Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Навчально-науковий інститут комп'ютерних систем
Кафедра інженерії програмного забезпечення

Бліжніков Максим Олегович
студент групи АС-172

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Програма для зміни змісту таблиць бази даних для реалізації денормалізації в умовах багаторівневої ієрархії

Спеціальність:
121 – Інженерія програмного забезпечення
Освітня програма:
Інженерія програмного забезпечення

Керівник:
Зіноватна Світлана Леонідівна,
кандидат технічних наук, доцент

Одеса – 2022

ЗМІСТ

Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Навчально-науковий інститут комп'ютерних систем
Кафедра інженерії програмного забезпечення

Рівень вищої освіти: другий (магістерський)
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____доц. Комлева Н.О.
"_____" _____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Бліжнікову Максиму Олеговичу, АС-172
(прізвище, ім'я та по батькові)

1. Тема проекту (роботи) Програма для зміни змісту таблиць бази даних для реалізації денормалізації в умовах багаторівневої ієрархії
Керівник проекту Зіноватна Світлана Леонідівна, канд. техн. наук, доцент
затверджені наказом ректора від "20"жовтня 2022 р.№399-в
2. Строк подання студентом проекту (роботи) 30.11.2022
3. Вихідні дані по проекту (роботі) Технічне завдання
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належіть розробити)
Вступ. Аналіз предметної області. Модель перенесення даних в денормалізовану таблицю. Вимоги до програми формування змісту денормалізованих таблиць. Проектування програми для зміни змісту таблиць бази даних для реалізації денормалізації. Реалізація програми. Тестування програми Висновки. Список літератури
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Мета роботи. Порівняльні характеристики систем з денормалізованими структурами. Порівняння підходів до завантаження даних. Нисхідна та висхідна денормалізація. Введені поняття. Теоретико-множинне представлення даних. Загальні схеми для додавання надлишкового атрибуту та запиту для зміни даних. Діаграма станів. Діаграма активності для визначення загального алгоритму роботи програми. Діаграма варіантів використання. Діаграма послідовності для процесу створення сесії. Концептуальна модель даних. Схема алгоритму створення запиту на зміну змісту денормалізованої БД. Специфікації класів. Інтерфейс програми.Результати експерименту. Висновки

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Результат прийняв

7. Дата видачі завдання 28.08.2022

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1.	Аналіз програмних засобів для денормалізації структури бази даних	10.09.2022	виконано
2.	Визначення функціональних та нефункціональних вимог	20.09.2022	виконано
3.	Розробка та реалізація структури сервісної бази даних	10.10.2022	виконано
4.	Розробка алгоритмів функцій програми	15.10.2022	виконано
5.	Визначення специфікацій класів	20.10.2022	
6.	Програмна реалізація функціоналу програми	15.11.2022	
7.	Тестування програми	20.11.2022	виконано
8.	Оформлення пояснювальної записки	25.11.2022	виконано

Здобувач вищої освіти _____ *М. О. Бліжніков*

Керівник роботи _____ *С. Л. Зіноватна*

РЕФЕРАТ

Метою роботи є зменшення часу на зміну змісту таблиць БД, до яких застосовується багаторівнева нисхідна або висхідна денормалізація за рахунок автоматизації побудови запитів для оновлення даних.

Розглянуто варіанти прискорення продуктивності інформаційної системи за рахунок зберігання надлишкових даних. Розроблено алгоритми зміни структури та змісту бази даних у разі реалізації нисхідної та висхідної денормалізації. Визначено функціональні та нефункціональні вимоги до програми. Розроблено структуру базі даних для зберігання сервісної інформації. Спроектовані класи для виконання функцій програми. Виконана програмна реалізація відповідно заданих вимог.

Ключові слова: структура бази даних, денормалізація, сервісна база даних.

ABSTRACT

The purpose of the work is to reduce the time it takes to change the content of database tables to which multi-level downward or upward denormalization is applied by automating the construction of queries to update data.

Variants of accelerating the productivity of the information system due to the storage of redundant data are considered. Algorithms for changing the structure and content of the database in case of implementation of downward and upward denormalization have been developed. Functional and non-functional requirements for the program are defined. A database structure for storing service information has been developed. Designed classes to perform program functions. The software implementation was completed in accordance with the specified requirements.

Keywords: database structure, denormalization, service database.

ВСТУП

В теперішній час майже ніяка організація не може обійтися без автоматизації обліку своєї діяльності. Для цього використовуються інформаційні системи, в основі яких лежить база даних, найчастіше реляційна. Якщо інформаційна система (ІС) працює тривалий час, звичайно, зростає обсяг таблиць БД, що може приводити до суттєвого зниження продуктивності виконання запитів користувача. Одним з варіантів підвищення продуктивності є денормалізація [1]. Також денормалізація широко використовується при проєктуванні структури сховищ даних [2].

Метою роботи є зниження часу на зміну змісту таблиць БД, до яких застосовується багаторівнева нисхідна або висхідна денормалізація за рахунок автоматизації побудови запитів для оновлення даних.

Для досягнення мети потрібно вирішити наступні задачі під час виконання роботи:

- визначити, які є проблеми при зміні структури бази даних в результаті використання денормалізації;
- проаналізувати існуючі інструменти автоматизованої зміни вмісту таблиць бази даних;
- проаналізувати ризики, які виникають при використанні денормалізації структури бази даних;
- виконати формалізований опис процедури зміни структури бази даних;
- створити базу даних для зберігання інформації, необхідної для виконання зміни змісту таблиць;
- розробити програмний інтерфейс для зручного виконання функцій програми.

Об'єктом дослідження є структура бази даних.

Предметом дослідження є засоби автоматизованої зміни даних в таблицях реструктиризованої бази даних.

Методи дослідження. Модель денормалізації структури бази даних базуються на принципах об’єктно-орієнтованого проектування, теорії множин.

Наукова новизна полягає в наступному: формалізовано представлення денормалізованої структури бази даних для багаторівневої ієрархії зв’язків між таблицями, що дає можливість автоматизувати заповнення доданих полів з використанням існуючої інформації.

У першому розділі розглянуті способи підвищення продуктивності баз даних. Проведено аналіз існуючих систем з використанням денормалізованих структур, зроблено висновок про відсутність в них універсальної складової.

В другому розділі обґрунтовано вибір варіантів денормалізації для автоматизації. Формалізовано загальне представлення обраних варіантів з точки зору зміни структури БД та формування запитів для заповнення доданих надлишкових атрибутів.

В третьому розділі наведено вимоги до програми формування змісту денормалізованих таблиць. Описані функціональні вимоги, які описують поводження системи в певних умовах, у вигляді варіантів використання. Для кожного варіанту використання наданий сценарій. Надані нефункціональні умови, які описують, як має працювати система. В розділі також описані шляхи вирішення можливих міжсесійних конфліктів які можуть привести до помилок під час виконання функцій в рамках однієї сесії.

В четвертому розділі описано, яким чином спроєктовано архітектуру програми, показано фрагменти модулів. за допомогою яких реалізуються окремі частини функціоналу застосунку; наданий детальний опис структури сервісної БД, яка зберігає дані, необхідні для автоматизованого виконання денормалізації, а саме, дані для підключення до БД, послідовність таблиць у ієрархії взаємозв’язків для транзитивного перенесення атрибуту з надлишковими даними. Наявність сервісної БД дозволяє рознести в часі виконання різних етапів підготовки денормалізаціїї. Надана схема алгоритму

для автоматизованого формування тексту запиту, за допомогою якого можливо внести дані в доданий атрибут з надлишковими даними.

В п’ятому розділі описана структура класів, за допомогою методів яких реалізується створення компонентів вхідних даних для реструктуризації БД та безпосередньо проведення денормалізації. Детально описаний зовнішній вигляд вікон інтерфейсу програми, за допомогою яких виконується перелічені функції.

В шостому розділі наведені чек-аркуші для проведення тестування програми зміни змісту денормалізованих таблиць. Представлені чек-аркуші дозволяють в повному обсязі перевірити правильну роботу всіх складових програми. Наведені дані експерименту, який показав значне скорочення часу на проведення денормалізації, з урахуванням наявності даних в БД і необхідності формувати запити, які вимагають залучення кількох пов’язаних таблиць.

Впровадження розробленого програмного забезпечення дозволить заповнювати нові поля у структурі таблиць бази даних, відповідних виконаній денормалізації, на основі заданої ієрархії відношень. Науково-практична цінність полягає в тому, що користувач може швидко отримувати актуальний вміст бази даних з коректними даними, відповідними виконаній реструктуризації.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Значення інформаційних систем для організацій

До поширення персональних комп'ютерів і Інтернету, компанії використали інформаційні системи для керування обробкою даних і веденням обліку, пов'язаними з бізнесами-транзакціями. Сьогодні основним завданням компаній є збереження конкурентоспроможності на глобальному рівні за рахунок використання можливостей сучасних інформаційних і комунікаційних технологій. Організації можуть використати такі технології для надання продуктів і послуг високої якості за доступними цінами й з першокласним обслуговуванням клієнтів. Електронна комерція може допомогти організаціям вийти на нові ринки. Тому, щоб бути конкурентоспроможними, компанії інвестують у сучасні інформаційні системи [3].

В [4] названо основні переваги повністю інтегрованої інформаційної системи:

– завдяки інтеграції інформації в одну систему всі дані актуальні, що важливо для всіх елементів організації;

– члени робочих груп зможуть краще спілкуватися, використовуючи ту саму інформацію, що усуває необхідність зіставляти дані між підрозділами й системами організації;

– дані не потрібно копіювати, таким чином, знижується ймовірність помилок, отримана інформація є більше точною;

– у співробітників стає більше часу на вирішення важливих завдань для росту бізнесу;

– оскільки всі необхідні дані зберігаються в одній інформаційній системі, співробітники можуть легко одержати до них доступ, при цьому конфіденційну інформацію можуть побачити тільки тих, кому дозволений доступ.

В [5] описано категорії можливостей інформаційних систем:

– доступ до інформації: співробітники повинні мати простий і швидкий доступ до інформації, щоб приймати обґрунтовані рішення;

– збір даних: інформаційні системи збирають і зіставляють дані як зовні, так і усередині організації, дані поєднуються й розміщаються в сховищах даних, які потім поєднуються в мережу для цілей аналітики;

– спільна робота: різні відділи й розподілені команди можуть спільно приймати рішення, не дивлячись на більші обсяги даних з різних джерел;

– інтерпретація: після ухвалення рішення інформаційні системи можуть допомогти менеджерам зрозуміти потенційні наслідки цих рішень, прогнозуючи майбутні вигоди або проблеми;

– презентація: більшість інформаційних систем містять у собі інструменти, призначені для створення простих для розуміння звітів для перегляду співробітниками більше високого рівня або керівниками вищої ланки.

1.2 Засоби підвищення продуктивності баз даних

В [6] підкреслений ступінь впливу продуктивності баз даних на бізнес в цілому: «бізнес може бути настільки успішним, наскільки дозволяють йому ваші IT-операції». Тобто добре функціонуюча база даних може мати великий вплив на прибутковість компанії. Оскільки, у випадках, коли витяг даних уповільнюється через те, що є невдало написаний запит або проблеми з індексацією, може виникнути вузьке місце, яке знижує продуктивність всієї організації. Оскільки клієнтські веб-сайти й застосунки витягають дані з централізованої бази даних, неефективні індекси й неоптимальні запити можуть мати такий же великий вплив на клієнтів, як і на внутрішніх кінцевих користувачів. У результаті задоволеність клієнтів прямо пов'язана із продуктивністю бази даних. Таким чином, знання того, як підвищити

продуктивність бази даних, може стати одним з найважливіших інструментів обслуговування клієнтів у вашому наборі інструментів.

В [7] вказується, що настроювання продуктивності бази даних дозволяє розроблювачам або адміністраторам баз даних максимально використати системні ресурси для постійного підвищення продуктивності. Також в [6] перелічені шляхи такого настроювання:

– перевірка серверу баз даних – потрібно перевірити процесор, пам'ять й дисковий простір, щоб визначити потенційні проблеми;

– необхідність оцінки різних аспектів продуктивності центрального процесору (ЦП), наприклад, часу готовності ЦП; через постійне базове навантаження серверів баз даних може бути необхідним збільшити кількість ядер ЦП, щоб підтримувати швидкість відгуку сервера; перехід на потужніший ЦП може знизити навантаження, створюване декількома застосунками й запитами, підвищуючи швидкість і ефективність бази даних;

– оцінка потреб у пам'яті: містить у собі оцінку використання пам'яті й відмов сторінок у секунду; кількість відмов сторінок свідчить про те, що хостам не вистачає доступного місця в пам'яті, і необхідно збільшити його; нестача доступної пам'яті є потенційною причиною проблем із продуктивністю бази даних;

– наявність достатньо великого обсягу доступного сховища для сервера бази даних: має вирішальне значення, оскільки індекси й інші поліпшення продуктивності змушують бази даних споживати більше дискового простору; виділення набору дисків для файлів даних, файлів журналів, файлів резервних копій і бази даних tempdb підвищує продуктивність, а також служить резервною копією у випадку аварійного відновлення;

– відстеження показників, пов'язаних із затримкою диска, для зменшення проблеми із затримкою слід використовувати механізми кешування; перехід на твердотілі диски може підвищити продуктивність SQL-запитів; слід обирати модель SSD, призначену для використання баз даних;

– настроювання індексів: може організувати структури даних таким чином, щоб сприяти більше ефективному витягу даних і скоротити час відгуку; індексування може оптимізувати виконання запитів;

– перенастроювання пулу з'єднань, якщо одержання з'єднання поглинає більшу частину часу відгуку бази даних;

– оптимізація запитів: використання підзапиту може зробити кодування більше зручним, але це може знизити продуктивність бази даних; цикли кодування можуть привести до великої кількості непотрібних запитів, які можуть зменшити навантаження на вашу базу даних; слід уникати додавання курсорів, які використаються для зациклення на серверах SQL, замість операторів SQL.

В [6] також описані поради, як підвищити продуктивність баз даних:

– оптимізація запитів: при виконанні оптимізації вручну можуть виникнути дилеми при виборі того, як краще підвищити ефективність запитів, наприклад, варто писати з'єднання або підзапит, використати EXISTS або IN і інше; можна використати інструмент для аналізу продуктивності бази даних, який може направити до найбільш ефективних запитів, пропонуючи експертні поради про те, який варіант кращий;

– поліпшення індексів: якщо індекси зроблені правильно, індексування може підвищити продуктивність бази даних і допомогти оптимізувати тривалість виконання запиту; індексація створює структуру даних, яка спрощує пошук інформації; оскільки дані легше знаходити, індексування підвищує ефективність витягу даних і прискорює весь процес, заощаджуючи час для системи;

– дефрагментація даних: коли багато даних постійно записується й видаляється з бази даних, дані можуть стати фрагментованими, що може сповільнити процес витягу даних, оскільки заважає запиту швидко знаходити потрібну інформацію; після дефрагментації даних відповідні дані можуть бути

згруповані й видаляються проблеми зі сторінкою індексу, тобто операції вводу-виведення будуть виконуватися швидше.

– збільшення обсягу пам'яті: для з'ясування, чи потрібно більше пам'яті, необхідно перевірити, скільки помилок сторінок є в системі; коли кількість збоїв велика, це означає, що хостам або не вистачає пам'яті, або повністю закінчилася доступна пам'ять; збільшення обсягу пам'яті допоможе підвищити ефективність і загальну продуктивність;

посилення процесора: чим потужніше процесор, тим менше в нього навантаження при роботі з декількома запитами й застосунками; необхідно відслідковувати всі елементи продуктивності ЦП , включаючи час готовності ЦП, що говорить про те, скільки разів система намагалася використати ЦП, але не могла, тому що ресурси були зайняті іншими завданнями;

перевірка доступу: необхідно перевірити, які застосунки звертаються до бази даних; якщо проблема виникла тільки з однією службою, необхідно вивчити її показники, щоб знайти основну причину.

Можна побачити, що переліки засобів підвищення продуктивності є майже однаковими, тобто список є вичерпним.

1.3 Денормалізація структури бази даних

Частина наведених вище порад з підвищення продуктивності БД відносяться до апаратурної частини ІС або стосуються налаштування системи. Безпосередньо роботи з БД стосуються пропозиції оптимізувати запити та індексувати таблиці БД.

Однак існує ще один засіб підвищення продуктивності, який стосується саме структури БД. Це денормалізація.

Нормальна форма зберігання даних припускає виключення дублювання даних [8]. При цьому повинні виконуватися два ключові правила:

– атомарність: всі сутності зберігаються в неподільному виді;

– унікальність: кожна сутність була визначена тільки один раз.

Однак з погляду продуктивності нормалізація викликає складнощі, тому що для вибору даних з різних таблиць необхідно зробити кілька запитів замість одного або використати операції з’єднання JOIN, які негативно впливають на продуктивність застосунка. Поступовий процес рятування від правил нормалізації там, де це необхідно, називається денормалізацією. Зазвичай це випадки, у яких є часті повторні запити до логічно зв'язаних даних.

Існує два основних підходи при денормалізації даних [8]:

– дублювання;

– попередня підготовка.

Відповідно [9] «денормалізація - навмисне приведення структури бази даних у стан, що не відповідає критеріям нормалізації, зазвичай проведене з метою прискорення операцій читання з бази за рахунок додавання надлишкових даних».

БД повинна була нормалізована, тобто відповідати вимогам нормальних форм, для усунення аномалій даних. При цьому мінімізується надмірність даних у базі даних і забезпечується відсутність логічних помилок відновлення й вибірки даних.

Однак при запитах великої кількості даних операція з'єднання нормалізованих відношень виконується неприйнятно довго. У ситуаціях, коли продуктивність таких запитів неможливо підвищити іншими засобами, може проводитися денормалізація, наприклад, композиція декількох таблиць в одну, яка фактично є збереженим результатом операції з'єднання вхідних відношень.

За рахунок такого перепроектування операція з'єднання при вибірці стає непотрібною, й запити вибірки, які раніше вимагали з'єднання, працюють швидше.

Денормалізація завжди виконується за рахунок підвищення ризику порушення цілісності даних при операціях модифікації. Тому денормалізацію варто проводити в крайньому випадку, якщо інші міри підвищення продуктивності неможливі.

Крім того можливо, що прискорення одних запитів на денормалізованій БД може супроводжуватися вповільненням інших запитів, які раніше виконувалися окремо на нормалізованих відносинах.

Практично існує багато випадків використання денормалізованих структур даних (табл.1.1). Але вони або створюють додатково денормалізовані таблиці, не змінюючи структуру базових таблиць [10], або працюють вже с денормалізованими структурами від початку, використовуючи автоматичні засоби (наприклад, тригери) для формування надлишкових даних під час внесення нових записів [11]. Іноді рішення про зміну структури даних приймається вже під час функціювання ІС [12], в такому випадку необхідно заповнити додані атрибути таблиць відповідно обраному варіанту денормалізаціїї, використовуючи вже існуючий зміст таблиць.

Таблиця 1.1- Порівняльні характеристики систем з денормалізованими структурами

Назва системи/характеристика	Зміна даних в базових таблицях	Автоматизоване заповнення денормалізованої таблиці
Nintex Promapp	Ні	Заповнюються періодично
Siebel EIM	Так	Тільки для Insert

1.4 Підходи до завантаження даних

1.4.1 SQL-запити. Передбачається формулювання інтерактивних SQL-запитів до бази даних з використанням INSERT і UPDATE виразів. Необхідне знання мови SQL і схеми бази даних. Тому найчастіше безпосередньо формулюванням запитів займається розроблювач. Вставки повинні бути

зроблені в певному порядку: спочатку записи, які не містять зовнішніх ключів, потім ті, які посилаються на вже вставлені дані. У цьому випадку необхідно відключати перевірки цілісності при виконанні завантаження з і включати їх після виконання вставок. На швидкість завантаження можуть впливати тригери. При виконанні завантаження великого обсягу даних варто зробити цю умову спрацьовування тригера максимально строгою [13].

Недоліки підходу:

- необхідне знання схеми бази даних і мови запитів до БД (SQL);

- легко припуститися помилок в даних і в створенні SQL-запиту, якщо процес створення запиту не автоматизований;

- необхідне збереження запитів для наступного використання;

- зміна схеми бази даних вимагає зміни всіх запитів до неї;

- складна обробка й модифікація даних, які сформульовані у вигляді SQL-запитів;

- низька швидкість через генерацію запитів і їхнє виконання з перевірками цілісності.

Переваги:

- переносимость між СУБД;

- помилки видні після виконання одиночного запиту;

- версії файлу зі збереженими запитами можна порівнювати між собою за допомогою стандартних засобів систем контролю версій, і при необхідності виконувати злиття.

1.4.2 Використання утиліти Import для СКБД Oracle. Можуть бути використані механізми, які роблять завантаження даних з більшою швидкістю, наприклад, завантаження даних у деякому бінарному форматі [14]. При цьому використається заздалегідь створений файл - дамп, що генерується окремою утилітою (Export).

Недоліки підходу:

- зміна файлу дампа неможлива;

- робота із двома БД із однаковою схемою;
- дамп не є переносимим між різними СКБД;
- не можна порівнювати дампи між собою;
- дані з різних файлів не можна об'єднати.

Переваги:

- простота завантаження даних з файлу-дампа;
- дамп має незначний розмір;
- відносно висока швидкість завантаження;
- мала ймовірність помилки при завантаженні даних.

1.4.3. Використання SQL*Loader. Утиліта [15] завантажує дані із зовнішніх текстових файлів у таблиці бази даних Oracle. Можливе завантаження з декількох файлів із заповненням множини таблиць за одну сесію завантаження. Формування зовнішніх ключів ускладнено, тому що завантаження виконується послідовно для кожної таблиці й із цієї причини неможливо одержати проміжне значення послідовності значень, на яку посилається зовнішній ключ.

Має два режими: звичайний і прямій. У першому випадку з даних формуються SQL-запити для завантаження в БД, у другому SQL не використається, безпосередньо форматуються блоки даних.

Недоліки підходу:

- необхідність знання структури БД;
- складність редагування файлів з даними;
- імовірність помилки при уведенні або редагуванні файлів з даними;
- необхідність формування конфігураційного файлу;
- часткова переносимість між СКБД.

Переваги:

- висока швидкість завантаження;
- гнучкий формат файлів з даними;
- багаторазове використання файлів з даними.

Таблиця 1.2 – Порівняння підходів до завантаження даних

Підхід / Можливості	SQL-запити	Утиліта Import	SQL*Loader
Формування запитів	Вручну	Автоматизовано	Вручну й автоматизовано
Простота використання	Вимагає спеціальних знань - знання SQL	Необхідна робота з командним рядком	Необхідна робота з командним рядком
Дії при зміні БД	Переписати всі SQL запити	Заново генерувати дамп файл	Переписати конфігураційний файл
Робота в єдиній БД	Можлива	Вимагає декількох БД	Можлива
Корекція збережених запитів	Автоматична	Не можлива	Можлива
Обмеження СКБД	Відсутнє	Oracle	Oracle
Обробка зовнішніх ключів	Складна	Не можлива	Складна

Аналіз наведених підходів показує, що використання утиліти Import є зовсім негнучким і не дозволяє вносити зміни до завантаження ніяким способом. Використання утиліти SQL*Loader є набагато гнучкішим, але може використовуватися лише в конкретній СКБД.

Таким чином, перспективним є підхід використання SQL-запитів у разі автоматизації формування запитів та обробки зовнішніх ключей, що і є задачею даної роботи.

1.5 Висновки до розділу

В розділі розглянуті засоби підвищення продуктивності ІС, в основі яких лежить БД.

Проведено аналіз існуючих випадків використання денормалізованих структур даних. Але проаналізовані випадки не носять універсального характеру, оскільки або додають надлишкові таблиці без реструктуризації існуючих таблиць, або передбачають введення деноралізації вже на етапі створення БД, а не після отримання висновків, що існуюча структура БД не дає досягти потрібного рівня продуктивності. Виділено два способи денормалізації як найбільш поширені.

Таким чином, метою роботи є зниження часу на зміну змісту таблиць БД, до яких застосовується багаторівнева нисхідна або висхідна денормалізація за рахунок автоматизації побудови запитів для оновлення даних.

2 МОДЕЛЬ ПЕРЕНЕСЕННЯ ДАНИХ В ДЕНОРМАЛІЗОВАНУ ТАБЛИЦЮ

2.1 Аналіз нисхідної та висхідної денормалізації

В [16] перераховані достоїнства й недоліки виконання денормалізації баз даних:

- достоїнства денормалізації
  - більш швидке читання денормалізованих даних;
  - спрощені запити для розроблювачів застосунків;
  - менше обчислень при операціях читання;
- недоліки денормалізації:
  - більш повільні операції запису;
  - додаткова складність бази даних;
  - можлива непогодженість даних;
  - для надлишкових таблиць потрібне додаткова пам’ять.

Для обробки обрані два найбільш поширених варіанти денормалізації: нисхідна та висхідна, з урахуванням можливості перенесення надлишкових даних через кілька рівнів ієрархії зв’язків між таблицями (рис. 2.1).

В [17] додавання надлишкових стовпців: «в основну таблицю додається тільки надлишковий стовпець, що часто використається в об'єднаннях, інша таблиця зберігається як є». Варіант висхідної денормалізації визначений як додавання похідних стовпців.

Висхідна денормалізація припускає перенос атрибута з підлеглої (дочірньої) таблиці в батьківську таблицю, зазвичай у формі підсумкових даних. Висхідна денормалізація - це процес введення надлишкових атрибутів у батьківських таблицях з метою усунення операцій з'єднання з операціями агрегування.

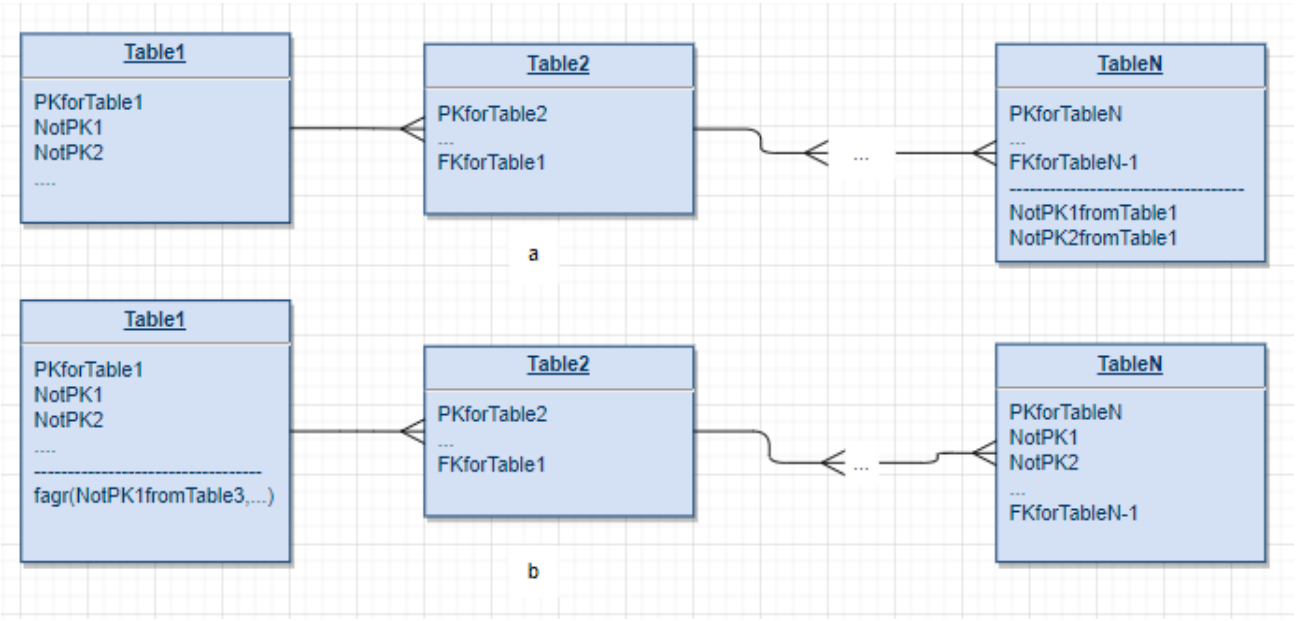


Рисунок 2.1 – Перенесення надлишкового атрибуту по ієрархії зв'язків:

a – у нисхідному напряму;

b – у висхідному напряму

Обидва варіанти передбачають створення додаткового атрибуту в одній з таблиць.

У випадку нисхідної денормалізації передбачається заповнення такого атрибуту даними з основної таблиці. Оскільки дублювання атрибуту можливо не тільки в безпосередньо зв'язаних таблицях, але й за послідовністю зв'язків, при формуванні запиту необхідно виконати операцію з'єднання таблиць, які присутні у цієї послідовності.

У випадку висхідної денормалізації передбачається заповнення такого атрибуту даними, обчисленими на основі даних з підлеглих таблиць. Тут також може бути задіяна послідовність пов'язаних таблиць. Надлишкові дані завжди обчислюються на основі заданої агрегованої функції. Але параметрами цієї функції в загальному випадку може бути вираз, в якому можуть приймати участь атрибути з будь-якої таблиці послідовності.

2.2 Формальне представлення запиту на заповнення даними денормалізованої таблиці

Введемо поняття цільових та вхідних таблиць в ієрархії зв’язків, які приймають участь у реалізації денормалізації. Цільова таблиця – така, для якої виконується зміна структури. Вхідна таблиця – така, дані з якої приймають участь у формуванні значень в доданих атрибутах.

Для нисхідного порядку цільовою таблицею є TableN (N≥2), вхідною - Table1.

Для висхідного порядку цільовою таблицею є Table1, вхідними – в загальному вигляді множина таблиць, елементами якої є TableI, I=2..N, можуть бути задіяні не всі таблиці з множини, але TableN – обов’язково присутня в множині.

Також введено поняття d-атрибуту, який є результатом денормалізації. У разі нисхідного порядку тип d-атрибуту співпадає з типом обраного для денормалізації NotPK з вхідної таблиці. У разі висхідного порядку d-атрибут має тип, який залежить від обраної агрегованої функції.

Таким чином, програма для автоматизованого створення таких запитів має виконувати наступні функції:

– з’єднання з базою даних;

– отримання списку таблиць бази даних;

– введення послідовності рівнів ієрархії зв’язків «один до багатьох» («один до одного») для завдання денормалізації структури бази даних;

– визначення атрибуту (вираження та агрегованої функції) для реалізації денормалізації;

– автоматизоване формування запиту на додавання атрибуту для реалізації денормалізації (нисхідної або висхідної);

– автоматизоване формування запиту на заповнення доданих атрибутів для реалізації денормалізації (нисхідної або висхідної);

– виконання сформованих запитів;

– перегляд змінених даних в таблицях, які приймають участь у денормалізації.

Для вирішення задачі у випадку і нисхідної, і висхідної денормалізації потрібно задати

– послідовність пов’язаних таблиць;

– для кожного зв’язку вказати атрибути первинного та зовнішнього ключів, за якими виконується зв’язок;

– назву доданого атрібуту.

Це є загальною частиною для обох варіантів.

Окремо для нисхідної денормалізації необхідно вказати атрибут вхідної таблиці, дані з якого будуть дублюватися.

Окремо для висхідної денормалізації необхідно вказати вид агрегатної функції та вираз, який є параметром цієї функції. Для формування виразу зручно мати конструктор. який дозволяє обирати назви атрибутів з множини атрибутів таблиць ланцюгу.

В загальному вигляді вхідні дані можна представити кортежем

$$D=<T, L, d, r>,$$

де T – упорядкована множина таблиць $|T| =N$;

L – множина пар для завдання зв’язків < $PKforTable_i$, $FKforTable_{i+1}$>;

d – назва доданого атрибуту;

r – правило формування даних:

$$r = \begin{cases} T_1.a \text{ у випадку нисхідної денормалізації} \\ < fa, expr > \text{ у випадку } d\text{исхідної денормалізації} \end{cases}$$

де a – атрибут, значення якого дублюється;

fa – агрегуюча функція;

$expr$ – вираз, який є параметром агрегуючої функції.

Загальна схема для додавання надлишкового атрибуту:

– для нисхідної денормалізації

alter table TableN

add d typea,

де *typea* співпадає з типом атрибуту $T_1.a$;

– для висхідної денормалізації

alter table Table1

add d typea,

де typea визначається видом агрегуючої функції: для COUNT це завжди ціле число, для інших функцій залежить від *expr*.

Загальна схема запиту для зміни даних має наступний вигляд:

– для нисхідної денормалізації

update TableN set d=

(select distinct a

from Table1 join Table2 join ... join TableN-1

on PKforTable1=FKforTable1 and ... and PKforTableN-2=FKforTableN-1

where PKforTableN-1= TableN.FKforTableN-1)

– для висхідної денормалізації

update Table1 set d=

(select fa(expr)

from Table2 join Table3 join ... join TableN

on PKforTable2=FKforTable2 and ... and PKforTableN-1=FKforTableN

where PKforTable1= Table2.FKforTable1)

Кортеж *D* може бути створений за частинами:

стан 1 - спочатку має бути створена множина таблиць *T*;

стан 2 - далі може бути створена множина пар атрибутів для завдання зв’язків *L*;

стан 3 – далі задаються характеристики додаткового атрибуту *d* та відбувається його створення;

стан 4 – задається правило заповнення атрибуту *d*;

стан 5 (фінальний) – заповнюється атрибут *d* відповідно правилу *r*.

Користувач може зупинитися та будь-якому етапі і продовжити роботу з варіантом денормалізації в інший час. Послідовність станів показана на рис. 2.2.

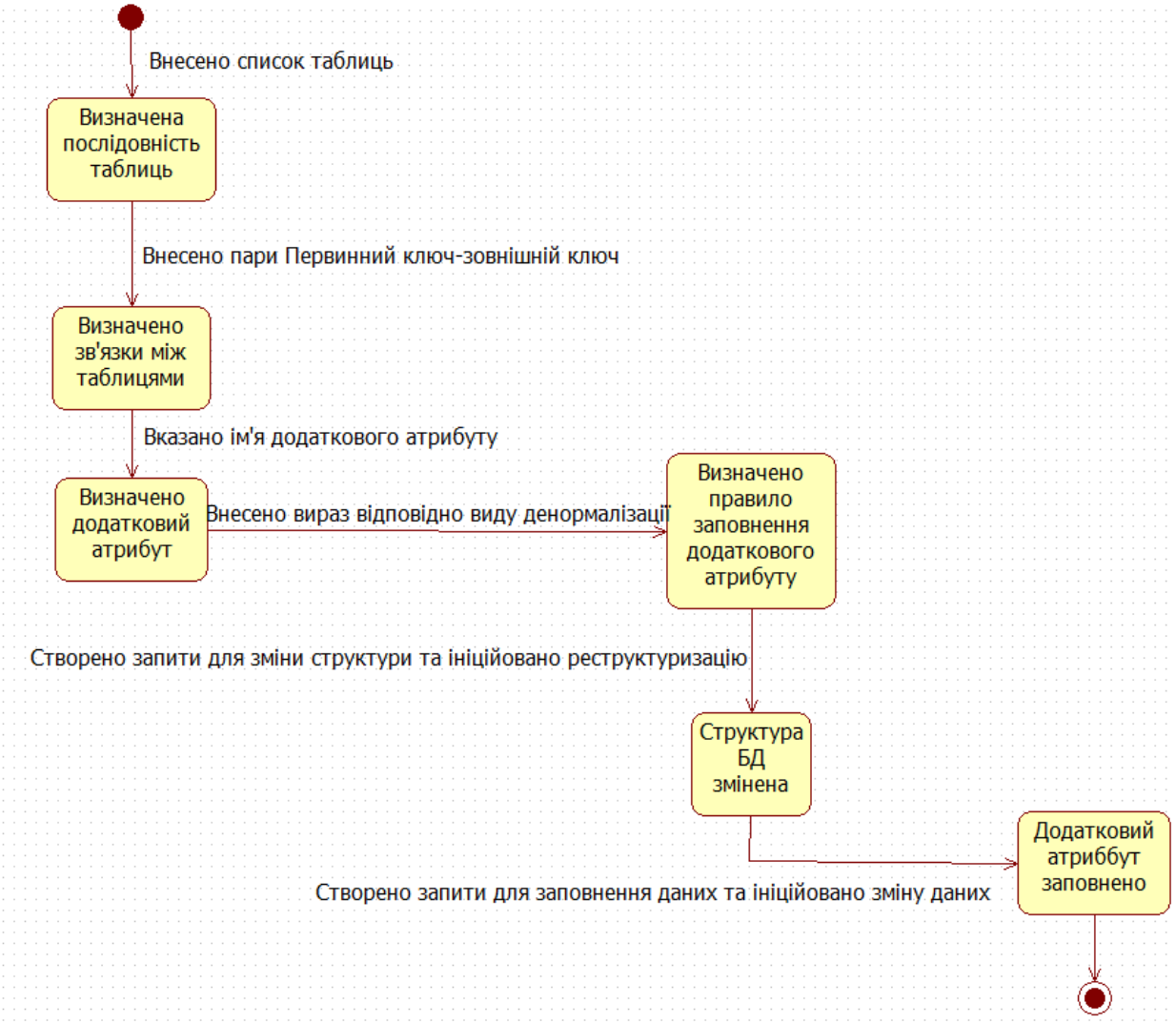


Рисунок 2.2 – Діаграма станів

2.3 Висновки до розділу

В розділі обгрунтовано вибір варіантів денормалізації для автоматизації. Формалізовано загальне представлення обраних варіантів з точки зору зміни структури БД та формування запитів для заповнення доданих надлишкових атрибутів.

3 ВИМОГИ ДО ПРОГРАМИ ФОРМУВАННЯ ЗМІСТУ ДЕНОРМАЛІЗОВАНИХ ТАБЛИЦЬ

3.1 Узагальнений опис роботи програми

Для опису роботи програми визначено діаграму активності (рис.3.1).

Діаграма активності описує, як координуються дії для надання послуги, що може перебувати на різних рівнях абстракції. Подія вимагає для досягнення деяких операцій, особливо коли операція призначена для досягнення ряду різних цілей, що вимагають координації, або коли події в одному варіанті використання співвідносяться одна з одною, зокрема, у варіантах використання, коли дії можуть перетинатися й вимагати координації. Діаграма активності також підходить для моделювання того, як набір варіантів використання координується для представлення бізнесів-процесів.

Діаграма активності відображає послідовні й паралельні процеси. Використається для моделювання бізнесів-процесів, послідовностей виконання завдань і потоків даних, а також складних алгоритмів. Діаграма діяльності встановлює основні правила послідовності дій, тобто показує, що відбувається.

Діаграма активності служить для:

– виявлення можливих варіантів використання шляхом вивчення бізнесів-процесів;

– визначення попередніх і наступних умов (контекст) для варіантів використання;

– моделювання робочих процесів між/усередині варіантів використання;

– моделювання складних робочих процесів в операціях над об'єктами;

– детального моделювання складних дій на високорівневій діаграмі дій.

Рисунок 3.1 – Діаграма активності для моделювання роботи програми

3.2 Функціональні вимоги

Щоб забезпечити універсальність роботи програми, необхідно забезпечити роботу різних користувачів з їх власними БД. Для цього необхідно створити кабінети для кожного користувача. Це потребує ідентифікації користувача, що може бути зроблено за унікальними логіном та паролем.

Для роботи з базою даних необхідно підключення до неї, що можна зробити за допомогою стандартного рядка підключення, який має вказати користувач.

Далі програма автоматично може отримувати списки таблиць та множини атрибутів цих таблиць, що дозволяє отримати конструктор виразів та конструктор послідовності таблиць для виконання денормалізації.

Таким чином, можна визначити користувача лише одного типу.

Повна сесія роботи користувача по заповненню даних включає такі стадії:

- визначення БД;
- визначення послідовності таблиць;
- визначення виду денормалізації;
- визначення та створення додаткового атрибуту;
- визначення джерела формування даних в додатковому атрибуті та заповнення даними.

Користувач може створювати багато сесій, сесія може бути пов’язана з однією БД, але різні сесії користувача можуть підключатися до різних БД.

Користувач може залишити сесію незакінченою на будь-якій стадії та продовжити її у будь-який час.

Для кожного користувача, таким чином, існує поняття особистого кабінету, у якому він може переглядати створені сесії та продовжувати незакінчені сесії.

Функціональні вимоги описані у вигляді множини варіантів використання (рис.3.2).

В табл. 3.1-3.8 надані детальні описи варіантів використання для системи.

Рисунок 3.2 – Діаграма варіантів використання

Таблиця 3.1 - Варіант використання «Підключення до БД»

Назва варіанту	Підключення до БД	
Ціль	Отримання доступу до схеми БД та можливість відправляти запити до БД	
Діючі особи (актори)	Адміністратор БД	
Короткий опис	Вказівка рядка для підключення до БД; перевірка встановлення зв’язку з БД	
Тип варіанту	Основний	
Передумова	Користувач авторизований	
Результат	Користувач має змогу працювати зі схемою БД	
№ кроку	Дії користувача	Відгук системи
Основний сценарій		
1	Користувач вводить рядок для підключення до БД (драйвер. назва БД, ім’я та пароль користувача для доступу до БД)	Система встановлює зв’язок з БД та виводить повідомлення про успішне з’єднання
2	Користувач хоче побачити список таблиць	Система формує список таблиць БД

Продовження табл. 3.1

№ кроку	Дії користувача	Відгук системи
3	Користувач хоче побачити список атрибутів таблиці	Система формує список атрибутів обраної таблиці
Альтернативні сценарії		
1а	Користувач не ввів рядок	Система виводить повідомлення про необхідність вказати дані
1б	Система не може з’єднатися з БД	Система виводить повідомлення про невірні дані
3а	Користувач не обрав таблицю у списку	Система виводить повідомлення про необхідність визначити таблицю

Таблиця 3.2 - Варіант використання «Створення послідовності таблиць»

Назва варіанту	Створення послідовності таблиць
Ціль	Формування багаторівневої ієрархії для переносу надлишкових даних
Діючі особи (актори)	Адміністратор БД
Короткий опис	Завдання упорядкованої множини таблиць шляхом вибору назв таблиць із автоматично отриманого списку після підключення до БД
Тип варіанту	Основний
Передумова	Виконано підключення до БД
Результат	Користувач визначив вхідну та цільову таблиці та визначив шлях між ними

Продовження табл. 3.2

№ кроку	Дії користувача	Відгук системи
Основний сценарій		
1	Користувач обирає таблицю, найвищу в ієрархії, для завдання послідовності	Система фіксує першу таблицю в послідовності та пропонує визначити наступну таблицю
2	Користувач обирає таблицю, підлеглу для останньої в послідовності таблиці	Система фіксує поточну таблицю в послідовності та пропонує визначити зв’язок між таблицями
3	Користувач визначає пару атрибутів «первинний ключ – зовнішній ключ»	Система фіксує зв’язок між останньою та передостанньою таблицями послідовності та пропонує визначити наступну таблицю
	Кроки 2 та 3 повторюється потрібну кількість разів	
4	Користувач обирає таблицю, найнижчу в ієрархії, для завдання послідовності	Система фіксує останню таблицю в послідовності
5	Користувач хоче зберегти послідовність	Система зберігає введену послідовність
Альтернативні сценарії		
3а	Вказані користувачем атрибути мають різний тип	Система виводить повідомлення про те, що вказана невірна пара атрибутів
3б	Послідовність включає лише 2 таблиці	Система фіксує останню таблицю в послідовності

Продовження табл. 3.2

№ кроку	Дії користувача	Відгук системи
5а	Користувач не визначив жодної таблиці в послідовності	Система виводить повідомлення про необхідність визначити послідовність таблиць

Таблиця 3.3 - Варіант використання «Створення додаткового атрибуту»

Назва варіанту		Створення додаткового атрибуту
Ціль		Отримання готової структури для подальшого внесення даних
Діючі особи (актори)		Адміністратор БД
Короткий опис		Вказується вид денормалізації та назва і тип доданого атрибуту з надлишковими даними
Тип варіанту		Основний
Передумова		Створена послідовність таблиць
Результат		Користувач має змінену структуру цільової таблиці
№ кроку	Дії користувача	Відгук системи
Основний сценарій		
1	Користувач обирає вид денормалізації	Система визначає цільову таблицю в послідовності
2	Користувач задає назву і тип даних додаткового атрибуту	Система фіксує назву та тип
3	Користувач хоче створити новий атрибут	Система формує запит на створення атрибуту, зберігає сформований запит у журналі та виконує запит

Продовження табл. 3.3

№ кроку	Дії користувача	Відгук системи
Альтернативні сценарії		
2а	Користувач не вказав назву атрибуту	Система виводить повідомлення про необхідність вказати назву
2б	Користувач вказав невірну назву атрибуту	Система фіксує помилку при виконанні запиту та виводить повідомлення про необхідність змінити назву
2в	Користувач ввів назву атрибуту, яка співпадає з існуючим атрибутом в цільовій таблиці	Система виводить повідомлення про необхідність змінити назву

Таблиця 3.4 - Варіант використання «Заповнення реструктурованої БД»

Назва варіанту	Заповнення реструктурованої БД
Ціль	Автоматично заповнити даними доданий атрибут
Діючі особи (актори)	Адміністратор БД
Короткий опис	Вказується атрибут – джерело даних або вираз та агрегуюча функція, за використанням цих даних виконується запит для заповнення доданого атрибуту
Тип варіанту	Основний
Передумова	Створений додатковий атрибут
Результат	Користувач має заповнений даними додатковий атрибут в цільовій таблиці

Продовження табл. 3.4

№ кроку	Дії користувача	Відгук системи
Основний сценарій		
1	Користувач формує вираз для обчислення значення додаткового атрибуту	Система фіксує вираз
2	Користувач хоче заповнити додатковий атрибут даними	Система формує запит на оновлення цільової таблиці, зберігає сформований запит у журналі та виконує запит
Альтернативні сценарії		
1а	Користувач не вказав необхідні дані	Система виводить повідомлення про необхідність визначити вираз
2б	Користувач вказав невірний вираз	Система фіксує помилку при виконанні запиту та виводить повідомлення про необхідність скорегувати вираз

Таблиця 3.5 - Варіант використання «Перегляд запитів для реструктуризації»

Назва варіанту	Перегляд запитів для реструктуризації
Ціль	Отримати інформацію про автоматично створений запит
Діючі особи (актори)	Адміністратор БД
Короткий опис	Користувач може переглянути та скопіювати запит, який був автоматично створений системою
Тип варіанту	Основний

Продовження табл. 3.5

Передумова	Створений додатковий атрибут і/або додатковий атрибут заповнений даними	
Результат	Користувач бачить на екрані текст запиту	
№ кроку	Дії користувача	Відгук системи
Основний сценарій		
1	Користувач обирає послідовність	Система знаходить в журналі відповідні послідовності запити
2	Користувач хоче побачити запити	Система виводить тексти запитів на екран
Альтернативні сценарії		
2а	Користувач не обрав послідовність	Система виводить повідомлення про необхідність визначити послідовність

Таблиця 3.6 - Варіант використання «Перегляд власних сесій»

Назва варіанту	Перегляд власних сесій
Ціль	Мати можливість переглянути історію власних дій в системі
Діючі особи (актори)	Адміністратор БД
Короткий опис	Користувач має дані про всі свої сесії в системі
Тип варіанту	Основний
Передумова	Користувач авторизований
Результат	Користувач отримує список сесій

Продовження табл. 3.6

№ кроку	Дії користувача	Відгук системи
Основний сценарій		
1	Користувач хоче побачити сесії	Система виводить список, який включає дані БД та створену під час сесії послідовність
Альтернативні сценарії		
1а	Користувач ще не створював сесії	Система виводить повідомлення про відсутність сесій у користувача

Таблиця 3.7 - Варіант використання «Перегляд журналу сесії»

Назва варіанту	Перегляд журналу сесії
Ціль	Мати можливість отримати детальні дані про власні дії в рамках конкретної сесії
Діючі особи (актори)	Адміністратор БД
Короткий опис	Користувач має повні дані про свої дії в системі
Тип варіанту	Основний
Передумова	Користувач обрав сесію
Результат	Користувач отримує детальну інформацію про сесію – стан сесії, введену послідовність, вид денормалізації, додатковий атрибут, вираз для заповнення даних, тексти запитів

Продовження табл. 3.7

№ кроку	Дії користувача	Відгук системи
Основний сценарій		
1	Користувач обирає сесію	Система визначає деталі сесії
2	Користувач хоче побачити деталі сесії	Система виводить на екран відповідні стану сесії дані послідовність таблиць, обраний вид денормалізації, метадані додаткового атрибуту, джерело даних для заповнення додаткового атрибуту, тексти запитів для створення та заповнення додаткового атрибуту
Альтернативні сценарії		
1а	Користувач ще не створював сесії	Система виводить повідомлення про відсутність сесій у користувача

Таблиця 3.8 - Варіант використання «Авторизція»

Назва варіанту	Авторизція
Ціль	Мати можливість працювати з власними даними. захищеними від доступу інших користувачів
Діючі особи (актори)	Адміністратор БД
Короткий опис	Користувач отримує доступ до власних сесій, може створювати нову сесію або переглядати існуючі сесії
Тип варіанту	Основний
Передумова	Користувач зайшов в систему
Результат	Користувач має власний кабінет

Продовження табл. 3.8

№ кроку	Дії користувача	Відгук системи
Основний сценарій		
1	Користувач хоче війти в свій кабінет	Система виводить поля для внесення логіну та паролю
2	Користувач вводить логін та пароль	Система перевіряє правильність даних та відкриває вікно для роботи з новою сесією
Альтернативні сценарії		
2а	Користувач не вказав логін або пароль	Система виводить повідомлення про необхідність вказати дані
2б	Користувач вказав невірний логін або пароль	Система виводить повідомлення про відсутність таких логіну або паролю

3.3 Нефункціональні вимоги

Як функціональні, так і нефункціональні вимоги описують конкретні характеристики, які має мати продукт для задоволення потреб зацікавлених сторін і самого бізнесу [19].

Функціональні вимоги визначають, що повинен робити програмний продукт: його властивості й функції.

Нефункціональні вимоги, не пов'язані з функціональністю системи, визначають, як повинна працювати система. Нефункціональні вимоги визначають атрибути якості системи, тому існує їх друга назва - атрибути якості .

Нижче перелічені основні нефункціональні вимоги до програми формування змісту денормалізованих таблиць.

1. Зручність використання

Ефективність використання: середній час, необхідний користувачеві для досягнення цілей, залежить від кількості рядків у таблицях БД, що хоче промоделювати користувач; всі завдання користувач може виконати без сторонньої допомоги, усе транзакцій виконуються без помилок завдяки системі повідомлень про помилки.

Інтуїтивність - програма має систему підказок, елементи керування з написами, які дозволяють просто розібратися у функціоналі.

Передбачуване робоче навантаження низьке - заповнення таблиць виконується з однієї спроби.

2. Безпека

Програмне забезпечення захищене від несанкціонованого доступу до системи й збережених у ній даним за рахунок системи логінов і паролів і зберігання паролів у зашифрованому виді. Кожний користувач має доступ тільки до своїм даних завдяки особистому кабінету. Ніхто, крім користувача, не може створювати, переглядати, копіювати, змінювати або видаляти інформацію.

3. Надійність

Програма має працювати без збоїв протягом усього періоду часу з імовірністю 99%. База даних може бути відновлена завдяки створюваному із заданим користувачем періодом архіву.

4. Продуктивність

Час завантаження кожної сторінки для користувачів - не більше 2 секунд.

5. Доступність

Функціональні можливості й послуги системи доступні для використання у всіх операціях 24/7.

Відновлення модулів програми не повинне впливати на доступність сторінок.

6. Масштабованість

Ліміт відвідуваності користувачами повинен підтримувати 1 000 користувачів одночасно.

3.4 Аналіз міжсесійних конфліктів

Можливі міжсесійні конфлікти наступного виду.

1. Дублювання додаткових атрибутів

Користувач може у різних сесіях, пов’язаних з однією БД, виконати спробу створити додатковий атрибут з іменем, яке вже використане в іншій сесії.

Оскільки сесія не може бути залишена у стані завдання імені атрибуту, але до його створення (завдання імені та створення атрибуту є єдиним етапом), то проблема не виникає, оскільки перед створенням атрибуту виконується перевірка існування такого атрибуту.

2. Дублювання надлишкових атрибутів

Можливо, що користувач спробує у різних сесіях, пов’язаних з однією БД, створити додаткові атрибути в одній цільовій таблиці, які мають різні імена, але заповнюються даними з одного атрибуту вхідної таблиці.

Система перевіряє, чи не існує сесії з однаковими правилами заповнення даними, і у разі виникнення конфлікту попереджає користувача.

3. Дублювання послідовності

Можливо, користувач спробує створити однакові послідовності за однаковими правилами. але зупинитися на стадіях різних стадіях.

Система має відслідкувати ідентичність введених частин та попередити користувача, що, можливо, такий варіант вже було почато створювати.

3.5 Висновки до розділу

В розділі наведено вимоги до програми формування змісту денормалізованих таблиць. Описані функціональні вимоги, які описують поводження системи в певних умовах.

Функціональні вимоги описані у вигляді варіантів використання. Для кожного варіанту використання наданий сценарій, який є послідовністю подій разом з іншою інформацією, що відноситься до варіанта використання, тобто включає наступну інформацію: опис, стан до та після взаємодії, основний шлях взаємодії, альтернативний шлях взаємодії. Така форма є зручною та зрозумілою як для команди розробників, так і для зацікавлених сторін.

Також надані нефункціональні умови, які описують, як має працювати система.

В розділі також описані шляхи вирішення можливих міжсесійних конфліктів які можуть привести до помилок під час виконання функцій в рамках однієї сесії. Конфлікти можуть виникати лише між сесіями, пов’язаними з однією БД. Теоретично кілька користувачів системи можуть працювати з єдиною БД, що приводить до необхідності виконувати перевірки конфліктів не лише між сесіями одного користувача. але й між сесіями різних користувачів.

4 ПРОЕКТУВАННЯ ПРОГРАМИ ДЛЯ ЗМІНИ ЗМІСТУ ТАБЛИЦЬ БАЗИ ДАНИХ ДЛЯ РЕАЛІЗАЦІЇ ДЕНОРМАЛІЗАЦІЇ

4.1 Структура застосунку

Для програми обрано найбільш поширену для веб-застосунків архітектуру Клієнт-Сервер-БД.

Клієнт - це те, із чим взаємодіє користувач, тобто код клієнта відповідає за більшу частину того, що бачить користувач. Код включає:

– визначення структури веб-сторінки;

– настроювання зовнішнього вигляду веб-сторінки;

– реалізацію способу користувальницької взаємодії, зокрема, натискання кнопок, уведення тексту й т.д.

Структура веб-сторінки визначається за допомогою HTML. На рис.4.1 показано фрагмент формування сторінки застосунку.

```
<table><tr><td class=bg><center>
<img class=logo src='images/logo.jpg'>
</td><td class=bg><?=$usertext?></td></tr>
<tr><td><div class='allone'>
<input type=button class=<?=$ccurrentmenu?> value='Поточна схема' onclick='scheme();'><br>
<input type=button class=<?=$carchivemenu?> value='Архів' onclick='archive();'><br><br>
<input type=button class='menu' value='Нова схема' onclick='newscheme();'>
</div></td>
<td><div class='alltwo'>
<?if($user==0)
    echo 'Необхідно авторизуватися';
else{
    if($menu=='current'){
        include_once ("view/current.php");
    }
    else{
        include_once ("view/archive.php");
    }
}
include_once ("view/addupdate.php");
?>
</div></td></tr></table>
<div class=footer>
    Blizhnikov Maksym - Qualification work
```

Рисунок 4.1 – Формування головної сторінки

Для визначення зовнішнього вигляду веб-сторінки використається CSS - мова, що дозволяє описати стиль елементів, певних в HTML. На рис.4.2 показано фрагмент визначення стилю елемента керування за допомогою CSS.

```
.button{
 color: #4682B4;
 background-color:#DCDCDC;
 font-weight:600;
 border:2px solid #DCDCDC;
}
```

Рисунок 4.2 – Опис зовнішнього виду
кнопки застосунку за допомогою CSS

Для реалізації механізму взаємодії з користувачем використовується JavaScript на стороні клієнта, щоб скоротити звертання до сервера. На рис.4.3 показано визначення функції формування списку полів обраної користувачем таблиці при визначенні виду денормалізації за допомогою JavaScript.

```
function selecttableup(){
tbl=document.getElementById('tableup').value
 arg='entity=field&table='+tbl+
 '&database='+document.getElementById('database').value+
 '&session='+document.getElementById('session').value;
 $.ajax({
   type: "GET",
   url: '../addmodules/addupdate.php',
   data: arg,
   success: function(data){
       document.getElementById('tablefieldup').innerHTML=data;
   }
 });
}
```

Рисунок 4.3 – Реалізація функції застосунку за допомогою JavaScript

Сервер прослуховує запити, що надходять від клієнта. На рис.4.4 показано, як обробляються дані, отримані за допомогою запиту.

```php
$entity=$_GET['entity'];
if($entity=='currentuser'){
    $cuser=new User;
    $user=$cuser->checkuser($_GET['login'],$_GET['pass']);
    echo $user;
}
```

Рисунок 4.4 – Обробка запитів до серверу

База даних призначена для зберігання інформації, щоб її можна було просто читати, обновляти, у цілому управляти нею. На рис. 4.5 показано фрагмент коду для підключення до БД.

```php
Class Controller{
    static public $host='localhost';
    static public $user='root';
    static public $password='root';
    public static $database='denorm';
}

Class Session extends Controller{
    public function getCurrentSession($user){
        $connection=mysqli_connect(Controller::$host, Controller::$user, Controller::$password, Controller::$database);
        $qr="SELECT sessionid FROM `session`
```

Рисунок 4.5 – Підключення до БД

База даних реалізована за допомогою системи керування базами даних MySQL (рис. 4.6).

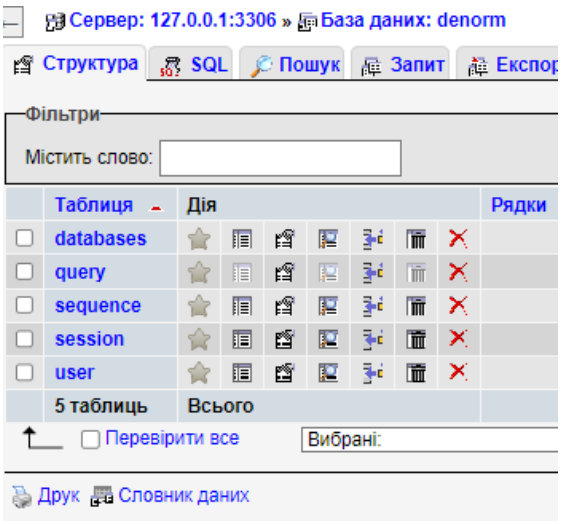


Рисунок 4.6 – Таблиці БД

4.2 Діаграми взаємодії

Діаграми взаємодії призначені для візуалізації інтерактивного поводження системи, що є частиною динамічного поводження системи. Інтерактивне поводження представлене в UML діаграмами послідовності (наголошує на часовій послідовності повідомлень) і діаграмами співробітництва (підкреслює структурну організацію об'єктів, які відправляють і одержують повідомлення) [19].

Ціль діаграми взаємодії:

- зафіксувати динамічне поводження системи;
- описати потік повідомлень у системі;
- описувати структурну організацію об'єктів;
- описувати взаємодії між об'єктами.

На рис.4.7 наведена діаграма послідовностей для процесу створення сесії для виконання денормалізації структури БД. Діаграма описує загальну послідовність дій в системі. Відповідно [20], «як правило, в цій діаграмі демонструється, як користувачі (актори з діаграми варіантів використання) взаємодіють з іншими компонентами програми під час реалізації тих чи інших варіантів використання програми, та як при цьому взаємодіють інші компоненти програмної системи».

На рис. 4.8-4.10 наведені діаграми співробітництва, які надають більш детальну інформацію про методи класів. Відповідно [21] «діаграми співробітництва являють собою сукупність об’єктів, поведінка яких значуща для досягнення складових мети системи, та взаємовідношення тих ролей, які об’єкти відіграють у співробітництві; на даному вигляді діаграм моделюється статична взаємодія об’єктів, при цьому фактор часу не враховується і не відображається на діаграмі співробітництва».

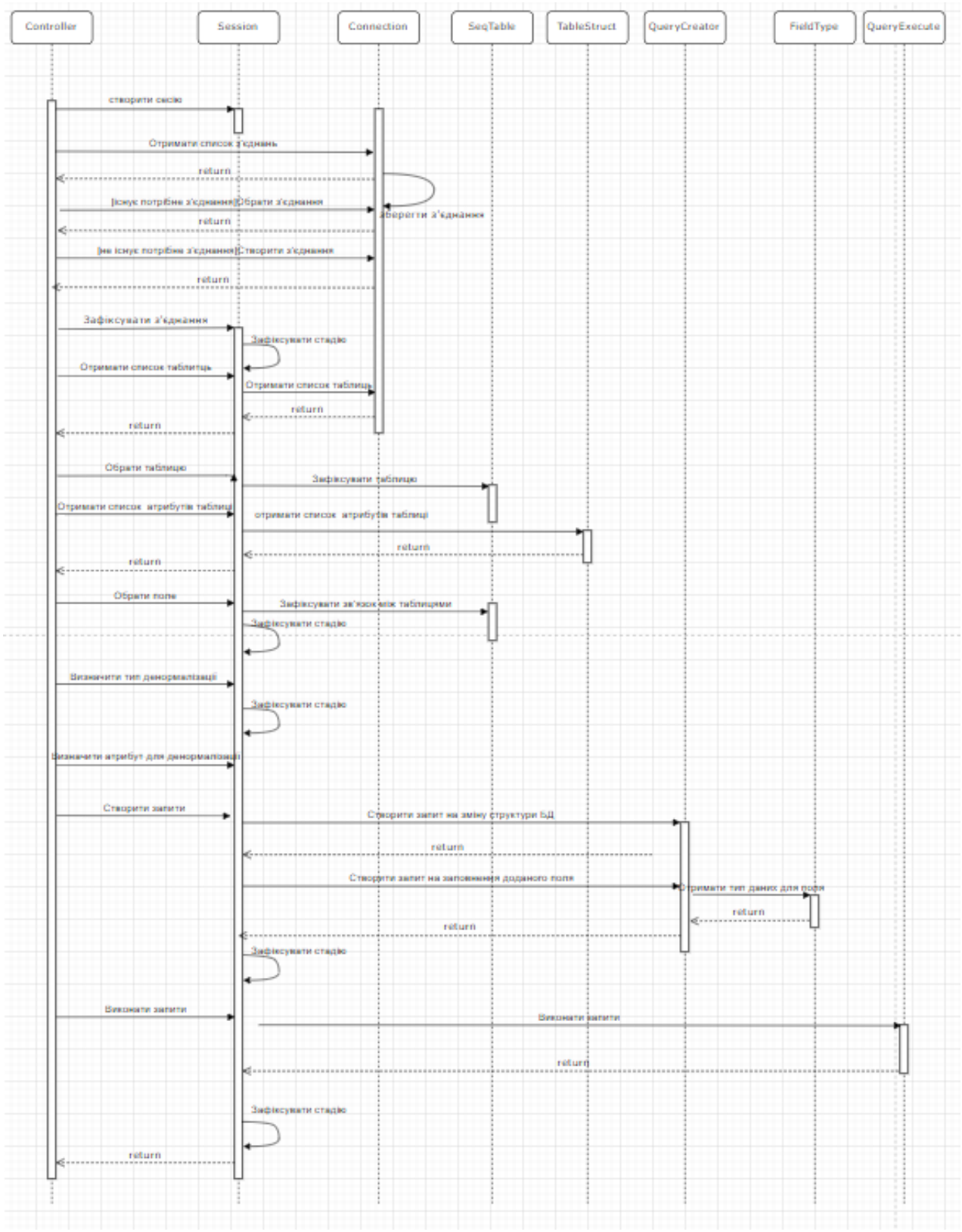


Рисунок 4.7 - Діаграма послідовності для процесу створення сесії

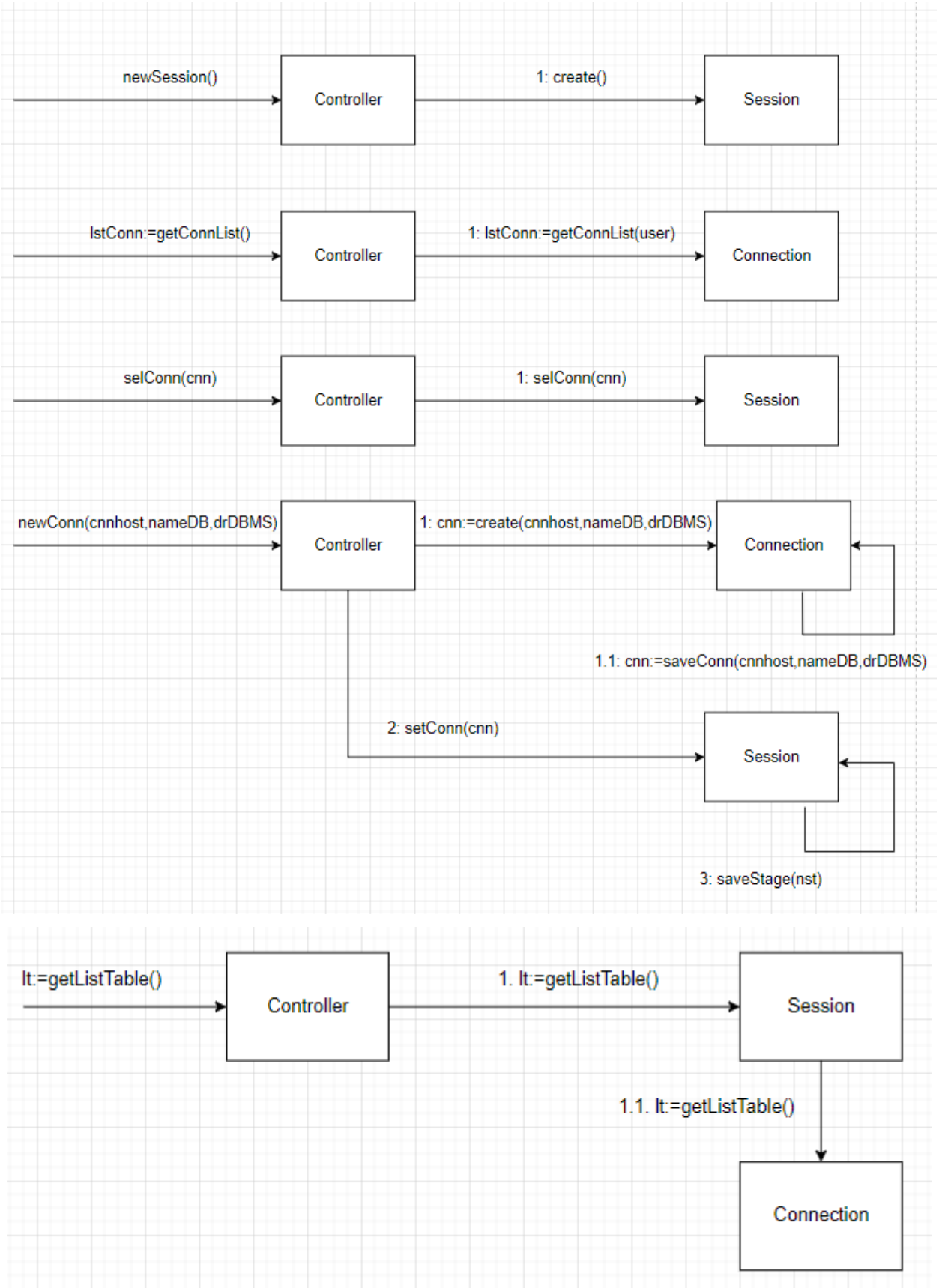


Рисунок 4.8 - Діаграма співробітництва для процесу створення сесії (частина1)

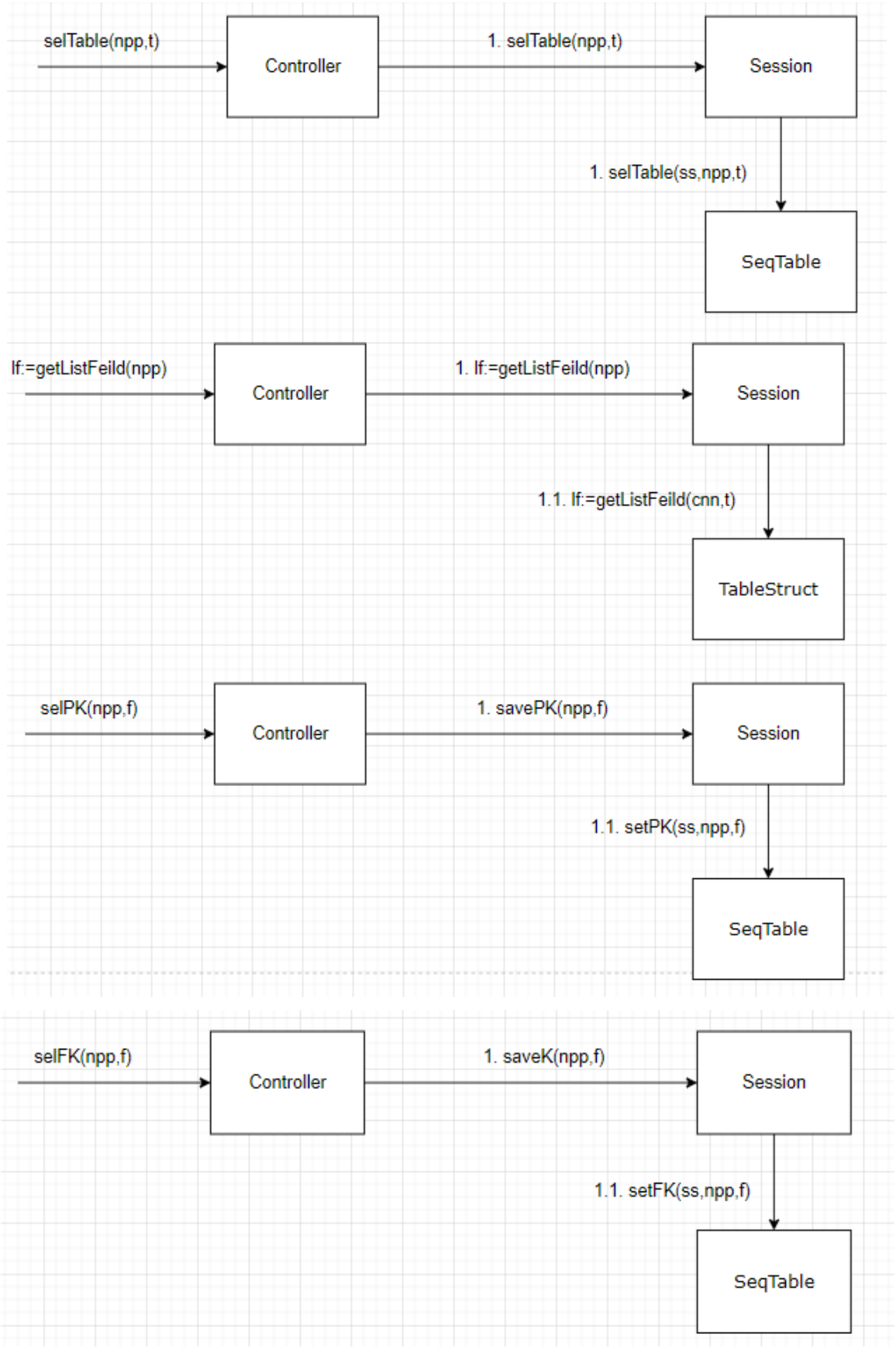


Рисунок 4.9 - Діаграма співробітництва для процесу створення сесії (частина2)

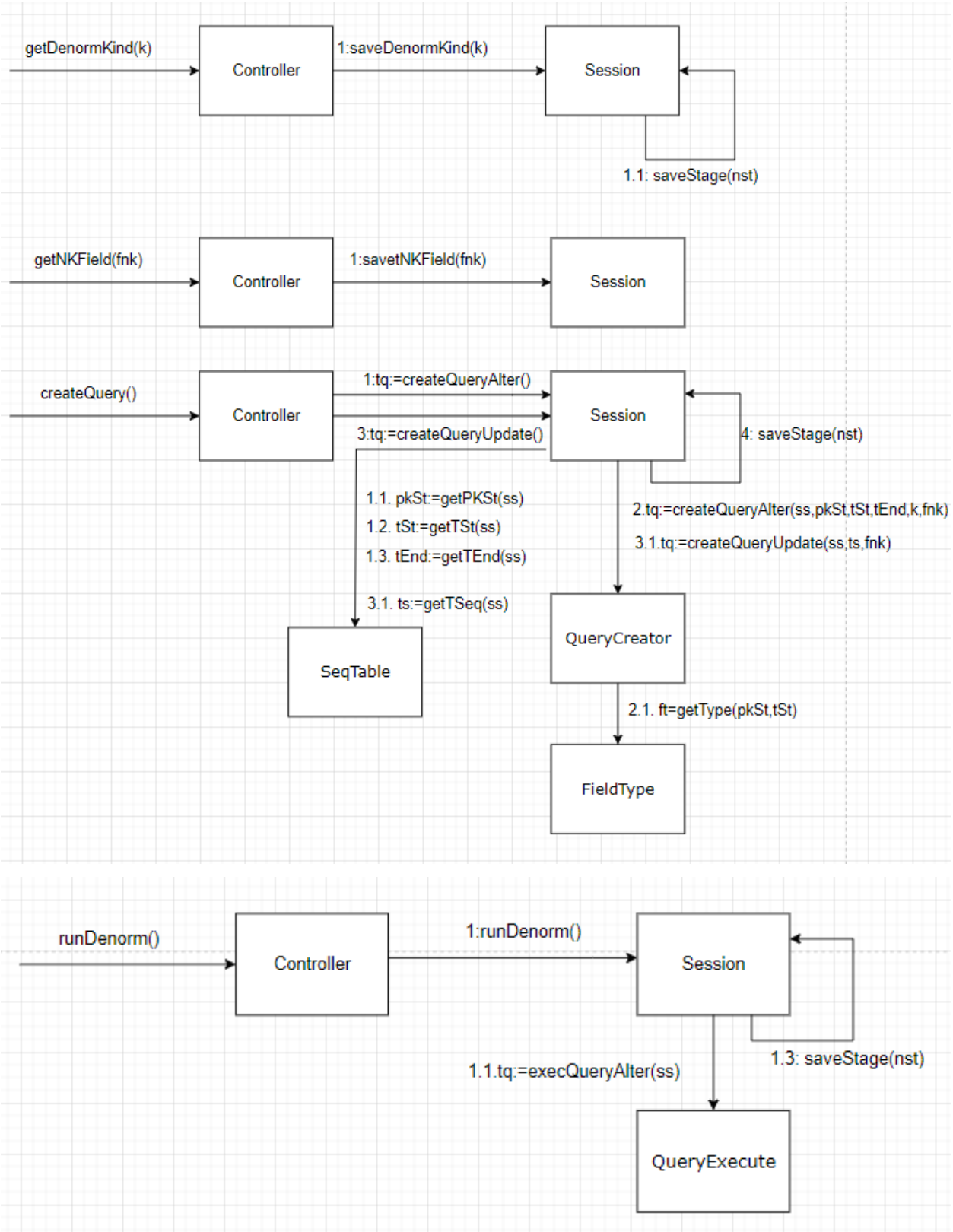


Рисунок 4.10 - Діаграма співробітництва для процесу створення сесії (частина3)

4.3 Структура бази даних

Концептуальна модель даних представляє сутності , необхідні для збереження інформації про функціювання користувачів системи. а також зв’язки між цими сутностями (рис.4.10).

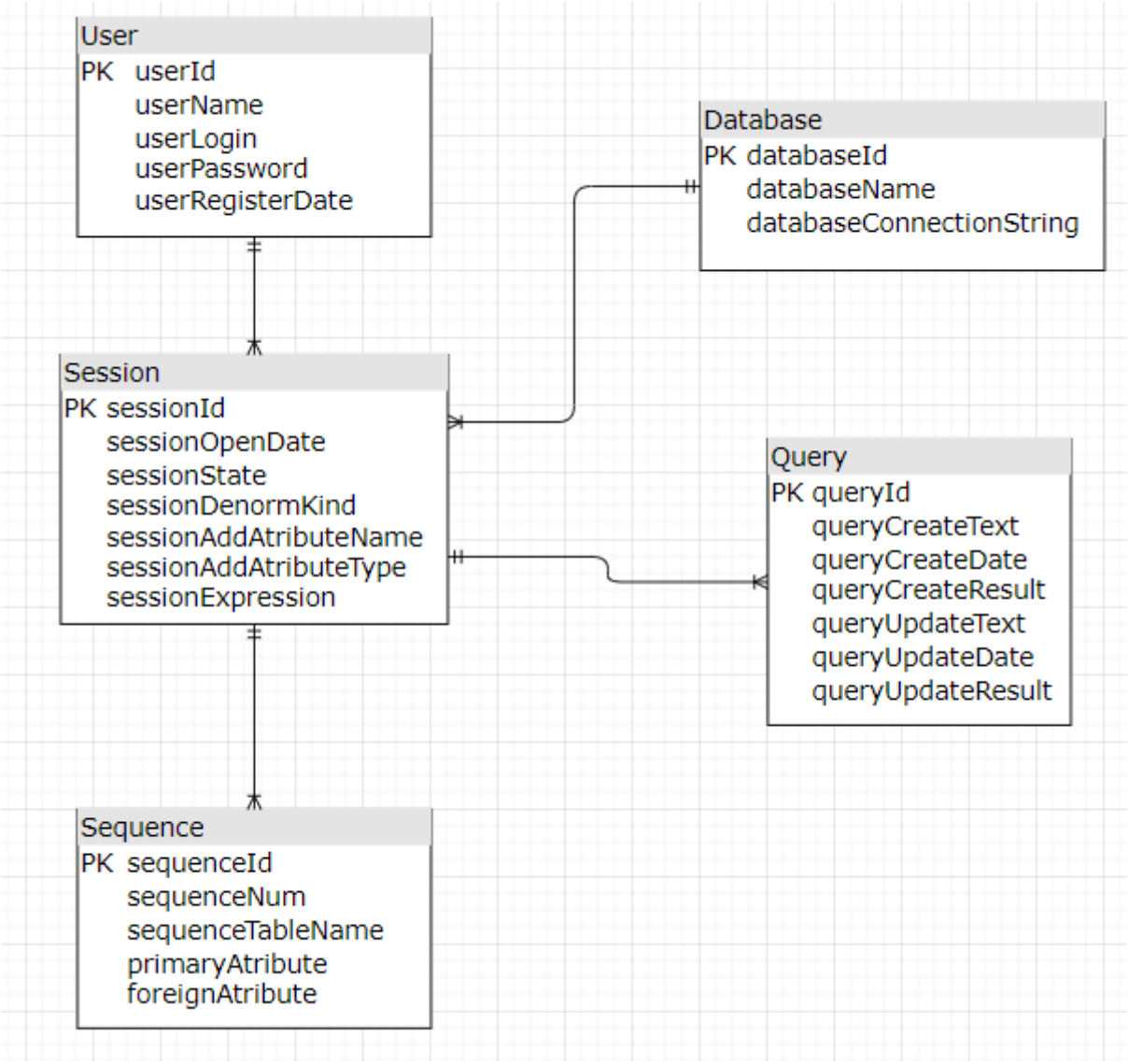


Рисунок 4.10 – Концептуальна модель даних

На основі розробленої концептуальної моделі даних створена реляційна модель даних в СКБД MySQL.

Реляційна база даних включає такі таблиці.

Таблиця User зберігає список зареєстрованих користувачів (табл.4.1).

Таблиця 4.1 – Атрибути таблиці User

Поле	Опис	Тип даних	Розмір	Ключ
userId	Ідентифікатор користувача	int	3	Первинний ключ
userName	Ім’я користувача	varchar	30	
userLogin	Логін користувача	varchar	30	
userPassword	Пароль користувача	varchar	30	
userRegisterDate	Дата реєстрації	date		

Таблиця Session зберігає множину сесій користувачів (табл.4.2).

Таблиця 4.2 – Поля таблиці Session

Поле	Опис	Тип даних	Розмір	Ключ
sessionId	Ідентифікатор сесії	int	5	Первинний ключ
sessionOpenDate	Дата відкриття сесії	date	30	
sessionState	Стан сесії	int	1	
sessionDenormKind	Вид денормалізації	int	1	
sessionAddAtributeName	Назва додаткового атрибуту	varchar	50	
sessionAddAtributeType	Тип додаткового атрибуту	varchar	20	
sessionExpression	Вираз для заповнення даними	varchar	100	

Продовження табл. 4.2

Поле	Опис	Тип даних	Розмір	Ключ
userId	Ідентифікатор користувача	int	2	Зовнішній ключ
databaseId	Ідентифікатор бази даних	int	2	Зовнішній ключ

Таблиця Sequence зберігає списки таблиць, які формують послідовності для багаторівневої ієрархії (табл.4.3).

Таблиця 4.3 – Поля таблиці Sequence

Поле	Опис	Тип даних	Розмір	Ключ
sequenceId	Ідентифікатор послідовності	int	9	Первинний ключ
sequenceTableName	Назва таблиці	varchar	30	
primaryAtribute	Атрибут зв’язку у основній таблиці	varchar	30	
foreignAtribute	Атрибут зв’язку у підлеглій таблиці	varchar	30	
sessionId	Ідентифікатор сесії	int	5	Зовнішній ключ

Таблиця Database зберігає дані про підключення до БД (табл.4.4).

Таблиця 4.4 – Поля таблиці Database

Поле	Опис	Тип даних	Розмір	Ключ
databaseId	Ідентифікатор бази даних	int	2	Первинний ключ
databaseName	Назва бази даних	varchar	30	
databaseConnectionString	Рядок для з’єднання з базою даних	varchar	200	

Таблиця Query зберігає інформацію, про можливість та особливості виконання досліджень показників в лабораторіях. Реалізує зв'язок «багато до багатьох» між лабораторією та показником (табл.4.5).

Таблиця 4.5 – Поля таблиці Query

Поле	Опис	Тип даних	Розмір	Ключ
queryId	Ідентифікатор запиту	int	5	Первинний
queryCreateText	Текст запиту на створення додаткового атрибуту	varchar	200	Зовнішній
queryCreateDate	Дата виконання запиту на створення додаткового атрибуту	date		Зовнішній
queryCreateResult	Результат виконання запиту на створення додаткового атрибуту	int	1	

Продовження табл. 4.5

Поле	Опис	Тип даних	Розмір	Ключ
queryUpdateText	Текст запиту на заповнення додаткового атрибуту	varchar	500	
queryUpdateDate	Дата виконання запиту на заповнення додаткового атрибуту	date		
queryUpdateResult	Результат виконання запиту на заповнення додаткового атрибуту	int	1	
sessionId	Ідентифікатор сесії	int	5	Зовнішній ключ

4.4 Алгоритм створення запиту на зміну змісту денормалізованої БД у випадку нисхідної денормалізації

Алгоритм складається з наступних кроків. Запит формується з трьох частин.

Перша частина містить текст, який визначає тип запиту (update), таблицю, зміст якої має бути зміненим, атрибут, значення якого має бути заповнено.

Друга частина містить перелік таблиць, між якими виконується операція з’єднання (join).

Третя частина містить перелік умов з’єднання.

Схема алгоритму показана на рис. 4.11.

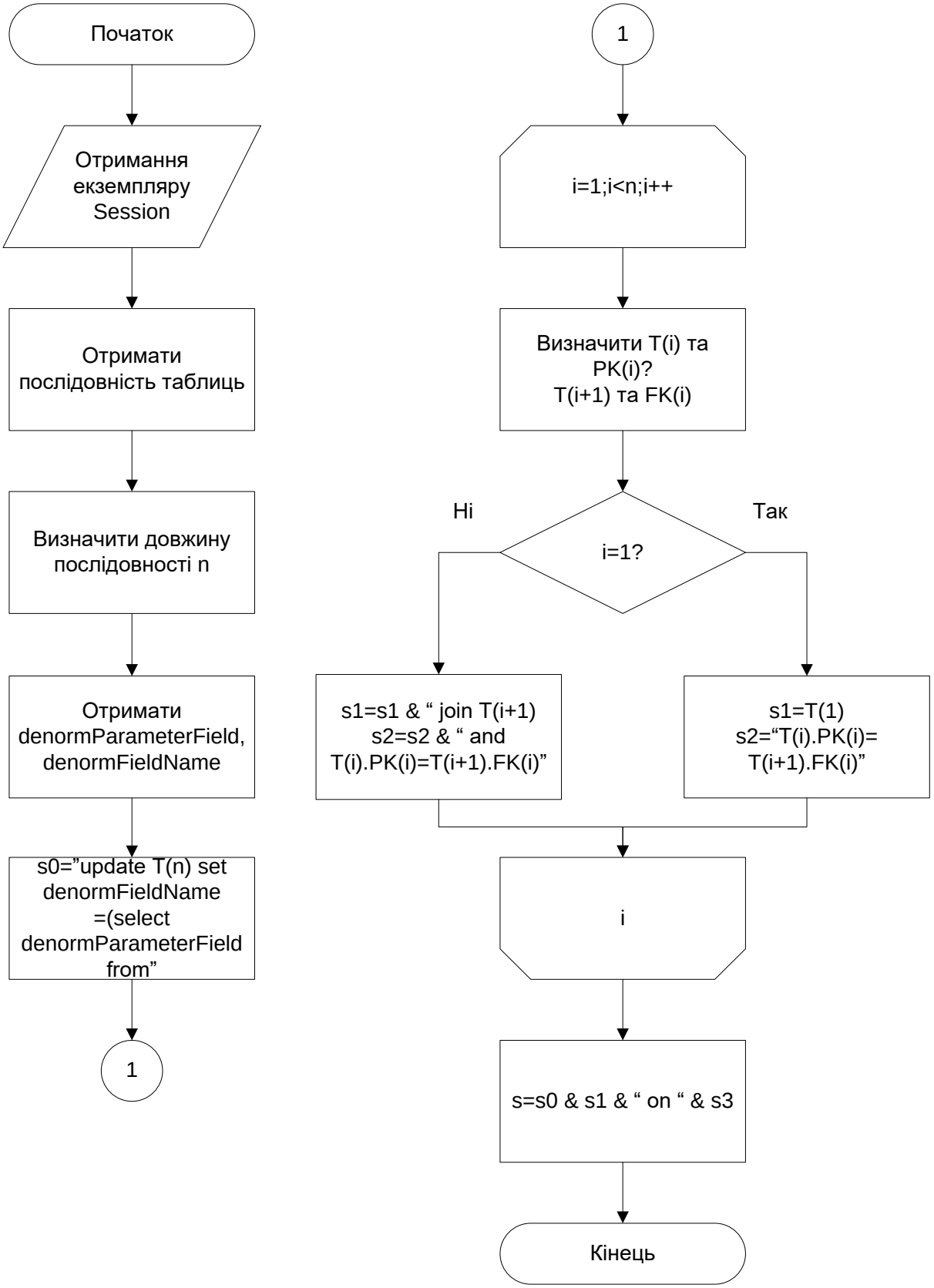


Рисунок 4.11 – Схема алгоритму створення запиту на зміну змісту денормалізованої БД

Друга та третя частини формуються у циклі, перебором усіх частин з послідовності таблиць, заданої під час визначення сесії.

4.5 Висновки до розділу

В розділі описано, яким чином спроєктовано архітектуру програми, показано фрагменти модулів. за допомогою яких реалізуються окремі частини функціоналу застосунку (з використанням різних мов та засобів програмування).

В розділі наданий детальний опис структури сервісної БД, яка зберігає дані, необхідні для автоматизованого виконання денормалізації, а саме, дані для підключення до БД, послідовність таблиць у ієрархії взаємозв’язків для транзитивного перенесення атрибуту з надлишковими даними. Наявність сервісної БД дозволяє рознести в часі виконання різних етапів підготовки денормалізаціїї.

Надана схема алгоритму для автоматизованого формування тексту запиту, за допомогою якого можливо внести дані в доданий атрибут з надлишковими даними.

5 РЕАЛІЗАЦІЯ ПРОГРАМИ ДЛЯ ЗМІНИ ЗМІСТУ ТАБЛИЦЬ БАЗИ ДАНИХ ДЛЯ РЕАЛІЗАЦІЇ ДЕНОРМАЛІЗАЦІЇ

5.1 Опис класів

На основі діаграм співробітництва розроблено структуру класів. Нижче наведені специфікації для окремих класів.

Клас Controller призначений для управління діями системи (рис.5.1).

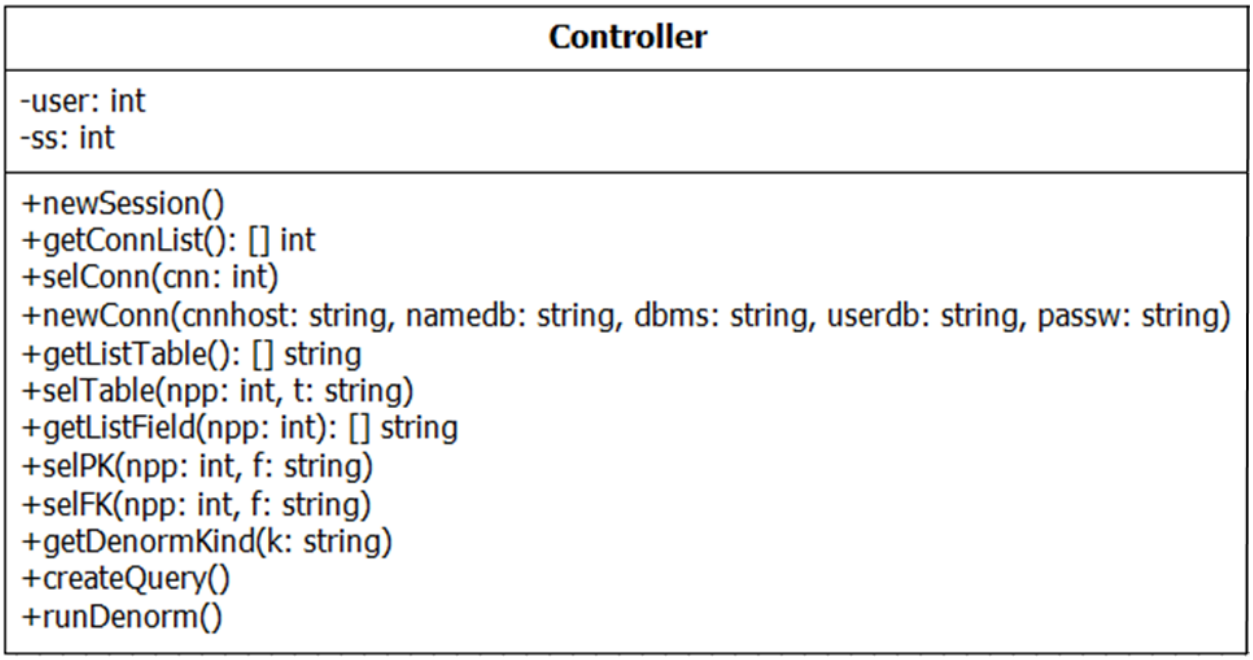


Рисунок 5.1 – Структура класу Controller

Клас Controller містить такі атрибути:

user – має тип int, призначений для збереження ідентифікатору поточного користувача;

ss – має тип int, призначений для збереження ідентифікатору іiЖ штепоточної сесії користувача.

Клас Controller має наступні методи:

newSession() – створює нову сесію для користувача з ідентифікатором user; вхідні параметри - немає; значення, яке повертається – немає;

getConnList() - запитує список існуючих підключень до БД для користувача з ідентифікатором user; вхідні параметри - немає; значення, яке повертається – масив ідентифікаторів підключень, які були створені користувачем з ідентифікатором user (array of int);

selConn(cnn) – передає обране одне з підключень з масиву, який сформований за допомогою метода getConnList(); вхідні параметри - cnn (int)- ідентифікатор підключення; значення, яке повертається – немає;

newConn(cnnhost, namedb, dbms, userdb, passw) – створює нове підключеннння для користувача з ідентифікатором user; вхідні параметри - cnnhost (string) – хост для БД, namedb (string) – назва БД, dbms (string) – СКБД, userdb (string) – ім’я користувача, passw (string) – пароль для доступу до БД; значення, яке повертається – ідентифікатор нового підключення (int);

getListTable() – запитує список таблиць з підключення сесії ss; вхідні параметри - немає; значення, яке повертається – масив назв таблиць (array of string);

selTable(npp, t) – передає обрану одну з таблиць з масиву, який сформований за допомогою метода getListTable (); вхідні параметри - npp (int) – номер у послідовності таблиць, t (string) – назва таблиці; значення, яке повертається – немає;

getListField(npp) – запитує список атрибутів таблиці, яка відповідає номеру npp в послідовності таблиць; вхідні параметри – npp (int) – номер у послідовності таблиць; значення, яке повертається – масив назв атрибутів(array of string);

selPK(npp,f) – передає обраний один з атрибутів з масиву, який сформований за допомогою метода getListField() для визначення батьківського ключа у зв’язку таблиць npp та (npp+1); вхідні параметри - npp (int) – номер у послідовності таблиць, f (string) – назва атрибуту; значення, яке повертається – немає;

selFK(npp, f) – передає обраний один з атрибутів з масиву, який сформований за допомогою метода getListField() для визначення дочірнього

ключа у зв’язку таблиць npp та (npp+1); вхідні параметри - npp (int) – номер у послідовності таблиць, f (string) – назва атрибуту; значення, яке повертається – немає;

getDenormKind(k) – визначає тип денормалізації; вхідні параметри - k (string) – тип денормалізації, обраний користувачем, значення down відповідає нисхідній денормалізації, значення up - висхідній; значення, яке повертається – немає;

createQuery() – запитує створення запитів для реалізації денормалізації; вхідні параметри – немає; значення, яке повертається – немає;

runDenorm() – запускає реалізацію денормалізації; вхідні параметри – немає; значення, яке повертається – немає.

Клас Session призначений для роботи з сесією користувача (рис.5.2).

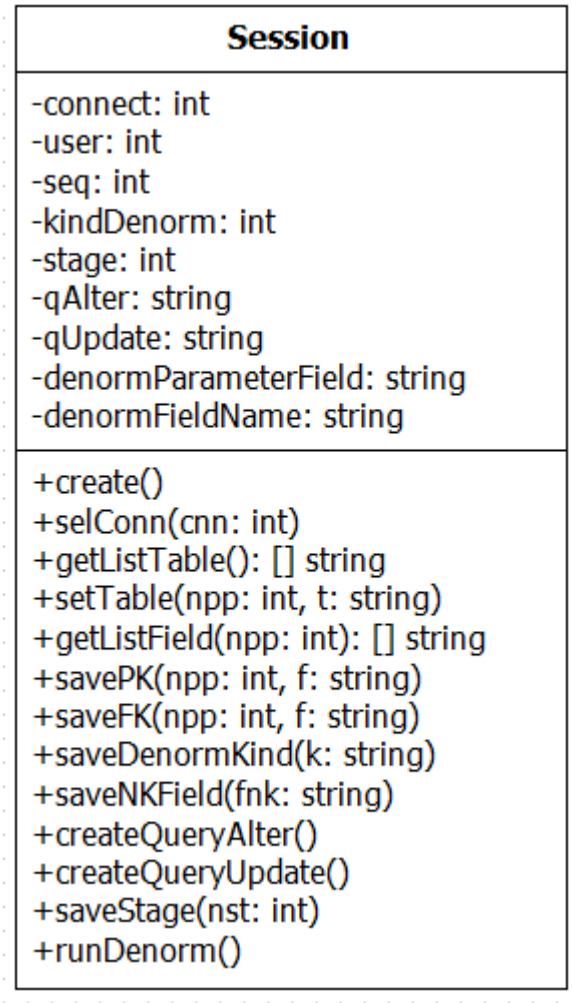


Рисунок 5.2 – Структура класу Session

Клас Session містить такі атрибути:

connect – має тип int, призначений для збереження ідентифікатору підключення до БД для поточної сесії;

user – має тип int, призначений для збереження ідентифікатору користувача. який створив поточну сесію;

seq – має тип int, призначений для збереження ідентифікатору послідовності таблиць, яка відповідає ієрархії денормалізації для поточної сесії;

kindDenorm – має тип string, визначає тип денормалізації;

stage – має тип int, визначає стадію, на якій знаходиться сесія;

qAlter – має тип string, визначає запит на зміну структури БД відповідно денормалізації поточної сесії;

qUpdate – має тип string, визначає запит на зміну даних денормалізованої БД;

denormParameterField – має тип string, визначає правило для створення денормалізованого атрибуту, відповідає назві неключового атрибуту з першої таблиці в послідовності, який дублюється у останній таблиці, у випадку нисхідної денормалізації або виразу, за яким формується агрегатне значення у випадку висхідної денормалізації;

denormFieldName – має тип string, визначає назву денормалізованого атрибуту.

Клас Session має наступні методи:

setConn(cnn) – встановлює підключення для сесії; вхідні параметри - cnn (int) – ідентифікатор підключення; значення, яке повертається – немає;

getListTable() – запитує список таблиць з підключення сесії ss; вхідні параметри - немає; значення, яке повертається – масив назв таблиць (array of string);

setTable(npp, t) – фіксує таблицю з заданим номером у послідовності таблиць; вхідні параметри - npp (int) – номер у послідовності таблиць, t (string) – назва таблиці; значення, яке повертається – немає;

getListField(npp) – запитує список атрибутів таблиці, яка відповідає номеру npp в послідовності таблиць; вхідні параметри – npp (int) – номер у послідовності таблиць; значення, яке повертається – масив назв атрибутів(array of string);

savePK(npp,f) – фіксує батьківський ключ у зв'язку таблиць npp та (npp+1); вхідні параметри – npp (int) – номер у послідовності таблиць, f (string) – назва атрибуту; значення, яке повертається – немає;

saveFK(npp,f) – фіксує дочірній ключ у зв'язку таблиць npp та (npp+1); вхідні параметри – npp (int) – номер у послідовності таблиць, f (string) – назва атрибуту; значення, яке повертається – немає;

saveDenormKind(k) – фіксує тип денормалізації; вхідні параметри - k (string) – тип денормалізації, обраний користувачем, значення down відповідає нисхідній денормалізації, значення up - висхідній; значення, яке повертається – немає;

saveNKField(fnk) – фіксує денормалізований атрибут; вхідні параметри – fnk (string) – відповідає назві неключового атрибуту з першої таблиці в послідовності, який дублюється у останній таблиці, у випадку нисхідної денормалізації або виразу, за яким формується агрегатне значення у випадку висхідної денормалізації;

createQueryAlter() – запитує створення запиту на зміну структури БД; вхідні параметри – немає; значення, яке повертається – немає;

createQueryUpdate() – запитує створення запиту на зміну даних в денормалізованій БД; вхідні параметри – немає; значення, яке повертається – немає;

saveStage(nst) – фіксує стадію виконання денормалізації; вхідні параметри – nst (int) – номер стадії; значення, яке повертається – немає;

runDenorm() – запускає реалізацію денормалізації; вхідні параметри – немає; значення, яке повертається – немає.

Клас SeqTable призначений для роботи з сесією користувача (рис.5.3).

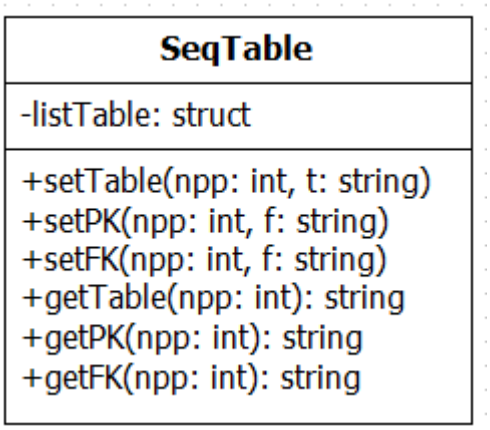


Рисунок 5.3 – Структура класу Session

Клас SeqTable містить такі атрибути:

listTable – має тип array of struct, призначений для збереження послідовності таблиць: стуктура містить поля {npp (int); t (string); pk (string): fk (string)}.

Клас SeqTable має наступні методи:

setTable(npp, t) – фіксує таблицю з заданим номером у послідовності таблиць; вхідні параметри - npp (int) – номер у послідовності таблиць, t (string) – назва таблиці; значення, яке повертається – немає;

setPK(npp, f) – фіксує батьківський ключ у зв'язку таблиць npp та (npp+1); вхідні параметри – npp (int) – номер у послідовності таблиць, f (string) – назва атрибуту; значення, яке повертається – немає;

setFK(npp, f) – фіксує дочірній ключ у зв'язку таблиць npp та (npp+1); вхідні параметри – npp (int) – номер у послідовності таблиць, f (string) – назва атрибуту; значення, яке повертається – немає;

getTable(npp) – визначає назву таблиці за заданим номером у послідовності; вхідні параметри – npp (int) – номер у послідовності таблиць,; значення, яке повертається – t (string) – назва таблиці;

getPK(npp) – визначає батьківський ключ у зв'язку таблиць npp та (npp+1); вхідні параметри – npp (int) – номер у послідовності таблиць,; значення, яке повертається – t (string) – назва атрибуту;

getFK(npp) – визначає дочірній ключ у зв’язку таблиць npp та (npp+1); вхідні параметри – npp (int) – номер у послідовності таблиць,; значення, яке повертається – t (string) – назва атрибуту.

5.2 Опис інтерфейсу програми

Дані про схему денормалізації належать окремому користувачу та не можуть використовуватися іншими користувачами. Відповідно для доступу до функціоналу програми необхідно зареєструватися та потім кожного разу авторизовуватися.

Зовнішній вигляд вікна при відкритті програми, до авторизації, показаний на рис. 5.4.

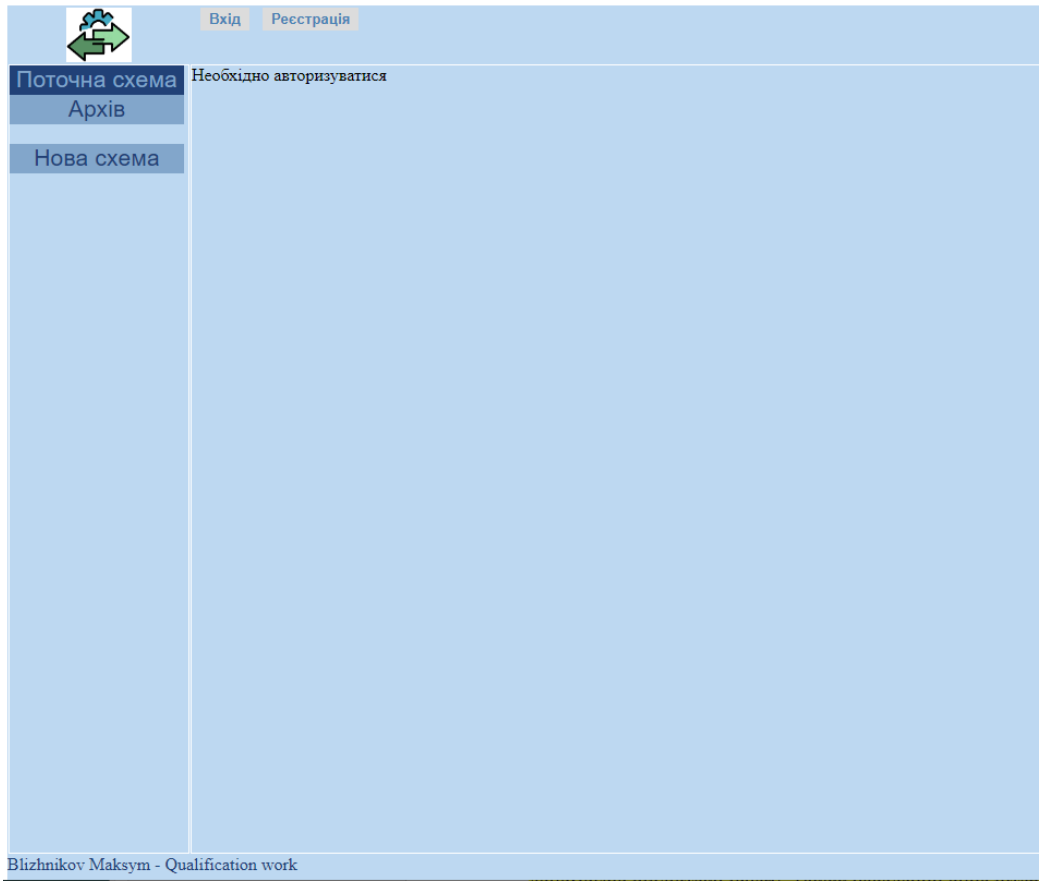


Рисунок 5.4 – Головне вікно програми прі відкритті

Зареєструватися може будь-який користувач, його дані ніяким чином не будуть перетинатися з даними інших користувачів.

Для реєстрації необхідно натиснути кнопку «Реєстрація», в головному вікні з’явиться форма для внесення імені, логіну та паролю нового користувача (рис.5.5).

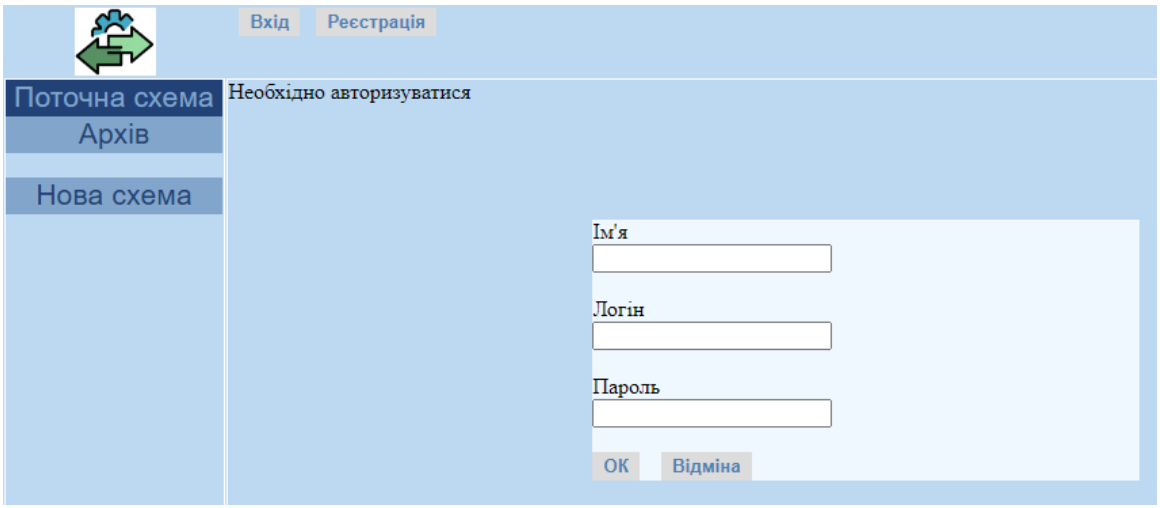


Рисунок 5.5 – Реєстрація користувача

Після збереження даних (кнопка «ОК») зареєстрований користувач стає поточним. В подальшому можна використовувати збережений логін та пароль для авторизації по кнопці «Вхід» (рис.5.6).

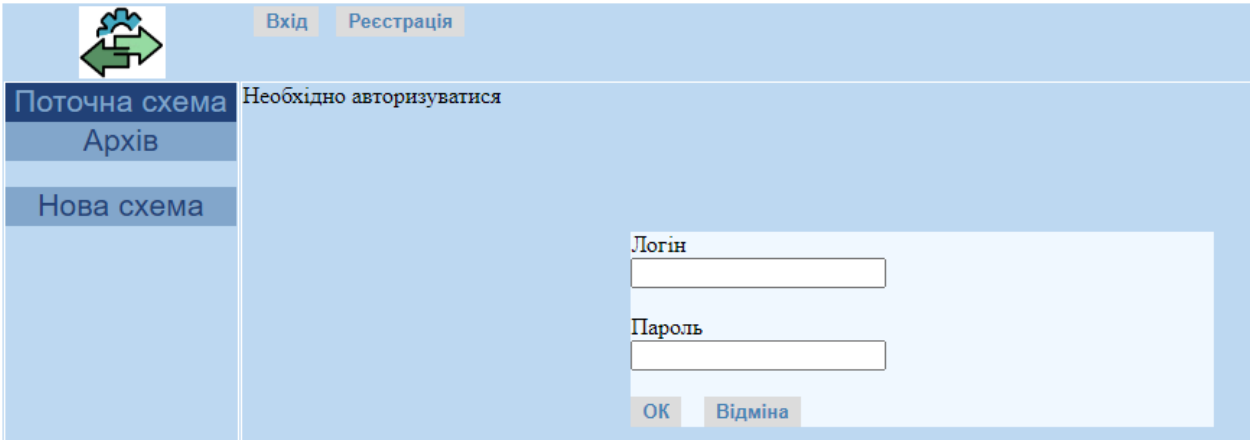


Рисунок 5.6 – Авторизація користувача

Після авторизації, у випадку внесення правильних логіну та паролю зареєстрованого раніше користувача програма демонструє ім’я поточного користувача (рис. 5.7).

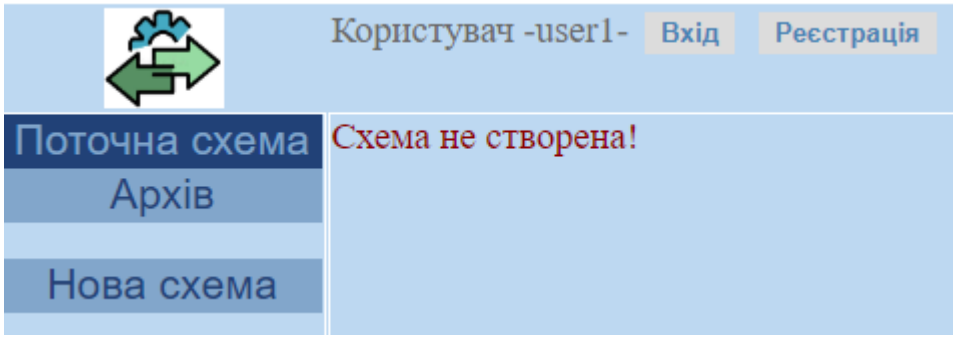


Рисунок 5.7 – Авторизація користувача

Для створення схеми необхідно натиснути кнопку «Нова схема», на екрані з’явиться інтерфейс для визначення складових, необхідних для проведення денормалізації (рис.5.8). Після цього необхідно послідовно сформувати складові в наступному порядку:

1) задати БД, структура якої має бути змінена;

2) задати послідовність таблиць, які утворюють ієрархію зв’язків для створення надлишкових даних;

3) задати правила зв’язків між таблицями, а саме вказати назви полів, через які відбувається зв’язок (первинний ключ – зовнішній ключ);

4) вид денормалізації, назву доданого атрибуту, який містить надлишкові дані та назву атрибуту, значення якого будуть дублюватися (нисхідна денормалізація) або вираз, за яким будуть формуватися агреговані дані (висхідна денормалізація).

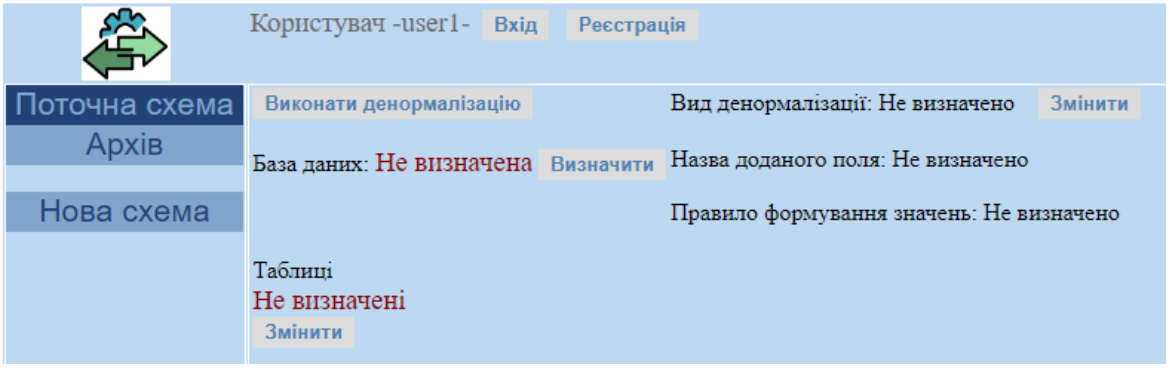


Рисунок 5.8 – Вікно для формування складових проведення денормалізації

Для визначення БД необхідно натиснути кнопку «Визначити» поряд з заголовком База даних.

Для визначення ієрархії таблицю необхідно натиснути кнопку «Змінити» поряд з заголовком Таблиці. Відкриється вікно (рис. 5.9), на якому міститься список таблиць, які є у вказаній БД, кнопка для перенесення таблиці у ієрархію для денормалізації, поле для вказівки номеру рівня фєрархії та список вже доданих таблиць.

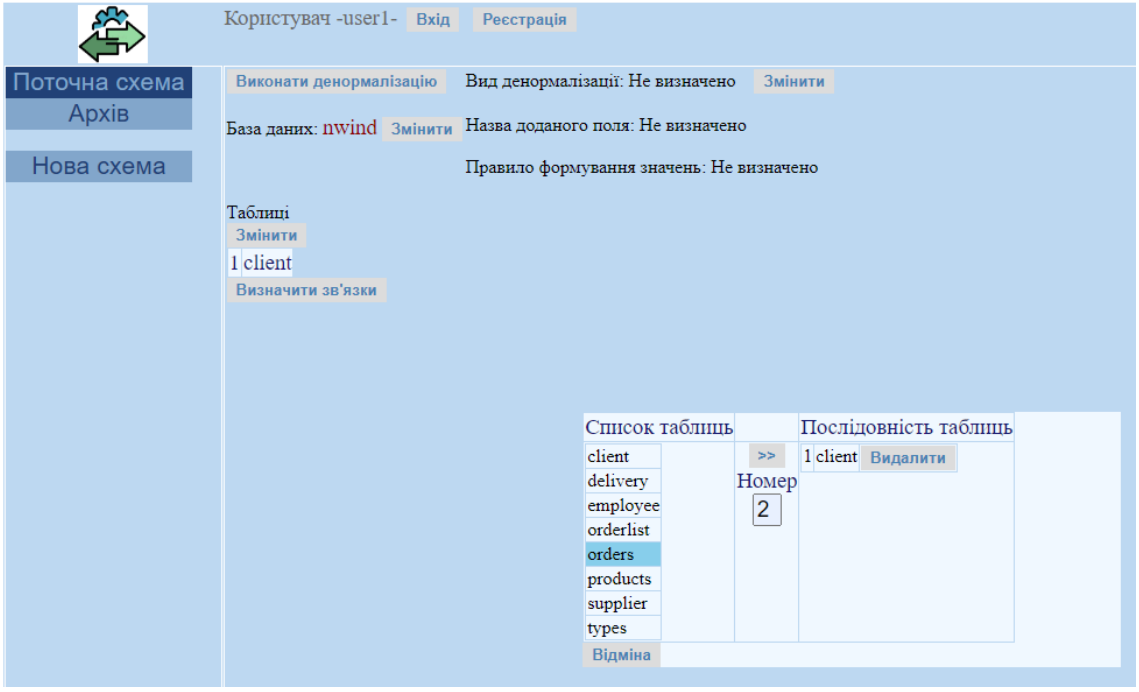


Рисунок 5.9 – Визначення ієрархії таблиць

Створена ієрархія відображається у головному вікні (рис.5.10) після заголовку Таблиці.

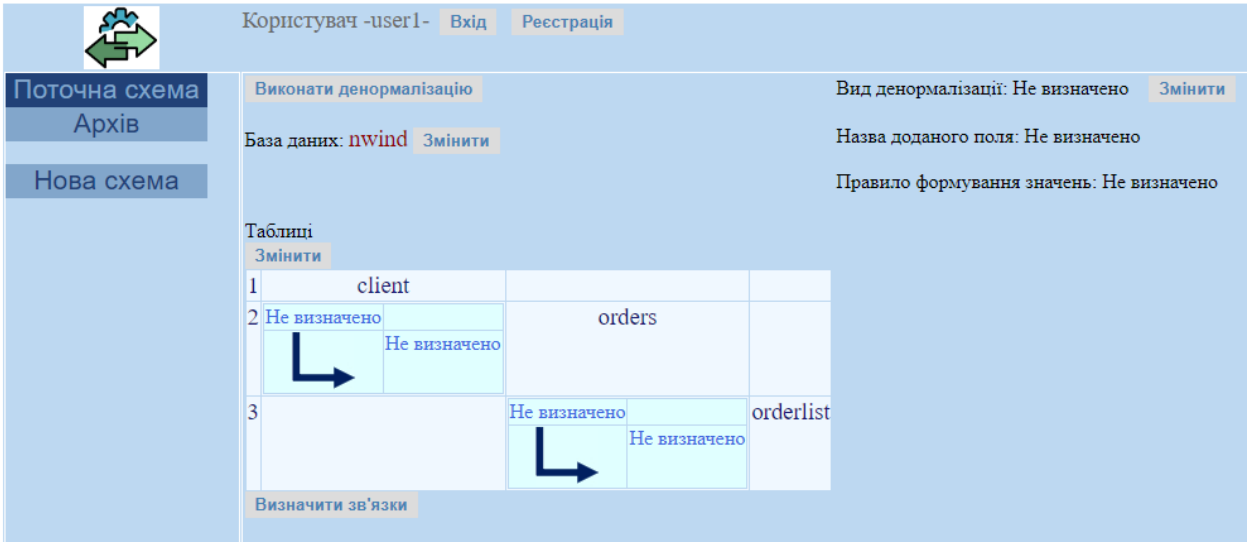


Рисунок 5.10 – Відображення сформованої ієрархії таблиць

Для визначення зв’язків між таблицями необхідно натиснути кнопку «Визначити зв’язки», після натискання у відповідному вікні (рис. 5.11) міститься список пар таблиць заданої ієрархії.

Після обрання пари кліком по списку у вікні відображається список полів батьківської таблиці та список полів підлеглої таблиці.

При обранні поля з боку батьківської таблиці кліком по списку назва поля потрапляє в поле Первинний ключ. При обранні поля з боку підлеглої таблиці кліком по списку назва поля потрапляє в поле Зовнішній ключ.

Для збереження зв’язку необхідно натиснути кнопку «ОК».

Збережені зв’язки відображаються в частині вікна з заголовком таблиці (рис. 5.12).

На наступному етапі необхідно встановити вид денормалізації. Для цього необхідно натиснути кнопку «Змінити» в групі полів «Види денормалізації».

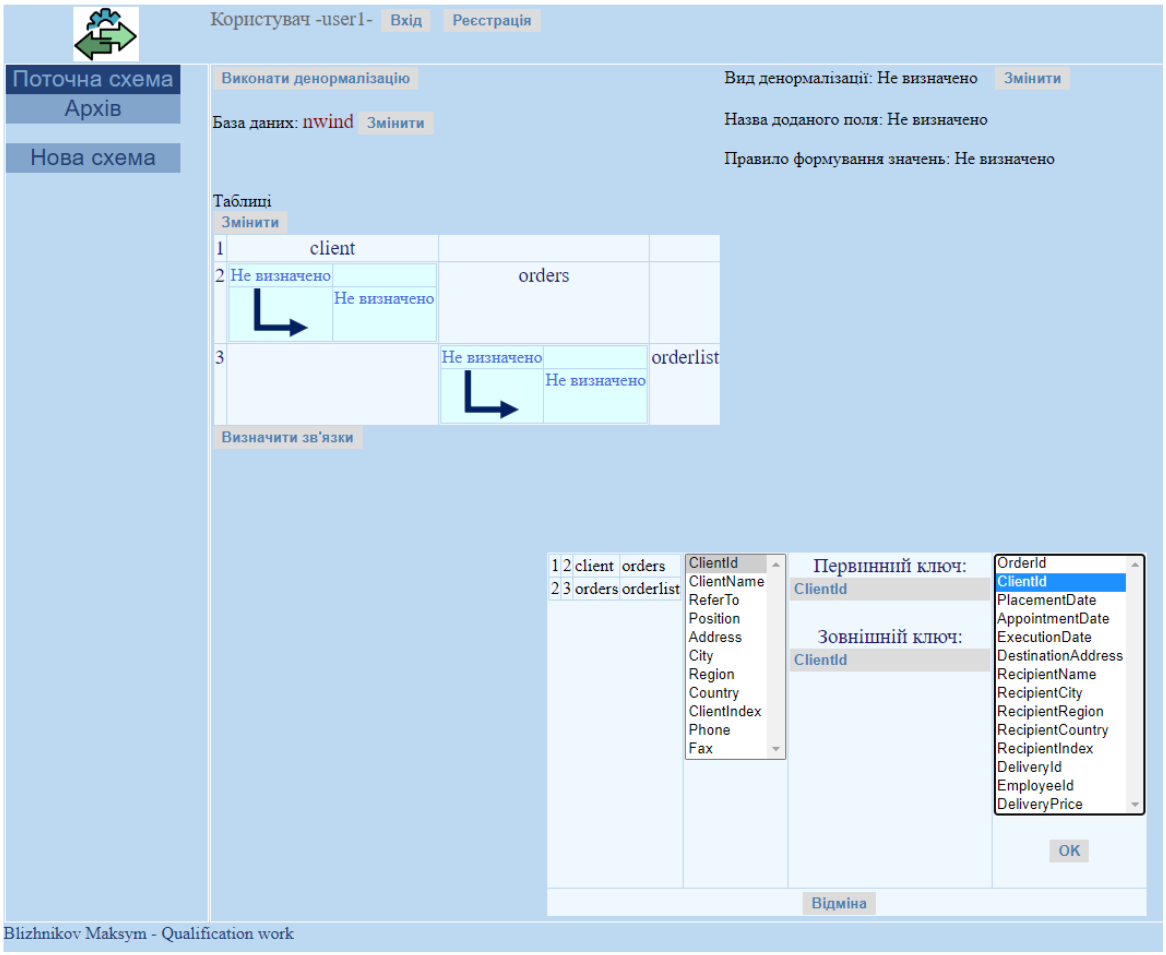


Рисунок 5.11 – Визначення зв’язків між таблицями

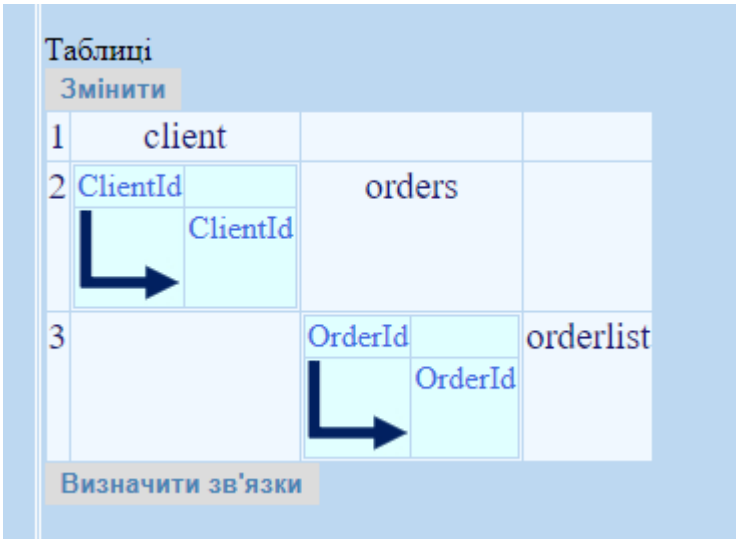
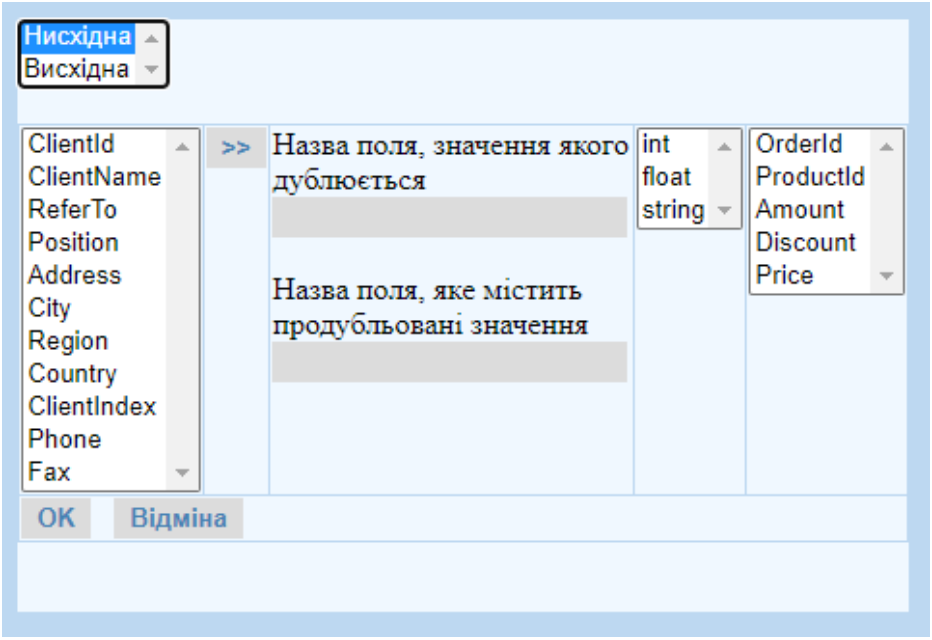
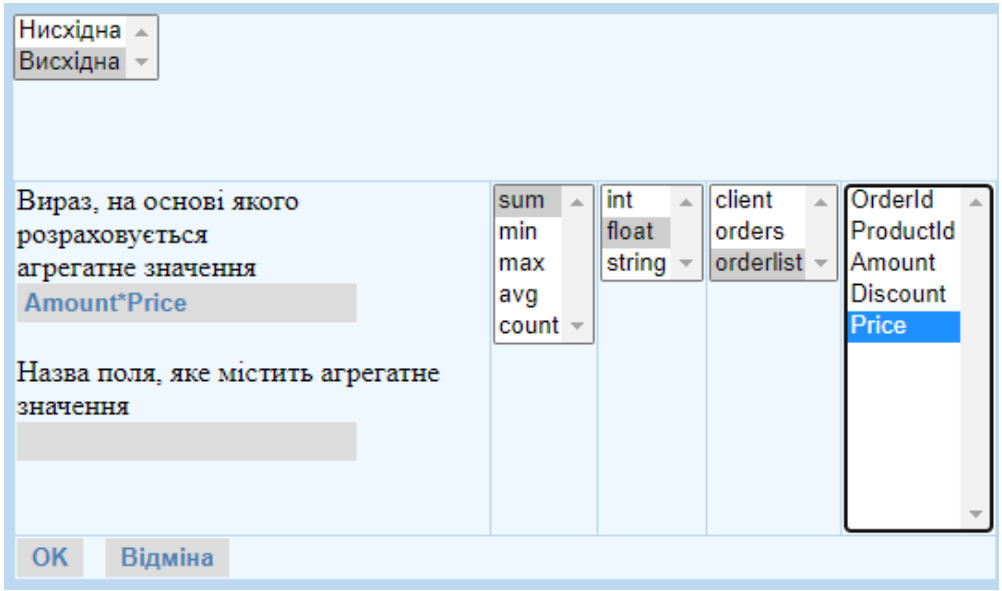


Рисунок 5.11 – Відображення заданих зв’язків між таблицями

Відкриється вікно для встановлення виду денормалізації та полів, необхідних для виконання реструктуризації. Після обрання виду денормалізації вікно змінює склад в залежності від обраного виду. Для нисхідної денормалізації вікно має вид, як показано на рис. 5.12 (а), а висхідної денормалізації - як показано на рис. 5.12 (б).



(а)



(б)

Рисунок 5.12 – Завдання виду денормалізації: (а) нисхідна; (б) висхідна

Для нисхідного виду формується список полів таблиці, яка визначена з номером у ієрархії таблиць. Необхідно обрати поле та за допомогою кнопки «>>» перемістити його в поле «Назва поля, значення якого дублюється». Також необхідно вказати тип даних для поля та назву поля, яке буде містити продубльовані значення. Для довідки, щоб запобігти повторення назви поля, в правій частині вікна розміщено список полів таблиці, яка визначена останньою у ієрархії.

Для висхідного виду необхідно сформувати вираз, на основі якого будуть обчислюватися агрегатні дані. Для спрощення формування виразу в правій частині вікна розташовано список таблиць з ієрархії. При обранні таблиці в списку формується список полів для обраної таблиці. Після кліку по списку обране поле поміщається в поле для формування виразу. Також необхідно обрати вид агрегуючої функції та тип даних для результуючого значення.

Для збереження вказаних даних необхідно натиснути кнопку «ОК».

Після послідовного заповнення необхідних складових можна виконати реструктуризацію БД, натиснувши кнопку «Виконати денормалізацію».

5.3 Висновки до розділу

В розділі описана структура класів, за допомогою методів яких реалізується створення компонентів вхідних даних для реструктуризації БД та безпосередньо проведення денормалізації. Детально описаний зовнішній вигляд вікон інтерфейсу програми, за допомогою яких виконується перелічені функції.

6 ТЕСТУВАННЯ ПРОГРАМИ ДЛЯ ЗМІНИ ЗМІСТУ ТАБЛИЦЬ БАЗИ ДАНИХ ДЛЯ РЕАЛІЗАЦІЇ ДЕНОРМАЛІЗАЦІЇ

6.1 Опис структури бази даних для проведення тестування

Для проведення використовується база даних testdenorm, структура якої може бути створена за скріптом, показаним на рис.6.1

```
-- Структура таблиці `t1`
CREATE TABLE `t1` (
  `id1` int(11) NOT NULL,
  `npk1_1` int(11) NOT NULL,
  `npk2_2` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
-- Дамп даних таблиці `t1`
INSERT INTO `t1` (`id1`, `npk1_1`, `npk2_2`) VALUES
(1, 1100, 345),
(2, 2000, 910);
-- Структура таблиці `t2`
CREATE TABLE `t2` (
  `id2` int(11) NOT NULL,
  `npk1_2` int(11) NOT NULL,
  `npk2_2` int(11) NOT NULL,
  `fk2` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
-- Дамп даних таблиці `t2`
INSERT INTO `t2` (`id2`, `npk1_2`, `npk2_2`, `fk2`) VALUES
(11, 11, 30, 1),
(22, 22, 50, 2);
-- Структура таблиці `t3`
CREATE TABLE `t3` (
  `id3` int(11) NOT NULL,
  `npk1_3` int(11) NOT NULL,
  `npk2_3` int(11) NOT NULL,
  `fk3` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
-- Дамп даних таблиці `t3`
INSERT INTO `t3` (`id3`, `npk1_3`, `npk2_3`, `fk3`) VALUES
(111, 120, 10, 22),
(222, 500, 20, 22);
-- Індекси таблиці `t1`
ALTER TABLE `t1`
  ADD PRIMARY KEY (`id1`);
-- Індекси таблиці `t2`
ALTER TABLE `t2`
  ADD PRIMARY KEY (`id2`);
-- Індекси таблиці `t3`
ALTER TABLE `t3`
  ADD PRIMARY KEY (`id3`);
```

Рисунок 6.1 – Створення БД для проведення тестування

БД містить три таблиці, пов’язаних між собою: t1 – t2, t2 – t3.

Кожна таблиця має первинний ключ. та два неключових поля, таблиці t2 та t3 містять також поля-зовнішні ключи.

6.2 Опис тестування

В табл. 6.1 надано чек-аркуш для функцій реєстрації та авторизації користувача.

Таблиця 6.1 - Чек-аркуш для перевірки реєстрації та авторизації користувача

Крок	Опис	Очікуваний результат
1	Запустити програму	На екрані з’явилися вікно, яке містить кнопку «Реєстрація»
2	Натиснути кнопку «Реєстрація»	На екрані з’явилося вікно з полями для введення імені, логіну та паролю користувача та кнопки «ОК» та «Відміна»
3	Натиснути кнопку «Відміна»	З екрану зникло вікно з полями для введення імені, логіну та паролю користувача
4	Натиснути кнопку «Реєстрація»	На екрані з’явилося вікно з полями для введення імені, логіну та паролю користувача та кнопки «ОК» та «Відміна»
5	Натиснути кнопку «ОК»	З’явилося попередження, що необхідно ввести дані

Продовження табл. 6.1

Крок	Опис	Очікуваний результат
6	Заповнити поля імені, логіну та паролю («Username», «User», «User2022» відповідно) та натиснути кнопку «ОК»	На екрані з’явилося ім’я введеного користувача Username
7	Натиснути кнопку «Реєстрація»	На екрані з’явилося вікно з полями для введення імені, логіну та паролю користувача та кнопки «ОК» та «Відміна»
8	Заповнити поля імені, логіну та паролю («Username1», «User1», «User1_2022» відповідно) та натиснути кнопку «ОК»	На екрані з’явилося ім’я введеного користувача Username1
9	Натиснути кнопку «Вхід»	На екрані з’явилося вікно з полями для введення логіну та паролю
10	Заповнити поля імені, логіну та паролю «User5», «User5_2022» відповідно) та натиснути кнопку «ОК»	На екрані з’явилося повідомлення. що користувач не знайдений
11	Заповнити поля логіну та паролю («User», «User2022» відповідно) та натиснути кнопку «ОК»	На екрані з’явилося ім’я введеного користувача Username

В табл. 6.2 надано чек-аркуш для функції створення схеми денормалізації (користувач з логіном User має бути авторизований).

Таблиця 6.2 - Чек-аркуш для перевірки створення схеми денормалізації

Крок	Опис	Очікуваний результат
1	Натиснути кнопку «Нова схема»	На екрані з’явилися вікно з кнопками для формування компонент реструктуризації БД.

В табл. 6.3 надано чек-аркуш для функції визначення компоненту БД (користувач з логіном User має бути авторизований, схема створена).

Таблиця 6.3 - Чек-аркуш для перевірки визначення компоненту БД

Крок	Опис	Очікуваний результат
1	Натиснути кнопку «Визначити» для розділу «База даних»	На екрані з’явилося вікно для внесення назви БД та даних для підключення до неї
2	Внести назву БД testdenorm, хост localhost та натиснути кнопку «ОК»	Назва БД відображена в розділі «База даних»

В табл. 6.4 надано чек-аркуш для функції визначення компоненту ієрархії таблиць (користувач з логіном User має бути авторизований, схема створена, БД визначена).

Таблиця 6.4 - Чек-аркуш для перевірки визначення компоненту ієрархії таблиць

Крок	Опис	Очікуваний результат
1	Натиснути кнопку «Змінити» для розділу «Таблиці»	З’явилося вікно зі списком таблиць з БД testdenorm (t1, t2, t3)
2	Ввести номер 1 та обрати таблицю t1, натиснути кнопку «>>»	В послідовності таблиць з’явилася назва t1, у розділі «Таблиці» також відображена t1
3	Ввести номер 2 та обрати таблицю t2, натиснути кнопку «>>»	В послідовності таблиць з’явилася назва t2, у розділі «Таблиці» також відображена t2
4	Ввести номер 3 та обрати таблицю t3, натиснути кнопку «>>»	В послідовності таблиць з’явилася назва t3, у розділі «Таблиці» також відображена t3

В табл. 6.5 надано чек-аркуш для функції визначення компоненту зв’язків між таблицями (користувач з логіном User має бути авторизований, схема створена, БД визначена, ієрархія таблиць визначена).

Таблиця 6.5 - Чек-аркуш для перевірки визначення компоненту ієрархії таблиць

Крок	Опис	Очікуваний результат
1	Натиснути кнопку «Визначити зв’язки»	З’явилося вікно з парами таблиць t1 – t2 та t2 – t3

Продовження табл. 6.5

Крок	Опис	Очікуваний результат
2	Обрати пару таблиць t1 – t2	З’явилися списки полів таблиць t1 та t2
3	Обрати поле id1 з лівого списку	Назва поля id1 з’явилася у полі Первинний ключ
4	Обрати поле fk2 з правого списку	Назва поля fk2 з’явилася у полі Зовнішній ключ
5	Натиснути кнопку «ОК»	Пара полів id1- fk2 відображена в розділі Таблиці
6	Обрати пару таблиць t2 – t3	З’явилися списки полів таблиць t2 та t3
7	Обрати поле id2 з лівого списку	Назва поля id2 з’явилася у полі Первинний ключ
8	Обрати поле fk3 з правого списку	Назва поля fk3 з’явилася у полі Зовнішній ключ
9	Натиснути кнопку «ОК»	Пара полів id2- fk3 відображена в розділі Таблиці

В табл. 6.6 надано чек-аркуш для функції визначення компоненту завдання виду денормалізації (користувач з логіном User має бути авторизований, схема створена, БД визначена, ієрархія таблиць визначена, зв’язки між таблицями визначені).

Таблиця 6.6 - Чек-аркуш для перевірки визначення компоненту ієрархії таблиць

Крок	Опис	Очікуваний результат
1	Натиснути кнопку «Змінити» для розділу «Вид денормалізації»	На екрані з’явилися вікно можливими видами денормалізації.
2	Обрати у списку Нисхідна	На екрані з’явилося вікно зі списками полів таблиць t1 та t3
3	Обрати поле npk1_1 з лівого списку	Назва поля npk1_1 відображена в полі «Назва поля, значення якого дублюється»
4	Ввести назву npk_from1 в поле «Назва поля, яке містить продубльовані значення», обрати тип даних int та натиснути кнопку «ОК»	Введені назви полів відображені у розділі «Вид денормалізації

В табл. 6.7 надано чек-аркуш для функції реструктуризації (користувач з логіном User має бути авторизований, всі компоненти визначені).

Таблиця 6.7 - Чек-аркуш для перевірки виконання реструктуризації

Крок	Опис	Очікуваний результат
1	Натиснути кнопку «Виконати денормалізацію»	В таблиці t3 з’явилося поле npk_from1, значення дорівнює 910

За чек-аркушами проведено тестування. визначено, що всі функції виконуються без помилок.

6.3 Проведення експерименту

Проведено наступний експеримент для перевірки досягнення мети - зменшення часу на зміну змісту таблиць БД, до яких застосовується багаторівнева нисхідна або висхідна денормалізація за рахунок автоматизації побудови запитів для оновлення даних.

Умови експерименту наступні.

Для 5 осіб запропоновано виконати денормалізацію (3 таблиці в ієрархії) для тестової БД та бази даних, створеної на основі учбової БД Борей MicroSoft.

Передбачено створити додаткові поля за допомогою запитів на SQL, написати запити для заповнення створених полів та виконати ці запити.

Дії проведені для двох видів денормалізації (нисхідна та висхідна).

Визначено середній час, необхідний для виконання перелічений дій.

Не враховано час на ознайомлення з структурою БД, яких приймали участь в експерименті. Також не враховано час виконання запитів на формування даних в полях, які містять надлишкову інформацію, оскільки час виконання цих запитів не залежить від користувача. а залежить від розміру таблиць.

Для тих же осіб виконати дії реєстрації користувача, введення компонентів реструктуризації та власно проведення денормалізації за допомогою розробленої програми. Створено по дві схема. окремо для нисхідної та висхідної денормалізації відповідно.

Визначено середній час, необхідний для виконання перелічений дій.

Результати експерименту показані в табл. 6.8.

Таблиця 6.8 – Результати експерименту

Середній час виконання дій, хв	Ручне формування та виконання запитів		Автоматизована денормалізація	
	Нисхідна	Висхідна	Нисхідна	Висхідна
	11,5	12,5	2,7	3,6

Результати експерименту показали скорочення часу проведення дій з нормалізації за рахунок автоматизованого формування запитів.

6.4 Висновки до розділу

В розділі наведені чек-аркуші для проведення тестування програми зміни змісту денормалізованих таблиць. Представлені чек-аркуші дозволяють в повному обсязі перевірити правильну роботу всіх складових програми. Наведені дані експерименту, який показав значне скорочення часу на проведення денормалізації, з урахуванням наявності даних в БД і необхідності формувати запити, які вимагають залучення кількох пов’язаних таблиць.

ВИСНОВКИ

В роботі проаналізовані можливі підвищення продуктивності баз даних. Вивчено та проаналізовано функції існуючих систем з використанням денормалізованих структур.

Обґрунтовано вибір варіантів денормалізації для автоматизації. Надане формалізоване представлення обраних варіантів з точки зору зміни структури БД та алгоритмів формування запитів для заповнення доданих атрибутів, які зберігають надлишкові дані.

Сформовано вимоги до програми для формування змісту доданих атрибутів, які зберігають надлишкові дані. Описані функціональні вимоги у вигляді варіантів використання. Для кожного варіанту використання наданий детальний сценарій. Також сформовані нефункціональні умови, які описують, як має працювати система. Розглянуто шляхи вирішення можливих міжсесійних конфліктів, які можуть привести до помилок під час виконання функцій в рамках однієї сесії.

Спроєктовано архітектуру програми, зокрема, структуру модулів, за допомогою яких реалізуються окремі частини функціоналу застосунку.

Розроблено структуру спеціальної сервісної БД, яка зберігає дані, необхідні для автоматизованого виконання денормалізації, а саме, дані для підключення до БД, послідовність таблиць у ієрархії взаємозв’язків для транзитивного перенесення атрибуту з надлишковими даними. Наявність сервісної БД дозволяє рознести в часі виконання різних етапів підготовки денормалізаціїї.

Розроблені алгоритми для виконання усіх функцій програми.

Сконструйована структура класів, за допомогою методів яких реалізується створення компонентів вхідних даних для реструктуризації БД та власне виконується проведення денормалізації.

Розроблено інтерфейс програми, за допомогою яких виконується її функції. Інтерфейс містить окремі вікна, які максимально спрощують отримання необхідних даних від користувача.

Складені чек-аркуші для перевірки працездатності програми відносно всіх її функцій.

Проведений експеримент, який підтвердив досягнення мети розробки. Показано, що час на проведення денормалізації скорочено в середньому в 3,8 рази за рахунок того, що програма надає максимально зручний інструмент для внесення вхідних даних та автоматизує створення тексту запитів та їх запуск.

Розробка представлена на XII міжнародній конференції студентів та молодих вчених «Сучасні Інформаційні Технології 2022» 19-20 травня 2022 р.

СПИСОК ЛІТЕРАТУРИ

1. Denormalization in Databases. URL: https://www.geeksforgeeks.org/denormalization-in-databases. – (дата звернення: 05.05.2022).

2. Teradata Documentation. Denormalization, Data Marts, and Data Warehouses. URL: https://docs.teradata.com/r/w4DJnG9u9GdDlXzsTXyItA/O5nTOgKYw168FKfcRspUsQ. – (дата звернення: 05.05.2022).

3. The Role of Information Systems in Running the 21st Century Organization. URL: https://elearning.scranton.edu/resources/article/the-role-of-information-systems-in-increasing-productivity/. – (дата звернення: 05.09.2022).

4. 5 benefits of a fully integrated information system. URL: https://eventura.com/netsuite/five-benefits-fully-integrated-information-system/. – (дата звернення: 05.09.2022).

5. What Are Information Systems, and How Do They Benefit Business? URL: https://onlinemba.wsu.edu/blog/what-are-information-systems-and-how-do-they-benefit-business/ – (дата звернення: 05.09.2022).

6. How to Increase Database Performance – 6 Easy Tips. URL: https://www.dnsstuff.com/how-to-increase-database-performance. – (дата звернення: 05.09.2022).

7. How to Improve Database Performance? https://www.appdynamics.com/topics/how-to-improve-database-performance. – (дата звернення: 05.09.2022).

8. Грегорченко І Денормализація даних. URL: https://highload.today/denormalizatsiya-dannykh/. – (дата звернення: 15.08.2022).

9. Денормалізація як засіб підвищення продуктивності і її реалізація. URL: https://intellect.icu/denormalizatsiya-kak-sredstvo-povysheniya-proizvoditelnosti-i-ee-realizatsiya-7877. – (дата звернення: 15.08.2022).

10. Working with denormalized data at Nintex. URL: https://www.nintex.com/blog/working-with-denormalized-data-at-nintex. – (дата звернення: 05.05.2022).

11. Siebel EIM Does Not Update Denormalized Table Columns such as S_SRV_REQ_BU. URL: https://www.toolbox.com/tech/siebel/blogs/siebel-eim-does-not-update-denormalized-table-columns-such-as-s-srv-req-bu-072809. – (дата звернення: 05.05.2022).

12. Velykodniy S. S..Analysis and Synthesis of the Results of Complex Experimental Research on Reengineering of Open Cad Systems. Applied Aspects of Information Technology. 2019; Vol.2 No.3: 186–205. DOI: https://doi.org/10.15276/aait.03.2019.2

13. Kimball R., Caserta J. The data warehouse ETL toolkit: practical techniques for extracting, cleaning, conforming, and delivering data. – Canada: Wiley Publishing, Inc., 2004. – 491 p.

14. Kyte, T. Expert Oracle Database Architecture: 9i and 10g Programming Techniques and Solutions. - Apress, 2005. – 768 p.

15. SQL*Loader Concepts. URL: https://docs.oracle.com/cd/B19306_01/server.102/b14215/ldr_concepts.htm. – (дата звернення: 17.08.2022).

16. Wright G., Vaughan J. Denormalization. URL: https://www.techtarget.com/searchdatamanagement/definition/denormalization. – (дата звернення: 16.09.2022).

17. De-normalization in Database. URL: https://www.tutorialcup.com/dbms/denormalization.htm. – (дата звернення: 16.09.2022).

18. Non-functional Requirements: Examples, Types, How to Approach. URL: https://www.altexsoft.com/blog/non-functional-requirements. – (дата звернення: 16.09.2022).

19. UML - Interaction Diagrams. URL: https://www.tutorialspoint.com/uml/uml_interaction_diagram.htm. – (дата звернення: 16.09.2022).

20. Застосування UML (частина 2). Діаграма послідовності - sequence diagram. URL: https://dut.edu.ua/ua/news-1-626-7897-zastosuvannya-uml-chastina-2-diagrama-poslidovnosti---sequence-diagram_kafedra-kompyuternih-nauk-ta-informaciynih-tehnologiy. – (дата звернення: 16.09.2022).

21. Діаграми співробітництва. URL: https://studfile.net/preview/9153239/page:13. – (дата звернення: 10.10.2022).

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

Модуль index.php

```
<form id=formdenorm method=post>
<?
include_once ("init.php");

if(isset($_POST['user']))
	$user=$_POST['user'];
else
	$user=0;
if(isset($_POST['menu']))
	$menu=$_POST['menu'];
else
	$menu='current';

if(isset($_POST['session']))
	$session=$_POST['session'];
else{
	if($user==0)
		$session=0;
	else{
		$session=$csession->getCurrentSession($user);
	}
}
if($user>0)
	$session=$csession->getCurrentSession($user);

	if($session==0)
		$database=0;
	else{
		$database=$csession->getCurrentDatabase($session);
	}
if($menu=='current'){
	$ccurrentmenu='activemenu';
	$carchivemenu='menu';
}
else{
	$ccurrentmenu='menu';
	$carchivemenu='activemenu';
}
if($user==0)
	$usertext="";
else{
	$username=$cuser->getName($user);
	$usertext="<font class='user'>Користувач -".$username."-</font>";
}
$usertext.="  <input type=button class=button value='Вхід' onclick='getuser();'>
  <input type=button class=button value='Реєстрація' onclick='reg();'>";
?>
us<input type=input id=user  name=user value='<?=$user?>'>
mn<input type=input id=menu  name=menu value='<?=$menu?>'>
```

```
db<input type=input id=database  name=database value='<?=$database?>'>
ss<input type=input id=session  name=session value='<?=$session?>'><br><br>
<table><tr><td class=bg><center>
<img class=logo src='images/logo.jpg'>
</td><td class=bg><?=$usertext?></td></tr>
<tr><td><div class='allone'>
<input type=button class=<?=$ccurrentmenu?> value='Поточна схема' onclick='scheme();'><br>
<input type=button class=<?=$carchivemenu?> value='Архів' onclick='archive();'><br><br>
<input type=button class='menu' value='Нова схема' onclick='newscheme();'>
</div></td>
<td><div class='alltwo'>
<?if($user==0)
        echo 'Необхідно авторизуватися';
else{
        if($menu=='current'){
                include_once ("view/current.php");
        }
        else{
                include_once ("view/archive.php");
        }
}
include_once ("view/addupdate.php");
?>
</div></td></tr></table>
<div class=footer>
        Blizhnikov Maksym - Qualification work
</div>
</body>
</html>
```

Модуль current.php

```
<?
if($session==0)
        echo "<font class=message>Схема не створена!</font><br>";
else{
        if($database==0)
                $databasename="Не визначена";
        else
                $databasename=$csession->getNameDatabase($database);
?>

<table>
<tr>
<td>
<input type=button class=button value='Виконати денормалізацію'
onclick='executedenorm();'><br><br>
База даних: <font class=message><?=$databasename?></font>
<?
if($database==0)
        echo "<input type=button class=button value='Визначити' onclick='adddb();'>";
else{
        $state=$csession->getState($session);
        if($state==1)
                echo "<input type=button class=button value='Змінити' onclick='updatedb();'>";
}
```

```
?>
</td>
<td>
<?
$denorm=$csession->getDenorm($session);
if($stat==0) {
	$txt="Не визначено   
<input type=button class=button value='Змінити' onclick='changeDenorm();'>";
$addfield='Не визначено ';
$rule='Не визначено ';
}
else{
if($denorm==1)$txt='Нисхідна';
else $txt='Висхідна';
$addfield=$csession->getDenormField($session);
$rule=$csession->getAddField($session);
}
?>
<div>
Вид денормалізації: <?=$txt?><br><br>
Назва доданого поля: <?=$addfield?><br><br>
Правило формування значень: <?=$rule?><br><br>
</div>
</td></tr><tr>
<td>
Таблиці<br>
<?
$list=$cseqtable->getListTable($session);
if(count($list)>0){
	if($state<3)
		echo "<input type=button class=button value='Змінити' onclick='updatetable();'>";

	echo "<table class=seqtable>";
	foreach($list as $l){
		echo "<tr><td>".$l['n']."</td>";
			for($i=1;$i<$l['n']-1;$i++)
				echo "<td></td>";
		if($l['n']>1){
			echo "<td><table class='rule'>
			<tr><td>".$pk."</td><td></td></tr><tr><td><img src='../images/rule.jpg'
style=</td><td>".$fk."</td></tr>
			</table>
			</td>";
		}
			echo "<td>".$l['sequenceTableName']."</td>";
			for($i=1;$i<=count($list)-$l['n'];$i++)
				echo "<td></td>";
		$pk=$l['primaryAtribute'];if($pk==null)$pk='Не визначено';
		$fk=$l['foreignAtribute'];if($fk==null)$fk='Не визначено';

		echo "</tr>";
	}
	echo "</table>";
	echo '<input type=button class=button value="Визначити зв\'язки"
onclick="updaterule();">';
}
```

```
else {echo "<font class=message>Не визначені</font><br>";
echo "<input type=button class=button value='Змінити' onclick='updatetable();'>";
}

?>
</td>
<td>
</td>
</tr>
</table>
<?
}
?>
```

Модуль addupdate.php

```
<div id=inputuser class=addupdate>
Логін<br>
<input type=text id=inputlogin name=inputlogin class='' ><br><br>
Пароль<br>
<input type=password id=inputpassword name=inputpassword class='' ><br><br>
<input type=button class=button value='ОК' onclick='input();'>   
<input type=button class=button value='Відміна'
onclick='cancelinputuser();'>   
</div>

<div id=reg class=addupdate>
Ім'я<br>
<input type=text id=regusername name=regusername class='' ><br><br>
Логін<br>
<input type=text id=reglogin name=reglogin class='' ><br><br>
Пароль<br>
<input type=password id=regpassword name=regpassword class='' ><br><br>
<input type=button class=button value='ОК' onclick='adduser();'>   
<input type=button class=button value='Відміна' onclick='cancelreg();'>   
</div>

<?
if($database>0){
        $textdbname=$databasename;
        $textdbhost=$csession->getHostDatabase($database);
}
else{
        $textdbname='';
        $textdbhost='';
}
?>
<div id=db class=addupdate>
<input type=hidden id=databaseid name=databaseid class='' ><br>
Назва<br>
<input type=text id=dbname name=dbname class='' value=<?=$textdbname?>><br><br>
Хост<br>
<input type=text id=dbhost name=dbhost class='' value=<?=$textdbhost?>><br><br>
<input type=button class=button value='ОК' onclick='addupdatedb();'>   
<input type=button class=button value='Відміна' onclick='canceldb();'>   
</div>
```

```
<div id=seqtable class=addupdate>
<table class='seqtable'>
<tr><td>Список таблиць</td><td></td><td>Послідовність таблиць</td></tr>
<tr><td>
<input type=hidden id=selecttable name=selecttable value=''>
<?
$list=$csession->getListTable($database);
if(count($list)>0){
	echo "<table>";
	foreach($list as $l){
		echo "<tr><td id='t_".$l['Tables_in_'.$databasename]."'

	onclick=selecttable('".$l['Tables_in_'.$databasename]."');>".$l['Tables_in_'.$databasena
me]."</td></tr>";
	}
	echo "</table>";
}
?>
</td>
<td><center>
<input type=button class=button value='>>' onclick='addtable();'><br>
Номер<br>
<input type=text id=ntable name=ntable class='ntable' value=''>
</td>
<td>
<?
$list=$cseqtable->getListTable($session);
if(count($list)>0){
	echo "<table>";
	foreach($list as $l){
		echo "<tr><td>".$l['n']."</td>";
		echo "<td>".$l['sequenceTableName']."</td><td>
		<input type=button class=button value='Видалити'
onclick='deletetable(".$l['n'].");'></td>";
		echo "</tr>";
	}
	echo "</table>";
}
?>
</td></tr>
</table>
<input type=button class=button value='Відміна' onclick='canceltable();'>   
</div>



<div id=denorm class=addupdate>
<select id=kinddenorm size=2 onchange='selectkinddenorm();'>
<option value=1>Нисхідна</option>
<option value=2>Висхідна</option>
</select><br><br>
<div id=kinddenormdown class='displaynone'>

<table><tr><td>
<?
```

```
$list=$cseqtable->getListFieldFirstTable($session,$database);
if(count($list)>0){
        echo "<select id=tablefielddown name=tablefielddown size=".count($list)."
ondblclick='getAddField();'>";
        foreach($list as $l){
                if($l['Field']==$selectedfield)$selectlist=' selected '; else $selectlist='';
                echo "<option value=".$l['Field'].$selectlist.">".$l['Field']."</option>";
        }
        echo "</select>";
   }
?>
</td><td>
<input type=button class=button value='>>' onclick='getAddField()'>
</td>
<td>
Назва поля, значення якого дублюється<br>
<input type=text id=copyfielddown name=copyfielddown class=button disabled><br><br>

Назва поля, яке містить продубльовані значення<br>
<input type=text id=addfielddown name=addfielddown class=button ><br><br>
</td>
<td>
<select id=typefielddenormdown name=typefielddenorm size=3 >
<option value='int'1>int</option>
<option value='float'>float</option>
<option value='string'>string</option>

</select>
</td><td>
<?
$list=$cseqtable->getListFieldLastTable($session,$database);
if(count($list)>0){
        echo "<select id=lasttablefielddown name=lasttablefielddown size=".count($list)." >";
        foreach($list as $l){
                if($l['Field']==$selectedfield)$selectlist=' selected '; else $selectlist='';
                echo "<option value=".$l['Field'].$selectlist.">".$l['Field']."</option>";
        }
        echo "</select>";
   }


?>
</td>
</tr>
<tr><td colspan=5><input type=button class=button value='OK'
onclick='addupdatedenorm()'>  
<input type=button class=button value='Відміна' onclick='canceldenorm()'></td>
</tr>
</table>
</div><br><br>
<div id=kinddenormup class='displaynone'>
<table><tr>
<td>
Вираз, на основі якого розраховується <br>агрегатне значення<br>
<input type=text id=copyfieldup name=copyfieldup class=button ><br><br>
```

```
Назва поля, яке містить агрегатне значення<br>
<input type=text id=addfieldup name=addfieldup class=button ><br><br>
</td>
<td>
<select id=sumfunc name=sumfunc size=5 ondblclick='addfunc();'>
<option value=1>sum</option>
<option value=2>min</option>
<option value=3>max</option>
<option value=4>avg</option>
<option value=5>count</option>
</select>
</td>
<td>
<select id=typefielddenormtype name=typefielddenorm size=3 >
<option value='int'1>int</option>
<option value='float'>float</option>
<option value='string'>string</option>

</select>
</td>
<td>
<?
$list=$cseqtable->getListTable($session);
if(count($list)>0){
        echo "<select id=tableup name=tableup size=".count($list)." onclick='selecttableup();'>";
        foreach($list as $l){
                //if($l['sequenceTableName']==$selectedfield)$selectlist=' selected '; else
      $selectlist='';
                echo "<option
value=".$l['sequenceTableName'].$selectlist.">".$l['sequenceTableName']."</option>";
        }
        echo "</select>";
  }
?>
</td><td><div id=lsstfieldup>
<?
$list=$cseqtable->getListFieldFirstTable($session,$database);
if(count($list)>0){
        echo "<select id=tablefieldup name=tablefieldup size=".count($list)."
ondblclick='selectfieldup();'>";
        foreach($list as $l){
                if($l['Field']==$selectedfield)$selectlist=' selected '; else $selectlist='';
                echo "<option value=".$l['Field'].$selectlist.">".$l['Field']."</option>";
        }
        echo "</select>";
  }


?>
</div></td>
</tr>
<tr><td colspan=5><input type=button class=button value='OK'
onclick='addupdatedenorm()'>  
<input type=button class=button value='Відміна' onclick='canceldenorm()'></td>
</tr>
</table>
```

```
</div>
</div>
<div id=rule class=addupdate>
<table class='seqtable'>
<tr><td>
<input type=hidden id=selectrule name=selectrule value=''>
<?
$list=$cseqtable->getListTable($session);
if(count($list)>0){
        echo "<table>";
        $t1=$list[0]['sequenceTableName'];$id1=$list[0]['sequenceId'];
        for($i=1;$i<count($list);$i++){
                echo "<tr
onclick='selectrule(\"".$t1."\",\"".$list[$i]['sequenceTableName']."\",".$i.");'>
                <td
id='t_".$id1."'>".$i."</td><td>".($i+1)."</td><td>".$t1."</td><td>".$list[$i]['sequenceTableName'].
"</td></tr>";
                $t1=$list[$i]['sequenceTableName'];
                $id1=$list[$i]['sequenceId'];
        }
        echo "</table>";
}
?>
</td>

<td>
<div id=listfieldforpk></div>
</td>
<td>
Первинний ключ:<br>
<input type=text id=pkrule name=pkrule class=button><br><br>
Зовнішній ключ:<br>
<input type=text id=fkrule name=fkrule class=button><br><br>

</td>
<td>
<div id=listfieldforfk></div><br>
</td></tr><tr>
<td colspan=4>
<input type=button class=button value='Відміна' onclick='cancelrule();'>   
</td>
</tr>
</table>

</div>
```

Модуль classlist.php

```
<?
Class Session extends Controller{
        public function getCurrentSession($user){
                $connection=mysqli_connect(Controller::$host, Controller::$user,
Controller::$password, Controller::$database);
                $qr="SELECT sessionid FROM  `session`
                WHERE userid=".$user." and sessionid=(select max(sessionid) from
`session`WHERE userid=".$user.")";
                //echo $qr;
```

```
                $rowset=mysqli_query ($connection,$qr);
        if ($rowset)
            if(mysqli_num_rows($rowset)>0){
            while($a = mysqli_fetch_assoc($rowset)){
                $sessionid=$a['sessionid'];
            }
            return $sessionid;
        }
                else return 0;
            }

            public function getCurrentDatabase($session){
                $connection=mysqli_connect( Controller::$host, Controller::$user,
Controller::$password, Controller::$database);
                $qr="SELECT databaseid FROM `session` a WHERE sessionid=".$session;
                $rowset=mysqli_query ($connection, $qr);
        if ($rowset)
            if(mysqli_num_rows($rowset)>0){
                                while($a = mysqli_fetch_assoc($rowset)) {
                                    if($a['databaseid']==null)
                                        $databaseid=0;
                                    else
                                        $databaseid=$a['databaseid'];
                                }
                                return $databaseid;
                            }
                else return 0;
            }

            public function getNameDatabase($database){
                $connection=mysqli_connect( Controller::$host, Controller::$user,
Controller::$password, Controller::$database);
                $qr="SELECT databasename FROM `databases`
                WHERE databaseid=".$database;
                $rowset=mysqli_query ($connection, $qr);
        if ($rowset)
            while($a = mysqli_fetch_assoc($rowset)) {
                $databasename=$a['databasename'];
            }
            return $databasename;
            }

            public function getHostDatabase($database){
                $connection=mysqli_connect( Controller::$host, Controller::$user,
Controller::$password, Controller::$database);
                $qr="SELECT databasehost FROM `databases`
                WHERE databaseid=".$database;
                $rowset=mysqli_query ($connection, $qr);
        if ($rowset)
            while($a = mysqli_fetch_assoc($rowset)) {
                $databasehost=$a['databasehost'];
            }
            return $databasehost;
            }

            public function getState($session){
```

```
                $connection=mysqli_connect( Controller::$host, Controller::$user,
Controller::$password, Controller::$database);
                $qr="SELECT sessionstate FROM `session`  WHERE sessionid=".$session;
                $rowset=mysqli_query ($connection, $qr);
      if ($rowset)
         while($a = mysqli_fetch_assoc($rowset)) {
            $sessionstate=$a['sessionstate'];
         }
         return $sessionstate;
          }
          public function getDenorm($session){
                $connection=mysqli_connect( Controller::$host, Controller::$user,
Controller::$password, Controller::$database);
                $qr="SELECT sessionDenormKind FROM `session`  WHERE
sessionid=".$session;
                $rowset=mysqli_query ($connection, $qr);
      if ($rowset)
         while($a = mysqli_fetch_assoc($rowset)) {
            $sessionDenormKind=$a['sessionDenormKind'];
         }
         return $sessionDenormKind;
          }

          public function getDenormField($session){
                $connection=mysqli_connect( Controller::$host, Controller::$user,
Controller::$password, Controller::$database);
                $qr="SELECT sessionDenormAtributeName FROM `session`  WHERE
sessionid=".$session;
                $rowset=mysqli_query ($connection, $qr);
      if ($rowset)
         while($a = mysqli_fetch_assoc($rowset)) {
            $sessionDenormAtributeName=$a['sessionDenormAtributeName'];
         }
         return $sessionDenormAtributeName;
          }
          public function getAddField($session){
                $connection=mysqli_connect( Controller::$host, Controller::$user,
Controller::$password, Controller::$database);
                $qr="SELECT sessionAddAtributeName FROM `session`  WHERE
sessionid=".$session;
                $rowset=mysqli_query ($connection, $qr);
      if ($rowset)
         while($a = mysqli_fetch_assoc($rowset)) {
            $sessionAddAtributeName=$a['sessionAddAtributeName'];
         }
         return $sessionAddAtributeName;
          }
          public function insertsession($user){
                $connection=mysqli_connect( Controller::$host, Controller::$user,
Controller::$password, Controller::$database);
                $qr="insert into `session` values(0,'".date('Y-m-d')."',0,0,'','','','".$user."',null)";
                mysqli_query ($connection, $qr);
                $sessionid=mysqli_insert_id($connection);
      return $sessionid;
        }
```

```
        public function insertdenorm($copyfield,$addfield,$session,$type,$kind){
                $connection=mysqli_connect( Controller::$host, Controller::$user,
Controller::$password, Controller::$database);
                $qr="update `session` set
sessionDenormKind=".$kind.",sessionDenormAtributeName='".$copyfield."',
                        sessionAddAtributeName='".$addfield."',
                        sessionAddAtributeType='".$type."',sessionstate=3 where
sessionid=".$session;
                mysqli_query ($connection, $qr);
        }

        public function getListTable($database){
                $connection=mysqli_connect( Controller::$host, Controller::$user,
Controller::$password, Controller::$database);
                $qr="SELECT databasename,databasehost FROM `databases` WHERE
databaseid=".$database;
                $rowset=mysqli_query ($connection, $qr);
      if ($rowset)
         while($a = mysqli_fetch_assoc($rowset)) {
            $databasehost=$a['databasehost'];
            $databasename=$a['databasename'];
         }
                $connection=mysqli_connect( $databasehost, 'root', 'root', $databasename);
                $qr="show tables";
                $rowset=mysqli_query ($connection, $qr);
      if ($rowset)
         if(mysqli_num_rows($rowset)>0){
         while($arr= mysqli_fetch_assoc($rowset)) {
            $list[]=$arr;
         }
         mysqli_free_result($rowset);
         return $list;
          }
        }

        public function getListFieldTable($session,$database,$table){
                $connection=mysqli_connect( Controller::$host, Controller::$user,
Controller::$password, Controller::$database);

                $qr="SELECT databasename,databasehost FROM `databases` WHERE
databaseid=".$database;
                $rowset=mysqli_query ($connection, $qr);
      if ($rowset)
         while($a = mysqli_fetch_assoc($rowset)) {
            $databasehost=$a['databasehost'];
            $databasename=$a['databasename'];
         }
                $connection=mysqli_connect( $databasehost, 'root', 'root', $databasename);
                $qr="SHOW COLUMNS FROM ".$table;
                $rowset=mysqli_query ($connection, $qr);
      if ($rowset)
         if(mysqli_num_rows($rowset)>0){
         while($arr= mysqli_fetch_assoc($rowset)) {
            $list[]=$arr;
         }
         mysqli_free_result($rowset);
```

```
        return $list;
        }
       }

        public function addupdatedb($databaseid,$name,$host,$sessionid){
                $connection=mysqli_connect( Controller::$host, Controller::$user,
Controller::$password, Controller::$database);
                if($databaseid==0){
                        $qr="SELECT databaseid FROM `databases`  WHERE
databasename='".$name."' and databasehost='".$host."'";
                        $rowset=mysqli_query ($connection, $qr);
                        if ($rowset)
                                if(mysqli_num_rows($rowset)>0){
                                        while($a = mysqli_fetch_assoc($rowset))
                                                $databaseid=$a['databaseid'];
                                }
                                else{
                                        $qr="insert into `databases`
values(0,'".$name."','','".$host."')";
                                        mysqli_query ($connection, $qr);
                                        $databaseid=mysqli_insert_id($connection);
                                }
                        $qr="update `session` set databaseid=".$databaseid.",sessionstate=1
where sessionid=".$sessionid;

                        mysqli_query($connection, $qr);
                }
                else{
                        $qr="SELECT count(*) as cnt FROM `session`  WHERE
databaseid=".$databaseid;
                        $rowset=mysqli_query ($connection, $qr);
                        if ($rowset)
                                while($a = mysqli_fetch_assoc($rowset))
                                        $cnt=$a['cnt'];
                        if($cnt>1){
                                $qr="insert into `databases` values(0,'".$name."','','".$host."')";
                                mysqli_query ($connection, $qr);
                                $databaseid=mysqli_insert_id($connection);
                                $qr="update `session` set databaseid=".$databaseid." where
sessionid=".$sessionid;
                                mysqli_query($connection, $qr);
                        }
                        else{
                                $qr="update `databases` set databasename='".$name."',
databasehost='".$host."' where databaseid=".$databaseid;
                                mysqli_query($connection, $qr);
                        }
                }
        }
}

Class User extends Controller{
        public function checkuser($login,$password){
                $connection=mysqli_connect( Controller::$host, Controller::$user,
Controller::$password, Controller::$database);
```

```
                $qr="select userid from `user` where userlogin='".$login."' and
userpassword='".$password."'";
                $rowset=mysqli_query ($connection, $qr);
        if ($rowset)
           if(mysqli_num_rows($rowset)>0){
           while($a = mysqli_fetch_assoc($rowset)) {
              $userid=$a['userid'];
           }
           return $userid;
        }
                else return 0;
          }
          public function getName($user){
                $connection=mysqli_connect( Controller::$host, Controller::$user,
Controller::$password, Controller::$database);
                $qr="select username from `user` where userid=".$user;
                $rowset=mysqli_query ($connection, $qr);
        if ($rowset)
           if(mysqli_num_rows($rowset)>0){
           while($a = mysqli_fetch_assoc($rowset)) {
              $username=$a['username'];
           }
           return $username;
        }
                else return 0;
          }
          public function insertuser($name,$login,$password){
                $connection=mysqli_connect( Controller::$host, Controller::$user,
Controller::$password, Controller::$database);
                $qr="insert into `user` values(0,'".$name."','".$login."','".$password."','".date('Y-m-
d')."')";
                mysqli_query ($connection, $qr);
                $userid=mysqli_insert_id($connection);
        return $userid;
          }
}

Class SeqTable{
          public function getListTable($session){
                $connection=mysqli_connect( Controller::$host, Controller::$user,
Controller::$password, Controller::$database);
                $qr="SELECT * FROM `sequence`
                WHERE sessionid=".$session." order by n";
                $rowset=mysqli_query ($connection, $qr);
        if ($rowset)
           if(mysqli_num_rows($rowset)>0){
           while($arr= mysqli_fetch_assoc($rowset)) {
              $list[]=$arr;
           }
           mysqli_free_result($rowset);
           return $list;
            }
          }
          public function inserttable($n,$name,$session){
                $connection=mysqli_connect( Controller::$host, Controller::$user,
Controller::$password, Controller::$database);
```

```
                $qr="insert into `sequence` values(0,'".$name."',",",".$session.",".$n.")";
                mysqli_query ($connection, $qr);
                $userid=mysqli_insert_id($connection);
        return $userid;
          }
          public function deletetable($n,$session){
                $connection=mysqli_connect( Controller::$host, Controller::$user,
Controller::$password, Controller::$database);
                $qr="delete from `sequence` where sessionid=".$session." and n=".$n;
                mysqli_query ($connection, $qr);

          }
          public function getListFieldFirstTable($session, $database){
                $connection=mysqli_connect( Controller::$host, Controller::$user,
Controller::$password, Controller::$database);
                $qr="select sequenceTableName from `sequence` where n=1 and
sessionid=".$session;
                $rowset=mysqli_query ($connection, $qr);
        if ($rowset)
           while($a = mysqli_fetch_assoc($rowset)) {
              $table=$a['sequenceTableName'];
           }
                $qr="SELECT databasename,databasehost FROM `databases` WHERE
databaseid=".$database;
                $rowset=mysqli_query ($connection, $qr);
        if ($rowset)
           while($a = mysqli_fetch_assoc($rowset)) {
              $databasehost=$a['databasehost'];
              $databasename=$a['databasename'];
           }
                $connection=mysqli_connect( $databasehost, 'root', 'root', $databasename);
                $qr="SHOW COLUMNS FROM ".$table;
                $rowset=mysqli_query ($connection, $qr);
        if ($rowset)
           if(mysqli_num_rows($rowset)>0){
           while($arr= mysqli_fetch_assoc($rowset)) {
              $list[]=$arr;
           }
           mysqli_free_result($rowset);
           return $list;
            }
          }
          public function getListFieldLastTable($session, $database){
                $connection=mysqli_connect( Controller::$host, Controller::$user,
Controller::$password, Controller::$database);
                $qr="select sequenceTableName from `sequence` where n=(select max(n) from
sequence where sessionid=".$session.") and sessionid=".$session;
                $rowset=mysqli_query ($connection, $qr);
        if ($rowset)
           while($a = mysqli_fetch_assoc($rowset)) {
              $table=$a['sequenceTableName'];
           }
                $qr="SELECT databasename,databasehost FROM `databases` WHERE
databaseid=".$database;
                $rowset=mysqli_query ($connection, $qr);
        if ($rowset)
```

```
            while($a = mysqli_fetch_assoc($rowset)) {
               $databasehost=$a['databasehost'];
               $databasename=$a['databasename'];
            }
                    $connection=mysqli_connect( $databasehost, 'root', 'root', $databasename);
                    $qr="SHOW COLUMNS FROM ".$table;
                    $rowset=mysqli_query ($connection, $qr);
        if ($rowset)
            if(mysqli_num_rows($rowset)>0){
            while($arr= mysqli_fetch_assoc($rowset)) {
               $list[]=$arr;
            }
            mysqli_free_result($rowset);
            return $list;
             }
            }
            public function insertrule($n,$session,$pk,$fk)        {
                    $connection=mysqli_connect( Controller::$host, Controller::$user,
Controller::$password, Controller::$database);
                    $qr="update `sequence` set primaryAtribute='".$pk."',foreignAtribute='".$fk."'
where n=".$n." and sessionid=".$session;
                    mysqli_query($connection, $qr);
return  $qr      ;
            }
}
```

Модуль script.js

```
function cancelall(){
            document.getElementById('reg').style.display='none';
            document.getElementById('inputuser').style.display='none';
            document.getElementById('db').style.display='none';
            document.getElementById('seqtable').style.display='none';
            document.getElementById('denorm').style.display='none';
            document.getElementById('rule').style.display='none';
}
function cancelinputuser(){
            document.getElementById('inputuser').style.display='none';
}
function cancelreg(){
            document.getElementById('reg').style.display='none';
}
function canceldb(){
            document.getElementById('db').style.display='none';
}
function canceltable(){
            document.getElementById('seqtable').style.display='none';
}
function canceldenorm(){
            document.getElementById('denorm').style.display='none';
}
function cancelrule(){
            document.getElementById('rule').style.display='none';
}
```

```
function  adddb(){
	cancelall();
	document.getElementById('databaseid').value=0;
	document.getElementById('db').style.display='block';
}
function updatedb(){
	cancelall();
	document.getElementById('databaseid').value=document.getElementById('database').va
lue;
	document.getElementById('db').style.display='block';
}

function updatetable(){
	cancelall();
	document.getElementById('seqtable').style.display='block';
}

function changeDenorm(){
	cancelall();
	document.getElementById('denorm').style.display='block';
}
function updaterule(){
	cancelall();
	document.getElementById('rule').style.display='block';
}
function selectkinddenorm(){
	document.getElementById('kinddenormdown').style.display='none';
	document.getElementById('kinddenormup').style.display='none';
	if(document.getElementById('kinddenorm').value==1)
	document.getElementById('kinddenormdown').style.display='block';
else
	document.getElementById('kinddenormup').style.display='block';
}

function selecttable(t){
	if(document.getElementById('selecttable').value!=''){

	document.getElementById('t_'+document.getElementById('selecttable').value).style.back
ground='#F0F8FF';
	}
	document.getElementById('selecttable').value=t;
	document.getElementById('t_'+document.getElementById('selecttable').value).style.back
ground='#87CEEB';
}

function getAddField(){
document.getElementById('copyfielddown').value=document.getElementById('tablefielddown').v
alue;
}

function selectfieldup(){
document.getElementById('copyfieldup').value+=document.getElementById('tablefieldup').value;
}

function selecttableup(){
```

```
tbl=document.getElementById('tableup').value
  arg='entity=field&table='+tbl+
  '&database='+document.getElementById('database').value+
  '&session='+document.getElementById('session').value;
   $.ajax({
      type: "GET",
      url: '../addmodules/addupdate.php',
      data: arg,
      success: function(data){
                  document.getElementById('tablefieldup').innerHTML=data;
      }
   });
}

function selectfieldrulepk(p){
document.getElementById('pkrule').value=p;
}
function selectfieldrulefk(p){
document.getElementById('fkrule').value=p;
}

function getuser(){
         cancelall();
         document.getElementById('inputuser').style.display='block';
}
function reg(){
         cancelall();
         document.getElementById('reg').style.display='block';
}
function scheme(){
         document.getElementById('menu').value='current';
         document.forms["formdenorm"].submit();
}
function archive(){
         document.getElementById('menu').value='archive';
         document.forms["formdenorm"].submit();
}


function input(){
         if(document.getElementById('inputlogin').value==''){
                  alert('Не вказаний логін!');
                  exit();
         }else
         if(document.getElementById('inputpassword').value==''){
                  alert('Не вказаний пароль!');
                  exit();
         }
  arg='entity=currentuser&login='+document.getElementById('inputlogin').value+
                  '&pass='+document.getElementById('inputpassword').value;
   $.ajax({
      type: "GET",
      url: '../addmodules/addupdate.php',
      data: arg,
      success: function(data){
                  if(data==0){
```

```
                        alert('Користувач не знайдений');
                }
                else{
                        document.getElementById('user').value=data;
                        document.forms["formdenorm"].submit();
                }
        }
    });
}
function adduser(){
        if(document.getElementById('regusername').value==''){
                alert('Не вказано ім\'я!');
                exit();
        }else
        if(document.getElementById('reglogin').value==''){
                alert('Не вказаний логін!');
                exit();
        }else
        if(document.getElementById('regpassword').value==''){
                alert('Не вказаний пароль!');
                exit();
        }
   arg='entity=reg&name='+document.getElementById('regusername').value+
                '&login='+document.getElementById('reglogin').value+
                '&pass='+document.getElementById('regpassword').value;
    $.ajax({
        type: "GET",
        url: '../addmodules/addupdate.php',
        data: arg,
        success: function(data){
                document.getElementById('user').value=data;
                document.forms["formdenorm"].submit();
        }
    });
}

function newscheme(){
        if(document.getElementById('user').value==0){
                alert('Не визначений користувач!');
                exit();
        }
         arg='entity=scheme&user='+document.getElementById('user').value;
        $.ajax({
                type: "GET",
                url: '../addmodules/addupdate.php',
                data: arg,
                success: function(data){
                        alert(data);
                        document.getElementById('session').value=data;
                        document.forms["formdenorm"].submit();
                }
        });
}
function addupdatedb(){
            arg='entity=db&databaseid='+document.getElementById('databaseid').value+
            '&databasename='+document.getElementById('dbname').value+
```

```
            '&databasehost='+document.getElementById('dbhost').value+
            '&session='+document.getElementById('session').value;
                $.ajax({
                        type: "GET",
                        url: '../addmodules/addupdate.php',
                        data: arg,
                        success: function(data){
                                document.forms["formdenorm"].submit();
                        }
                });
}
function addtable(){
          arg='entity=table&n='+document.getElementById('ntable').value+
                '&tablename='+document.getElementById('selecttable').value+
                '&session='+document.getElementById('session').value;
   $.ajax({
      type: "GET",
      url: '../addmodules/addupdate.php',
      data: arg,
      success: function(data){
                document.forms["formdenorm"].submit();
                document.getElementById('seqtable').style.display='block';
      }
   });
}

function addupdatedenorm(){
            arg='entity=denorm&kind='+document.getElementById('kinddenorm').value;
            if(document.getElementById('kinddenorm').value==1)
                  arg+=   '&copyfield='+document.getElementById('copyfielddown').value+
                  '&addfield='+document.getElementById('addfielddown').value+
                  '&type='+document.getElementById('typefielddenormdown').value+
                  '&session='+document.getElementById('session').value;
          else
                  arg+=   '&copyfield='+document.getElementById('copyfieldup').value+
                  '&addfield='+document.getElementById('addfieldup').value+
                  '&type='+document.getElementById('typefielddenormup').value+
                  '&session='+document.getElementById('session').value;

   $.ajax({
      type: "GET",
      url: '../addmodules/addupdate.php',
      data: arg,
      success: function(data){
                        //alert(data);
                document.forms["formdenorm"].submit();
      }
   });
}
function addrule(){
  arg='entity=rule&n='+document.getElementById('selectrule').value+
  '&pk='+document.getElementById('pkrule').value+
  '&fk='+document.getElementById('fkrule').value+
  '&session='+document.getElementById('session').value;
   $.ajax({
      type: "GET",
```

```
        url: '../addmodules/addupdate.php',
        data: arg,
        success: function(data){
                document.forms["formdenorm"].submit();
        }
    });
}
function selectrule(t1,t2,n){
document.getElementById('selectrule').value          =n;
   arg='entity=fieldrule&t='+t1+
   '&session='+document.getElementById('session').value+
   '&type=1'+
   '&database='+document.getElementById('database').value;
    $.ajax({
        type: "GET",
        url: '../addmodules/addupdate.php',
        data: arg,
        success: function(data){
                document.getElementById('listfieldforpk').innerHTML=data;
        }
    });
   arg='entity=fieldrule&t='+t2+
   '&session='+document.getElementById('session').value+
   '&type=2'+
   '&database='+document.getElementById('database').value;
$.ajax({
        type: "GET",
        url: '../addmodules/addupdate.php',
        data: arg,
        success: function(data){
                document.getElementById('listfieldforfk').innerHTML=data;

        }
    });

}
function deletetable(n){
   arg='entity=table&n='+n+
   '&session='+document.getElementById('session').value;
    $.ajax({
        type: "GET",
        url: '../addmodules/delete.php',
        data: arg,
        success: function(data){

                document.forms["formdenorm"].submit();
                document.getElementById('seqtable').style.display='block';
        }
    });
}
```

Скрипт сервісної БД

```
-- phpMyAdmin SQL Dump


CREATE TABLE `databases` (
```

```
  `databaseId` int(2) NOT NULL,
  `databaseName` varchar(30) COLLATE utf8mb4_unicode_ci NOT NULL,
  `databaseConnectionString` varchar(200) COLLATE utf8mb4_unicode_ci NOT NULL,
  `databasehost` varchar(30) COLLATE utf8mb4_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE `query` (
  `queryId` int(2) NOT NULL,
  `queryCreateText` varchar(500) COLLATE utf8mb4_unicode_ci NOT NULL,
  `queryCreateDate` date NOT NULL,
  `queryUpdateText` varchar(500) COLLATE utf8mb4_unicode_ci NOT NULL,
  `queryUpdateDate` date NOT NULL,
  `queryUpdateResult` int(1) NOT NULL,
  `sessionId` int(5) NOT NULL,
  `queryCreateResult` int(1) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE `sequence` (
  `sequenceId` int(9) NOT NULL,
  `sequenceTableName` varchar(30) COLLATE utf8mb4_unicode_ci NOT NULL,
  `primaryAtribute` varchar(30) COLLATE utf8mb4_unicode_ci NOT NULL,
  `foreignAtribute` varchar(30) COLLATE utf8mb4_unicode_ci NOT NULL,
  `sessionId` int(5) NOT NULL,
  `n` int(3) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE `session` (
  `sessionId` int(5) NOT NULL,
  `sessionOpenDate` date NOT NULL,
  `sessionState` int(1) NOT NULL,
  `sessionDenormKind` int(1) NOT NULL,
  `sessionDenormAtributeName` varchar(30) COLLATE utf8mb4_unicode_ci NOT NULL,
  `sessionAddAtributeName` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
  `sessionAddAtributeType` varchar(20) COLLATE utf8mb4_unicode_ci NOT NULL,
  `sessionExpression` varchar(100) COLLATE utf8mb4_unicode_ci NOT NULL,
  `userId` int(2) NOT NULL,
  `databaseId` int(2) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE `user` (
  `userId` int(3) NOT NULL,
  `userName` varchar(30) COLLATE utf8mb4_unicode_ci NOT NULL,
  `userLogin` varchar(30) COLLATE utf8mb4_unicode_ci NOT NULL,
  `userPassword` varchar(30) COLLATE utf8mb4_unicode_ci NOT NULL,
  `userRegisterDate` date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

ALTER TABLE `databases`
  ADD PRIMARY KEY (`databaseId`);
ALTER TABLE `query`
  ADD PRIMARY KEY (`queryId`),
  ADD KEY `sessionId` (`sessionId`);
ALTER TABLE `sequence`
  ADD PRIMARY KEY (`sequenceId`),
  ADD KEY `sessionId` (`sessionId`);
ALTER TABLE `session`
```

```
  ADD PRIMARY KEY (`sessionId`),
  ADD KEY `userId` (`userId`),
  ADD KEY `databaseId` (`databaseId`);
ALTER TABLE `user`
  ADD PRIMARY KEY (`userId`);
ALTER TABLE `databases`
  MODIFY `databaseId` int(2) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=1;
ALTER TABLE `query`
  MODIFY `queryId` int(2) NOT NULL AUTO_INCREMENT;
ALTER TABLE `sequence`
  MODIFY `sequenceId` int(9) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=1;
ALTER TABLE `session`
  MODIFY `sessionId` int(5) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=1;
ALTER TABLE `user`
  MODIFY `userId` int(3) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;
ALTER TABLE `query`
  ADD CONSTRAINT `query_ibfk_1` FOREIGN KEY (`sessionId`) REFERENCES `session`
(`sessionId`);
ALTER TABLE `sequence`
  ADD CONSTRAINT `sequence_ibfk_1` FOREIGN KEY (`sessionId`) REFERENCES `session`
(`sessionId`);
ALTER TABLE `session`
  ADD CONSTRAINT `session_ibfk_1` FOREIGN KEY (`userId`) REFERENCES `user`
(`userId`),
  ADD CONSTRAINT `session_ibfk_2` FOREIGN KEY (`databaseId`) REFERENCES
`databases` (`databaseId`);
```