

Чим більше число суб'єктів, які охоплюються предметною областю, тим з більшого числа відносин буде складатися нормалізована БД. Бази даних в складі великих систем, які задіяні в життєдіяльності великих організацій і підприємств, можуть містити сотні пов'язаних між собою відносин. Оскільки поріг людського сприйняття не дозволяє одночасно охопити велику кількість об'єктів з урахуванням їх взаємозв'язків, то можна стверджувати, що зі збільшенням числа нормалізованих відносин

цілісне сприйняття БД як системи взаємопов'язаних даних зменшується.

Тому при розробці та експлуатації великих систем існують ситуації, коли кожен співробітник уявляє собі процеси, що протікають тільки в частині системи. Відомі випадки еволюційного створення таких систем, коли принципи їх функціонування згодом виходили за межі розуміння.

Іншим недоліком нормалізованої БД є необхідність зчитувати з відносин пов'язані дані при виконанні складних запитів, які надають

інформацію про взаємодію сутностей технологічного процесу між собою. При великих обсягах даних це призводить до збільшення часу доступу до даних.

Третя нормальна форма є теоретичними конструкціями. Тому доречно зробити кілька зауважень про недоліки, властивих відноsinам, представленим в 3НФ. Існують варіанти, коли має сенс розділити ставлення на більш дрібні, якщо частина представлених в ньому даних непостійна і часто оновлюється (оперативна інформація), а інші дані пасивні і змінюються в рідкісних випадках (довідкова інформація). Також є сенс об'єднати відносини, коли необхідно забезпечити високу швидкість реакції на запит. Можна навіть піти на дублювання даних в таблицях, якщо це дозволить знизити витрати на обробку запитів

Необхідно також відзначити, що зв'язок між відносинами доцільно здійснювати по атрибутам, повністю звільненим від семантичної залежності, яка породжується особливостями реалізації реальних процесів, що відображаються в БД. Для цього використовують спеціально виділені інкрементні атрибути однозначної ідентифікації кортежів усередині відносин. Такий прийом дозволяє уникнути необхідності перевизначення зв'язків між відносинами при зміні в реальному житті правил обліку різних суб'єктів діяльності організації.

2.3 Визначення типів даних

Умовно таблиці, що складають всю БД для інформаційної системи курсів з програмування, можна поділити на два типи: таблиці-словники, та таблиці з інформацією, що може часто змінюватися, або доповнюватись. Таблиці-словники найчастіше містять в собі статичну інформацію, яка не змінюється протягом довгого періоду часу. Найпростішим прикладом такої таблиці в даній базі даних будуть співвідношення «ключ – розшифрування». Вони використовуються для визначення певних даних, як наприклад, назва предмету, чи тип контенту.

2.3.1 Перелік таблиць бази даних

Таблицями бази даних є:

authors - таблиця, яка містить інформацію про авторів курсів.

comments - таблиця, яка містить інформацію залишені коментарі про курси.

coursecontent - таблиця, яка містить інформацію про контент курсу.

courses - таблиця, яка містить інформацію про курс.

coursesstudents - таблиця, яка містить інформацію про студентів курсів.

students - таблиця, яка містить інформацію про студентів.

subjects - таблиця, яка містить інформацію про предмет курсу.

users - таблиця, яка містить інформацію про реєстраційні дані всіх користувачів.

2.3.2 Перелік полів таблиць бази даних

Поля, які ідентифікують властивості таблиці “authors”:

id - ідентифікатор автора

rating - рейтинг автора

name - ім'я під яким автор створює курси

user_id - ідентифікатор даних автора у таблиці всіх зареєстрованих користувачів

Поля, які ідентифікують властивості таблиці «comments »:

id - ідентифікатор коментаря

text - текст коментаря

date_created - дата створення коментаря

course_id - ідентифікатор курсу до якого відноситься коментар

sender_id - ідентифікатор користувача який написав коментар

Поля, які ідентифікують властивості таблиці «coursecontent»:

id - ідентифікатор контенту

name_content - ім'я контенту

content_type - тип контенту

course_id - ідентифікатор курсу до якого належить контент

Поля, які ідентифікують властивості таблиці «courses»:

id - ідентифікатор курсу

title - заголовок курсу

description - опис курсу

created datetime - час створення курсу

rating - рейтинг курсу

level - рівень складності курсу

owner_id - ідентифікатор автора курсу

subject_id - ідентифікатор предмету курсу

Поля, які ідентифікують властивості таблиці «coursesstudents»:

course_id - ідентифікатор курсу

student_id - ідентифікатор студента зареєстрованого на курс

Поля, які ідентифікують властивості таблиці «students »:

id - ідентифікатор студента

user_id - ідентифікатор даних студента у таблиці всіх зареєстрованих

користувачів

Поля, які ідентифікують властивості таблиці «subjects »:

id - ідентифікатор предмета

title - назва курсу

Поля, які ідентифікують властивості таблиці «users »:

id - ідентифікатор користувача

username - зареєстрований никнейм

password - пароль користувача

mail - email адрес користувача

reg_date - дата реєстрації користувача

2.3.3 Склад таблиць бази даних

База даних з точки зору MySQL – це звичайний каталог, що містить бінарні файли певного формату – таблиці. Таблиці складаються із записів, а записи, у свою чергу, складаються з полів. Поле має два атрибути – ім'я та тип.

Тип поля може бути:

числовим цілим;

числовим дійсним;

рядковим;

бінарним;

дата й час;

переліком і множиною.

Далі перераховані деякі з доступних типів у MySQL:

Числові типи даних.

Ці типи даних підтримують цілі числа без будь-якого десяткового подання. Існують різні підтипи, такі як: INT, TINYINT, MEDIUMINT, SMALLINT, BIGINT.

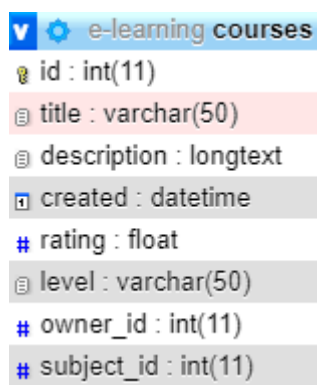
Типи з плаваючою комою є приблизними типами значень, і це залежить від значення по. десяткової точності, визначеної під час оголошення типу стовпця. Існує 2 типи типів даних із плаваючою комою: FLOAT та DOUBLE, які підтримують різні діапазони та споживають пам'ять / пам'ять.

Типи даних DateTime

Типи даних DateTime у MySQL, як випливає з назви, використовуються для зберігання значень дати та часу в базі даних MySQL. MySQL підтримує 2 часові типи - DATETIME і TIMESTAMP.

Типи даних рядка

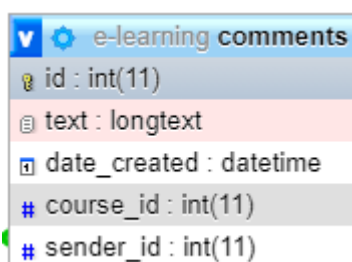
Рядкові типи даних, як випливає з назви, використовуються для зберігання рядків та текстів або крапок текстової інформації в базі даних. Залежно від варіанту використання доступні різні типи даних -CHAR, VARCHAR, BINARY, VARBINARY, TEXT, ENUM, SET & BLOB.



The screenshot shows the 'e-learning courses' table structure. It includes a primary key 'id' of type int(11). Other fields are 'title' (varchar(50)), 'description' (longtext), 'created' (datetime), 'rating' (float), 'level' (varchar(50)), 'owner_id' (int(11)), and 'subject_id' (int(11)).

e-learning courses	
id	int(11)
title	varchar(50)
description	longtext
created	datetime
rating	float
level	varchar(50)
owner_id	int(11)
subject_id	int(11)

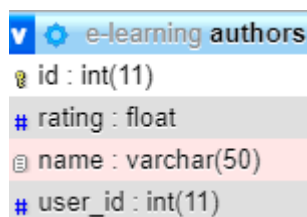
рис 2.2 Типи даних таблиці courses



The screenshot shows the 'e-learning comments' table structure. It includes a primary key 'id' of type int(11). Other fields are 'text' (longtext), 'date_created' (datetime), 'course_id' (int(11)), and 'sender_id' (int(11)).

e-learning comments	
id	int(11)
text	longtext
date_created	datetime
course_id	int(11)
sender_id	int(11)

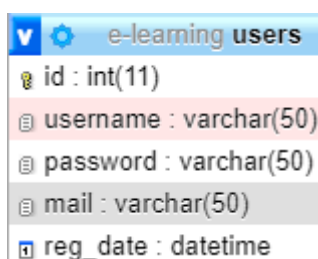
Рис 2.3 Типи даних таблиці comments



The screenshot shows the 'e-learning authors' table structure. It includes a primary key 'id' of type int(11). Other fields are 'rating' (float), 'name' (varchar(50)), and 'user_id' (int(11)).

e-learning authors	
id	int(11)
rating	float
name	varchar(50)
user_id	int(11)

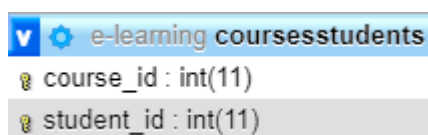
Рис 2.4 Типи даних таблиці authors



The screenshot shows the 'e-learning users' table structure. It includes a primary key 'id' of type int(11). Other fields are 'username' (varchar(50)), 'password' (varchar(50)), 'mail' (varchar(50)), and 'reg_date' (datetime).

e-learning users	
id	int(11)
username	varchar(50)
password	varchar(50)
mail	varchar(50)
reg_date	datetime

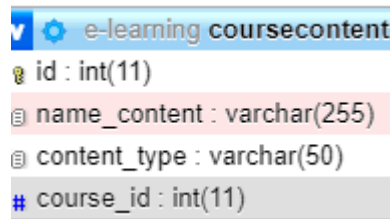
Рис 2.5 Типи даних таблиці users



The screenshot shows the 'e-learning coursesstudents' table structure. It includes two primary keys: 'course_id' and 'student_id', both of type int(11).

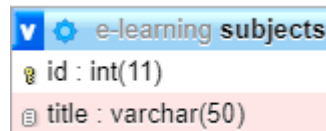
e-learning coursesstudents	
course_id	int(11)
student_id	int(11)

Рис 2.6 Типи даних таблиці coursestudents



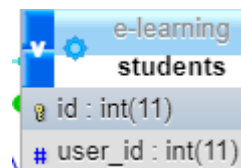
e-learning coursecontent	
id	int(11)
name_content	varchar(255)
content_type	varchar(50)
course_id	int(11)

Рис 2.7 Типи даних таблиці coursecontent



e-learning subjects	
id	int(11)
title	varchar(50)

Рис 2.8 Типи даних таблиці subjects



e-learning students	
id	int(11)
user_id	int(11)

Рис 2.9 Типи даних таблиці subjects

2.4 Реалізація SQL-скрипту

```
SELECT * FROM Courses
```

```
SELECT title, level, rating FROM Courses
```

```
WHERE level LIKE 'Beginner' and rating > 5
```

```
ORDER BY created DESC;
```

```
SELECT Courses.title, Courses.rating, Comments.text FROM Courses,  
Comments
```

```
WHERE Comments.course_id = Courses.id
```

```
ORDER BY Courses.rating DESC;
```



```

SELECT Courses.title as 'Most popularity courses', S.title as 'Subject'
FROM Courses
JOIN Subjects as S on Courses.subject_id = S.id
ORDER BY Courses.rating DESC
LIMIT 10;

```

```

SELECT Courses.* FROM Courses
WHERE Courses.title LIKE 'P%'
ORDER BY Courses.created DESC;

```

```

SELECT Courses.title, A.name, A.rating FROM Courses
JOIN Authors as A on A.id = Courses.owner_id;

```

```

SELECT Courses.title, A.name FROM Courses
JOIN Authors AS A on Courses.owner_id = A.id
JOIN Comments AS C on Courses.id = C.course_id
WHERE Courses.rating BETWEEN 2 and 6;

```

```

SELECT Authors.name, C.title, C.rating FROM Authors
JOIN Courses as C on Authors.id = C.owner_id
SELECT count(*) as 'Count courses', min(Courses.rating),
max(Courses.rating) FROM Courses

```

```

SELECT Users.username, C.text, C.date_created FROM Users
JOIN Comments as C on Users.id = C.sender_id
ORDER BY C.date_created

```

```

SELECT * FROM Comments
WHERE Comments.id % 2 = 0
ORDER BY course_id;

```

```
SELECT * FROM Users
WHERE Users.mail LIKE '%@gmail.com'
ORDER BY Users.reg_date DESC;
```

```
SELECT * FROM Users
WHERE length(Users.password) < 6
ORDER BY length(Users.password);
```

```
SELECT Courses.title, A.name FROM Courses
JOIN Authors as A on A.id = Courses.owner_id
WHERE Courses.level = 'Beginner';
```

```
SELECT CourseContent.name_content, C.title FROM CourseContent
JOIN Courses as C on C.id = CourseContent.course_id
WHERE CourseContent.name_content LIKE '%book%';
```

```
SELECT * FROM CourseContent
WHERE content_type = 'picture' or content_type='file'
```

title	name
Learn to Program: The Fundamentals	Paris-Sorbonne University Abu Dhabi
HTML, CSS for Web Developers	Rochester Institute of Technology, Dubai
PHP for Beginners	Rochester Institute of Technology, Dubai
PHP with Laravel for beginners	Rochester Institute of Technology, Dubai
Modern JavaScript From The Beginning	Rochester Institute of Technology, Dubai
C# Basics for Beginners	Rochester Institute of Technology, Dubai
C Programming For Beginners	The Petroleum Institute
he Complete Java Certification Course	Rak Medical & Health Sciences University
Programming Java for Beginners	Rak Medical & Health Sciences University
Perl Programming for Beginners	University Of Dubai

Рис 2.10 Виконання запиту:

“SELECT Courses.title, A.name FROM Courses JOIN Authors as A on A.id = Courses.owner_id WHERE Courses.level = 'Beginner'”

ВИСНОВКИ

У даній курсовій роботі було проведено теоретичний огляд методів та засобів БД для інформаційної системи курсів програмування, а також проведено аналіз необхідного для збереження обсягу даних, з метою розробки оптимальної структури БД для системи, що розробляється. Було виділено основні інформаційні сутності, що сформували основу для абстрактної моделі даних, яка допомогла у визначенні подальших дій з розробки безпосередньо таблиць БД. При розробці таблиць головною метою було охопити увесь обсяг потрібних для збереження даних та запобігти логічним суперечкам, чи надлишковому повторенню інформації.

Також було проведено дослідження наявних моделей баз даних, та обрано найбільш зручну для реалізації поставленого завдання – реляційну модель. Спираючись на це, було проведено аналіз доступних для використання в розробці систем управління базами даних. Обрана СУБД MySQL задовольняє своїм функціоналом, допоміжним програмним забезпеченням, а також доступністю довідкових матеріалів потреби, що виникають при вирішенні даної задачі. Також дана СУБД є безкоштовною, тобто не накладає фінансового навантаження

Отже, у даній роботі було розроблено повноцінну Базу даних для інформаційної системи курсів програмування, яка являє собою ґрунтовну основу для подальшого розширення функціональності та оптимізації здобутих результатів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Database [Електронний ресурс] // Wikipedia. – 2017. – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/Database>.
2. Database [Електронний ресурс] // Oracle FAQ. – 2008. – Режим доступу до ресурсу: <http://www.orafaq.com/wiki/Database>.
3. PostgreSQL Wiki [Електронний ресурс] // postgresql.org – Режим доступу до ресурсу: https://wiki.postgresql.org/wiki/Main_Page.
4. Introduction to the Oracle Database [Електронний ресурс] // Oracle. – 2017. – Режим доступу до ресурсу: https://docs.oracle.com/cd/B19306_01/server.102/b14220/intro.htm.
5. Microsoft SQL Server [Електронний ресурс] // Wikipedia. – 2017. – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Microsoft_SQL_Server.
6. MySQL [Електронний ресурс] // Wikipedia. – 2017. – Режим доступу до ресурсу: <https://ru.wikipedia.org/wiki/MySQL>.
7. DB-Engines Ranking [Електронний ресурс] // DB-engines. – 2017. – Режим доступу до ресурсу: <https://db-engines.com/en/ranking>

ДОДАТКИ

Додаток (А)

Реалізація SQL-скрипту

```
USE `e-learning`;
```

```
CREATE TABLE Comments (  
    id INT PRIMARY KEY AUTO_INCREMENT,  
    text LONGTEXT NOT NULL,  
    date_created DATETIME DEFAULT CURRENT_TIMESTAMP,  
    course_id INT NOT NULL,  
    sender_id INT  
);
```

```
CREATE TABLE Courses (  
    id INT PRIMARY KEY AUTO_INCREMENT,  
    title VARCHAR(50) NOT NULL,  
    description LONGTEXT NOT NULL,  
    created DATETIME DEFAULT CURRENT_TIMESTAMP,  
    rating FLOAT,  
    level VARCHAR(50),  
    owner_id INT NOT NULL,  
    subject_id INT NOT NULL  
);
```

```
CREATE TABLE Students (  
    id INT PRIMARY KEY AUTO_INCREMENT,  
    user_id INT  
);
```

```
CREATE TABLE CourseContent (  
    id INT PRIMARY KEY AUTO_INCREMENT,  
    name_content VARCHAR(255) NOT NULL,  
    content_type VARCHAR(50) NOT NULL,  
  
    course_id INT NOT NULL  
);
```

```
CREATE TABLE Subjects (  
    id INT PRIMARY KEY AUTO_INCREMENT,  
    title VARCHAR(50) NOT NULL  
);
```

```
CREATE TABLE Authors (  
    id INT PRIMARY KEY AUTO_INCREMENT,  
    rating FLOAT NOT NULL,  
    name VARCHAR(50),  
    user_id INT  
);
```

```
CREATE TABLE Users (  
    id INT PRIMARY KEY AUTO_INCREMENT,  
    username VARCHAR(50) NOT NULL,  
    password VARCHAR(50) NOT NULL,  
    mail VARCHAR(50) NOT NULL,  
    reg_date DATETIME DEFAULT CURRENT_TIMESTAMP  
);
```

```
CREATE TABLE CoursesStudents (  
    course_id INT,
```

```
student_id INT,  
primary key (course_id, student_id),  
FOREIGN KEY (course_id) REFERENCES Courses(id),  
FOREIGN KEY (student_id) REFERENCES Students(id)  
);
```

```
ALTER TABLE Comments  
ADD FOREIGN KEY (course_id) REFERENCES Courses(id),  
ADD FOREIGN KEY (sender_id) REFERENCES Users(id);
```

```
ALTER TABLE Courses  
ADD FOREIGN KEY (owner_id) REFERENCES Authors(id),  
ADD FOREIGN KEY (subject_id) REFERENCES Subjects(id);
```

```
ALTER TABLE CourseContent  
ADD FOREIGN KEY (course_id) REFERENCES Courses(id);
```

```
ALTER TABLE Students  
ADD FOREIGN KEY (user_id) REFERENCES Users(id);
```

```
ALTER TABLE Authors  
ADD FOREIGN KEY (user_id) REFERENCES Users(id);
```

```
INSERT INTO Subjects(title) VALUES  
('Python'),  
('Java'),  
('PHP'),  
('Javascript'),  
('C#'),  
('C++'),
```

```
('C'),
('Assembler'),
('Perl'),
('HTML5/CSS3');
```

```
INSERT INTO Users(username, password, mail) VALUES
```

```
('Andrew', 'qwerty', 'andrii@gmail.com'),
('Kate', 'bestkate123', 'kate123@gmail.com'),
('Bill', 'bill6467', 'bill355@gmail.com'),
('Wade', 'wade351A', 'darkwade@gmail.com'),
('Dave', 'daviebest', 'daview@gmail.com'),
('Seth', 'sethilo', 'sethlol@gmail.com'),
('Ivan', 'ivanov2', 'ivanivanov@gmail.com'),
('Robert', 'Robo567', 'robin@gmail.com'),
('William', 'willie1', 'william@gmail.com'),
('Alego', 'legofriend', 'olegov@gmail.com'),
('And54w', 'thesimple6', 'anderwrii@gmail.com'),
('Katebelle', 'bestkate123', 'kate123belle@gmail.com'),
('Billyn', 'bill6467', 'biynll355@gmail.com'),
('Wandy', 'wade351A', 'darkwanade@gmail.com'),
('Davie', 'daviebest', 'daviewoll@gmail.com'),
('Sethy', 'sethilo', 'seth@gmail.com'),
('Ivanov', 'ivan2', 'ivaniva@gmail.com'),
('Robertos', 'Riba2', 'robin_good@gmail.com'),
('William_James', 'willie_james34', 'william_james@gmail.com'),
('Alegomonth', 'legofriend', 'olegovoy@gmail.com');
```

```
INSERT INTO Authors(rating, name, user_id) VALUES
```

```
(5.6, 'Paris-Sorbonne University Abu Dhabi', 1),
(2.0, 'Rak Medical & Health Sciences University', 2),
```



```
(9.9, 'Rochester Institute of Technology, Dubai', 3),
(9.8, 'Skyline University College, Sharjah', 4),
(7.7, 'Synergy University, Dubai Campus', 5),
(3.7, 'The Emirates Academy of Hotel Managment', 6),
(6.0, 'The Petroleum Institute', 7),
(8.1, 'United Arab Emirates University', 8),
(1.0, 'University Of Dubai', 9),
(10.0, 'University of Jazeera', 10);
```

```
INSERT INTO Students(user_id) VALUES
```

```
(11),
(12),
(13),
(14),
(15),
(16),
(17),
(18),
(19),
(20);
```

```
INSERT INTO Courses(title, description, rating, owner_id, subject_id,
level) VALUES
```

```
('Learn to Program: The Fundamentals', 'Behind every mouse click and
touch-screen tap, there is a computer program that makes things happen. This course
introduces the fundamental building blocks of programming and teaches you how to
write fun and useful programs using the Python language.', 10.0, 1, 1, 'Beginner'),
```

```
('Python Data Structures', 'This course will introduce the core data
structures of the Python programming language. We will move past the basics of
procedural programming and explore how we can use the Python built-in data
```

structures such as lists, dictionaries, and tuples to perform increasingly complex data analysis. This course will cover Chapters 6-10 of the textbook “Python for Everybody”. This course covers Python 3., 8.5, 2, 1, 'Advanced'),

('HTML, CSS for Web Developers', 'The web today is almost unrecognizable from the early days of white pages with lists of blue links. Now, sites are designed with complex layouts, unique fonts, and customized color schemes. This course will show you the basics of Cascading Style Sheets (CSS3). The emphasis will be on learning how to write CSS rules, how to test code, and how to establish good programming habits.', 6.4, 3, 10, 'Beginner'),

('C++ For C Programmers', 'This course is for experienced C programmers who want to program in C++. The examples and exercises require a basic understanding of algorithms and object-oriented software.', 2.5, 3, 6, 'Intermediate'),

('Object Oriented Programming in Java', 'Welcome to our course on Object Oriented Programming in Java using data visualization. People come to this course with many different goals -- and we are really excited to work with all of you! Some of you want to be professional software developers, others want to improve your programming skills to implement that cool personal project that you’ve been thinking about, while others of you might not yet know why you’re here and are trying to figure out what this course is all about.', 3.4, 2, 2, 'Intermediate'),

('PHP for Beginners', ' Then this course will help you get all the fundamentals of Procedural PHP, Object Oriented PHP, MySQLi, and ending the course by building a CMS system similar to WordPress, Joomla, or Drupal.', 7.5, 3, 3, 'Beginner'),

('PHP with Laravel for beginners', 'Laravel has become one of the most popular if not the most popular PHP framework. Employers are asking for this skill for all web programming jobs and in this course we have put together all of them, to give you the best chance of landing that job; or taking it to the next level.', 3.4, 3, 10, 'Beginner'),

('The Modern JavaScript Bootcamp', 'JavaScript is the most popular programming language out there, but that doesn't mean it's easy to learn. You end up wasting time on out-of-date courses and incomplete YouTube tutorials that talk about a JavaScript features without showing how to use them when building real-world applications.', 3.4, 3, 4, 'Intermediate'),

('Modern JavaScript From The Beginning', 'This is a front to back JavaScript course for absolutely everybody. We start with the basic fundamentals and work our way to advanced programming WITHOUT relying on frameworks or libraries at all. You will learn a ton of pure JavaScript, whether you are a beginner or an established JS programmer. There is something for everyone...This is a front to back JavaScript course for absolutely everybody. We start with the basic fundamentals and work our way to advanced programming WITHOUT relying on frameworks or libraries at all. You will learn a ton of pure JavaScript, whether you are a beginner or an established JS programmer. There is something for everyone...', 3.4, 3, 4, 'Beginner'),

('C# Intermediate', 'Whether you want to use C# to build web apps, mobile apps, desktop apps or games, understanding C# classes, interfaces and principles of object-oriented programming is crucial.', 6.4, 3, 5, 'Intermediate'),

('C# Basics for Beginners', 'C# is a beautiful cross-platform language that can be used to build variety of applications. With C#, you can build mobile apps (for Windows, Android and iOS), games, web sites and desktop applications.', 2.4, 3, 5, 'Beginner'),

('C# Advanced', 'Chances are you're familiar with the basics of C# and are hungry to learn more. Or you've been out of touch with C# for a while and are looking for a quick course as a refresher to get you up to speed with advanced C# constructs. If so, then this course is for you.', 3.4, 3, 5, 'Advanced'),

('Advanced C Programming', 'The C programming language in 2020 is still one of the most popular and widely used languages. Having C programming skills gives you great career options, but learning the C language, particularly some of the trickier advanced stuff can be really difficult.', 5.4, 3, 7, 'Advanced'),

('C Programming For Beginners', 'The fastest, easiest way to learn to program C on a Mac or Windows. This course will teach you to program the C language from the ground up. You will learn everything from the very fundamentals of programming right through to the complexities of pointers, addresses and File IO. Maybe you've tried to master C before but failed. Or maybe you are new to C or new to programming. If so, this is the course for you', 3.4, 7, 7, 'Beginner'),

('he Complete Java Certification Course', 'Welcome to Master Practical Java Development. This course is designed to help you master the most in-demand and critical components for becoming a Core Java developer. Especially if you're going for a job interview or have a Java Project that needs your best performance. This course assumes no prior java experience so prior Java so it will take you from zero to hero!', 4.4, 2, 2, 'Beginner'),

('Programming Java for Beginners', 'Learn the basic concepts, tools, and functions that you will need to build fully functional programs with the popular programming language, Java.', 3.4, 2, 2, 'Beginner'),

('Perl Programming for Beginners', 'The world of programming has become almost saturated with different languages, all created for different purposes but developed for use in multiple applications. For those just delving into the world of programming, this can be a little overwhelming. Luckily, there are plenty of languages that are simple to learn, highly versatile to use, and make a great starting point for gaining fluency in the coding universe. Perl is one of those languages, and this course will teach you everything you need to know.', 3.4, 9, 9, 'Beginner');

```
INSERT INTO CourseContent(name_content, content_type, course_id)
VALUES
```

```
    ('PythonBook', 'file', 1),
    ('Java ', 'picture', 2),
    ('PHP begin', 'video',3),
    ('Javascript book', 'file',4),
    ('C#book','file', 5),
```

```

('C++begin', 'picture',6),
('C begin', 'file',7),
('Assemblerbook','file', 8 ),
('Perl','video', 9),
('HTML5/CSS3book' , 'file',10),
('PythonBook3', 'file', 1),
('Java', 'picture',2),
('PHPbook', 'file',3),
('Javascript', 'site',4),
('C#book','file', 5),
('C++book', 'file',6),
('C begin', 'video',7),
('Assembler advance','file', 8 ),
('Perlbook','file', 9),
('HTML5/CSS3 advance' , 'picture',10),
('PythonBook', 'file', 1),
('Java begin', 'file',2),
('PHP begin', 'file',3),
('Javascript', 'scheme',4),
('C# begin','file', 5),
('C++', 'file',6),
('C begin', 'scheme',7),
('Assembler','file', 8 ),
('Perl advance','file', 9),
('HTML5/CSS3' , 'scheme',10);

```

```

INSERT INTO Comments(text, course_id, sender_id) VALUES

```

```

('The course was absolutely worth the money.',2,11),
('This is the best course I've ever taken on computer programming',2,12),
('I was really blown away by how fun this course was.',3,13),

```

('This course was amazing. Your detailed explanations, thoughtful exercises and challenge problems were all wonderful. Thank you!',4,10),

('It helps beginner for sure. It would have been better if we had a project which is done.',5,11),

('I found the material to be presented in a clear, and understandable, format. The instructor provided numerous examples to clarify the concepts presented and the quizzes and their solutions were also extremely helpful in illustrating various approaches that could be taken to solve the problems.',6,12),

('The course is charted in a brilliant way to enable the understanding of people from non-computers background. Also, the exercises given helped to understand the theory and put well to use.',1,17),

('Excellent course. I indeed went from zero to hero. It took me a lot to go through all the videos, notes and exercises, but it was completely worthwhile.',1,18),

('This course touches many areas, but lack in depth.',6,13),

('This is a very basic course. Most of the topics are discussed at a surface level.',9,15),

('This course doesn't cover much material and there are too few exercises. Most of what it teaches is just manipulating strings or objects in an elementary way. Hard to see applying this elementary knowledge to most real world situations.',8,14),

('It is painful to watch someone try and explain something that they do not fully understand.',6,19),

('he instructor covers many essential topics and provides in-depth examples and alternatives. I feel I have gained a sound foundation on which to build future applications.',5,17),

('This is an honest rating. Got what i paid and singed up for',10,20),

('I really enjoy the energy that is put into this course - I have learned a lot in just a few days, thank you',9,15),

('Yes, he is very thorough in his explanations. The video is of high quality and easy to see what he is typing etc. Easy to follow along very good so far.',8,14),

('Great course! teacher style is straight forward, touches on all the critical points and takes us through the dev process - hardly any editing which brings us in to show the iterative process that dev actually is with mistakes and all. ',6,19),

('Great Course!',5,17),

('Absolutely amazing course! and best teacher ever!',10,20);

-- Запросить все курсы

SELECT * FROM Courses;

-- Запросить курсы, где уровень "Начальный", а рейтинг больше 5, и отсортировать по убыванию даты создания

SELECT title, level, rating FROM Courses

WHERE level LIKE 'Beginner' and rating > 5

ORDER BY created DESC;

-- Запросить курсы, их рейтинг и текст комментариев с помощью

-- неявного соединения таблиц.

SELECT Courses.title, Courses.rating, Comments.text FROM Courses,
Comments

WHERE Comments.course_id = Courses.id

ORDER BY Courses.rating DESC;

-- Запросить курсы и предмет, к которому они относятся, используя внутренний Join

SELECT Courses.title as 'Most popularity courses', S.title as 'Subject'
FROM Courses

JOIN Subjects as S on Courses.subject_id = S.id

ORDER BY Courses.rating DESC

LIMIT 10;

```
SELECT Courses.* FROM Courses
WHERE Courses.title LIKE 'P%'
ORDER BY Courses.created DESC;
```

```
SELECT Courses.title, A.name, A.rating FROM Courses
JOIN Authors as A on A.id = Courses.owner_id;
```

```
SELECT Courses.title, A.name FROM Courses
JOIN Authors AS A on Courses.owner_id = A.id
JOIN Comments AS C on Courses.id = C.course_id
WHERE Courses.rating BETWEEN 2 and 6;
```

```
SELECT Authors.name, C.title, C.rating FROM Authors
JOIN Courses as C on Authors.id = C.owner_id
ORDER BY C.rating ASC;
```

```
SELECT count(*) as 'Count courses', min(Courses.rating),
max(Courses.rating) FROM Courses
```

```
SELECT Users.username, C.text, C.date_created FROM Users
JOIN Comments as C on Users.id = C.sender_id
ORDER BY C.date_created
```

```
SELECT * FROM Comments
WHERE Comments.id % 2 = 0
ORDER BY course_id;
```

```
SELECT * FROM Users
WHERE Users.mail LIKE '%@gmail.com'
ORDER BY Users.reg_date DESC;
```



```
SELECT * FROM Users
WHERE length(Users.password) < 6
ORDER BY length(Users.password);
```

```
SELECT Courses.title, A.name FROM Courses
JOIN Authors as A on A.id = Courses.owner_id
WHERE Courses.level = 'Beginner';
```

```
SELECT CourseContent.name_content, C.title FROM CourseContent
JOIN Courses as C on C.id = CourseContent.course_id
WHERE CourseContent.name_content LIKE '%book%';
```

```
SELECT * FROM CourseContent
WHERE content_type = 'picture' or content_type='file'
```