Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматизації та комп'ютерно-інтегрованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Розроблення імітаційної моделі розподіленої бази даних для системи керування комп'ютерно-інтегрованим виробництвом
(тема)

Виконав:
студент II курсу, групи КІТПВм-21-1
Іванюк О.А.
(прізвище, ініціали)

Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Комп'ютерно-інтегровані технологічні процеси і виробництва
(повна назва освітньої програми)

Керівник проф. Безкоровайний В. В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри КІТАМ _____ Невлюдов І. Ш.
(підпис) (прізвище, ініціали)

2022 р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет *Автоматизації та комп'ютерно-інтегрованих технологій*

Кафедра *КІТАМ*

Рівень вищої освіти *другий (магістерський)*

Спеціальність *151 Автоматизація та комп'ютерно-інтегровані технології*
(код і повна назва)

Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)

Освітня програма *Комп'ютерно-інтегровані технологічні процеси і виробництва*
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Іванюк Ользі Андріївні*
(прізвище, ім'я, по батькові)

1. Тема роботи *«Розроблення імітаційної моделі розподіленої бази даних для системи керування комп'ютерно-інтегрованим виробництвом»*
затверджена наказом університету від *«07» листопада 2022 р. № 1464 Ст*
2. Термін подання студентом роботи до екзаменаційної комісії *«07» грудня 2022 р.*
3. Вихідні дані до роботи *Об'єкт дослідження – розподілені бази даних (РБД) систем керування комп'ютерно-інтегрованим виробництвом (КІВ). Предмет дослідження – функціональні характеристики РБД систем керування КІВ. Предмет розробки – засіб моделювання процесів функціонування РБД. Функція – оцінювання характеристик процесу функціонування РБД. Кількість локальних баз – до 10. Технічне забезпечення: ІВМ-сумісний персональний комп'ютер. Перелік використовуваних програмних засобів: ОС Microsoft Windows 10 та вище, мова або пакет імітаційного моделювання.*
4. Перелік питань, що потрібно опрацювати в роботі *Вступ. Огляд та аналіз сучасного стану проблеми моделювання розподілених баз даних у системах керування комп'ютерно-інтегрованим виробництвом. Опис комп'ютерно-інтегрованого виробництва та систем керування ним. Архітектура розподілених баз даних. Моделі та методи моделювання РБД. Постановка мети та задач дослідження. Математичне забезпечення задачі моделювання процесу функціонування РБД. Розподіл обробки запитів. Постановка задачі моделювання РБД. Концептуальна модель розподіленої бази даних. Структурна схема моделі та її опис. Q-схема процесу функціонування РБД. Опис моделювального алгоритму. Розробка програмного забезпечення та експерименти. Вибір мови програмування. Вибір середовища розробки. Постановка задачі контрольного прикладу. Опис програмного забезпечення. Планування експериментів та аналіз результатів. Охорона праці. Висновки. Перелік джерел посилання.*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри): *демонстраційний матеріал, представлений у форматі презентації PowerPoint (*.ppt) на аркушах формату А4 (16 сторінок).*

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	*Отримання, аналіз завдання, уточнення плану*		
	роботи	*01.09.22*	*Виконано*
2	*Аналіз проблеми моделювання РБД у системах*		
	керування комп'ютерно-інтегрованим		
	виробництвом	*08.09.22*	*Виконано*
3	*Огляд сучасних засобів моделювання РБД*	*15.09.22*	*Виконано*
4	*Розробка математичного забезпечення задачі*	*29.09.22*	*Виконано*
5	*Розробка програмного забезпечення задачі*	*13.10.22*	*Виконано*
6	*Проведення експериментальних досліджень*	*27.10.22*	*Виконано*
7	*Підготовка публікацій за результатами*		
	дослідження	*10.11.22*	*Виконано*
8	*Оформлення пояснювальної записки*	*24.11.22*	*Виконано*
9	*Подання закінченої роботи науковому керівникові*	*26.11.22*	*Виконано*
10	*Усунення зауважень наукового керівника*	*01.12.22*	*Виконано*
11	*Подання роботи на рецензування*	*02.12.22*	*Виконано*
12	*Підготовка презентації*	*03.12.22*	*Виконано*
13	*Попередній захист*	*05.12.22*	*Виконано*
14	*Подання роботи до екзаменаційної комісії*	*07.12.22*	*Виконано*

Дата видачі завдання *«01» вересня 2022 р.*

Студент _____
(підпис)

Керівник роботи _____ *проф. Безкоровайний В. В.*
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 83 с., 25 рис., 3 табл., 1 дод., 34 джерела.

ІМІТАЦІЙНА МОДЕЛЬ, КОМП'ЮТЕРНО-ІНТЕГРОВАНЕ ВИРОБНИЦТВО, МОДЕЛЮВАННЯ, РОЗПОДІЛЕНА БАЗА ДАНИХ, СИСТЕМА КЕРУВАННЯ.

Об'єкт дослідження – розподілені бази даних систем керування комп'ютерно-інтегрованим виробництвом.

Предмет дослідження – функціональні характеристики розподілених баз даних систем керування комп'ютерно-інтегрованим виробництвом.

Мета кваліфікаційної роботи – підвищення точності визначення основних характеристик процесу функціонування розподіленої бази даних системи керування комп'ютерно-інтегрованим виробництвом.

Методи дослідження: системний підхід, теорія баз даних, планування комп'ютерних експериментів, імітаційне моделювання.

Запропоновано рішення актуальної науково-прикладної задачі підвищення точності визначення характеристик процесу функціонування розподіленої бази даних системи керування комп'ютерно-інтегрованим виробництвом за рахунок розробки адекватної імітаційної моделі процесу. Для програмування задачі моделювання було розроблено кросплатформений додаток у середовищі Visual Studio Code на мові програмування JavaScript з використанням бібліотеки react.js.

Практичне використання моделі за рахунок врахування стохастичного характеру складових процесу дозволить підвищити точність визначення його функціонально-часових характеристик.

Результати кваліфікаційної роботи апробовані на 3-х міжнародних і Всеукраїнській наукових конференціях.

ABSTRACT

Explanatory note: 83 pp., 25 fig., 3 tab., 1 add., 34 dzherel.

SIMULATION MODEL, COMPUTER-INTEGROVANIE VIROBNITSTVODY, MODELING, REPAIRED DATABASE, KERUVANNIA SYSTEM.

The object of follow-up is the subdivision of the databases of these systems for the maintenance of computer-integrated virobnitstvom.

The subject of the research is the functional characteristics of the distribution of data bases of systems for the maintenance of computer-integrated scanning.

Meta-qualification work - improvement of the accuracy of the designation of the main characteristics of the process of functioning of the distributed base of the data system, keruvannya computer-integrated virobnitstvom.

Follow-up methods: system analysis, database theory, planning of computer experiments, simulation modeling.

The solution of the actual scientific and applied problems and the improvement of the accuracy of the characteristics of the process of functioning of the distribution of the data base of the system were proposed by computer-integration versatility for the calculation of the development of an adequate simulation model for the process. For the programming tasks of the modeling, cross-platform add-ons were developed for the Visual Studio Code middleware on the JavaScript language library with the react.js library.

It is practical to vary the model for the calculation of the stochastic nature of the warehouse process in order to improve the accuracy of the designation of its functional and hourly characteristics.

The results of the qualification work were tested at 3 international and All-Ukrainian scientific conferences.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних.

ЛБ – локальна база.

РБД – розподілена база даних.

СМО – система масового обслуговування.

СУБД – система управління базою даних.

СУБМД – система управління мультибазами даних.

ВСТУП

Використання баз даних стало невід'ємною частиною життя кожної людини. Кожного дня створюється та зберігається велика кількість даних по всьому світу. Тому виникає необхідність в зберіганні, обробці та захисту такої великої кількості даних. Людина не має змоги обробляти таку велику кількість даних. Найбільш ефективним способом рішення цієї проблеми є автоматизація предметної області засобами систем управління базами даних (СУБД). Бази даних використовуються практично у всіх сферах діяльності. Виникає гостра проблема проектувати географічно розподілені БД (РБД).

Сучасні комп'ютерно-інтегровані виробництва об'єднують у собі різнорідні додатки і технології (автоматизоване проектування, робототехніка, планування виробничих ресурсів, управління підприємством тощо), що взаємодіють із загальним сховищем даних. Наслідком зростаючої складності засобів вимірювання, обробки та подання інформації у системах керування виробництвом є збільшення потреб в управлінні дедалі більшими обсягами інформації. У таких умовах усе більшу значимість набувають процеси децентралізації, що вимагають створення додатків, доступ до яких здійснюється з територіально розосереджених місць. З цією метою використовуються технології РБД.

Це обумовлює актуальність завдань оцінки часу доступу до інформаційних ресурсів (ІР), що розв'язуються на етапах багатокритеріальної оптимізації РБД, зокрема, систем керування виробництвом в процесах їх проєктування, планування розвитку чи реінжинірингу.

Метою дослідження є підвищення точності визначення основних характеристик процесу функціонування розподіленої бази даних системи керування комп'ютерно-інтегрованим виробництвом.

Об'єкт дослідження – розподілені бази даних систем керування комп'ютерно-інтегрованим виробництвом.

Предмет дослідження – функціональні характеристики розподілених баз даних систем керування комп’ютерно-інтегрованим виробництвом.

Методи дослідження: системний підхід, теорія баз даних, теорія ймовірностей, математична статистика, планування комп’ютерних експериментів, імітаційне моделювання.

Для досягнення мети необхідно вирішити множину таких завдань:

– провести огляд та аналіз сучасного стану проблеми моделювання РБД у системах керування комп’ютерно-інтегрованим виробництвом;

– сформулювати постановку задачі моделювання РБД у системах керування комп’ютерно-інтегрованим виробництвом;

– обрати метод та побудувати еквівалентну схему процесу обробки запитів в РБД;

– розробити моделювальний алгоритм розв’язання задачі;

– обрати мову програмування;

– обрати середовище розробки програми моделювання;

– розробити програму, яка розв’язує поставлену задачу;

– провести перевірку працездатності програми;

– спланувати, провести модельні експерименти, оцінити точність результатів моделювання РБД;

– розглянути питання безпеки життєдіяльності співробітників у відділі проєктування РБД;

– оформити пояснювальну записку згідно з рекомендаціями методичних вказівок та вимогами діючих стандартів.

За темою кваліфікаційної роботи було підготовлено доповіді на три міжнародні та Всеукраїнську науково-практичні конференції.

Кваліфікаційна робота оформлена згідно з вимогами ДСТУ 3008:2015 [1], а також з рекомендаціями з підготовки і оформлення кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти [2].

1 ОГЛЯД ТА АНАЛІЗ СУЧАСНОГО СТАНУ ПРОБЛЕМИ МОДЕЛЮВАННЯ РОЗПОДІЛЕНИХ БАЗ ДАНИХ У СИСТЕМАХ КЕРУВАННЯ КОМП'ЮТЕРНО-ІНТЕГРОВАНИМ ВИРОБНИЦТВОМ

1.1 Опис комп'ютерно-інтегрованого виробництва та систем керування

Комп'ютерно-інтегроване виробництво – це виробничий підхід, що забезпечує повну автоматизацію виробництва. Усі операції контролюються комп'ютерами та мають сховища даних, а також розподіл. Процеси та компоненти, які залучаються до комп'ютерно-інтегрованого виробництва, наведені нижче [3]:

– системи автоматизованого проектування;

– виготовлення прототипу;

– визначення ефективного методу виробництва шляхом розрахунку витрат та обліку методів виробництва, обсягу продукції, зберігання та розподілу;

– замовлення необхідних матеріалів для виробничого процесу;

– автоматизоване виробництво виробів за допомогою цифрових комп'ютерних контролерів;

– контроль якості на кожному етапі розробки комп'ютерно-інтегрованого виробництва;

– складання продукту за допомогою роботів;

– перевірка якості та автоматизоване зберігання;

– автоматичний розподіл продуктів зі складських приміщень в очікуванні вантажівок;

– автоматичне оновлення журналів, фінансових даних та розрахунків у комп'ютерній системі.

Комп'ютерно-інтегроване виробництво є комбінацією різних додатків і технологій, таких як автоматизоване проектування, робототехніка, планування

виробничих ресурсів і рішення для управління підприємством. Це також можна розглядати як інтеграцію всіх корпоративних операцій, що працюють із загальним сховищем даних.

Основними компонентами комп’ютерно-інтегрованого виробництва є [3]:

– механізми зберігання, пошуку, обробки та подання даних;

– датчики реального часу для визначення поточного стану та зміни процесів комп’ютерно-інтегрованого виробництва;

– алгоритми обробки даних.

Підхід комп’ютерно-інтегрованого виробництва знайшов широкий спектр застосувань у промисловій та виробничій сфері, машинобудуванні та автоматизації електронного проектування. Комп’ютерно-інтегроване виробництво збільшує продуктивність виробництва та знижує загальну вартість виробництва. Це також пропонує велику гнучкість, якість та чутливість.

На шляху до створення безперебійно працюючої комп'ютерно-інтегрованої виробничої системи стоїть три основні завдання [3]:

– інтеграція компонентів від різних постачальників: коли різні верстати,

– такі як ЧПУ, конвеєри та роботи використовують різні протоколи зв'язку, може спричинити проблеми;

– цілісність даних: чим вище рівень автоматизації, тим важливіше цілісність даних, що використовуються для керування машинами. Хоча система комп’ютерно-інтегрованого виробництва заощаджує роботу з експлуатації машин, вона вимагає додаткових людських зусиль для забезпечення належного захисту сигналів даних, які використовуються для керування машинами;

– контроль над процесом: комп'ютери можуть використовуватися людиною-оператором виробничого об'єкта, але завжди повинен бути під рукою компетентний інженер, який міг би впоратися з обставинами, які не могли бути передбачені розробниками програмного забезпечення.

Система управління – це задана сукупність інструментів управління об'єктом. Об'єктами можуть бути технічні об'єкти, підприємства, люди. Об'єкт може складатися з підлеглих об'єктів із побудованими взаємозв'язками.

Серед головних завдань сучасної системи управління можна виділити наступні [3]:

– координація дій всіх співробітників для досягнення спільної мети;

– організація та підтримка контактів між окремими співробітниками та робочими групами;

– робота з інформацією – збирання, обробка, аналіз, зберігання;

– розподіл та перерозподіл ресурсів;

– кадрова діяльність – формування мотиваційної системи, залагодження конфліктів, контроль роботи;

– налагодження контактів із іншими компаніями;

– маркетинг та реклама;

– впровадження інновацій;

– планування, ухвалення рішень, контроль;

– залежно від ситуації – корекція діяльності.

Призначення системи управління полягає у виконанні наступних функцій [3]:

– створення необхідних умов самоорганізації співробітників, а саме, використання фінансової, правової та інших структур, грамотний розподіл управлінських функцій, регламентація бізнес-процесів, стимулювання;

– формування головної ідеї. Обґрунтування суті та призначення компанії для осіб, які працюють всередині компанії та за її межами, аналіз цільової аудиторії, формування стратегічно важливих товарних категорій, напрямів діяльності, корпоративного стилю;

– цілепокладання для підприємства в цілому, відділів та підрозділів, окремих проектів та бізнес-процесів. Для кожного формується власна схема короткострокових та довгострокових цілей, визначаються критерії, за якими робитиметься висновок про виконання поставлених перед підприємством завдань;

– формування системи планування досягнення завдань та виконання цілей;

– виконання поставлених планів. Визначається порядок знаходження, аналізу та усунення управлінських проблем, розробляється та впроваджується механізм розробки, впровадження та виконання управлінських рішень, організується товарообіг, комунікації всередині компанії та за її межами, розробляються та впроваджуються автоматизовані системи управління;

– зворотній зв'язок. Оцінюються довгострокові та короткострокові цілі, їх досягнення, причини недостатньої ефективності, проводиться внутрішній контроль, аудит, запроваджуються корпоративні інформаційні системи (КІС).

Види систем управління. Системи управління бувають відкритими та закритими. Відмінність між типами у тому, що у закритих СУ управлінський блок є невід'ємною частиною самої системи, а відкритих – не входить до її складу.

За характером впливу можлива наступна класифікація систем управління:

Програмні або жорсткі. Суб'єкт та об'єкт управління взаємодіють безпосередньо. Усі керуючі впливу одного до іншого обов'язкові до виконання. Однією з різновидів програмної системи є адміністративно-командна.

Регульована система. Використовує інформацію про те, як об'єкт реагує на інтереси клієнтів, різні адміністративні та економічні управлінські методи.

Саморегулююча система. Тут регулювання здійснюється без участі будь-яких зовнішніх факторів.

Адаптивні СУ. Вплив зовнішніх сил і чинників є, проте керований об'єкт реагує будь-які втручання досить адекватно. Суб'єкт найчастіше знаходиться або поза об'єкта, або вважається вищим.

Ефективність системи управління. Ефективність системи управління організацією оцінюється за кількома важливими компонентами:

– зворотній зв'язок: оперативність, повнота та точність оцінки сьогодення та прогнозованого найближчим часом стану об'єкта управління;

– якість суб'єкта: повнота, злагодженість та вартість бізнес-процесів;

– якість впливу: оперативність, вигода, точність ухвалення важливих управлінських рішень.

До джерел та головних факторів ефективності управлінської системи можна віднести [3]:

– якість зворотного зв'язку: оперативне надходження інформації, актуальність та достовірність відомостей, мінімальна невизначеність та абсолютна безпека та конфіденційність усіх процесів;

– функціональність та злагодженість: усі функції та завдання мають реалізовуватись у повному обсязі, персонал має бути готовий виконувати поставлені завдання, керівництво всіх рівнів активно залучається до управлінського процесу;

– ефективність прийнятих рішень: усі рішення мають відповідати поставленим цілям, враховувати альтернативні шляхи, задовольняти очікування бенефіціарів;

– уникнення кризових ситуацій: характеризується мінімальним рівнем ризиків, результативністю антикризових заходів;

– пошук інноваційних можливостей подальшого розвитку: підвищення результативності, зростання рентабельності.

1.2 Моделі та методи моделювання РБД

Розподілена база даних виробничої системи є аналогом виробничого об'єкта, процесу чи явища, використовувана як замінник оригіналу називається моделлю.

Типи інформаційних моделей. Інформаційні моделі відображають різні типи систем об'єктів, у яких реалізуються різні структури взаємодії та взаємозв'язку між елементами системи. Для відображення систем із різними структурами використовуються різні типи інформаційних моделей: табличні, ієрархічні та мережеві.

У табличній інформаційній моделі перелік однотипних об'єктів або властивостей розміщений у першому стовпці (або рядку) таблиці, а значення їх властивостей розміщуються у наступних стовпцях (або рядках) таблиці. Процес

систематизації об'єктів називається процесом класифікації.

У процесі класифікації об'єктів часто будуються інформаційні моделі, які мають ієрархічну структуру.

Статична ієрархічна модель. В ієрархічній інформаційній моделі об'єкти розподілені за рівнями. Кожен елемент вищого рівня може складатися з елементів нижнього рівня, а елемент нижнього рівня може входити до складу лише одного елемента найвищого рівня.

Мережеві інформаційні моделі застосовуються для відображення систем зі складною структурою, яких зв'язки між елементами мають довільний характер.

Заміна одного об'єкта іншим з метою отримання інформації про найважливіші властивості об'єкта-оригіналу за допомогою об'єкта-моделі називається моделюванням.

Узагальнено моделювання можна визначити як метод опосередкованого пізнання, при якому об'єкт-оригінал, що вивчається знаходиться у певній відповідності з іншим об'єктом-моделлю, причому модель здатна в тому чи іншому відношенні замінити оригінал на деяких стадіях пізнавального процесу.

Стадії пізнання, на яких відбувається така заміна, а також форми відповідності моделі та оригіналу можуть бути різними [4]:

– моделювання як пізнавальний процес містить переробку інформації, яка надходить із зовнішнього середовища, про явища, що відбуваються в ній, в результаті чого у свідомості з'являються образи, що відповідають об'єктам;

– моделювання, що полягає у побудові деякої системи-моделі (іншої системи), пов'язаної певними співвідношеннями подібності до системи оригіналу (першої системи), причому в цьому випадку відображення однієї системи в іншу є засобом виявлення залежностей між двома системами, відображеними у співвідношеннях подібності, а не результатом безпосереднього вивчення інформації, що надходить.

Цілі моделювання [4]:

– зрозуміти сутність об'єкта, що вивчається;

– навчитися управляти об'єктом та визначати найкращі способи управління;

– прогнозувати прямі чи непрямі наслідки;

– вирішувати прикладні завдання.

Способи моделювання процесів функціонування РБД. Все різноманіття способів моделювання РБД, що розглядається теорією моделювання, можна умовно поділити на дві групи [4]:

– аналітичне;

– імітаційне моделювання.

Аналітичне моделювання полягає у побудові моделі, заснованої на описі поведінки об'єкта (РБД) чи системи об'єктів (локальних баз – ЛБ) як аналітичних висловлювань – формул.

Імітаційне моделювання передбачає побудову моделі з характеристиками, адекватними оригіналу, з урахуванням будь-якого його фізичного чи інформаційного принципу. Це означає, що зовнішні впливи на модель та об'єкт викликають ідентичні зміни властивостей оригіналу та моделі.

1.3 Прозорість проектування й управління реплікованими та розподіленими даними

Розподілена база даних – це сукупність кількох логічно взаємопов’язаних баз даних, розташованих у вузлах розподіленої системи. Система керування розподіленою базою даних (розподілена СУБД) визначається, як система програмного забезпечення, яка дозволяє керувати розподіленою базою даних і робить розповсюдження прозорим для користувачів [5].

Іноді «система розподіленої бази даних» (розподілена СУБД) використовується для спільного посилання на розподілену базу даних і розподілену СУБД. Двома важливими характеристиками є те, що дані логічно взаємопов’язані і що вони знаходяться в розподіленій системі.

Розподілена СУБД є логічно інтегрованою, але фізично розподіленою. Це

означає, що розподілена СУБД надає користувачам уявлення про уніфіковану базу даних, тоді як базові дані розподілені фізично.

Розглядаємо два типи розподілених СУБД: територіально-розподілену (зазвичай називають георозподіленою) і єдину локацію (одновузлові). У першому випадку вузли з'єднані між собою глобальними мережами, які характеризуються великою затримкою повідомлень і вищим рівнем помилок. Останні складаються з систем, у яких обробні елементи розташовані в безпосередній близькості, що забезпечує набагато швидший обмін, що призводить до меншої (навіть незначної з новими технологіями) затримки повідомлень і дуже низького рівня помилок.

Одновузлові розподілені СУБД в одному місці зазвичай характеризуються комп'ютерними кластерами в одному центрі обробки даних, і зазвичай відомі як паралельні СУБД. Як зазначалося вище, зараз досить часто зустрічаються розподілені СУБД, які складаються з декількох одновузлових кластерів, з'єднаних глобальними мережами, що призводить до гібридних багатовузлових систем.

Прозорість означає відокремлення семантики вищого рівня системи від питань реалізації нижчого рівня. Іншими словами, прозора система «приховує» деталі реалізації від користувачів [6].

Перевагою повністю прозорої СУБД є високий рівень підтримки, який вона забезпечує для розробки складних програм. Прозорість у розподілених СУБД можна розглядати як розширення концепції незалежності даних у централізованих СУБД.

Незалежність даних. Це поняття переходить від централізованих СУБД і стосується незалежності програм користувача до змін у визначенні та організації даних, і навпаки.

Зазвичай називають два типи незалежності даних: логічна незалежність даних і фізична незалежність даних.

Логічна незалежність даних стосується несприйнятливості програм користувача до змін у логічній структурі (тобто схемі) бази даних.

Фізична незалежність даних, з іншого боку, має справу з приховуванням деталей структури зберігання від програм користувача. Коли користувацька програма написана, вона не повинна турбуватися про деталі фізичної організації даних.

Таким чином, програму користувача не потрібно змінювати, коли відбуваються зміни в організації даних через міркування продуктивності.

Прозорість мережі. Бажано, щоб користувачі були захищені від робочих деталей комунікаційної мережі, яка з'єднує вузли, можливо, навіть приховуючи існування мережі. Тоді не буде різниці між програмами баз даних, які працюватимуть у централізованій базі даних, і тими, які працюватимуть у розподіленій базі даних. Цей тип прозорості називається прозорістю мережі або прозорістю розповсюдження [4].

Іноді виділяють два типи прозорості розподілу: прозорість розташування та прозорість імен. Прозорість розташування стосується того факту, що команда, яка використовується для виконання завдання, не залежить як від розташування даних, так і від системи, у якій виконується операція. Прозорість іменування означає, що кожному об'єкту в базі даних надається унікальне ім'я.

За відсутності прозорості іменування користувачі повинні вставляти ім'я розташування (або ідентифікатор) як частину назви об'єкта.

Прозорість фрагментації. Зазвичай бажано розділити кожне відношення бази даних на менші фрагменти та розглядати кожен фрагмент як окремий об'єкт бази даних (тобто інше відношення). Зазвичай це робиться з міркувань продуктивності, доступності та надійності.

Було б краще, щоб користувачі не знали про фрагментацію даних під час визначення запитів, а система могла відобразити запит, що сформувався у повних відносинах, як зазначено в схемі, на множину запитів, що виконуються у часткових відносинах.

Іншими словами, проблема полягає в тому, щоб знайти стратегію обробки запитів, засновану на фрагментах, а не на відношеннях, навіть якщо запити вказуються на останніх [5].

Прозорість реплікації. З міркувань продуктивності, надійності та доступності зазвичай бажано мати можливість розповсюджувати дані в реплікований спосіб між машинами в мережі. Якщо припустити, що дані копіюються, проблема прозорості полягає в тому, чи повинні користувачі знати про існування копій, чи система повинна керувати копіями, а користувач повинен діяти так, ніби існує одна копія даних.

З точки зору користувача, бажано не займатися обробкою копій і не вказувати, що певну дію можна та/або потрібно виконувати з кількома копіями.

1.3.1 Проектування розподіленої бази даних

Питання, яке вирішується, полягає в тому, як дані розміщуються у вузлах. Відправною точкою є одна глобальна база даних, а кінцевим результатом є розподіл даних між вузлами. Це називається проектуванням зверху вниз.

Існує дві основні альтернативи розміщення даних: розділені (або нерепліковані) і репліковані. У розділеній схемі база даних розділена на кілька непересічних розділів, кожен з яких розміщений на іншому місці.

Репліковані проекти можуть бути або повністю реплікованими (також називаються повністю дубльованими), коли вся база даних зберігається на кожному вузлі, або частково реплікованими (або частково дубльованими), коли кожен розділ бази даних зберігається на кількох вузлах, але не на всіх вузлах.

Двома основними проблемами проектування є фрагментація, поділ бази даних на розділи, які називаються фрагментами, і розподіл, тобто оптимальний розподіл фрагментів [5].

Пов'язаною проблемою є проектування та керування системним каталогом. У централізованих СУБД каталог містить метаінформацію (тобто опис) про дані.

У розподіленій системі ми маємо каталог, який містить додаткову інформацію, наприклад, де розташовані дані.

Проблеми, пов'язані з керуванням каталогом, подібні за своєю природою

до проблеми розміщення бази даних. Каталог може бути глобальним для всієї розподіленої СУБД або локальним для кожного вузла; він може бути централізований на одній ділянці або розподілений на кількох ділянках; може бути одна копія або кілька копій.

1.3.2 Контроль розподілених даних

Важливою вимогою до СУБД є підтримка узгодженості даних, контролюючи спосіб доступу до даних. Це називається контролем даних і включає керування переглядами, контроль доступу та забезпечення цілісності.

Розповсюдження накладає додаткові труднощі, оскільки дані, необхідні для перевірки правил, розподіляються на різні вузли, що вимагає розподіленої перевірки та виконання правил [5].

1.4 Архітектура розподілених баз даних

Архітектура системи визначає її структуру. Це означає, що компоненти системи визначені, функція кожного компонента визначена, а також визначені взаємозв’язки та взаємодії між цими компонентами. Специфікація архітектури системи вимагає ідентифікації різних модулів з їхніми інтерфейсами та взаємозв’язками з точки зору потоку даних і керування через систему.

Існує чотири «еталонні» архітектури для розподіленої СУБД: клієнт/сервер, однорангова мережа, мультибаза даних і хмарові обчислення [4].

Клієнт/сервер архітектура. Загальна ідея дуже проста: відрізнити функціональні можливості, які потрібно надати на сервері, від тих, які потрібно надати на клієнті. Це забезпечує дворівневу архітектуру, яка полегшує керування складністю сучасних СУБД і складністю розподілу.

У реляційних клієнт/серверних СУБД сервер виконує більшу частину роботи з керування даними. Це означає, що вся обробка та оптимізація запитів, керування транзакціями та керування сховищами виконуються на сервері.

Клієнт, на додаток до програми та інтерфейсу користувача, має клієнтський модуль СУБД, який відповідає за керування даними, які кешуються клієнтом, і керування блокуванням транзакцій, які також можуть бути кешованими.

Також можна розмістити перевірку узгодженості запитів користувачів на стороні клієнта, але це не є поширеним явищем, оскільки вимагає реплікації системного каталогу на клієнтських машинах.

Ця архітектура, зображена на рис. 1.1, досить поширена в реляційних системах, де зв'язок між клієнтами та сервером (серверами), відбувається на рівні SQL.

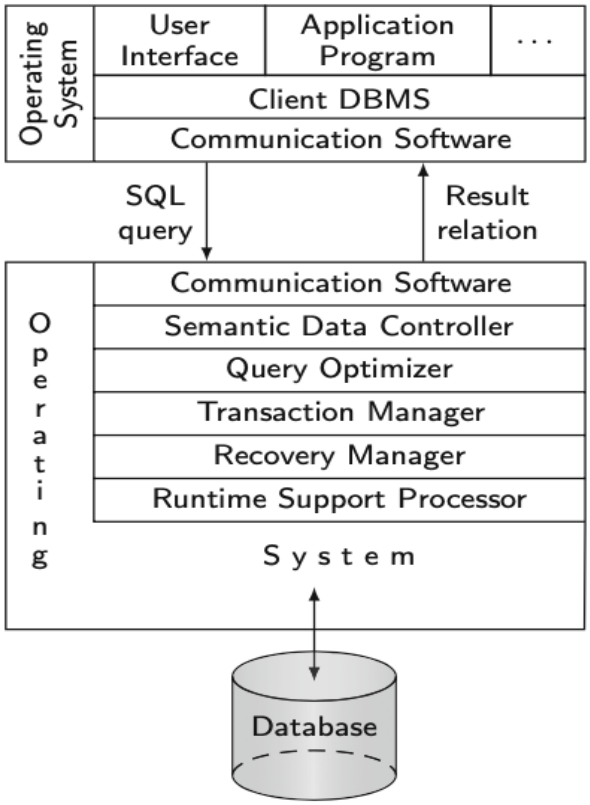


Рисунок 1.1 – Клієнт/сервер архітектура [5]

Існує кілька різних реалізацій архітектури клієнт/сервер. Найпростішим є випадок, коли є лише один сервер, до якого звертаються кілька клієнтів.

Більш складна архітектура клієнт/сервер – це така, у якій у системі є

кілька серверів (так званий підхід із кількома клієнтами та кількома серверами). У цьому випадку можливі дві альтернативні стратегії керування: або кожен клієнт керує своїм власним підключенням до відповідного сервера, або кожен клієнт знає лише про свій «домашній сервер», який потім спілкується з іншими серверами за потреби.

У системах з декількома серверами дані розділені та можуть бути відтворені між серверами. Це прозоро для клієнтів у разі легкого клієнтського підходу, і сервери можуть спілкуватися між собою, щоб відповісти на запит користувача. Цей підхід реалізовано в паралельній СУБД для підвищення продуктивності за рахунок паралельної обробки.

Клієнт/сервер можна природно розширити, щоб забезпечити більш ефективний розподіл функцій на різних типах серверів: клієнти запускають інтерфейс користувача (наприклад, веб-сервери), сервери додатків запускають прикладні програми, а сервери баз даних виконують функції керування базами даних. Це призводить до трирівневої розподіленої архітектури системи [5].

Підхід до сервера додатків (справді, n-рівневий розподілений підхід) можна розширити за допомогою введення кількох серверів баз даних і кількох серверів додатків (рис. 1.2), як це можна зробити в класичних архітектурах клієнт/сервер.

У цьому випадку, як правило, кожен сервер додатків призначений для однієї або кількох додатків, тоді як сервери баз даних працюють у режимі кількох серверів, описаному вище. Крім того, інтерфейс програми зазвичай здійснюється через балансувальник навантаження, який направляє запити клієнта на відповідні сервери.

Незважаючи на те, що ці переваги є значними, існують додаткові накладні витрати, які вводять інший рівень зв’язку між програмою та серверами даних.

Вартість зв’язку може бути амортизована, якщо інтерфейс сервера достатньо високий, щоб дозволити вираження складних запитів, що включають інтенсивну обробку даних.

Одноранговi системи. Ранні роботи над розподіленими СУБД були зосереджені на одноранговых архітектурах, де не було різниці між функціональністю кожного сайту в системі. Сучасні одноранговi системи мають дві важливі відмінності [5].

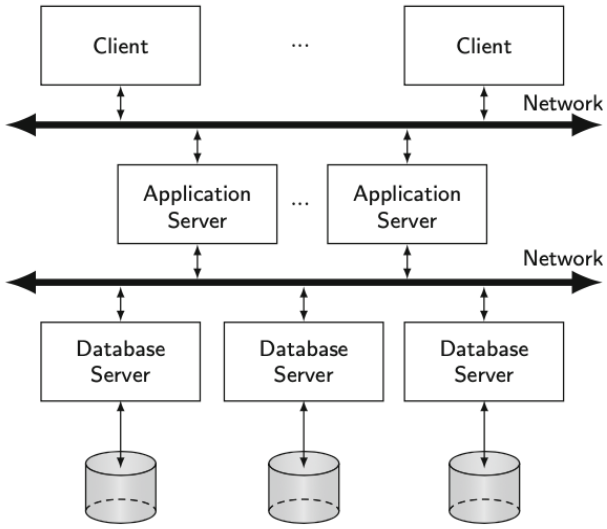


Рисунок 1.2 – Схема розширеного підходу клієнт/сервер архітектури [5]

У цих системах проектування бази даних слідує низхідному дизайну. Це означає, що вхідними даними є (централізована) база даних із власним визначенням схеми (глобальна концептуальна схема – GCS). Ця база даних розбита на розділи та розподілена між вузлами розподіленої СУБД.

Таким чином, на кожному вузлі існує ЛБ даних з власною схемою (так звана локальна концептуальна схема – LCS). Користувач формує запити відповідно до GCS, незалежно від його розташування.

Розподілена СУБД перетворює глобальні запити в групу локальних запитів, які виконуються розподіленими компонентами СУБД на різних вузлах, які спілкуються один з одним.

З точки зору запитів, одноранговi системи та клієнт/серверні СУБД забезпечують однаковий перегляд даних. Тобто вони надають користувачеві вигляд логічно єдиної бази даних, тоді як на фізичному рівні дані розподілені.

Детальні компоненти розподіленої СУБД показані на рис. 1.3. Один компонент відповідає за взаємодію з користувачами, а інший – за зберігання.

Перший основний компонент, а саме процесор користувача, складається з чотирьох елементів [5]:

– обробник інтерфейсу користувача відповідає за інтерпретацію команди користувача, коли вони надходять, і форматування даних результату, коли вони надсилаються користувачеві;

– контролер даних використовує обмеження цілісності та авторизації, визначені як частина глобальної концептуальної схеми, щоб перевірити, чи можна обробити запит користувача;

– глобальний оптимізатор і декомпозитор запитів визначає стратегії виконання для мінімізації функції витрат і переводить глобальні запити в локальні за допомогою глобальної та локальної концептуальних схем, а також глобального каталогу;

– монітор розподіленого виконання координує запити користувача.

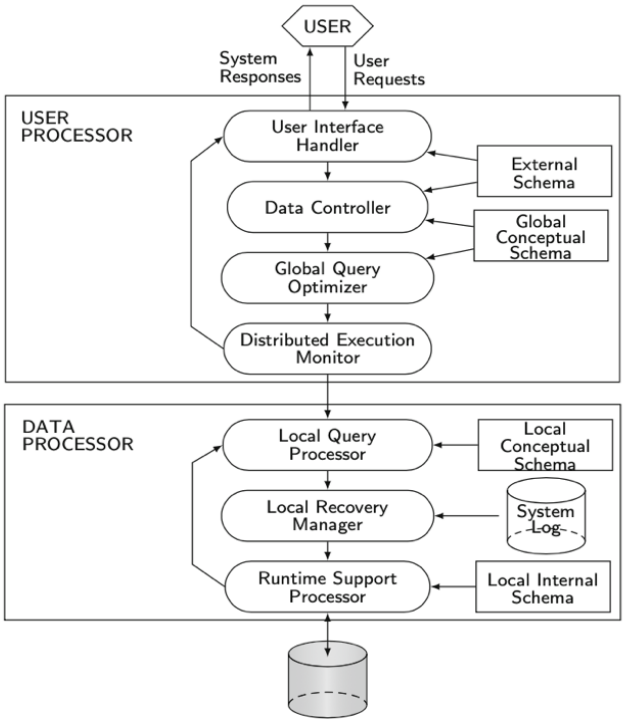


Рисунок 1.3 – Детальні компоненти розподіленої СУБД [5]

Монітор виконання також називають диспетчером розподілених

транзакцій. Виконуючи запити розподіленим способом, монітори виконання на різних сайтах можуть і зазвичай спілкуються один з одним.

Другим основним компонентом розподіленої СУБД є процесор даних і складається з наступних трьох елементів [4]:

– локальний оптимізатор запитів, який фактично діє, як селектор шляху до доступу, відповідає за вибір найкращого шляху доступу для доступу до будь якого елемента даних;

– адміністратор локального відновлення несе відповідальність за те, що локальна база даних залишалася б узгодженою навіть у разі збою;

– процесор підтримки виконання фізично отримує доступ до бази даних відповідно до фізичних команд у розкладі, створеному оптимізатором запитів.

Процесор підтримки виконання є інтерфейсом до операційної системи та містить менеджер буфера бази даних (або кеш-пам’ять), який відповідає за підтримку основних буферів пам’яті та керування доступом до даних.

У однорангових системах очікується, що на кожній машині будуть як модулі процесора користувача, так і модулі процесора даних. Однак можуть існувати «вузли лише для запитів», які мають лише процесор користувача.

Системи управління мультибазами даних. Системи з кількома базами даних (СУБМД) представляють випадок, коли окремі СУБД є повністю автономними і не мають концепції співпраці.

Відмінності в рівні автономності між СУБМД і розподіленими СУБД стосується визначення глобальної концептуальної схеми. У випадку логічно інтегрованих розподілених СУБД глобальна концептуальна схема визначає концептуальний вигляд усієї бази даних, у той час як у випадку СУБМД вона представляє лише набір деяких ЛБ даних, якими кожна локальна СУБД хоче спільно користуватися [5].

Окремі СУБД можуть зробити деякі свої дані доступними для доступу іншим. Таким чином, визначення глобальної бази даних відрізняється в СУБМД, ніж у розподілених СУБД. В останній глобальна база даних дорівнює об’єднанню ЛБ даних, тоді як у першій вона є лише підмножиною того самого

об’єднання. У СУБМД (яка також називається опосередкованою схемою) визначається інтеграцією локальних концептуальних схем.

Компонентна архітектурна модель розподіленої СУБМД істотно відрізняється від розподіленої СУБД, оскільки кожен вузол є повноцінною СУБД, яка керує іншою базою даних.

СУБМД забезпечує рівень програмного забезпечення, яке працює поверх цих окремих СУБД і надає користувачам засоби доступу до різних баз даних (рис. 1.4).

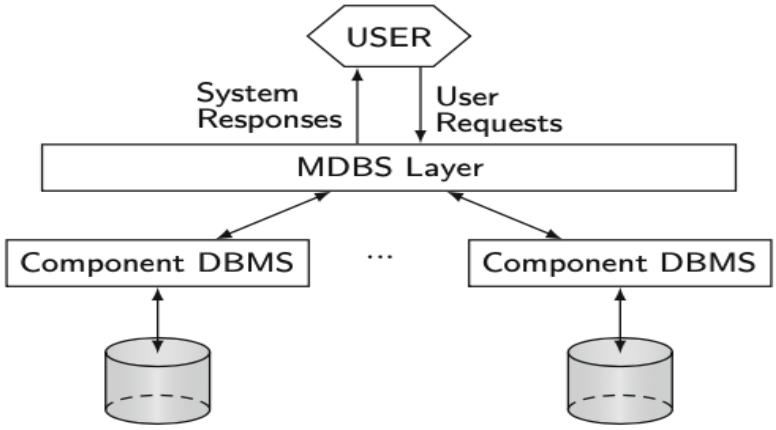


Рисунок 1.4 – Компоненти СУБМД [5]

Розподілена СУБМД може працювати на кількох вузлах або може існувати центральний вузл, де пропонуються ці послуги. Також що, що стосується окремих СУБД, рівень СУБМД – це просто інша програма, яка подає запити та отримує відповіді [5].

Популярною архітектурою реалізації для СУБМД є підхід посередника/обгортки (рис. 1.5).

Посередник «це програмний модуль, який використовує закодовані знання про певні набори або підмножини даних для створення інформації для вищого рівня програм».

Таким чином, кожен посередник виконує певну функцію з чітко визначеними інтерфейсами.

Використовуючи цю архітектуру для реалізації СУБМД, кожен модуль на рівні СУБМД на рис. 1.4 реалізується як посередник.

Оскільки медіатори можуть бути побудовані поверх інших медіаторів, можна побудувати багатошарову реалізацію.

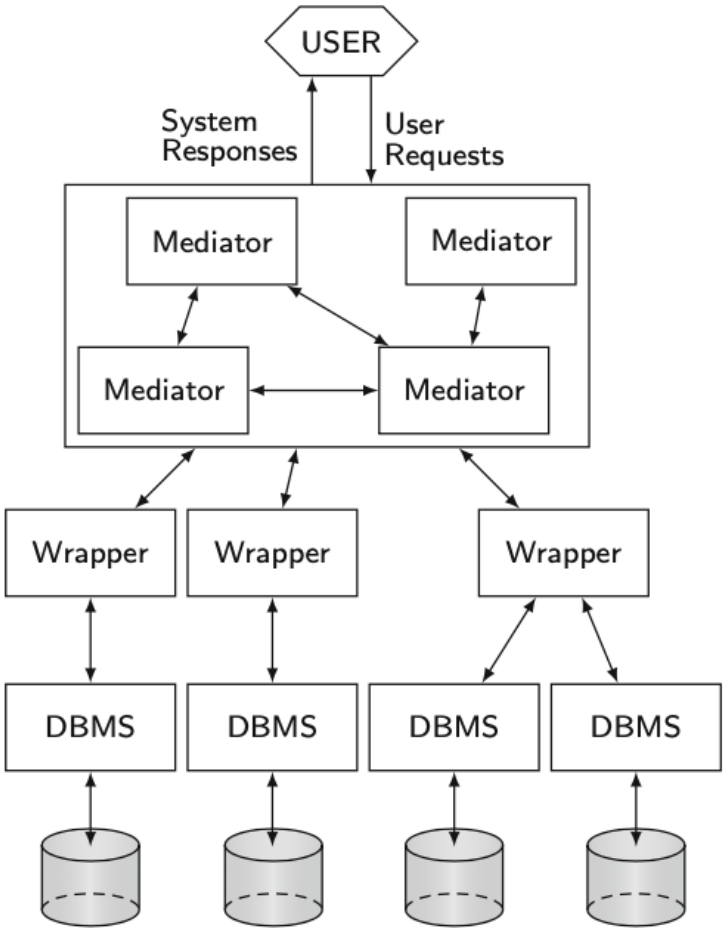


Рисунок 1.5 – Архітектура посередника/обгортки [5]

Посередники зазвичай працюють, використовуючи спільну модель даних і мову інтерфейсу. Щоб мати справу з потенційними неоднорідностями вихідних СУБД, оболонки реалізовані, завдання яких полягає в тому, щоб забезпечити відображення між виглядом вихідної СУБД і видом посередників.

Наприклад, якщо вихідна СУБД є реляційною, але реалізації посередника є об’єктно-орієнтованими, необхідні відображення встановлюються оболонками. Точна роль і функції посередників відрізняються від однієї реалізації до іншої. У деяких випадках посередники лише перекладають; їх

називають «тонкими» посередниками. В інших випадках обгортки беруть на себе виконання деяких функцій запиту.

Хмарні обчислення. Хмарні обчислення — це природна еволюція та комбінація різних обчислювальних моделей, запропонованих для підтримки додатків через Інтернет: сервіс-орієнтовані архітектури (SOA) для високорівневого обміну додатками через веб-сервіси, допоміжні обчислення для пакування обчислювальних ресурсів і ресурсів зберігання, як послуги, технології кластерів і віртуалізації для керування великою кількістю обчислювальних ресурсів і ресурсів зберігання, а також автономні обчислення для забезпечення самостійного керування складною інфраструктурою.

Хмарні обчислення надають різні рівні функціональності, наприклад [4]:

– інфраструктура як послуга (IaaS): надання інфраструктури для

– обчислення (тобто обчислювальних, мережевих і ресурсів зберігання) як послуги;

– платформа як послуга (PaaS): надання обчислювальної платформи разом з інструментами розробки та API як послуга;

– програмне забезпечення як послуга (SaaS): доставка прикладного програмного забезпечення як послуги;

– database-as-a-Service (DaaS): надання бази даних як послуги.

Основними функціями, які забезпечують хмарні обчислення є [4]: безпека, керування каталогами, керування ресурсами (надання, розподіл, моніторинг) і керування даними (зберігання, керування файлами, керування базами даних, реплікація даних). Крім того, хмарні обчислення забезпечують підтримку ціноутворення, бухгалтерського обліку та управління угодами про рівень обслуговування.

Типові переваги хмарних обчислень [5]:

– вартість. Витрати для клієнта можуть бути значно зменшені, тому що інфраструктура не потребує володіння та управління;

– виставлення рахунків здійснюється лише на основі споживання ресурсу;

– простота доступу та використання. Хмарне середовище приховує складність ІТ-інфраструктури та робить розташування та розподіл прозорими. Таким чином, клієнти можуть мати доступ до ІТ-послуг у будь-який час і з будь-якого місця, де є підключення до Інтернету;

– якість обслуговування. Експлуатація ІТ-інфраструктури за допомогою спеціалізованого провайдера, який має великий досвід роботи з дуже великими інфраструктурами (включаючи власну інфраструктуру), підвищує якість обслуговування та операційну ефективність;

– інноваційність. Використання найсучасніших інструментів і додатків, наданих хмарою, заохочує сучасну практику, таким чином збільшуючи інноваційні можливості клієнтів;

– еластичність. Здатність динамічно масштабувати ресурси відповідно до мінливих умов є великою перевагою. Зазвичай це досягається за допомогою віртуалізації сервера, технології, яка дозволяє запускати кілька програм на одному фізичному комп’ютері як віртуальні машини (ВМ), тобто так, ніби вони працюють на різних фізичних комп’ютерах. Тоді клієнти можуть вимагати обчислювальних екземплярів як віртуальних машин і приєднувати ресурси зберігання за потреби.

Однак є й недоліки, які слід добре зрозуміти перед переходом до хмарних обчислень. Ці недоліки подібні до випадків аутсорсингу програм і даних зовнішній компанії [5]:

– залежність від провайдера. Хмарні постачальники, як правило, обмежують доступ клієнтам за допомогою власного програмного забезпечення, власного формату або високих витрат на передачу вихідних даних, що ускладнює міграцію хмарних служб;

– втрата контролю. Клієнти можуть втратити управлінський контроль над критичними операціями, такими як простій системи, наприклад, для виконання оновлення програмного забезпечення;

– безпека. Оскільки хмарні дані клієнта доступні з будь-якої точки Інтернету, атаки на безпеку можуть скомпрометувати дані компанії;

– приховані витрати. Налаштування додатків, щоб зробити їх готовими для використання хмарним середовищем за допомогою SaaS/-PaaS, може призвести до значних витрат на розробку.

Хмарні обчислення, як правило, багатовузлові (рис. 1.6), тобто складається з кількох територіально розподілених вузлів (або центрів обробки даних), кожен зі своїми ресурсами та даними.

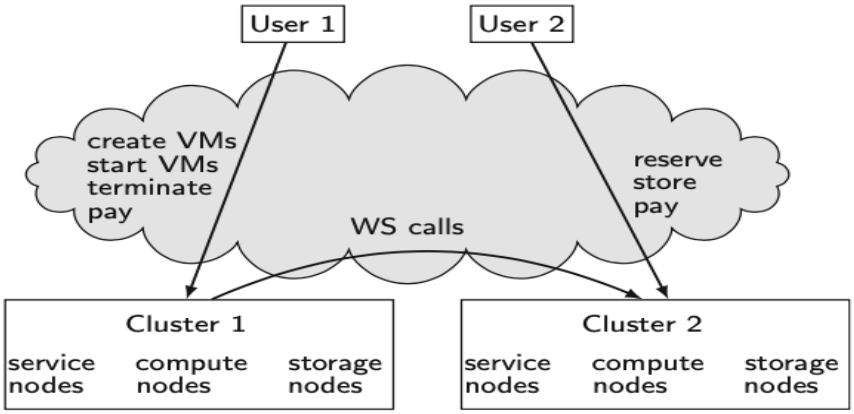


Рисунок 1.6 – Архітектура хмарних обчислень [5]

Архітектура хмарного вузла (центру обробки даних) зазвичай є трирівневою [5].

Перший рівень складається з веб-клієнтів, які отримують доступ до хмарних веб-серверів, як правило, через маршрутизатор або балансувальник навантаження на хмарному сайті.

Другий рівень складається з веб-серверів/серверів додатків, які підтримують клієнтів і забезпечують бізнес-логіку.

Третій рівень складається з серверів баз даних. Можуть існувати інші типи серверів, наприклад, сервери кешу між серверами додатків і серверами баз даних.

Таким чином, хмарна архітектура передбачає два рівні розподілу: географічний розподіл між вузлами, що використовують глобальну мережу та всередині сайту, розподіл між серверами, як правило, у комп’ютерному

кластері.

Техніки, що використовуються на першому рівні, належать до територіально розподілених СУБД, тоді як методи, що використовуються на другому рівні, належать до паралельних СУБД.

1.5 Постановка мети та задач дослідження

Проведений аналіз сучасного стану проблеми моделювання РБД показав, що вони знаходять усе більш широке застосування у системах керування комп'ютерно-інтегрованим виробництвом. З їх використанням забезпечується інформаційна підтримка проєктування й управління ТП, етапів життєвого циклу виробів у CALS-технологіях, інтегрованих систем керування виробництвом.

Для своєчасного та безперебійного забезпечення інформацією елементів системи керування в умовах обмежень на витрати здійснюється оптимізація РБД та їх фізичних структур. Оцінка функціональних характеристик варіантів побудови РБД в процесі їх оптимізації здійснюється засобами математичного моделювання. Це обумовлює актуальність завдань розробки ефективних засобів моделювання процесів функціонування РБД для систем керування.

Метою дослідження є підвищення точності визначення основних характеристик процесу функціонування розподіленої бази даних системи керування комп'ютерно-інтегрованим виробництвом.

Для досягнення мети необхідно вирішити множину завдань:

– провести огляд та аналіз сучасного стану проблеми моделювання РБД у системах керування комп'ютерно-інтегрованим виробництвом;

– сформулювати постановку задачі моделювання РБД у системах керування комп'ютерно-інтегрованим виробництвом;

– обрати метод та побудувати еквівалентну схему процесу обробки запитів в РБД;

– розробити моделювальний алгоритм розв'язання задачі;

– обрати мову програмування;

– обрати середовище розробки програми моделювання;

– розробити програму, яка розв’язує поставлену задачу;

– провести перевірку працездатності програми;

– спланувати, провести модельні експерименти, оцінити точність результатів моделювання РБД;

– розглянути питання безпеки життєдіяльності співробітників у відділі моделювання.

1.6 Висновки до першого розділу

У першому розділі було розглянуто комп’ютерно-інтегроване виробництво як об’єкт керування, процеси, основні компоненти виробництва, системи керування ним, їх призначення, основні задачі сучасних систем керування комп’ютерно-інтегрованим виробництвом. Детально описано цілі та способи моделювання РБД.

Основна увага була приділена опису архітектури розподілених баз даних, а саме чотирьом «еталонним» архітектурам для розподіленої СУБД таким як, клієнт/сервер, однорангова мережа, мультибаза даних і хмарові обчислення.

На цій основі дійшли висновку, що використання розподілених баз даних необхідне для систем керування комп’ютерно-інтегрованим виробництвом, а для оптимізації РБД актуальною задачею є моделювання процесу їхнього функціонування.

2 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ ЗАДАЧІ МОДЕЛЮВАННЯ ПРОЦЕСУ ФУНКЦІОНУВАННЯ РОЗПОДІЛЕНИХ БАЗ ДАНИХ

2.1 Розподіл обробки запитів

Приховуючи низькорівневі деталі фізичної організації даних, мови реляційних баз даних дозволяють виражати складні запити в стислому та простому вигляді. Зокрема, щоб побудувати відповідь на запит, користувач точно не вказує процедуру, якій слід слідувати, ця процедура насправді розроблена модулем, який називається процесором запитів.

Це звільняє користувача від оптимізації запитів, трудомісткого завдання, яке найкраще впорається з процесором запитів, оскільки він може використовувати велику кількість корисної інформації про дані. Ця критична проблема продуктивності, обробка запитів отримала значну увагу в контексті як централізованих, так і розподілених СУБД [7].

Однак проблема обробки запитів набагато складніша в розподілених середовищах, оскільки більша кількість параметрів впливає на продуктивність розподілених запитів. Зокрема, зв’язки, задіяні в розподіленому запиті, можуть бути фрагментованими та/або відтвореними, що спричиняє витрати на зв’язок. Крім того, з великою кількістю доступних вузлів час відповіді на запит може стати дуже високим.

Основною функцією процесора запитів є перетворення запиту високого рівня (як правило, у реляційному численні) в еквівалентний запит нижчого рівня (як правило, у певному варіанті реляційної алгебри). Запит низького рівня фактично реалізує стратегію виконання для запиту.

Трансформація повинна досягати як правильності, так і ефективності. Це правильно, якщо низькорівневий запит має таку саму семантику, як вихідний запит, тобто якщо обидва запити дають однаковий результат.

Чітко визначене відображення від реляційного числення до реляційної алгебри спрощує проблему правильності. Але розробити ефективну стратегію

виконання складніше. Запит на реляційне числення може мати багато еквівалентних і правильних перетворень у реляційну алгебру [8].

Оскільки кожна еквівалентна стратегія виконання може призвести до дуже різного споживання комп'ютерних ресурсів, основна складність полягає у виборі стратегії виконання, яка мінімізує споживання ресурсів.

Оптимізація запиту відноситься до процесу створення плану виконання запиту (QEP), який представляє стратегію виконання для запиту. Цей QEP мінімізує функцію об'єктивної вартості. Оптимізатор запитів, програмний модуль, який виконує оптимізацію запитів, зазвичай складається з трьох компонентів: простору пошуку, моделі витрат і стратегії пошуку [9].

Сама проблема обробки запитів може бути розкладена на кілька підпроблем, що відповідають різним рівням. На рис. 2.1 показана загальна схема шарів для обробки запитів, де кожен рівень вирішує чітко визначену підпроблему.

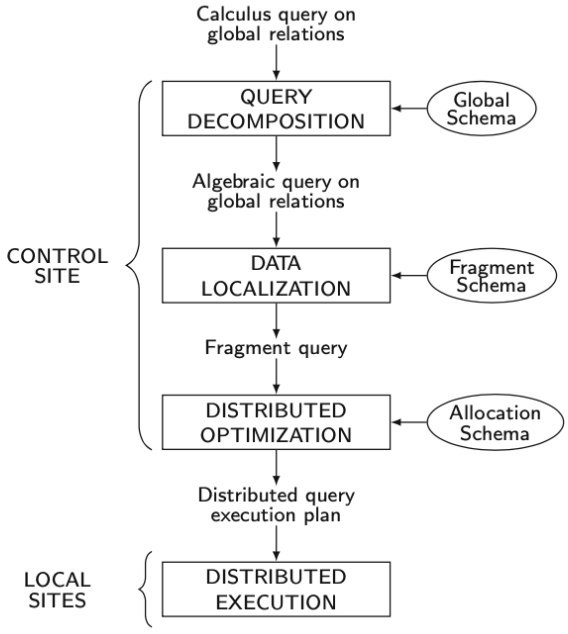


Рисунок 2.1 – Загальна схема розбиття розподіленої обробки запиту на рівні [5]

Вхідними даними є запитом на глобальні дані, виражені в реляційному численні. Цей запит створюється для глобальних (розподілених) зв'язків, тобто розподіл даних прихований.

Чотири основні рівні задіяні в обробці розподілених запитів. Перші три рівні відображають вхідний запит у розподілений план виконання запиту (розподілений QEP). Вони виконують функції декомпозиції запитів, локалізації даних і глобальної оптимізації запитів.

Декомпозиція запиту та локалізація даних відповідають перезапису запиту. Перші три рівні виконуються центральним вузлом керування та використовують інформацію про схему, що зберігається в глобальному каталозі. Четвертий рівень виконує розподілене виконання запиту шляхом виконання плану та повертає відповідь на запит. Це роблять локальні вузли та контрольний вузол [8].

Декомпозиція запиту. Перший рівень розкладає запит на обчислення на алгебраїчний запит щодо глобальних відносин. Інформація, необхідна для цієї трансформації, міститься в глобальній концептуальній схемі, що описує глобальні відносини. Однак інформація про розподіл даних використовується не тут, а на наступному рівні. Техніки, які використовує цей рівень, є методами централізованої СУБД.

Декомпозицію запиту можна розглядати як чотири послідовні кроки.

По-перше, запит на обчислення переписується в нормалізованій формі, придатній для подальших маніпуляцій. Нормалізація запиту зазвичай передбачає маніпулювання кванторами запиту та кваліфікацією запиту шляхом застосування пріоритету логічного оператора.

По-друге, нормалізований запит аналізується семантично, щоб некоректні запити виявлялися та відхилялися якомога раніше. Запит вважається семантично неправильним, якщо його компоненти жодним чином не сприяють формуванню результату. У контексті реляційного числення неможливо визначити семантичну коректність загальних запитів. Однак це можливо для великого класу реляційних запитів, тобто тих, які не містять диз’юнкції та заперечення [8]. Це засновано на представленні запиту у вигляді графа, який називається графом запиту або графом з’єднання. Цей граф використовується для найкорисніших типів запитів, що містять оператори вибору, проекції та

об’єднання. У графі запиту один вузол вказує на результат відношення, а будь-який інший вузол вказує на відношення операнда. Ребро між двома вузлами, які не є результатами, представляє об’єднання, тоді як ребро, кінцевим вузлом якого є результат, представляє проект.

Крім того, нерезультуючий вузол може бути позначений предикатом вибору або самооб’єднання (об’єднання відношення з самим собою). Важливим підграфом графа запиту є граф об’єднання, у якому розглядаються лише об’єднання [9].

По-третє, правильний запит (все ще виражений реляційним численням) спрощується. Одним із способів спростити запит є усунення зайвих предикатів.

Надлишкові запити можуть виникати, коли запит є результатом системних перетворень, застосованих до запиту користувача.

По-четверте, запит на обчислення реструктуровано як алгебраїчний запит. Якість алгебраїчного запиту визначається в термінах очікуваної продуктивності. Традиційний спосіб зробити це перетворення до «кращої» алгебраїчної специфікації полягає в тому, щоб почати з початкового алгебраїчного запиту та трансформувати його, щоб знайти «хороший» [10].

Початковий алгебраїчний запит одразу виводиться із запиту на обчислення шляхом перекладу предикатів і цільового оператора в оператори відношення, як вони з’являються в запиті. Цей безпосередньо перекладений запит потім реструктурується за допомогою правил перетворення. Алгебраїчний запит, створений цим рівнем, хороший у тому сенсі, що зазвичай уникають гірших виконання. Наприклад, до відношення буде доступ лише один раз, навіть якщо є кілька предикатів вибору. Однак цей запит, як правило, далекий від забезпечення оптимального виконання, оскільки інформація про розподіл даних і розміщення фрагментів не використовується на цьому рівні [10].

Локалізація даних. Вхідними даними для другого рівня є алгебраїчний запит про глобальні відносини. Основна роль другого рівня полягає в локалізації даних запиту за допомогою інформації про розподіл даних у схемі

фрагмента. Цей рівень визначає, які фрагменти беруть участь у запиті, і перетворює розподілений запит у запит на фрагменти. Фрагментація визначається правилами фрагментації, які можна виразити як оператори відношення.

Глобальне відношення можна реконструювати шляхом застосування правил фрагментації, а потім отримання програми, яка називається програмою матеріалізації, операторів реляційної алгебри, яка потім діє на фрагменти [11].

Локалізація складається з двох етапів. По-перше, запит відображається у запит фрагмента шляхом заміни кожного відношення його програмою матеріалізації. По-друге, запит на фрагменти спрощено та реструктуризовано для створення іншого запиту. Спрощення та реструктуризація може виконуватися за тими ж правилами, які використовуються в шарі декомпозиції. Як і на рівні декомпозиції, кінцевий запит фрагмента, як правило, далекий від оптимального, оскільки інформація щодо фрагментів не використовується.

Розподілена оптимізація. Вхідними даними третього рівня є алгебраїчний запит на фрагменти, тобто запит на фрагменти [12].

Метою оптимізації запиту є пошук стратегії виконання для запиту, яка є близькою до оптимальної. Стратегію виконання для розподіленого запиту можна описати за допомогою операторів реляційної алгебри та комунікаційних примітивів (операторів надсилання/отримання) для передачі даних між вузлами.

Попередні рівні вже оптимізували запит, наприклад, видаливши зайві вирази. Однак ця оптимізація не залежить від характеристик фрагментів, таких як розподіл фрагментів і кількість елементів. Крім того, оператори зв'язку поки не вказані [12]. Переставляючи порядок операторів у межах одного запиту на фрагменти, можна знайти багато еквівалентних запитів.

Оптимізація запиту полягає в пошуку «найкращого» порядку операторів у запиті, включаючи операторів зв’язку, який мінімізує функцію витрат. Функція вартості, часто визначена в термінах одиниць часу, відноситься до використання обчислювальних ресурсів, таких як диск, вартість ЦП і мережа.

Як правило, це зважена комбінація витрат на введення/виведення, ЦП і зв’язок.

Для вибору впорядкування операторів необхідно передбачити витрати на виконання альтернативних упорядкувань кандидатів. Визначення вартості виконання перед виконанням запиту (тобто статична оптимізація) базується на статистиці фрагментів і формулах для оцінки потужності результатів операторів відношення [13].

Таким чином, рішення щодо оптимізації залежать від розподілу фрагментів і доступної статистики щодо фрагментів, які реєструються в схемі розподілу.

Важливим аспектом оптимізації запитів є впорядкування об’єднань, оскільки перестановки об’єднань у запиті можуть призвести до покращень на порядки.

Результатом рівня оптимізації запиту є оптимізований алгебраїчний запит із операторами зв’язку, включеними до фрагментів [13]. Зазвичай він представляється та зберігається (для майбутніх виконань) як розподілений QEP.

Розподілене виконання. Останній рівень виконується всіма вузлами, які мають фрагменти, залучені до запиту. Кожен підзапит, що виконується на одному вузлі, називається локальним запитом, оптимізується за допомогою локальної схеми вузла та виконується. У цей час можуть бути обрані алгоритми для виконання операторів відношення. Локальна оптимізація використовує алгоритми централізованих систем [14].

Класична реалізація операторів відношення в системах баз даних базується на моделі ітератора, яка забезпечує конвеєрний паралелізм у межах дерев операторів. Це проста модель витягування, яка виконує оператори, починаючи від кореневого вузла оператора (який створює результат) до кінцевих вузлів (які отримують доступ до базових зв’язків) [14].

Таким чином, проміжні результати операторів не потрібно матеріалізувати, оскільки кортежі створюються на вимогу та можуть бути використані наступними операторами. Однак це вимагає, щоб оператори були реалізовані в конвеєрному режимі з використанням інтерфейсу відкриття-наступного-закриття. Кожен оператор має бути реалізований як ітератор із

трьома функціями [4]:

– open(): ініціалізує внутрішній стан оператора, наприклад, виділяє хеш таблицю;

– next(): створює та повертає наступний кортеж результатів або null;

– close(): очищає всі виділені ресурси після обробки всіх кортежів.

Таким чином, ітератор забезпечує компонент ітерації циклу while, тобто ініціалізацію, приріст, умову завершення циклу та остаточне очищення. Виконання QEP відбувається наступним чином. По-перше, виконання ініціалізується викликом Open() кореневого оператора дерева операторів, який потім пересилає виклик Open() через весь план за допомогою самих операторів. Тоді кореневий оператор ітеративно створює свій наступний запис результату, пересилаючи виклик Next() через дерево операторів за потреби. Виконання завершується, коли останній виклик Open() повертає кореневому оператору «end» [15].

2.2 Постановка задачі моделювання РБД

Об’єктом дослідження у роботі є розподілені бази даних систем керування комп’ютерно-інтегрованим виробництвом, у яких є множина користувачів, що відправляють свої запити для пошуку інформації. Кожен запит потрапляє до загальної черги очікування. Відомі інтенсивності надходження запитів до локальних баз та пропускна здатність каналів зв’язку. Будемо вважати, що всі запити приблизно мають однакову довжину. Час передачі запиту складає $\tau \pm \Delta\tau$ одиниць часу [16].

Після очікування у черзі, до кожної локальної бази даних надходять запити від користувачів. Після надходження запит реєструється та очікує у черзі, до конкретної локальної бази даних [16]. Після очікування запит надходить до локальної бази даних на обробку.

Під час обробки запиту з заданою ймовірністю, виникають ситуації, коли не вистачає даних для повної відповіді на запит, тому відбувається

перенаправлення запиту до інших локальних баз даних з метою пошуку більш детальної інформації (рис. 2.2). Пошук додаткової інформації відбувається одночасно в усіх локальних базах даних. Після перенаправлення відбувається повторна обробка запиту. Якщо запит оброблено від залишає систему.

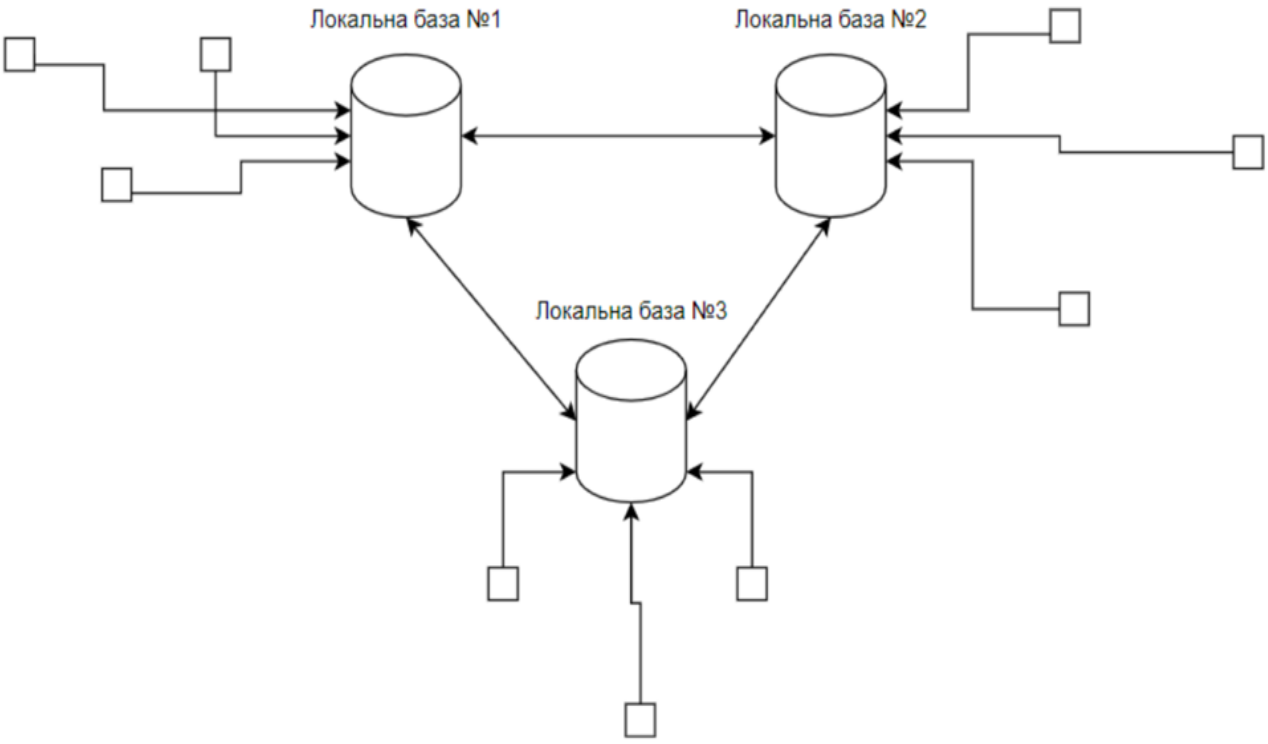


Рисунок 2.2 – Концептуальна модель РБД для трьох локальних баз даних

Розглядається задача оцінки часу доступу до інформаційних ресурсів (ІР), що розв'язується на етапі оптимізації РБД в процесі її проєктування (реінжинірингу) [16]. Задані: множина корпоративних користувачів РБД, що розміщуються у n вузлах комп'ютерної мережі; кількість ІР локальних баз m і місця їх розміщення у вузлах мережі; розподіл ІР по вузлах мережі x = [x_{ij}] (де x_{ij} – булева змінна: x_{ij} = 1, якщо в і-му вузлі зберігається j-й ІР; x_{ij} = 0 – в іншому випадку); інтенсивності (закони розподілу та параметри) надходження запитів від користувачів до ІР λ_{ij}, $i = \overline{1,n}$, $i = \overline{1,m}$; час обробки (закони розподілу та параметри) запитів до ІР $t_{ij}^{qp}(x)$; пропускні здатності каналів мережі h = [h_{ij}], $i,j = \overline{1,n}$; обсяги запитів користувачів до ІР баз a_{ij}, $i = \overline{1,n}$,

$j = \overline{1,m}$ і відповідей на запити b_{ij}, $i = \overline{1,n}$, $i = \overline{1,m}$).

Для заданого інтервалу моделювання необхідно визначити оцінки кількість запитів, що були оброблені та необроблені, та часу доступу користувачів до ІР РБД t(x).

Вимоги до розроблюваного засобу моделювання розподіленої бази даних системи керування комп'ютерно-інтегрованим виробництвом можна подати у такому вигляді.

Призначення розробки. Додаток призначений для підвищення точності визначення основних характеристик процесу функціонування розподіленої бази даних системи керування комп'ютерно-інтегрованим виробництвом.

Користувачем системи, що розроблюється є розробники розподіленої бази даних системи керування комп'ютерно-інтегрованим виробництвом.

Система використовується під час інтегрування розподіленої бази даних до виробництва або для вдосконалення вже існуючої БД [17].

Система, що була розроблена є універсальною та може використовуватись будь-якими виробництвами, які впроваджують розподілену базу даних [17].

Характеристика розробки. Програмне забезпечення, що буде реалізовано кросплатформений додаток, реалізований мовою програмування – JavaScript. За рахунок такої мови програмування додаток є універсальним.

Мета створення системи. З боку виробництва:

– оптимізація зберігання даних різних підрозділів виробництва;

– зменшення часу очікування, після звертання до локальної бази даних;

– можливість обміну запитами між локальними БД у випадку не компетентності БД до якої надійшло звернення.

З боку розробника розподіленої бази даних [18]:

– можливість задавати необхідну кількість баз даних та час роботи;

– отримання детального звіту після відпрацювання системи.

Основні функціональні можливості системи. Для розробників розподіленої бази даних [18]:

– можливість оцінити час доступу до ІР РБД;

– можливість оцінити завантаженість обладнання локальних баз.

Для власників комп’ютерно-інтегрованого виробництва:

– дана технологія дає можливість оптимізації зберігання даних в різних локальних базах даних;

– можливість керувати часом роботи системи;

– здійснення обміну даними між різними локальними базами даних, якщо локальна БД не може дати відповідь на запит, що прийшов;

– можливість отримати, після завершення роботи системи, звіт з повною інформацією про роботу системи.

Нефункціональні вимоги. Підтримка операційних систем: розроблена система має коректно працювати на таких операційних системах як, Windows 7 або вищі версії, IOS та Linux [19].

Вимоги до продуктивності:

– система має працювати на перелічених операційних системах;

– система повинна працювати при різній кількості локальних БД;

– система має опрацьовувати кожен запит, що надходить до системи;

– система має обробляти запит за час, який задано у вхідних даних;

– система має генерувати звіт після закінчення часу роботи.

Вхідні дані. Кількість локальних баз даних та час роботи системи – задаються користувачем.

Вихідні дані. Сформовано звіт в якому зазначено кількість запитів, що надійшли до системи, час роботи системи, кількість запитів, що було оброблено, кількість запитів, що були оброблені після обміну між локальними БД, кількість запитів, що не встигла обробити система через брак часу [19].

2.3 Концептуальна модель розподіленої бази даних

Створення розподіленої бази даних – це процес, який починається з набору вимог і призводить до визначення схеми, яка визначає набір зв’язків.

Дизайн розподілу починається з концептуальної схеми і виконує два завдання: розділення (фрагментація) і розподіл.

Деякі методи поєднують ці два завдання в одному алгоритмі, а інші реалізують їх у двох окремих алгоритмах [20].

Існує дві основні причини та цілі фрагментації в розподілених СУБД. У першому випадку основною причиною є локальність даних. Необхідно щоб запити отримували доступ до даних на одному вузлі, аби уникнути дорогого віддаленого доступу до даних. Друга основна причина полягає в тому, що фрагментація дає змогу одночасному виконанню кількох запитів (через паралелізм між запитами).

Фрагментація зв’язків також призводить до паралельного виконання одного запиту шляхом поділу його на набір підзапитів, які працюють над фрагментами [20].

Таким чином, у розподілених СУБД фрагментація може потенційно зменшити дорогий віддалений доступ до даних і підвищити паралелізм між і всередині запитів.

Фрагментація важлива для продуктивності системи, але вона також створює труднощі в розподілених СУБД. Не завжди можливо повністю локалізувати запити та транзакції для доступу до даних лише на одному вузлі – це називається розподіленими запитами та розподіленими транзакціями [21].

Їх обробка призводить до зниження продуктивності через, наприклад, необхідність виконання розподілених об’єднань і вартість зобов’язань розподілених транзакцій.

Один із способів подолати це зниження продуктивності для запитів реалізувати тиражування даних на кількох вузлах, але це ще більше посилює накладні витрати на розподілені транзакції. Друга проблема пов'язана з семантичним контролем даних, зокрема з перевіркою цілісності. В результаті фрагментації атрибути, що беруть участь в обмеженні, можуть бути розкладені на різні фрагменти, які розподіляються на різні вузли [22].

У цьому випадку сама перевірка цілісності передбачає розподілене

виконання, що є дорогим.

Таким чином, завдання полягає в тому, щоб розділити і виділити дані таким чином, щоб більшість запитів і транзакцій користувачів були локальними для одного вузла, мінімізуючи розподілені запити та транзакції.

На рисунку 2.2 представлена концептуальна модель розподіленої бази даних для трьох локальних баз даних.

У роботі розглядається та досліджується процес функціонування розподіленої бази даних комп'ютерно-інтегрованого виробництва. Система складається з заданої множини локальних БД [22]. Під час роботи системи користувач надсилає запит до необхідної локальної бази даних (це відображено на рисунку 2.2). Кожен запит, що надійшов до окремої локальної бази даних обробляється, але може виникнути ситуація, що даних для відповіді не вистачає, тому відбувається обмін з іншими локальними базами даних для повної відповіді на запит [23].

Час обробки кожного запиту в локальній базі даних при первинній та повторній обробці визначається складністю запиту та кількістю запитів, що надійшли до системи.

2.4 Структурна схема моделі та її опис

Для того щоб описати процеси системи керування розподіленої бази даних використовують структурні схеми. Структурна схема є для кожного проектованого об'єкта основним проектним документом [24].

У структурній схемі управління та контролю відображаються особливості технологічного характеру даного виробництва, а також технічні засоби, що використовуються при створенні локальних систем контролю та автоматизації.

Структурна схема моделювання розподіленої бази даних представлена на рисунку 2.3.

Об'єктом дослідження є імітаційна модель розподіленої бази даних системи керування комп'ютерно-інтегрованим виробництвом, на якій задана

множина локальних баз даних та час роботи системи [24].

Є множина користувачів, які відправляють свої запити до системи. Кожен запит потрапляє до загальної черги очікування. Відомі інтенсивності надходження запитів до локальних баз та пропускна здатність каналів зв'язку. Будемо вважати, що всі запити приблизно мають однакому довжину [25]. Час передачі запиту складає $\tau \pm \Delta\tau$ одиниць часу (рис. 2.3).

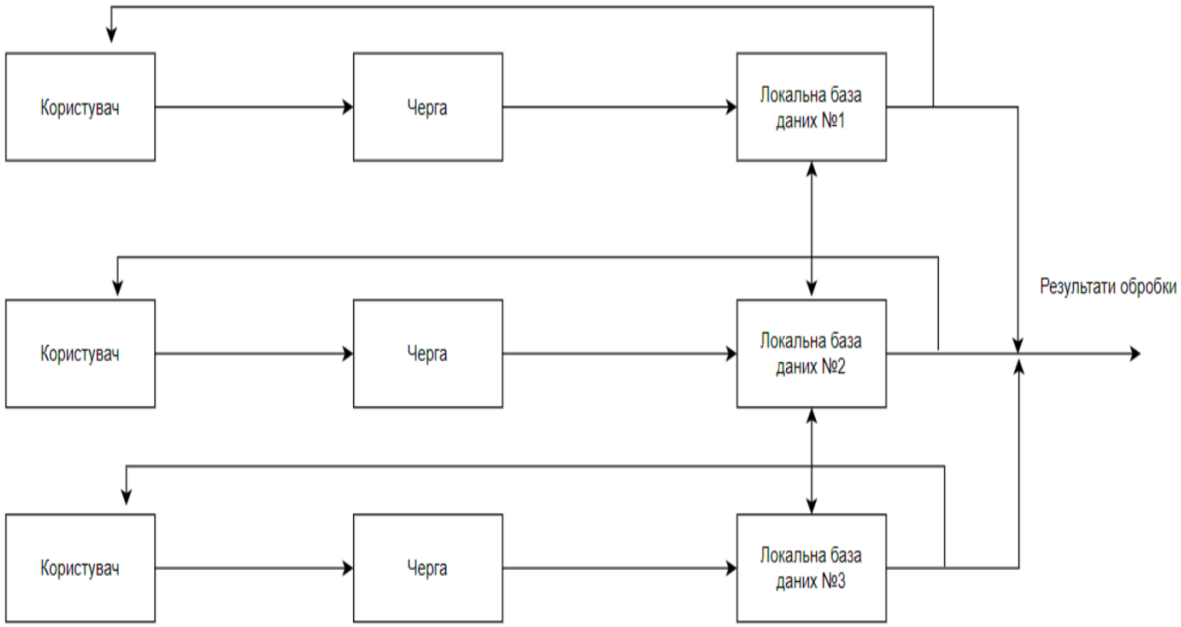


Рисунок 2.3 – Структурна схема РБД з трьома локальними базами

Після очікування у черзі, до кожної локальної бази даних надходять запити від користувачів. Через задану певну кількість секунд, після надходження, запит реєструється та очікує у черзі, до конкретної локальної бази даних. Після очікування запит надходить до локальної бази даних на обробку [25].

Під час обробки запиту з заданою ймовірністю, виникають ситуації, коли не вистачає даних для повної відповіді на запит, тому відбувається перенаправлення запиту до інших локальних баз даних для пошуку більш детальної інформації. Пошук додаткової інформації відбувається одночасно в усіх локальних базах даних. Після перенаправлення відбувається обробка

запиту, але вже повторна. Якщо запит оброблено від залишає систему [25].

2.5 Q-схема системи та її опис

Системою масового обслуговування (СМО) є будь-яка система призначена для обслуговування потоку заявок. Системи масового обслуговування є класом математичних схем, розроблених у теорії масового обслуговування, і різних додатків для формалізації процесів функціонування систем, які за своєю суттю є процесами обслуговування [26].

Характерним для роботи таких об’єктів є стохастичний характер процесу їхнього функціонування, тобто випадкова поява заявок (вимог) на обслуговування і завершення обслуговування у випадкові моменти часу. Засоби, що обслуговують вимоги називаються обслуговуючими пристроями чи каналами обслуговування.

Предметом теорії масового обслуговування є встановлення залежності між факторами, що визначають функціональні можливості системи масового обслуговування, та ефективністю її функціонування [26]. Основним завданням теорії СМО є вивчення режиму функціонування обслуговуючої системи та дослідження явищ, що виникають у процесі обслуговування.

Однією з характеристик обслуговуючої системи є час перебування вимоги у черзі [26]. Цей час можна скоротити за рахунок збільшення кількості обслуговуючих пристроїв. Однак кожен додатковий пристрій потребує певних матеріальних витрат, при цьому збільшується час бездіяльності обслуговуючого пристрою через відсутність вимог на обслуговування, що також є негативним явищем.

У багатоканальній СМО потік заявок, що надходять у систему, обслуговується ідентичними каналами (рис. 2.4). Вибір каналу, що обслуговуватиме чергову заявку, може здійснюватися у такій системі за різними правилами: перший, що звільнився; за жеребкуванням; за мінімальним коефіцієнтом завантаження тощо. Моделювальний алгоритм багатоканальної

СМО має процедуру визначення каналу, який обслуговуватиме чергову заявку.

Багатоканальні системи можуть складатися з обслуговуючих пристроїв як однакової, так і різної продуктивності. За часом перебування вимог у черзі до початку обслуговування системи діляться на три групи [27]: з очікуванням; з відмовами; змішаного типу.

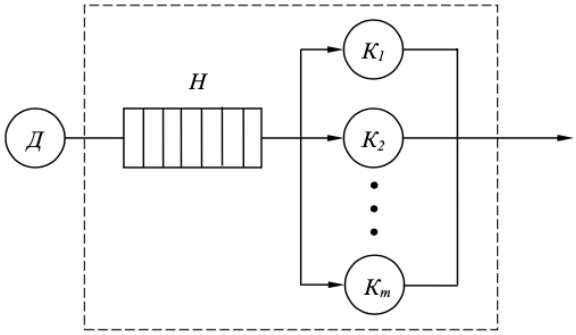


Рисунок 2.4 – Структурна схема багатоканальної СМО

Основними елементами СМО є: вхідний потік вимог; черга вимог; обслуговуючі пристрої (канали); вихідний потік вимог.

Безперервно-стохастичний підхід у моделюванні ґрунтується на застосуванні теорії систем масового обслуговування (СМО). Підхід виражається у побудові Q-схем.

Елементарна Q-схема зображена на рисунку 2.5.

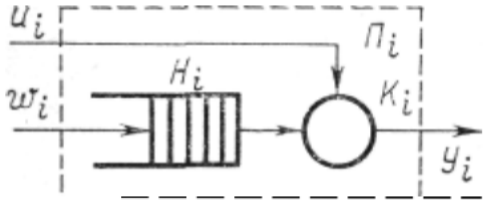


Рисунок 2.5 – Типовий вигляд Q-схеми найпростішої системи

Вона складається з каналу (приладу) обслуговування заявок (вимог), що надходять на його вхід, та черги або накопичувача. У будь-якому елементарному системі обслуговування можна виділити дві основні складові:

очікування обслуговування та обслуговування заявки [27].

Елементарна Q-схема системи керування комп'ютерно-інтегрованим виробництвом складається з: приладу обслуговування $П_i$; накопичувача заявок H_i, в якому одночасно може перебувати декілька заявок; каналу обслуговування заявок (або каналу) K_i.

На кожен елемент приладу обслуговування $Пі$ надходять потоки подій:

– потік заявок w_i в накопичувач H_i;

– потік обслуговування u_i на канал K_i.

Проектування розподілених баз даних обумовлює розв'язання множини різних взаємопов'язаних задач, які умовно можна об'єднати у етапи концептуального, фізичного та логічного проектування [11].

Під час проведення кожного етапу відбувається синтез та аналіз проектних рішень за деякою множиною функціональних та вартісних показників.

Реінжиніринг розподіленої бази даних чи ефективність функціонування зазвичай визначаються фізичною структурою. Саме це є причиною вирішувати множини задач їх топологічною оптимізацією за участю традиційних задач проектування БД.

Структура РБД, як територіально розподілений об'єкт та його властивості можуть бути подані у вигляді [16]:

$$s=<E,R,G>, \quad \varphi:(E,R,G)\rightarrow P(s), \tag{2.1}$$

де E,R,G – відповідно, множини елементів структури, зв'язків між елементами та топологія елементів і зв'язків РБД, які визначають її фізичну реалізацію на комп'ютерній мережі (топологію вузлів мережі й інформаційних ресурсів (ІР) бази G_E, каналів G_R і маршрутів передачі інформації G_A в РБД, $G=<G_E,G_R,G_A>$);

$P(s)$ – множина функціональних і вартісних характеристик РБД зі структурою s.

Обов'язковим етапом розв'язання кожної з задач є оцінка часу доступу до

інформаційних ресурсів РБД $t(s) \in P(s)$ [10]. У процесі визначення такої оцінки процес функціонування РБД подається у вигляді системи масового обслуговування (Q-схеми) [16]:

$$Q=<W,U,H,Z,R,A>, \tag{2.2}$$

де W – вхідний потік запитів;

U – потік обслуговувань;

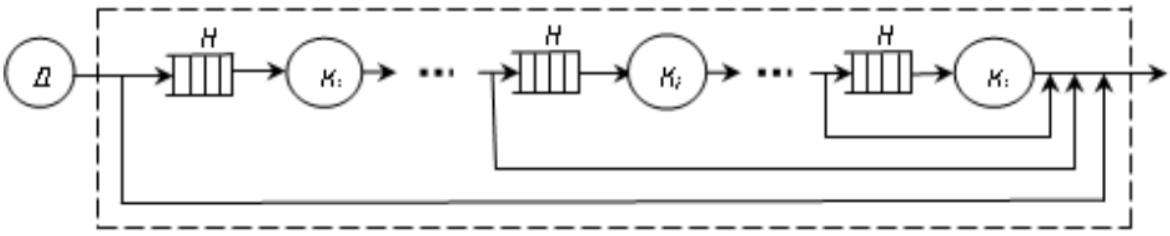
H – множина внутрішніх параметрів системи;

Z – множина станів системи;

R – схема зв'язків елементів системи;

A – алгоритм функціонування системи.

Побудуємо Q-схему згідно структурній схемі (рис. 2.3). Побудована Q-схема для моделювання розподіленої бази даних представлена на рисунку 2.6.



$Д$ – джерело запитів;

H_i, H_j – накопичувачі для черг запиті відповідей у вузлах комп'ютерної мережі;

K_i , K_j – канали обслуговування вузлів видачі запиту та розташування IP

Рисунок 2.6 – Фрагмент багатофазної СМО для моделювання РБД

Об'єктом дослідження є імітаційна модель розподіленої бази даних системи керування комп'ютерно-інтегрованим виробництвом, на якій задана множина локальних баз даних та час роботи системи.

Є множина користувачів, які відправляють свої запити до системи. Кожен запит потрапляє до загальної черги очікування [28]. Відомі інтенсивності надходження запитів до локальних баз та пропускна здатність каналів зв'язку.

Будемо вважати, що всі запити приблизно мають однакому довжину. Час передачі запиту складає $\tau \pm \Delta\tau$ одиниць часу.

Після очікування у черзі, до кожної локальної бази даних надходять запити від користувачів [16]. Через задану певну кількість секунд, після надходження, запит реєструється та очікує у черзі, до конкретної локальної бази даних. Після очікування запит надходить до локальної бази даних на обробку.

Під час обробки запиту з заданою ймовірністю, виникають ситуації, коли не вистачає даних для повної відповіді на запит, тому відбувається перенаправлення запиту до інших локальних баз даних для пошуку більш детальної інформації (приклад роботи системи приведено на рис. 2.7). Пошук додаткової інформації відбувається одночасно в усіх локальних базах даних. Після перенаправлення відбувається обробка запиту, але вже повторна. Якщо запит оброблено від залишає систему [28].

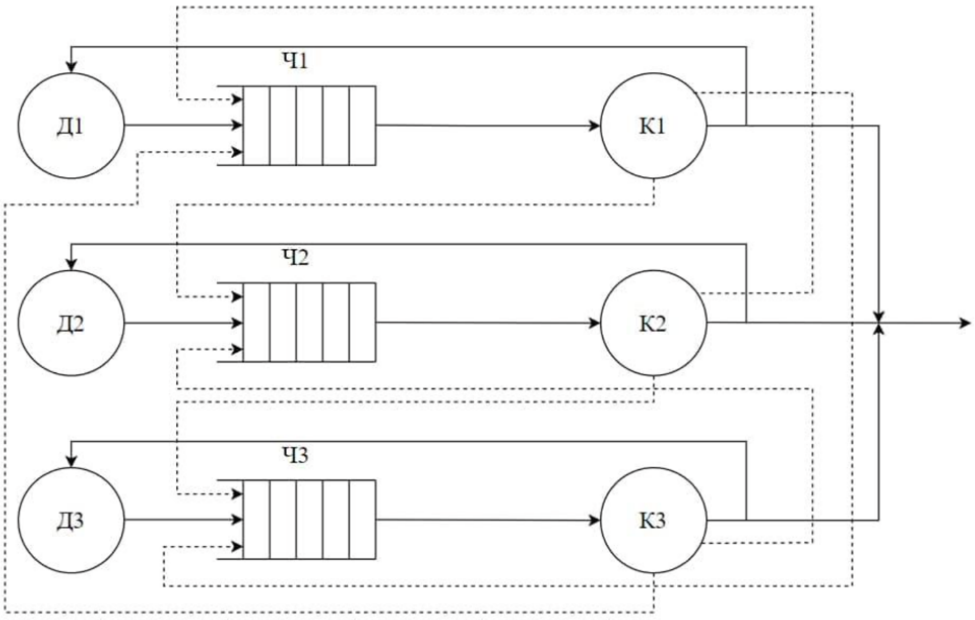


Рисунок 2.7 – Q-схема РБД з трьома локальними базами

Тривалість роботи локальної бази даних при первинній та повторній обробці запиту займає певну кількість часу.

2.6 Опис моделювального алгоритму розподіленої бази даних

Аналіз та побудова моделювального алгоритму дозволяє розв’язувати дуже велику кількість складних задач на моделювання розподіленої бази даних та допомагає знизити фінансові витрати та час.

Моделювальний алгоритм – це ефективний, дієвий і найкращий метод, який можна використовувати для вираження рішення будь-якої проблеми в межах обмеженого простору та часу на чітко визначеній формальній мові.

Для того щоб продемонструвати алгоритм можна використати блок-схеми [28]. Починаючи з початкового стану, інструкції описують процес моделювання розподіленої бази даних, який під час виконання проходить через кінцеву кількість чітко визначених послідовних станів, зрештою створюючи «вихід» і завершуючи в кінцевому стані.

Усі алгоритми мають відповідати наступним критеріям:

– вхід;
– вихід;
– визначеність;
– скінченність;
– ефективність.

Властивості алгоритму. Простого написання послідовності інструкцій як алгоритму недостатньо для виконання певного завдання. Необхідно мати такі властивості, пов'язані з моделювальним алгоритмом.

Відсутність двозначності. Кожен крок в алгоритмі має бути однозначним. Це означає, що кожна інструкція має бути чіткою та точною. Інструкція в будь-якому алгоритмі не повинна означати будь-якого конфліктного значення. Ця властивість також свідчить про ефективність алгоритму [29].

Діапазон введення. Необхідно вказати діапазон введення. Це пояснюється тим, що зазвичай алгоритм керується введенням, і якщо діапазон введення не вказано, алгоритм може перейти в нескінченний стан.

Кратність. Один і той самий алгоритм можна представити кількома

різними способами. Це означає, що не можна написати послідовність інструкцій простою англійською мовою, але ми можемо написати її у формі псевдокоду. Аналогічно, для розв'язання однієї задачі ми можемо написати кілька різних алгоритмів.

Швидкість. Алгоритм, написаний з використанням певних ідей. Такий моделювальний алгоритм повинен бути ефективним і виробляти вихід з високою швидкістю [25].

Скінченність. Алгоритм повинен бути кінцевим. Це означає, що після виконання необхідних операцій його слід припинити.

При розробці алгоритму слід враховувати наступні фактори:

– модульність: ця функція ідеально розроблена для алгоритму, якщо задачу розбивають на маленькі модулі або маленькі кроки, що є базовим визначенням алгоритму;

– правильність: правильність алгоритму визначається, коли задані вхідні дані дають бажаний вихід, що вказує на те, що алгоритм розроблено правильно. Аналіз алгоритму виконано правильно;

– супроводжуваність: це означає, що алгоритм має бути розроблений у зрозумілий, структурований спосіб, щоб під час перевизначення алгоритму в нього не вносилося значних змін;

– функціональність: враховує різні логічні кроки для вирішення проблеми поставленої задачі;

– надійність: надійність означає здатність алгоритму чітко визначати вашу проблему;

– зручність: якщо алгоритм складний для розуміння, розробник додатку не зможе чітко слідувати алгоритму;

– простота: якщо алгоритм простий, його легко зрозуміти;

– розширюваність: алгоритм має бути розширюваним, якщо інший розробник алгоритму або програміст захоче його використовувати.

Схема моделювального алгоритму розподіленої бази даних для трьох локальних баз даних представлена на рисунку 2.8.

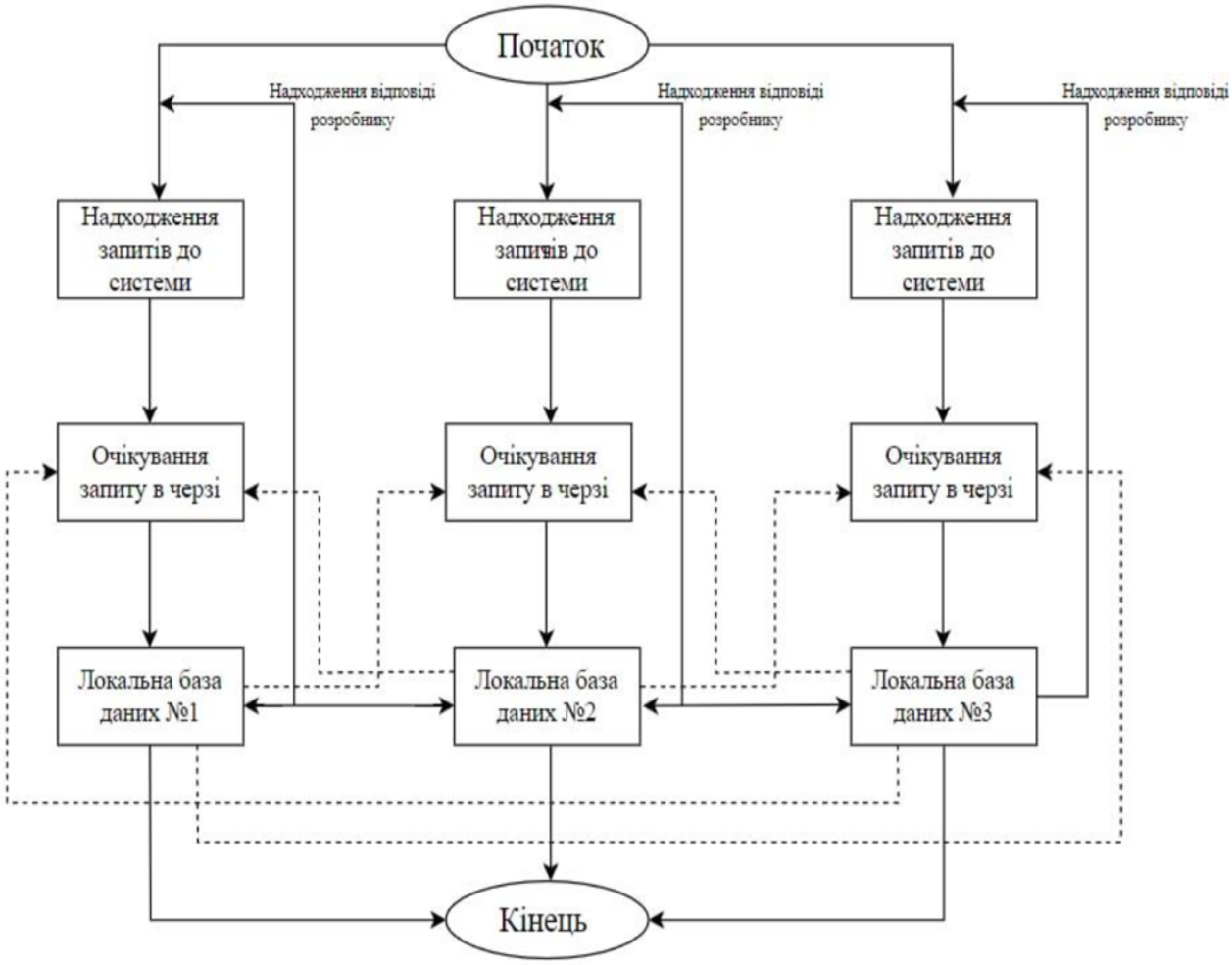


Рисунок 2.8 – Схема моделювального алгоритму РБД для трьох локальних баз

Опис алгоритму роботи програми з трьома локальними базами даних.

Робота кожної локальної бази даних ідентична, тому у описі моделювального алгоритму описано роботу однієї локальної бази даних.

Пункт 1. Відбувається надходження запитів від розробника до локальної бази даних системи розподіленої бази даних.

Пункт 2. Запити, що надходять до системи, потрапляють до черги, де очікують обробки. Кожна локальна база даних має свою чергу очікування.

Пункт 3. Запит потрапляє до локальної бази даних для обробки.

Пункт 4. У разі успішної обробки запиту локальною базою даних, до якої надійшов запит, він залишає систему РБД.

Пункт 5. Оброблена відповідь запиту надходить до розробника. Також після закінчення часу роботи системи формується звіт про результати обробки.

Пункт 6. Якщо локальна база даних, до якої надійшов запит, не може дати повну відповідь, відбувається обмін запиту між іншими локальними базами даних для повної відповіді на запит.

Пункт 7. Запит, якому необхідно здійснити обмін, надходить до черги очікування у іншій локальній БД.

Пункт 8. Після обслуговування запиту, для якого відбувся обмін залишає систему РБД, а відповідь так само надходить до розробника і формується звіт по закінченню часу роботи системи.

2.7 Висновки до другого розділу

На основі аналізу процесу розподілу запитів в РБД сформульовано постановку задачі її моделювання. Виходячи з дискретності та стохастичності процесів функціонування РБД, запропоновано розглядати їх як багатофазні системи масового обслуговування. Це дозволило обрати для їх моделювання дискретно-стохастичний підхід. В рамках цього підходу виконана формалізація процесу обробки запитів з використанням багатофазної Q-схеми.

Для розробленої Q-схеми побудовано моделювальний алгоритм, який відображає всі основні операції з обробки запитів та зв’язки між ними. Розроблений алгоритм слугує основою для розроблюваного програмного засобу імітаційного моделювання РБД.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ЕКСПЕРИМЕНТИ

3.1 Вибір мови програмування

У сучасному світі існує дуже велика кількість різних мов програмування, які швидко розвиваються. Це спричинено тим, що відбувається цифрова трансформація світу.

Сьогодні ми не можемо уявити собі перегляд веб-сторінок без інтерактивних веб-сайтів, кнопок, які можна натискати, та онлайн-відеоігор. Спосіб нашої взаємодії з Інтернетом радикально змінився протягом багатьох років і здебільшого, все зводиться до JavaScript [12].

Для реалізації імітаційної моделі розподіленої бази даних була обрана високорівнева мова програмування JavaScript з використанням бібліотеки React.js.

JavaScript є однією з найбільш використовуваних мов програмування. За даними Північно-Східного університету, це друга за популярністю мова кодування. Очевидно, що JavaScript є невід'ємною частиною нашого повсякденного онлайн-життя [12].

JavaScript – це текстова мова програмування, яка дозволяє створювати інтерактивні елементи на веб-сторінці.

Разом із HTML і CSS, JavaScript є одним із трьох основних компонентів Інтернету. Будь-який веб-сайт обов'язково використовує комбінацію цих трьох мов програмування, кожна з яких має свою особливу мету. Розглянемо кожну мову більш детально.

Мова розмітки гіпертексту (HTML): це основна мова програмування, яка є основою для всіх веб-сайтів. Це дозволяє створювати вміст, розміщувати його на сторінці та посилатися на інші сторінки [7].

Каскадні таблиці стилів (CSS): ця мова сценаріїв дозволяє створювати вміст на веб-сторінці. Він відповідає за стиль тексту, кольори, шрифти тощо.

JavaScript (JS): і останнє, але не менш важливе, JavaScript дозволяє

зробити веб-сайт інтерактивним. Це дозволяє зробити весь вміст, який створюється за допомогою HTML і стилізується за допомогою CSS, більш привабливим. Від кнопок, які можна натиснути, до параметрів входу, JavaScript перетворює веб-сайт зі стіни тексту на гарний сайт [15].

Звичайно, JavaScript не був би наріжним каменем у світі програмування, якби він стосувався лише натискання кнопок. Практичні застосування JavaScript майже нескінченні, але розглянемо деякі з найпоширеніших застосувань нижче.

Інтерактивні веб-сайти. Усюди, де користувач взаємодіє з елементами на веб-сайті, це завдяки JavaScript. Деякі з багатьох речей, у яких може допомогти JavaScript, включають:

– випадаючі меню, які допоможуть користувачам зробити правильний вибір;

– каруселі зображень для прокручування вмісту сторінки;

– анімації, щоб додати трохи цікавості сайту;

– аудіо та відео одним натисканням кнопки;

– вспливаючі вікна, щоб активувати відвідувачів;

– віджети чату, які дозволяють користувачам зв’язуватися з розробниками.

Найбільшою перевагою мови програмування JavaScript є створення гарних та зручних веб-сайтів [12].

Відео ігри. JavaScript – це взаємодія користувача та веб-сайту, відео-ігри це найбільший інтерактив. Останніми роками дуже часто використовують мову програмування JavaScript для написання сценаріїв, які дозволяє веб-сайту створювати насичені діями пригоди та карколомні стратегічні головоломки.

Веб і мобільні програми. Іншим потужним використанням мови програмування JavaScript є створення універсальних програм, які можуть задовольнити будь-які потреби. Фактично, швидше за все, мільйони людей щодня використовуєте одну з багатьох веб- або мобільних програм, створених за допомогою JavaScript.

Переваги мови програмування JavaScript [12]:

– простий − JavaScript простий для розуміння та сприйняття. Користувачі, і розробники знайдуть структуру простою. Крім того, його дуже легко реалізувати, заощаджуючи веб-розробникам купу грошей під час створення динамічного вмісту;

– швидкість – JavaScript це «інтерпретована» мова, вона скорочує час, необхідний для компіляції іншими мовами програмування, такими як Java. Ще один сценарій на стороні клієнта — JavaScript, який прискорює виконання програми, усуваючи час очікування підключення до сервера.

Незалежно від того, де розміщено JavaScript, він завжди виконується в клієнтському середовищі, щоб зменшити використання пропускної здатності та прискорити виконання.

– взаєморозуміння – оскільки JavaScript легко інтегрується з іншими

мовами програмування, багато розробників вважають за краще використовувати його для створення різноманітних програм. JavaScript може містити будь-яка веб-сторінка або скрипт іншої мови програмування;

– завантаження сервера – перевірку даних можна виконати в самому браузері, а не пересилати їх на сервер, оскільки JavaScript працює на стороні клієнта. Весь веб-сайт не потрібно перезавантажувати у разі будь-яких розбіжностей. Браузер оновлює лише обрану область сторінки.

Недоліки мови програмування JavaScript [12]:

– неможливо налагодити – хоча деякі редактори HTML дозволяють налагоджувати, вони не такі ефективні, як редактори для C або C++. Крім того, розробнику важко з’ясувати проблему, оскільки браузер не відображає жодних помилок;

– неочікувана зупинка візуалізації – весь JavaScript код веб-сайту може припинити візуалізацію через єдину помилку в коді. Тому що, помилка виявляються на етапі виконання коду;

– безпека на стороні клієнта − користувач може бачити код JavaScript; цим можуть зловживати інші розробники. Ці дії можуть передбачати анонімне

використання вихідного коду. Крім того, на веб-сайт дуже просто вставити код, який погіршує безпеку даних, що передаються через веб-сайт;

– успадкування − JavaScript не підтримує множинне успадкування, підтримується лише одне успадкування. Ця властивість об'єктно-орієнтованих мов може бути необхідною для деяких програм;

– підтримка браузера – залежно від браузера JavaScript інтерпретується по-різному. Тому перед публікацією, код потрібно запустити на різних платформах. Також потрібно перевірити старі браузери, оскільки вони не підтримують деякі нові функції.

3.2 Вибір середовища розробки

Для розробки кросплатформеного додатку обрано середовище розробки Visual Studio Code.

Visual Studio Code – це безкоштовний редактор коду, який допомагає програмісту писати код, налагоджувати та виправляє його за допомогою методу «intelli-sense». Це полегшує розробникам писати код простим способом.

З часом мови програмування потребували спеціального фреймворку та підтримки для подальшого кодування та розробки, що було неможливо за допомогою простих редакторів. VI Editor, Sublime Text Editor є одним із багатьох типів редакторів, які з’явилися на світ.

Найвідомішим, який підтримує майже всі мови кодування, є Visual Studio Code. Його функції дозволяють користувачеві змінювати редактор відповідно до використання, що означає, що користувач може завантажувати бібліотеки з Інтернету та інтегрувати їх із кодом відповідно до своїх вимог [19].

Visual Studio Code має кілька дуже унікальних функцій. Вони перераховані нижче [12]:

– підтримка кількох мов програмування: підтримує кілька мов програмування.

Редактор легко виявляє будь-яку помилку чи посилання на іншу мову

програмування.

– intelli-Sense: він може виявити, якщо якийсь фрагмент коду залишився незавершеним. Крім того, загальний синтаксис змінних і оголошення змінних створюються автоматично;

– підтримка між платформами: традиційно редактори підтримували системи Windows, Linux або Mac. Але код Visual Studio є кросплатформним. Тому він може працювати на всіх трьох платформах;

– розширення та підтримка: зазвичай підтримує всі мови програмування, але якщо користувач/програміст хоче використовувати мову програмування, яка не підтримується, він може завантажити розширення та використовувати його. З точки зору продуктивності, розширення не сповільнює роботу редактора, оскільки воно виглядає як інший процес;

– репозиторій: оскільки попит на код постійно зростає, безпечне та своєчасне зберігання є однаково важливим. Він підключений до Git або може бути підключений до будь-якого іншого сховища для отримання та збереження екземплярів;

– веб-підтримка: редактор поставляється з вбудованою підтримкою веб-додатків. Таким чином, веб-програми можна створювати та підтримувати у VSC;

– ієрархічна структура: файли коду розташовані у файлах і папках. Необхідні файли коду також містять деякі файли, які можуть знадобитися для інших складних проектів. Ці файли можна видалити як зручно;

– удосконалення коду: деякі фрагменти коду можна декларувати дещо інакше, що може допомогти користувачеві в коді. Ця функція пропонує користувачеві, якщо необхідно, змінити його на запропонований варіант;

– підтримка терміналу: у багатьох випадках користувачеві потрібно починати з кореня каталогу, щоб розпочати певну дію, вбудований термінал або консоль надає підтримку користувачам, щоб не перемикатися між двома екранами;

– кілька проектів: кілька проектів, що містять кілька файлів/папок, можна

відкривати одночасно. Ці проекти/папки можуть бути або не пов’язані один з одним;

– підтримка Git: ресурси можна отримати з Git Hub Repo онлайн і навпаки;

– вилучення ресурсів також означає клонування коду, доступного в Інтернеті. Пізніше цей код можна змінити та зберегти;

– коментування: загальна функція, але деякі мови її не підтримують. Коментування коду допомагає користувачеві згадати або відстежити відповідно до потрібної йому послідовності.

Переваги Visual Studio Code перед будь-якою іншою IDE:

– підтримка між платформами:Windows, Linux, mac IOS;

– редактор легкий у використанні;

– надійна архітектура;

– intelli-Sense;

– безкоштовне програмне забезпечення: безкоштовно - мабуть, найкраща функція для всіх програмістів, навіть більше для організацій;

– багато користувачів використовуватимуть його або використовували його лише для настільних програм, але він також забезпечує чудову підтримку інструментів для таких веб-технологій, як: HTML, CSS, JSON.

Сфера дії Visual Studio. Найпоширенішими мовами є: C#; Visual Basic; Java-Script; XML; Python; CSS; GO; PERL.

Ще одна особливість, яку користувачі миттєво помічають – це зручність Visual Studio Code. Зручність використання дуже проста. Файл упорядковано ієрархічно та має звичайне програмне забезпечення, таке як панель інструментів, рядок стану та бічна панель. Він також має плаваюче вікно провідника Windows, яке можна зафіксувати в одному місці відповідно до зручності, яке складається зі структури каталогів файлів. Ці файли (файли коду, папки із зображеннями тощо) можна відкривати або перейменовувати звідси, а зміни автоматично відображатимуться в сховищі [14].

До останнього часу навряд чи існувало IDE або редактор коду, який був

би настільки зручним для користувача, що навіть користувачі, які вперше знайомляться з ним, могли без жодних проблем використовувати кожну функцію [23]. Зручна функція кодування та розпізнавання помилок у коді також допомагають користувачам зробити код більш ефективним і безпомилковим.

Завдяки прогресу технологій Visual Studio Code відіграє ключову роль у розробці програмного забезпечення. Завдяки функціям, що постійно розвиваються, і новим налаштуванням, які незабаром будуть додані, і які дозволять користувачам працювати з ним звідки завгодно [27].

3.3 Постановка задачі контрольного прикладу

Задано n локальних баз даних розподіленої бази даних. Надходження запитів до кожної локальної бази даних описується пуассонівським потоком з заданою інтенсивністю m та пропускною здатністю k каналів зв'язку. Будемо вважати, що всі запити приблизно мають однакому довжину. Час передачі запиту складає $\tau \pm \Delta\tau$ одиниць часу. Час, який відведено на обслуговування задано експоненціально [14].

Є множина користувачів, які відправляють свої запити до системи. Кожен запит потрапляє до загальної черги очікування [16]. Якщо локально база даних вільна під час надходження запиту, він одразу потрапляє на обслуговування.

Після очікування у черзі, до кожної локальної бази даних надходять запити від користувачів. Через задану s кількість секунд, після надходження, запит реєструється та очікує у черзі, до конкретної локальної бази даних. Після очікування запит надходить до локальної бази даних на обробку [19].

Під час обробки запиту з заданою ймовірністю r, виникають ситуації, коли не вистачає даних для повної відповіді на запит, тому відбувається перенаправлення запиту до інших локальних баз даних для пошуку більш детальної інформації. Пошук додаткової інформації відбувається одночасно в усіх локальних базах даних. Після перенаправлення відбувається обробка

запиту, але вже повторна. Якщо запит оброблено від залишає систему [16].

Розробити кросплатформений додаток для проведення аналізу та експериментів роботи розподіленої бази даних комп’ютерно-інтегрованого виробництва протягом 4 год 30 хв. Необхідно провести оцінку зміни характеристики черги запитів та завантаженість РБД при різній кількості локальних баз даних. Вихідні дані: n = 5, m =30 запитів/с, k = 24 000 біт/с, s = ±3 с.

3.4 Опис програмного забезпечення

Розроблена програма моделює роботу системи обробки інформації на прикладі розподіленої бази даних системи керування комп’ютерно-інтегрованого виробництва. За допомогою програми можна дізнатися, як відбувається процес роботи розподіленої бази даних. На рисунку 3.1 представлено стартову сторінку програми.

Доступно до заповнення кількість локальних баз даних та час роботи системи. Час роботи системи можна задати у годинах та хвилинах.

Після старту програми, можна побачити на екрані кількість попередньо заданих локальних баз даних та таймер з заданим часом. Через декілька мілісекунд відбувається надходження запитів до черг кожної локальної бази даних. На рисунках 3.2 і 3.3 відображено стартову сторінку програми й екранну форму на початку роботи програми.

На рисунку 3.3 можна побачити екранну форму у процесі моделювання РБД.

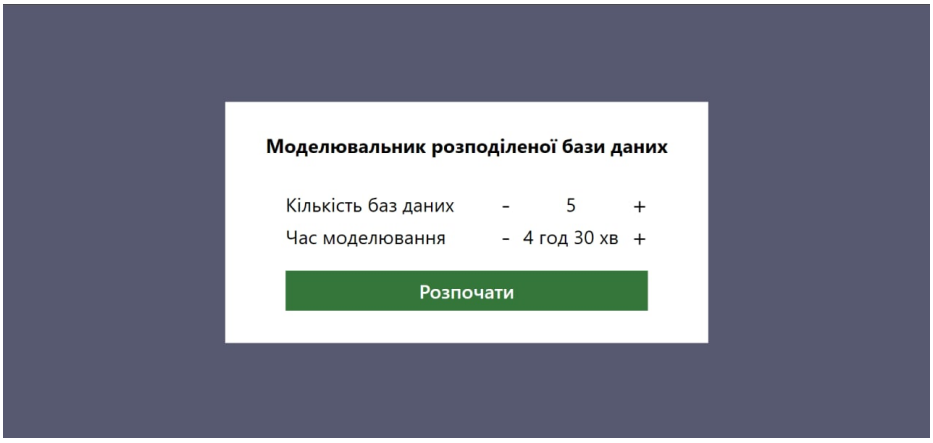


Рисунок 3.1 – Стартова сторінка програми

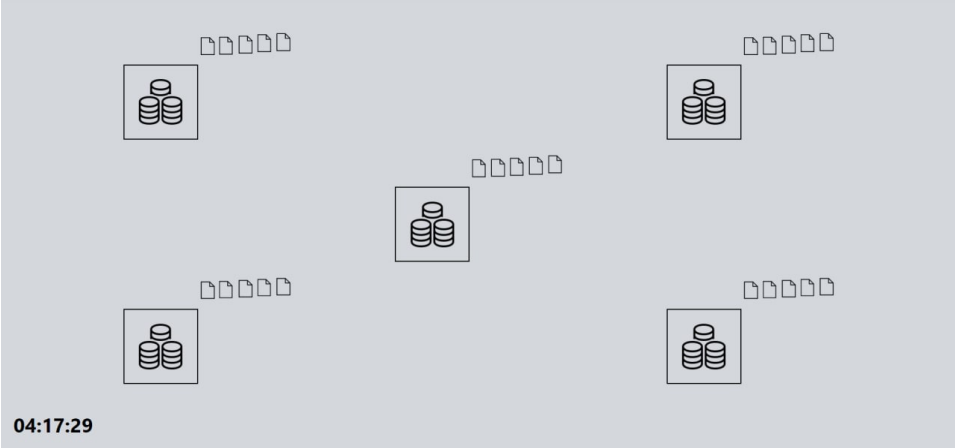


Рисунок 3.2 – Екранна форма початку роботи програми

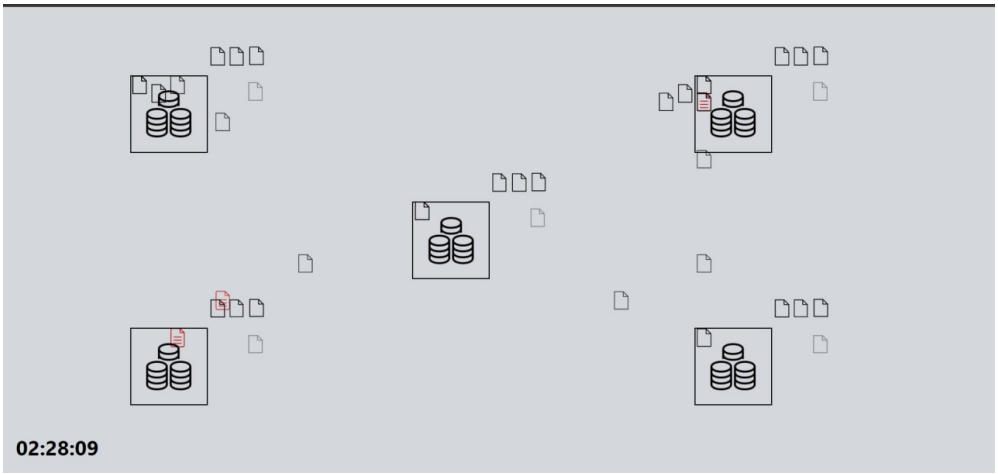


Рисунок 3.3 – Екранна форма у процесі роботи моделі

Після старту програми можна побачити, як відбувається надходження запитів до кожної локальної бази даних, перенаправлення запитів до інших локальних баз даних та вихід запитів з системи після обслуговування. Зліва внизу також відображено зворотній час процесу моделювання.

Коли час роботи закінчується, справа внизу з’являється кнопка «Отримати звіт». На рисунку 3.4 відображено екранну форму «Кінець роботи програми».

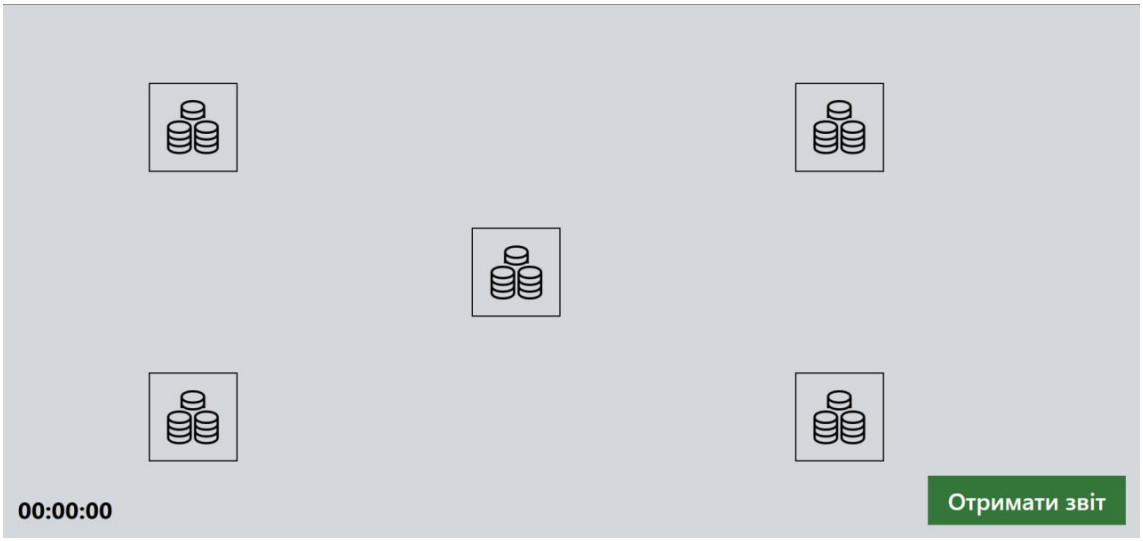
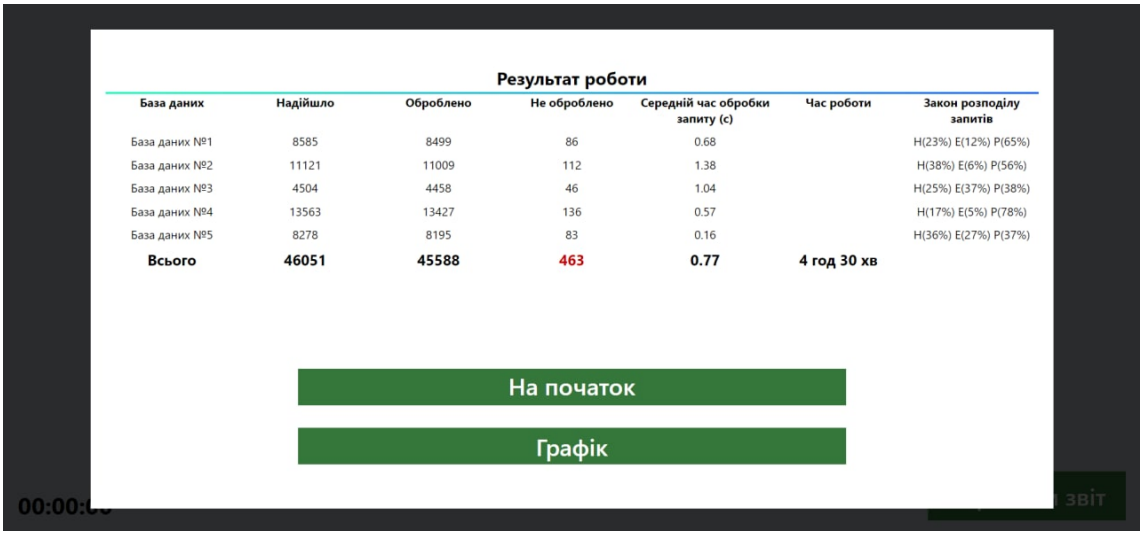


Рисунок 3.4 – Екранна форма «Кінець роботи програми»

Після натискання кнопки «Отримати звіт» з’являється таблиця з результатами моделювання (рис. 3.5).



База даних	Надійшло	Оброблено	Не оброблено	Середній час обробки запиту (с)	Час роботи	Закон розподілу запитів
База даних №1	8585	8499	86	0.68		H(23%) E(12%) P(65%)
База даних №2	11121	11009	112	1.38		H(38%) E(6%) P(56%)
База даних №3	4504	4458	46	1.04		H(25%) E(37%) P(38%)
База даних №4	13563	13427	136	0.57		H(17%) E(5%) P(78%)
База даних №5	8278	8195	83	0.16		H(36%) E(27%) P(37%)
Всього	**46051**	**45588**	**463**	**0.77**	**4 год 30 хв**	

Рисунок 3.5 – Таблиця з результатами моделювання

У таблиці можна побачити 7 колонок: кількість баз даних, загальна кількість запитів, що надійшли до кожної локальної бази даних, кількість оброблених запитів кожною локальною базою даних, кількість запитів, що не встигли обробитись за брак часу, середній час обробки запиту у кожній локальній базі даних, час роботи системи та відсоткове співвідношення законів розподілу запитів у кожній БД.

Екранні форми графічного подання результатів представлені на рисунках 3.6 та 3.7.

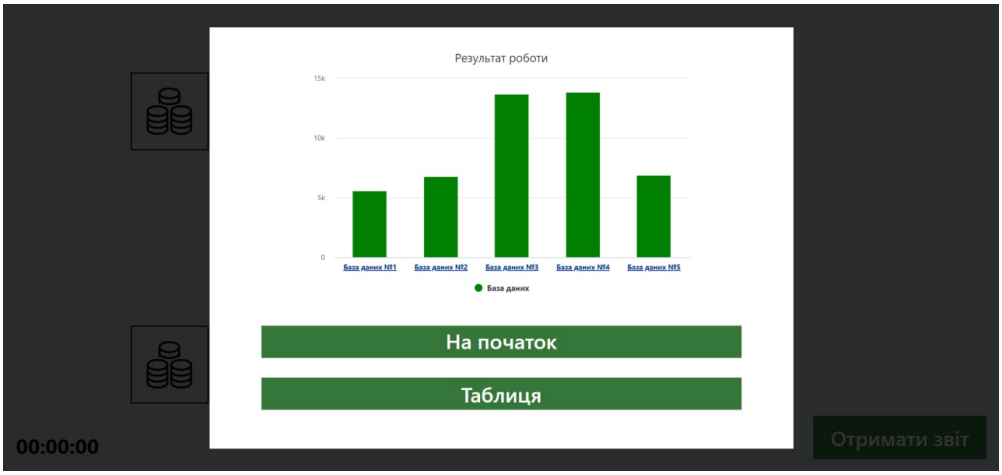


Рисунок 3.6 – Подання результатів у вигляді гістограми

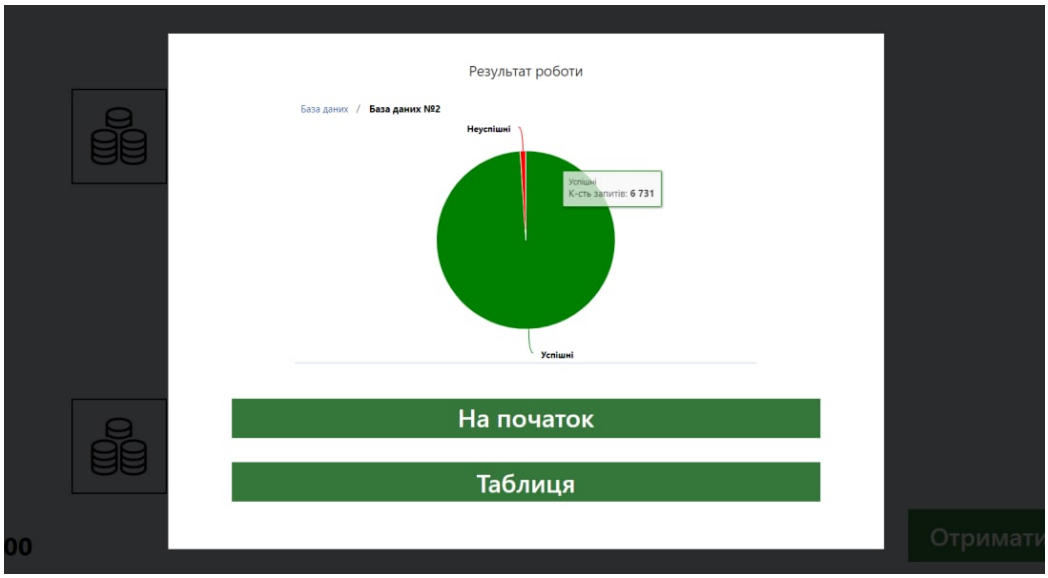


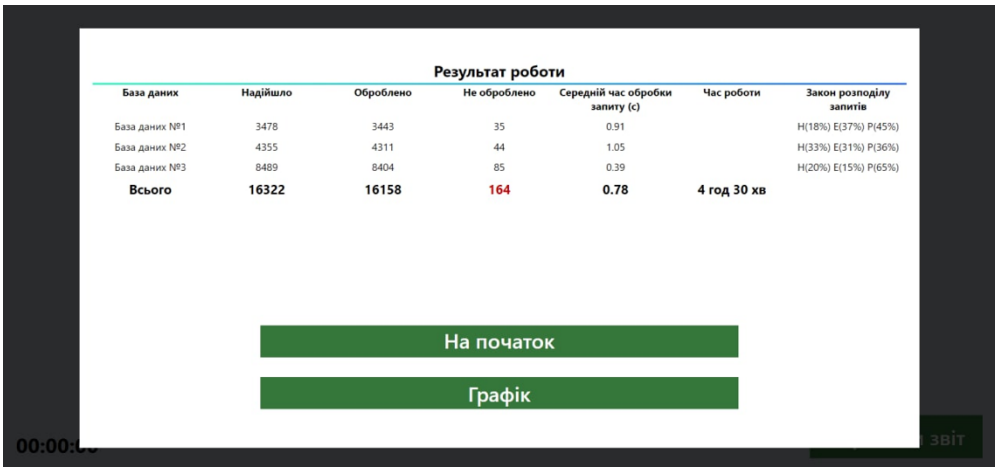
Рисунок 3.7 – Подання результатів у вигляді діаграми

Для того, щоб запустити програму знову та провести інші експерименти, доступна кнопка «На початок».

3.5 Огляд результатів експериментів та їх аналіз

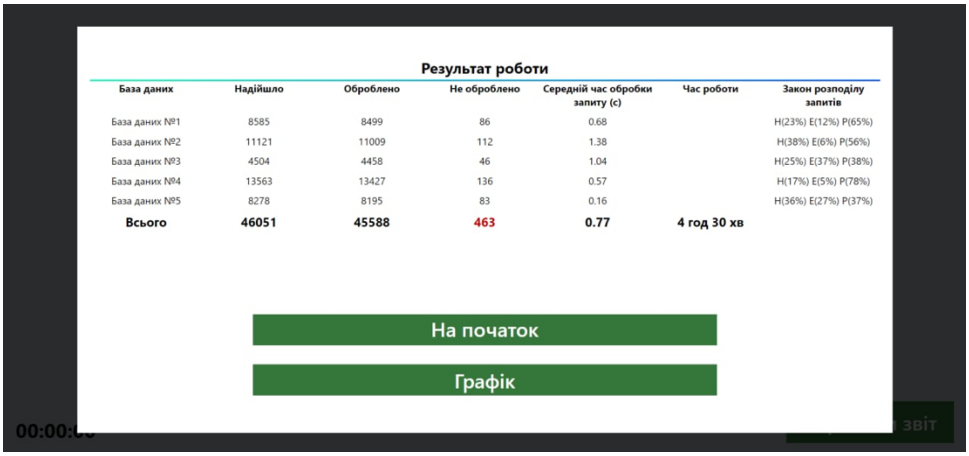
Було проведено 4 експерименти для різної кількості локальних баз даних 3, 5, 7 та 10 БД. Час роботи системи було задано однаковий 4 год 30 хв (при необхідності час можна задати будь-який). Нижче представлені скріншоти звітів результатів роботи системи у вигляді таблиць (рис. 3.8 – 3.11).

Для оцінки точності результатів було проведено 10 експериментів для 3 локальних баз даних. Результати експериментів наведені у таблиці 3.1.



База даних	Надійшло	Оброблено	Не оброблено	Середній час обробки запиту (с)	Час роботи	Закон розподілу запитів
База даних №1	3478	3443	35	0.91		H(18%) E(37%) P(45%)
База даних №2	4355	4311	44	1.05		H(33%) E(31%) P(36%)
База даних №3	8489	8404	85	0.39		H(20%) E(15%) P(65%)
Всього	**16322**	**16158**	**164**	**0.78**	**4 год 30 хв**	

Рисунок 3.8 – Результати моделювання для трьох ЛБ



База даних	Надійшло	Оброблено	Не оброблено	Середній час обробки запиту (с)	Час роботи	Закон розподілу запитів
База даних №1	8585	8499	86	0.68		H(23%) E(12%) P(65%)
База даних №2	11121	11009	112	1.38		H(38%) E(6%) P(56%)
База даних №3	4504	4458	46	1.04		H(25%) E(37%) P(38%)
База даних №4	13563	13427	136	0.57		H(17%) E(5%) P(78%)
База даних №5	8278	8195	83	0.16		H(36%) E(27%) P(37%)
Всього	**46051**	**45588**	**463**	**0.77**	**4 год 30 хв**	

Рисунок 3.9 – Результати моделювання для п'яти ЛБ

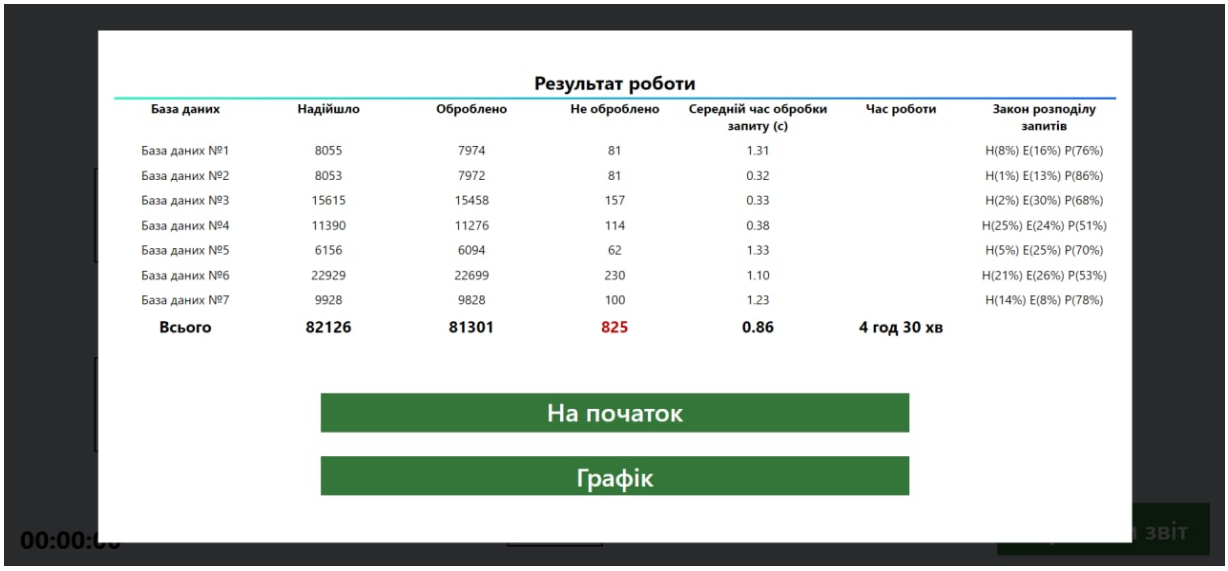


Рисунок 3.10 – Результати моделювання для семи ЛБ

Рисунок 3.11 – Результати моделювання для десяти ЛБ

Таблиця 3.1 – Результати експериментів

Номер експерименту	Завантаженість ЛБ №1	Завантаженість ЛБ №2	Завантаженість ЛБ№3	Час обробки запиту, с
1	0,954	0,933	0,915	0,76
2	0,972	0,973	0,967	0,60
3	0,974	0,964	0,943	0,94
4	0,978	0,963	0,958	0,52

Кінець таблиці 3.1

Номер експерименту	Завантаженість ЛБ №1	Завантаженість ЛБ №2	Завантаженість ЛБ№3	Час обробки запиту, с
5	0,973	0,978	0,941	0,94
6	0,975	0,966	0,949	1,09
7	0,971	0,956	0,943	0,75
8	0,972	0,967	0,969	0,68
9	0,979	0,946	0,934	1,00
10	0,969	0,951	0,938	0,62

Завантаженість кожної локальної БД – це відсоток часу протягом якого працюють локальні БД.

Середній час очікування – це час, який запит, що надійшов перебуває у черзі на обробку. Чим більше заявок надходить, тим більший час очікування у черзі.

Розрахуємо математичне сподівання після проведення 10 експериментів, для кожного коефіцієнту завантаження локальної БД, а також середній час очікування у черзі на обробку.

Для цього необхідно скористатися формулою (3.1):

$$\bar{k} = \frac{1}{N}\sum_{i=1}^{N} k_i, \tag{3.1}$$

де k_i – це коефіцієнт завантаження локальної БД;

N – кількість експериментів.

За формулою (3.2) розрахуємо дисперсію коефіцієнтів навантаження локальних БД та середнього часу очікування у черзі на обробку:

$$D(k) = \frac{1}{N}\sum_{i=1}^{N}(\bar{k} - k_i)^2. \tag{3.2}$$

Результати розрахунків оцінок математичного сподівання та дисперсії коефіцієнтів завантаження локальних баз даних, а також часу обробки запитів користувачів РБД з трьома локальними базами приведені у таблиці 3.2.

Таблиця 3.2 – Результати розрахунків дисперсії та математичного сподівання

Оцінювана характеристика	Коефіцієнти завантаження			Час обробки запиту, с
	ЛБ №1	ЛБ №2	ЛБ №3	
Математичне сподівання	0,9726	0,9577	0,9439	0,79
Дисперсія	0,00006	0,00021	0,000226	1,09

Для кожного з коефіцієнтів завантаження локальних БД, а також для середньої довжини черги необхідно розрахувати потрібну кількість експериментів. Дані розрахунків приведені у таблиці 3.3.

Необхідна кількість експериментів для завантаженості локальних БД розраховувалася так, щоб похибка не перевищувала 0,5% від математичного очікування завантаженості кожної локальної БД. У випадку з розрахунками кількості експериментів для середнього часу очікування у черзі похибка не повинна перевищувати 7% від математичного очікування.

Таблиця 3.3 – Результати розрахунків кількості експериментів

Оцінювана характеристика	Коефіцієнти завантаження			Час обробки запиту, с
	ЛБ №1	ЛБ №2	ЛБ №3	
Кількість експериментів	25	28	23	0,89

Після проведення експериментів можна побачити, що запити, які не були оброблені, через брак часу, не перевищують 10% від загальної кількості запитів, що надійшли до кожної локальної бази даних.

Також в ході експериментів можна зазначити, що при однаковому часі моделювання, але при збільшенні локальних БД збільшується і кількість запитів, що були те не були оброблені.

Було помічено, що при роботі системи локальні БД були навантажені нерівномірно. Це було спричинено тим, що відбувалося перенаправлення запитів до інших локальних БД, коли не вистачало інформації для відповіді на запит.

Проаналізувавши отримані результати експериментів, можна сміливо стверджувати, що хоч і система має необроблені запити, після закінчення часу роботи, запитів що були оброблені набагато більше. Для того, щоб необроблених запитів було менше, рекомендується збільшити час роботи системи.

3.6 Висновки до третього розділу

За результатами аналізу сучасних мов програмування для розв’язання задачі імітаційного моделювання процесу функціонування РБД обрано високорівневу мову програмування JavaScript з використанням бібліотеки React.js, а для розробки кросплатформеного додатку – середовище розробки програмування Visual Studio Code.

Використання цього середовища розробки допомагало створювати код, налагоджувати та виправляти його за допомогою методу «intelli-sense». Це дозволило прискорити процес програмування моделі, проведення експериментів та обробки їх результатів. З використанням розробленої програми розв’язано основні задачі тактичного планування комп’ютерних експериментів. Це дозволило визначити необхідну кількість прогонів моделі та оцінити точність отриманих статистичних оцінок характеристик процесу функціонування РБД. Таким чином було визначено необхідні кількості експериментів з моделлю, що забезпечить необхідну точність оцінок за результатами моделювання.

4 ОХОРОНА ПРАЦІ

4.1 Вимоги до приміщення

Розглядається питання роботи лабораторії для моделювання та програмування розподілених баз даних. Лабораторія містить персональні робочі місця для двох розробників з їх персональними комп’ютерами.

Під час робочого дня розробники майже весь час працюють сидячі, тому немає будь-якого фізичного навантаження. Така робота має відношення до категорії Іа, а саме енерговитрати не сягають більше ніж 120 ккал/год і має легкі фізичні роботи [31].

Встановлено такі метеорологічні параметри для забезпечення комфортних умов для розробника, які відповідають ДСН 3.3.6.042-99.

У холодний період року:

– температура повітря має бути від 22 °С до 24 °С;

– вологість повітря має бути від 40% до 60%;

– швидкість руху повітря ≤ 0,1 м / с.

У теплий період року:

– температура повітря має бути від 23 °С до 25 °С;

– вологість повітря має бути від 40% до 60%;

– швидкість руху повітря ≤ 0,1м/с.

У сучасному житті комп'ютер широко застосовується в житті людини: і вдома, і в офісі, і в магазині, і у виробництві, і навіть у побутовій техніці - іншими словами, комп'ютери міцно увійшли до повсякденного життя людей і їх використання постійно збільшується [31].

Не секрет, що і в офісах комп'ютери переважно використовуються як допоміжні засоби обробки інформації, і таке введення комп'ютерних технологій принципово змінило характер праці офісних працівників та вимоги до організації та охорони праці.

Недотримання вимог безпеки призводить до того, що при роботі за комп'ютером співробітник може відчувати дискомфорт: виникають головні болі та різь в очах, з'являються втома та дратівливість. У деяких людей порушується сон, апетит, погіршується зір, починають хворіти руки, шия, поперек тощо. При ненормованій роботі можливе нервове виснаження [31].

За проведеними дослідженнями, однією з причин поганого самопочуття користувачів персональних комп’ютерів є пульсація яскравості зображення на екрані монітора. Пульсація яскравості викликана особливістю роботи підсвічування плоских моніторів. На даний момент цей параметр не нормується, але його вплив має той самий ефект, що і пульсація загального та місцевого освітлення.

Поради, що до роботи за персональним комп’ютером для розробників [32]:

– не рекомендується працювати за комп'ютером понад 6 годин на зміну;

– рекомендується робити перерви у роботі за ПК тривалістю 10 хвилин через кожні 50 хвилин роботи;

– тривалість безперервної роботи за комп'ютером без регламентованої перерви не повинна перевищувати 2 години;

– під час регламентованих перерв доцільно виконувати комплекси вправ;

– при нерегламентованій роботі підвищеної інтенсивності можливі головний біль, нервові зриви.

Площа робочого місця користувача ПК з ЕПТ-дисплеєм має становити не менше 6 м^2, для ПК з плоским дисплеєм – 4,5 м^2. У приміщеннях повинно проводитися щоденне вологе прибирання та систематичне провітрювання після кожної години роботи. Шумне обладнання (друкарські пристрої, сканери, сервери тощо), рівні шуму якого перевищують нормативні, має розміщуватися поза робочими місцями співробітників [32].

Важливо, щоб розробник, сидячи за комп'ютером, знаходився за добре освітленим робочим столом. Найчастіше саме погане освітлення робочого місця надає згубніший для зору вплив, ніж сам факт знаходження за

комп'ютером [32].

Робочі столи слід розміщувати таким чином, щоб монітори були орієнтовані бічною стороною до світлових прорізів, щоб природне світло падало переважно зліва.

При розміщенні робочих місць відстань між робочими столами має бути не менше 2,0 м, а відстань між бічними поверхнями моніторів не менше 1,2 м.

Конструкція робочого столу повинна забезпечувати оптимальне розміщення на робочій поверхні устаткування, що використовується. Висота робочої поверхні столу повинна становити 725 мм, робоча поверхня стола повинна мати ширину 800...1400 мм і глибину 800...1000 мм. Робочий стіл повинен мати простір для ніг заввишки щонайменше 600 мм, шириною — щонайменше 500 мм, глибиною лише на рівні колін — щонайменше 450 мм і лише на рівні витягнутих ніг — щонайменше 650 мм [32].

Конструкція робочого випорожнення або крісла повинна забезпечувати підтримку раціональної робочої пози працівника і дозволяти змінювати позу з метою зниження статичної напруги м'язів шийно-плечової області та спини. Робочий стілець або крісло повинні бути підйомно-поворотними, регульованими по висоті та кутам нахилу сидіння та спинки, а також відстані спинки від переднього краю сидіння, при цьому регулювання кожного параметра має бути незалежним, легко здійснюваним і мати надійну фіксацію.

Клавіатуру слід розташовувати на поверхні столу на відстані 100...300 мм від краю, зверненого до користувача, або на спеціальній поверхні, відокремленій від основної стільниці.

Екран монітора повинен знаходитися від очей користувача на відстані 600...700 мм, але не ближче 500 мм.

Лабораторія, де виконується розробка розподіленої бази даних, має наступні характеристики:

– площа приміщення 18 м^2;

– висота 2,5 м;

– кількість робочих міст – 2;

– обладнання – стіл з ПК і периферією – 2 шт.

Приміщення з ПК необхідно мати природне та штучне освітлення відповідно до ДБН В.25-28-2006 «Природне і штучне освітлення». Природне світло повинно проникати через бічні світлові прорізи, зорієнтовані, як правило, на північ або північний схід, і забезпечувати коефіцієнт природної освітленості не нижче 1,5%.

Рівень загального штучного освітлення приміщення можна перевірити за допомогою методу питомої потужності, викладеної в.

Розрахункова формула методу розрахунку освітлення:

$$W = \frac{W_{\Sigma}}{S}, \tag{4.1}$$

де W – питома потужність, Вт/м^2;

S – площа приміщення, м^2;

W_{Σ} – загальна потужність освітлювальної установки, яка розраховується за формулою:

$$W_{\Sigma} = W_{св} \cdot n_{св}, \tag{4.2}$$

де $W_{св}$ – потужність одного світильника, Вт;

$n_{св}$ – кількість світильників в приміщенні, шт;

$$W_{\Sigma} = 110 \cdot 4 = 440 \text{ Вт}, \tag{4.3}$$
$$W = 24{,}4 \text{ Вт/м}^2.$$

Питомої потужності 24,4 Вт/м^2 відповідає освітленість в 350 лк. при мінімальній допустимої освітленості 300 лк.

Таким чином, створюються сприятливі зорові умови в лабораторії.

4.4 Висновки для четвертого розділу

Для забезпечення комфортних і безпечних умов праці в лабораторії інформаційного забезпечення систем керування комп’ютерно-інтегрованими об’єктами розглянуто питання охорони праці та безпеки життєдіяльності.

Виходячи з того, що в лабораторії моделювання не використовуються та не зберігаються небезпечне обладнання, речовини або матеріали у цьому розділі лише сформовано вимоги до приміщення лабораторії, встановлено параметри мікроклімату для холодної та теплої пір року, а також виконано розрахунок необхідної кількості світильників у приміщенні лабораторії.

ВИСНОВКИ

У кваліфікаційній роботі отримано рішення актуальної науково-прикладної задачі підвищення точності визначення характеристик процесу функціонування розподіленої бази даних системи керування комп'ютерно-інтегрованим виробництвом за рахунок розробки адекватної імітаційної моделі процесу.

За результатами огляду й аналізу сучасного стану проблеми інформаційного забезпечення системи керування комп'ютерно-інтегрованим виробництвом, моделей та методів моделювання РБД було обрано підхід для підвищення точності оцінок на основі імітаційного моделювання. Виходячи з дискретності та стохастичності процесів функціонування РБД, запропоновано розглядати їх як багатофазні системи масового обслуговування. Це дозволило обрати для їх моделювання дискретно-стохастичний підхід, в рамках якого виконана формалізація процесу обробки запитів з використанням багатофазної Q-схеми.

Для програмування задачі моделювання було розроблено кросплатформений додаток у середовищі Visual Studio Code на мові програмування JavaScript з використанням бібліотеки react.js. З використанням розробленої програми розв'язано основні задачі тактичного планування комп'ютерних експериментів. Це дозволило визначити необхідну кількість прогонів моделі та оцінити точність отриманих статистичних оцінок характеристик процесу функціонування РБД.

Практичне використання розробленої імітаційної моделі РБД комп'ютерно-інтегрованого виробництва дозволяє відтворювати процес її роботи: як відбувається обмін даними між різними локальними базами, відслідкувати кількість запитів, що надійшли до кожної локальної бази, якою була завантаженість кожної бази та кількість запитів, що залишились без обробки в кінці робочого дня.

Створена імітаційна модель може використовуватися на підприємствах і в організаціях, де вирішуються задачі проєктування чи реінжинірингу РБД для систем керування комп’ютерно-інтегрованими виробництвами та інших пошукових систем. Практичне використання моделі за рахунок врахування стохастичного характеру складових процесу дозволить підвищити точність визначення його функціонально-часових характеристик.

По темі кваліфікаційної роботи були підготовлені доповіді на: V Міжнародну конференцію «Виробництво & Мехатронні Системи 2021» (м. Харків, 21-22 жовтня 2021 р.) [23]; 11-ту Міжнародну науково-технічну конференцію «Інформаційні системи та технології» (м. Харків, 22-25 листопада 2022 р.) [33]; Всеукраїнську науково-практичну конференцію здобувачів вищої освіти і молодих учених «Комп’ютерно-інтегровані технології автоматизації технологічних процесів на транспорті та у виробництві» (м. Харків, 23 листопада 2022 р.) [34].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008-15. Документація. Звіти у сфері науки та техніки. Структура та правила оформлення. – Введ. 2015-06-22. – К. Держстандарт України, 2017 – 29 с.

2. Методичні вказівки з підготовки та захисту кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології, освітньо-професійних програм: «Автоматизоване управління технологічними процесами», «Комп'ютерно-інтегровані технологічні процеси і виробництва», «Комп'ютеризовані та робототехнічні системи» / Упоряд. І. Ш. Невлюдов, Р. В. Артюх, В. В. Безкоровайний, Н. П. Демська, В. В. Євсєєв, О. І. Филипенко, О. М. Цимбал. – Харків: ХНУРЕ, 2021. – 55 с.

3. Комп'ютерно-інтегровані технології автоматизації технологічних процесів на транспорті та у виробництві. Матеріали всеукраїнської науково-практичної конференції здобувачів вищої освіти і молодих учених. Харків, ХНАДУ, 2021. 246 с.

4. Product Data Management (PDM). URL: https://www.techtarget.com/searcherp/definition/product-data-management-PDM (дата звернення: 21.09.2022).

5. Невлюдов І.Ш. Основи наукових досліджень: Навч. посібник / І.Ш. Невлюдов, Ю.М. Олександров, А.О. Андрусевич, О.О. Чала. Кривий Ріг: Криворізький коледж НАУ. 2019. 396 с.

6. Технологічний процес та його структура. 2017. URL: https://tehnar.net.ua/tehnologicheskiy-protsess-i-ego-struktura/.

7. Квєтний Р. Н. Комп'ютерне моделювання систем та процесів. Методи обчислень. Частина 1 : навчальний посібник / Богач І. В. та інші; за заг. ред. Р.Н. Квєтного. Вінниця: ВНТУ, 2017.193 с.

8. Tamer-Ozsu M., Valduriez P. Principles of Distributed Database Systems //

Fourth Edition, Waterloo, Canada Montpellier, France June 2019.

9. Освітньо-професійна програма «Комп'ютерно-інтегровані технологічні процеси і виробництва». URL: https://nure.ua/wp-content/uploads/Education_ programs/2021/2021_mag_151_opp_kitpv.pdf.

10. Міхеєва Т. В. Огляд існуючих програмних засобів імітаційного моделювання під час дослідження механізмів функціонування та управління виробничими системами / Т. В. Міхеєва URL: https://cyberleninka.ru/article/n/obzor-suschestvuyuschih-programmnyh-sredstv-imitatsionnogo-modelirovaniya-pri-issledovanii-mehanizmov-funktsionirovaniya-i/viewer (дата звернення: 17.10.2022).

11. Безкоровайний В. В., Іванюк О. А. Інформаційна технологія моделювання розподілених баз даних // Міжнародна науково-практична конференція «Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку» / Збірник тез доповідей (м. Харків, 17 березня 2020 р.). Харків. 2020,159-161.

12. Разные языки программирования и их применение. Лекция в Google. URL: https://habr.com/company/yandex/blog/272759 (дата звернення: 12.10.22).

13. Bergsten B., Couprie M., Valduriez P. Prototyping DBS3, a Shared-Memory Parallel Database System // Proc. Int. Conf. on Parallel and Distributed Information Systems, Miami, Florida, December 2019, 226-234.

14. Безкоровайний В. В., Іванюк О. А. Інформаційна технологія моделювання розподілених баз даних // V Міжнародна науково-практична конференцію «Інформаційні технології в культурі, мистецтві, освіті, науці, економіці та бізнесі» (м. Київ) 2020. С.125–128.

15. Безкоровайний В. В., Іванюк О. А. Інформаційна технологія моделювання розподілених баз даних // Всеукраїнська науково-практична конференція здобувачів вищої освіти і молодих учених у розділі «Комп'ютерно- інтегровані технології автоматизації технологічних процесів на транспорті та у виробництві. (м. Харків, 25 листопада 2020 р.). Харків. 2020.

16. Безкоровайний В. В., Іванюк О. А., Прінь К. В.. Технологія реінжинірингу розподілених баз даних систем керування виробництвом // Виробництво & Мехатронні Системи 2021: матеріали V Міжнародної конференції, Харків, 21-22 жовтня 2021 р.: тези доповідей / [редкол. І.Ш. Невлюдов]. Харків: 2021. С. 104-106.).

17. Бескоровайный, В. В. Оценка времени доступа к информационным ресурсамраспределенных баз данных при решении задач синтеза их физических структур/ В. В. Бескоровайный, О.С. Ульянова // Системи управління, навігації та зв'язку. 2010.№3(15).С. 210-214.

18. Бескоровайный В. В., Подоляка К.Е. Разработка системологической модели проблемы структурно-топологического реинжиниринга систем крупномасштабного мониторинга // Восточно-Европейский журнал передовых технологий. 2015. С. 37-42.

19. Нестеренко С. А. Выбор оптимального плана энергоэффективного реинжиниринга корпоративной компьютерной сети // Електротехнічні та комп'ютерні системи. 2017. С. 341-346.

20. Gary Donahue, Denise Donohue. Network Warrior: Everything You Need to Know That Wasn't on the CCNA Exam. London, 2020, 620 p.

21. Безкоровайний В. В., Іванюк О. А., Прінь К. В. Технологія реінжинірингу розподілених баз даних систем керування виробництвом // Виробництво & Мехатронні Системи 2021: матеріали V-ої Міжнародної конференції, Харків, 21-22 жовтня 2021 р.: тези доповідей / [редкол. І.Ш. Невлюдов (відповідальний редактор)]. Харків: [електронний друк], 2021. С. 104-106.).

22. Newton O. Everything You Need to Ace Math in One Big Fat Notebook: The Complete Middle School Study Guide. Deggendorf, 2019, 1510 p.

23. Yocy M. T., Valduries P., Principles of organization of distributed databases. 2021.

24. Beskorovainyi V., Petryshyn L., Shevchenko O., «Specific subset effective option in technology design decisions», Applied Aspects of Information Technology,

Vol. 3., pp. 443-455, 2020.

25. Beskorovainyi V., «Combined method of ranking options in project decision support systems»,Innovative Technologies and Scientific Solutions for Industries, pp. 13-20, 2020.

26. Beskorovainyi V., Kolesnyk L., «Interval model of multi-criterion task of reengineering physical structures of distributed databases», Intelligent information systems for decision support in project and program management: Collective monograph, Riga: ISMA, 2021. pp. 7-14.

27. Безкоровайний В.В. Конспект лекцій з дисципліни "Моделювання систем" для студентів спеціальності 122 - Комп'ютерні науки. Харків, 2019. 174 с.

28. Бескоровайный В. В., Авраменко В. П. Евристичний метод структурного синтезу територіально розподілених систем // Технологія приладобудування. 2018. С. 69–74.

29. Юнич М. В., Загурський О.Б. Роль управління запасами у забезпеченні економічної безпеки підприємства // Пріоритети економічної науки XXI століття : матеріали наук.-практ. конф. з міжнар. участю (м. Івано-Франківськ, 17 червня. 2022 р.). Івано-Франківськ : НАІР, 2022. С. 144–146.

30. ДЕРЖАВНИЙ КОМІТЕТ УКРАЇНИ З НАГЛЯДУ ЗА ОХОРОНОЮ ПРАЦІ. URL: https://zakon.rada.gov.ua/laws/show/z0231-05 (дата звернення: 12.11.2022).

31. Особливості нормування роботи програмістів. URL: https://studfile.net/preview/2401216/page:4/ (дата звернення: 12.11.2022).

32. Закон України «Про охорону праці» // Відомості Верховної Ради України (ВВР). 28.07.2022. с.668. URL: http://zakon4.rada.gov.ua/laws/show/2694-12.

33. Іванюк О., Безкоровайний В., Імітаційне моделювання процесу функціонування розподіленої бази даних комп’ютерно-інтегрованого виробництва// Інформаційні системи та технології: матеріали 11-ї Міжнародної наук.-техн. конф. Ч.2(м. Харків, 22-25 листоп. 2022 р.), Х.:ХНУРЕ,2022.С.9-10.

URL: https://istconf.sedep.online/archive/ist_2022_part_2.pdf (дата звернення 03.12.2022).

34. Іванюк О.А., Безкоровайний В. В. Імітаційна модель розподіленої бази даних у системі керування комп'ютерно-інтегрованим виробництвом // Всеукраїнська наук.-практ. конф. здобувачів вищої освіти і молодих учених «Комп'ютерно-інтегровані технології автоматизації технологічних процесів на транспорті та у виробництві» (м. Харків, 23 листоп. 2022 р.), Харків: ХНАДУ. URL:https://mf.khadi.kharkov.ua/fileadmin/user_upload/_imported/uploads/%D0%9F%D0%A0%D0%9E%D0%93%D0%A0%D0%90%D0%9C%D0%90_%D0%90%D0%9A%D0%86%D0%A2_2022.pdf (дата звернення 03.12.2022).