

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ

к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2021 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "магістр"

на тему: Дослідження інформаційних технологій керування даними ІТ-компаній та
розробка проекту активного сховища даних ІТ-компанії

Виконав: студент 6144 м групи

_____ **Смірнов Д.О.**
(підпис)

Керівник роботи:

доцент кафедри ІУСТ, к.ф-м.н.,
(посада, науковий ступінь вчене звання)

_____ **Коваль С.С.**
(підпис)

м.Миколаїв – 2021 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій
Спеціальність 122 "Комп'ютерні науки"
Освітня програма "Комп'ютерні науки"

"ЗАТВЕРДЖУЮ"
Гарант освітньої програми

д.т.н, проф. Чернова Лб. С.
" ____ " _____ 2021 р.

**ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти "магістр"**

Студенту Смірнову Дмитру Олександровичу

(прізвище, ім'я, по батькові)

1. Тема роботи: Дослідження інформаційних технологій керування даними ІТ-компаній та розробка проекту активного сховища даних ІТ-компанії _____

Керівник роботи К.ф.-м.н. Коваль С.С.

Затверджені наказом ректора № 1232-уч. від "12" жовтня 2021 року.

2. Термін подання роботи: 21 грудня 2021 р.

3. Вихідні дані до проекту (роботи) ДСТУ щодо інформаційних технологій організації баз даних

4. Перелік питань, що належать до розробки (найменування розділів):
дослідження інформаційних технологій управління проектами; розробка моделей управління проектами; розробка проектних рішень по системі.

5. Перелік презентаційних матеріалів: Мета і постановка задачі дослідження; Узагальнена структура системи; Математична модель; Модель інформаційного забезпечення; Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	К.ф-м.н., доц. Коваль С.С.		
2	К.ф-м.н., доц. Коваль С.С.		
3	К.ф-м.н., доц. Коваль С.С.		
4			
5	Ст. викл. Гурець Н.В.		
6	Ст. викл. Гурець Н.В.		

7. Дата видачі завдання 04 вересня 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Вибір теми, визначення мети, завдань, об'єкта та предмета дослідження		
2	Підбір та опрацювання літератури		
3	Дослідження інформаційних технологій організації активних сховищ даних		
4	Розробка концепції і проекту АІС		
5	Розробка та реалізація проекту АІС		
6	Розробка додаткових розділів, оформлення роботи, підготовка презентаційних матеріалів		
7	Представлення до попереднього захисту	1 грудня 2021 р.	
8	Отримання відгуку керівника	13 грудня 2021 р.	
9	Отримання рецензії	15 грудня 2021 р.	
10	Захист кваліфікаційної роботи	21 грудня 2021 р.	

Студент

_____ (підпис)

Смірнов Д.О.
(ПІБ)

Керівник роботи

_____ (підпис)

Коваль С.С.
(ПІБ)

АНОТАЦІЯ

Робота “Дослідження інформаційних технологій керування даними ІТ-компаній та розробка проекту активного сховища даних ІТ-компанії” присвячена дослідженню технологій організації і бізнес-аналізу даних в інформаційних системах, визначенню критеріїв оцінки ефективності їх застосування.

В роботі представлені результати аналізу технологій збереження, організації та доступу до бізнес-даних, розглянуті та визначені особливості застосування інструментальних засобів бізнес-аналізу, визначені критерії оцінки відповідності технологій та інструментів типам сховищ даних інформаційних систем.

В роботі запропонований науково-обґрунтований підхід до оцінки інструментальних засобів та технологій проектування програмного забезпечення для аналізу даних в корпоративних базах даних, наведені результати застосування розроблених моделей та механізмів у реальних проектах.

Робота викладена на ____ сторінках друкованого тексту, містить ____ рисунків, ____ таблиць, перелік джерел з ____ найменувань, ____ додатків.

SUMMARY

The work “Research of information technologies of data management of IT companies and development of the project of active data repository of IT companies” is devoted to research of technologies of organization and access to data in Internet projects and definition of criteria for evaluating their effectiveness.

The work presents the results of analysis of technologies for storage, organization and access to data, considered and identified features of tools, defined criteria for dividing software Internet projects into categories and criteria for assessing the compliance of technologies and tools to types of software projects.

The work proposes a scientifically sound approach to the evaluation of tools and technologies for software design for Internet projects, the results of the application of developed models and mechanisms in real projects.

The work is presented on ____ pages of printed text, contains ____ figures, ____ tables, a list of sources from ____ titles, ____ appendices.

ВСТУП

РОЗДІЛ 1. АНАЛІЗ МЕТОДІВ І ЗАСОБІВ ЗБЕРЕЖЕННЯ ТА АНАЛІЗУ ДАНИХ В КОРПОРАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

1.1. Огляд методів і засобів збереження та аналізу даних в корпоративних інформаційних системах

1.1.1. Моделювання бізнес процесів в корпоративних інформаційних системах

1.1.2. Реляційні СУБД

1.1.3. NoSQL СУБД

1.1.4. ОО СУБД

1.1.5. Документо–орієнтовані бази даних

1.1.6. Розподілені бази даних

1.1.7. Висновки щодо методів та засобів збереження даних

1.2 Огляд інтернет–сервісів та протоколів доступу до даних корпоративних інформаційних систем

1.2.1. Механізми доступу та сервіси

1.2.2. Інтернет–сервери HTTP та HTTPS

1.2.3. Особливості використання файлових сховищ в задачах бізнес–аналітики

1.2.4. Задачі бізнес–аналітики і технологія OLAP

1.2.5. Висновки щодо методів та засобів передачі даних

1.3. Проблеми доступу до даних і аналізу даних в проектах корпоративних інформаційних систем

1.3.1. Проблеми захисту даних та каналів передачі даних

1.3.2. Проблеми, викликані модифікацією структур даних

1.3.3. Проблеми синхронізації даних в корпоративних сховищах даних

1.3.4. Проблеми, що визначаються предметною областю проектів

1.3.5. Проблеми, що визначаються інструментальним забезпеченням

1.3.6. Проблеми обсягу даних корпоративних інформаційних систем

1.4. Постановка задачі дослідження

1.5. Висновки до розділу 1

РОЗДІЛ 2. РОЗРОБКА МОДЕЛЕЙ ТА МЕХАНІЗМІВ АКТИВНИХ СХОВИЩ ДАНИХ КОРПОРАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

2.1. Визначення критеріїв оцінки методів та засобів розробки проектів сховищ даних корпоративних інформаційних систем

2.1.1. Визначення критеріїв оцінки методів та засобів збереження і аналізу даних в проектах корпоративних інформаційних систем

2.1.2. Визначення критеріїв оцінки методів та засобів розробки проектів активних сховищ даних корпоративних інформаційних систем

2.2. Розробка моделей оцінки якості механізмів збереження та доступу до активних сховищ даних

2.2.1. Критерії оцінки якості технологій збереження та доступу до активних сховищ даних корпоративних інформаційних систем

2.2.2. Механізми забезпечення якості активних сховищ даних корпоративних інформаційних систем

2.2.3. Механізми забезпечення якості доступу до активних сховищ даних корпоративних інформаційних систем

2.2.4. Механізми доступу та збереження даних клієнта активних сховищ даних корпоративних інформаційних систем

2.3. Розробка моделей вибору методів та засобів збереження та доступу і аналізу даних в проектах корпоративних інформаційних систем

2.4. Вибір інструментального забезпечення розробки проектів активних сховищ даних корпоративних інформаційних систем

2.5. Визначення властивостей мов веб–програмування корпоративних інформаційних систем з веб–доступом

2.6. Висновки до розділу 2

РОЗДІЛ 3. ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕНЬ АКТИВНИХ СХОВИЩ ДАНИХ

3.1. Розробка адаптивної моделі навантаженого сервера активного сховища даних корпоративної інформаційної системи

3.2. Результати тестування активного сховища даних для різних“engines”
РСУБД MySQL

3.3. Висновки до розділу 3

РОЗДІЛ 4. РОЗРАХУНОК ЕКОНОМІЧНОГО ЕФЕКТУ ВІД ВПРОВАДЖЕННЯ ІС

4.1. Загальні положення

4.2. Розрахунок витрат на створення і впровадження ІС управління програмними проектами компаній з розробки програмного забезпечення

4.3. Витрати на впровадження системи з урахуванням вартості навчання кадрів і витрати на монтаж і налаштування

4.4. Економічна ефективність розробки і впровадження ІС управління програмними проектами компаній з розробки програмного забезпечення

РОЗДІЛ 5. ОХОРОНА ПРАЦІ

5.1. Аналіз шкідливих та небезпечних факторів на робочому місці програміста

5.2. Розрахунок системи штучного освітлення приміщення

5.3. Основні положення техніки безпеки під час експлуатації ЕОМ

РОЗДІЛ 6. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

6.1. Вступ

6.2. Проблема екології

6.3. Облік дорогоцінних металів у виробках

6.4. Процес переробки

6.5. Утилізація комп'ютерів у інших країнах

6.6. Висновки

ЗАГАЛЬНІ ВИСНОВКИ ПО РОБОТІ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

Додаток А. Технічне завдання на розробку програмного забезпечення

підсистеми активного сховища даних

Додаток Б. Структура таблиць бази даних клієнтів активного сховища даних корпоративної інформаційної системи ІТ-компанії (фрагмент)

Додаток В. Приклад реалізації моделей даних з механізмами активного сховища даних інформаційної системи (фрагмент)

Додаток Г. Реалізація спрощеного алгоритму вибору інструментальних засобів відповідно до критеріїв оцінки вимог до ІС

ВСТУП

Останнім часом широко розвивається напрямок корпоративних сховищ даних, а системи управління (або керування) базами даних (СУБД, СКБД) є одним з найбільш поширених класів прикладних систем для корпоративних серверів даних. Це пов'язано з тим, що обсяги даних, що в той чи інший спосіб доступні в корпоративному секторі, в тому числі і в Інтернет, вже становлять сотні терабайт і продовжують зростати, а спеціальні алгоритми традиційних СУБД не можуть забезпечити прийнятний час відгуку на запитах даних в задачах аналізу та побудови корпоративної звітності. Саме тому для вирішення таких задач останнім часом використовують спеціалізовані рішення для аналізу і обробки даних на рівні систем збереження даних, які, у свою чергу, виконуються на спеціальних обчислювальних системах з масивно-паралельною архітектурою, що дозволяють здійснити такий аналіз даних на значних обсягах даних. Такі архітектури дозволяють розпаралелити обробку даних і винести частину обробки з рівня СУБД на рівень серверів зберігання.

Часткове вирішення таких задач також забезпечують хмарні обчислення (Cloud computing). Ця технологія дуже приваблива для користувачів, оскільки дозволяє відносно просто отримати через Інтернет у тимчасове користування деякий сервіс для зберігання і аналізу даних. Наприклад, можна замовити комп'ютер із заданим об'ємом пам'яті, дисків, кількістю процесорів, з попередньо встановленими ОС і СУБД або іншими пакетами ПЗ і працювати з ним зі свого офісу, хоча фізично і обладнання, і ПО розташовуються і обслуговуються в віддаленому чужому центрі обслуговування даних. В інтернет поширені служби, де можна замовити потрібний дисковий ресурс і зберігати на ньому свої файли. Прикладами Cloud Computing є сервіси зберігання (Amazon S3), обчислювальні сервіси (Google App Engine, Amazon EC2) і сервіси даних (Amazon SimpleDB, Microsoft SQL Server Data Services, Google's Datastore).

Зважаючи на вимоги Інтернет-сервісів до обробки значних обсягів даних та активного подання запитів на доступ до даних сучасні СУБД повинні

відповідати цим вимогам. Крім того, Інтернет-доступ до СУБД, в тому числі в середовищі cloud computing, висуває підвищені вимоги до СУБД: з точки зору захисту і розмежування доступу (дані розміщуються на чужих комп'ютерах); масштабованості (коли користувач сервісу може розширити вимоги до обсягів дискового простору, потужності та числа процесорів). Оскільки управління такими системами відбувається через Інтернет, це також визначає вимоги до методів та механізмів вибору технологій зберігання даних, вдосконалення засобів адміністрування баз даних та реорганізації їх структури відповідно до бізнес-процесів інтернет-користувачів СУБД.

Таким чином істотною проблемою розробки та експлуатації (у тому числі – розширення) корпоративних інформаційних систем є обмеження, пов'язані з механізмами доступу і аналізу до даних в мережі Інтернет в частині вибірки, передачі, обробки та захисту даних; обмеження, пов'язані зі складністю моделювання бізнес процесів, що постійно змінюються та вимагають модифікації алгоритмів обробки корпоративних даних та самих структур даних. Як наслідок, розширення функціональності таких систем часто супроводжується змінами в структурах даних корпоративної бази даних, які складаються в результаті розвитку таких систем, що для корпоративних систем є нетривіальним завданням і часто вимагає суттєвої переробки модулів, інтерфейсів і архітектури систем, які використовують такі дані і виконують їх аналіз.

Зважаючи на значну кількість технологій організації, збереження і аналізу даних в системах баз даних, механізмів доступу до даних та можливостей їх застосування в корпоративних системах та, зокрема, інтернет-проектах, постає задача науково-обґрунтованого вибору механізмів збереження даних в сховищах даних та механізмів доступу і аналізу до цих даних.

Актуальність роботи визначається поширенням орієнтованих на користувача корпоративних інформаційних систем та наявними проблемами, викликаними бажанням надавати інтелектуальні адресні послуги

на основі результатів бізнес-аналізу в умовах значних обсягів даних.

Активне сховище даних (ADW) являє собою поєднання продуктів, функцій, сервісів та робочих партнерств, які підтримують бізнес-стратегію Active Enterprise Intelligence і дозволяють частково подолати зазначені проблеми.

Метою дослідження є підвищення ефективності процесів прийняття рішень щодо вибору технологій організації активних сховищ даних в клієнто-орієнтованих корпоративних системах на основі науково обгрунтованих моделей та механізмів.

Об'єктом дослідження є технології збереження та доступу до даних в клієнто-орієнтованих корпоративних системах.

Предметом дослідження є процеси прийняття рішень щодо вибору технологій організації і доступу до даних в клієнто-орієнтованих корпоративних системах.

Задачі дослідження: виконати аналіз існуючих технологій збереження та доступу до даних, визначити критерії оцінки інструментальних засобів, розробити моделі та механізми вибору інструментальних засобів та технологій проектування сховищ даних і програмного забезпечення роботи з даними в клієнто-орієнтованих корпоративних системах.

Наукова новизна роботи полягає у розробці моделей адаптивного вибору та оцінки активних механізмів збереження даних та механізмів доступу до даних клієнто-орієнтованих корпоративних систем на основі активних сховищ даних.

РОЗДІЛ 1. АНАЛІЗ МЕТОДІВ І ЗАСОБІВ ЗБЕРЕЖЕННЯ ТА АНАЛІЗУ ДАНИХ В КОРПОРАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

1.1. Огляд методів і засобів збереження та аналізу даних в корпоративних інформаційних системах

1.1.1. Моделювання бізнес процесів в корпоративних інформаційних системах

Складність вирішення завдання моделювання бізнес процесів загалом та реорганізації структур даних зокрема в системах з реляційними базами даних (які набули найбільшого поширення) зростає з ускладненням структури даних внаслідок невідповідності методів організації даних в програмах, що використовують базу даних (вони, зазвичай, розробляються відповідно до методологій об'єктно-орієнтованого програмування) і методів організації даних в необ'єктних сховищах даних, що, в цілому, призводить до необхідності побудови і виконання над базами даних вкрай складних запитів. Значна кількість методологій щодо проектування та організації баз даних лише підтверджують складність та різноплановість процесів створення баз даних та їх істотну залежність від бізнес логіки задач, для яких такі бази даних створюють. Наслідком спрощеного підходу до проектування баз даних та невиправдане використання найбільш поширених реляційних баз даних призводить до невідповідності функціоналу, що забезпечується СУБД, вимогам до механізмів доступу і аналізу даних, які визначаються програмним забезпеченням зокрема та бізнес-логікою в цілому. При цьому розробка запитів для формування даних для бізнес-аналізу на складних структурах даних є невиправдано трудомістким завданням і, в окремих випадках, вимагає створення допоміжних таблиць даних (Temporary tables) або видів (Views), що, в свою чергу, ускладнює роботу з базою даних через зростання вимог до обсягів оперативної пам'яті. При цьому будь-які зміни структури бази даних внаслідок змін у бізнес логіці або розвитку систем змушує розробників виконувати переробку раніше створених запитів.

Серед різних компонентів апаратури сервера СУБД конфігурація пам'яті системи зазвичай має найбільше вплив на її продуктивності — більша частина основної пам'яті в системах зберігання і обробки даних використовується як буфер (кеш) даних, що дозволяє в багатьох випадках усунути необхідність виконання фізичного введення-виведення з диска. Оскільки звернення до основної пам'яті виконується приблизно в 30000 разів швидше, ніж звернення до швидкого диску, мінімізація дискового вводу-виводу є першочергово важливим завданням.

Існуюче в даний час співвідношення між цінами підсистеми пам'яті і дискової підсистеми показує, що є економічно доцільним кешувати дані, до яких здійснюються звернення більше одного разу кожні п'ять хвилин. Це означає, що дані, до яких звернення відбуваються більш часто, ніж один раз на п'ять хвилин, повинні кешуватися в основній пам'яті. Відповідно, щоб оцінити розмір кеша даних необхідно підсумувати об'єми всіх даних, які додаток припускає використовувати більш часто, ніж один раз на п'ять хвилин на рівні всієї системи. Тому при визначенні необхідного обсягу основної пам'яті розраховують, принаймні, на такий розмір кешу даних. Додатково зазвичай резервують ще 5-10% для розміщення верхніх рівнів індексів В-дерев, збережених процедур та іншої керуючої та допоміжної інформації СУБД.

Кожна комерційна СУБД забезпечує механізм для визначення вартості чи переваг зміни розміру різних кешів СУБД. Коли система інстальована і починає працювати, необхідно використовувати ці механізми для визначення ефекту від зміни розмірів кешу вводу-виводу СУБД.

Хоча основною метою СУБД на макрорівні є управління томами даних, які за визначенням мають дуже великий обсяг і неминуче у багато разів перевищують обсяг основної пам'яті, проведені дослідження показали, що доступ до даних в загальному випадку підкоряється правилу "80/20" або "90/10": 80-90% всіх звернень виконуються для отримання 10-20% даних. Більше того, виявилось, що до постійно потрібних даних, звернення до яких

становлять 80-90% від усіх звернень, знову застосовано правило 80/20” або “90/10”. Таким чином, приблизно 80-90% усіх звернень до даних пов’язані з доступом до приблизно 1% агрегованих даних. Слід зазначити, що це відношення включає звернення до внутрішніх метаданих СУБД (що включають індекси В-дерев і т. п.), отримання і обробка яких зазвичай приховані від прикладного програміста. Хоча бажано мати коефіцієнт попадань в кеш приблизно на рівні 95%, з економічних міркувань неможливо сліпо забезпечити обсяг кешу в пам’яті, що дозволяє розмістити 10-20% всіх даних: навіть для скромних баз даних об’ємом 5 Гбайт це зажадало б збільшення розміру основної пам’яті до 500 Мбайт. Проте зазвичай завжди можливо забезпечити кеш обсягом в 1% всіх даних навіть для дуже великих баз даних. Максимальний обсяг основної пам’яті сучасних серверів може бути дуже великим.

Тут слід зазначити, що попередні оцінки є доволі “середніми” і в реальних задачах можуть значно відрізнятись як в більшу, так і в меншу сторону.

Природно, конфігурація системи повинна забезпечувати також простір для традиційного використання пам’яті. В СУБД на базі Unix і MS Windows завжди необхідно забезпечити принаймні 100-200 Мбайт для ядра базової операційної системи та системних служб. Далі, необхідно передбачити окремий простір для ряду програм бізнес-аналітики на рівні СУБД а також програм обслуговування сховища даних: ведення журналу, перевірки узгодженого стану, архіваторів і т. п.

Оброблювана кожним процесором додаткова інформація вимагає виділення деякого простору для того, щоб обійти інтенсивну фрагментацію пам’яті на зовнішніх носіях. Ці задачі виявляються тим більше складними, що помилки вибору структури бази даних важко виправляти через істотні розбіжності в механізмах організації даних в базах даних.

Наслідком вище зазначеного є обмеження на обробку даних в системах збереження даних, коли обробка базується на всіх або значній кількості даних — така обробка потребуватиме обсягів оперативної пам’яті рівних або навіть

більше, ніж обсяг даних у БД, який в бізнес секторі може становити одиниці або навіть сотні терабайт. Саме тому така обробка даних потребує обсягів обчислювальних ресурсів, які є недосяжними в класичних сховищах даних і потребує спеціальних механізмів. Модель доступу до класичного сховища даних корпоративної інформаційної системи (пасивному, без можливостей бізнес-аналітики) представлена на рисунку 1.1.

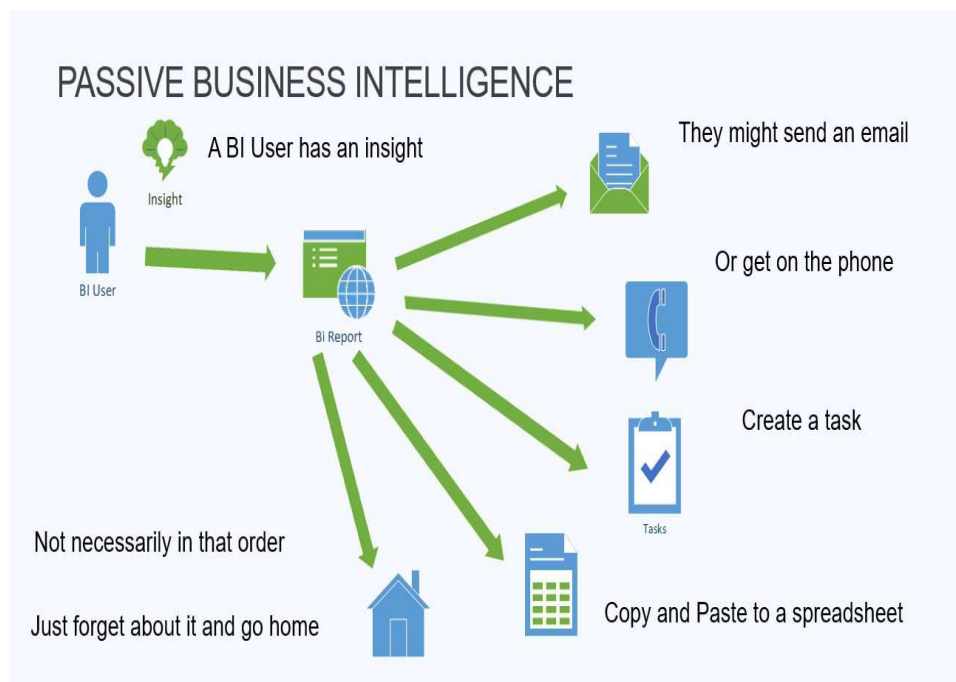


Рис. 1.1. Модель доступу до даних у пасивному сховищі даних корпоративної БД

Для вирішення задач бізнес-аналізу даних традиційно використовують системи онлайнової аналітичної обробки (OLAP) – це технологія, яка використовується для організації великих бізнес-баз даних і підтримки бізнес-аналітики. Бази даних OLAP розділено на один або кілька кубів, кожен із яких упорядковано та розроблено адміністратором куба відповідно до способу отримання і аналізу даних, що спрощує створення й використання звітів зведених таблиць і зведена діаграма потрібних звітів.

З метою визначення особливостей реалізації бізнес логіки в корпоративних

інформаційних системах та її відображення в моделі та механізми збереження та доступу до даних для СУБД розглянемо особливості основних типів сучасних баз даних та визначимо їх переваги та недоліки стосовно розробки навантажених систем збереження даних та їх бізнес-аналізу.

1.1.2. Реляційні СУБД

Корпоративні рішення на базі реляційних СУБД є традиційними для багатьох проектів через поширення цих СУБД та високий рівень формалізації реляційної алгебри. Вони також дозволяють забезпечити відображення даних предметної області в структури даних сховища у найбільш простий спосіб (Рис. 1.2).

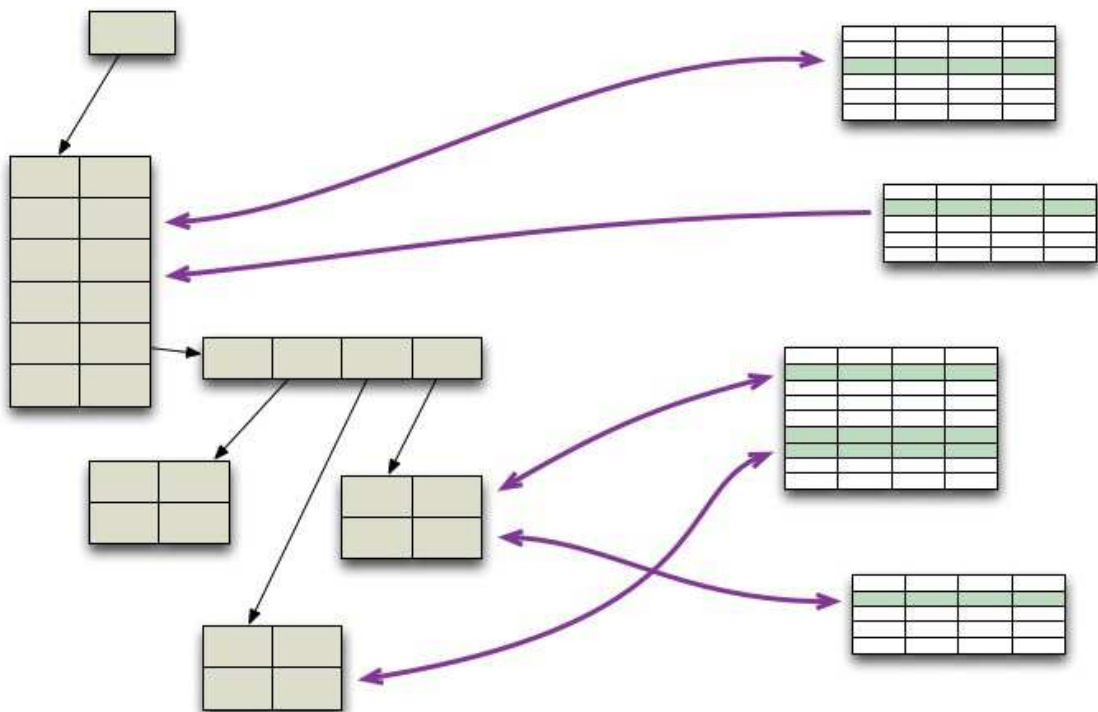


Рис. 1.2. Відображення бази даних (праворуч) на предметну область (ліворуч) в реалізацію інформаційної системи для РСУБД

Найбільш поширеними базами даних є наступні реляційні – MySQL, PostgreSQL, MSSQL і Oracle. Перші дві безкоштовні і найчастіше використовуються для корпоративних систем з інтернет доступом. Другі дві – платні і рідше використовуються в корпоративних системах і, зокрема, інтернет-проектах, а частіше в корпоративних інформаційних системах з бізнес-аналітикою. По суті, для звичайних проектів в технічному плані немає істотної різниці, яку СКБД використовувати, але в економічному плані вигідніше використовувати найпоширенішу MySQL (та її відгалудження MariaDB) або трохи менш поширену PostgreSQL, для яких більше розробників, ніжче вартість підтримки і розробки.

MariaDB – відгалуження реляційної СУБД MySQL, розроблювальне сообществом під ліцензією GPL. MariaDB повністю сумісна з додатками, що використовують MySQL, а перехід на цю СУБД виправданий тим, що MySQL вже не так активно розвивається.

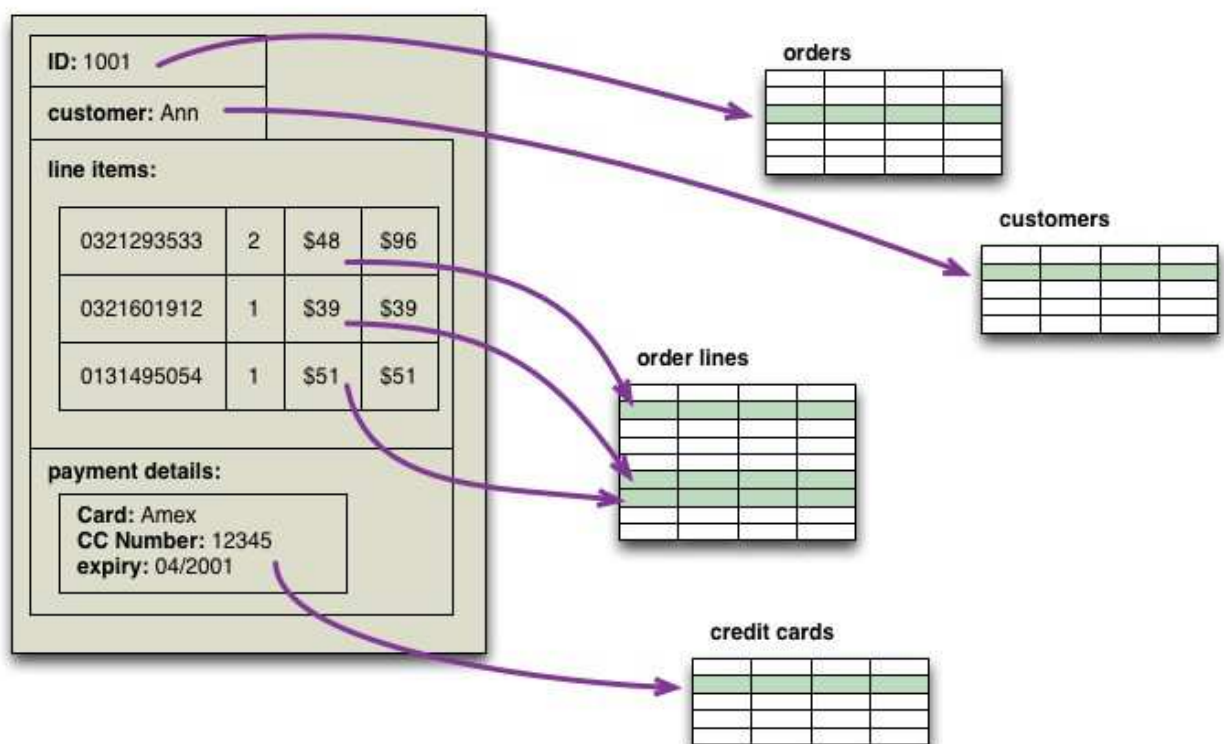


Рис. 1.3. Відображення предметної області в реалізації інформаційної системи для РСУБД

Крім зрозумілої політики поширення продукту, дана СУБД відрізняється від MySQL більш високою продуктивністю, новими можливостями і меншою кількістю помилок в коді (зокрема, за рахунок кращого покриття тестами і більш активного розвитку).

У MariaDB вбудовані покращений оптимізатор запитів, безпечна і швидка реплікація, більш швидкі індекси для механізму зберігання даних MEMORY, більш висока продуктивність перекодування символів, використання пулу потоків, а також багато інших покращання, позитивно впливають на продуктивність.

У MariaDB відбулася відмова від підсистеми зберігання даних InnoDB і його заміна на XtraDB. Також на додаток до MyISAM, Blackhole, CSV, Memory і Archive включені підсистеми зберігання Aria, PBXT (до версії 5.5), FederatedX, OQGRAPH, SphinxSE, IBMDB2I (до версії 5.5), TokuDB, а також з версії 10.0 – Cassandra, CONNECT, SEQUENCE, Spider.

1.1.3. NoSQL СУБД

Останнім часом для вирішення завдань, пов'язаних зі складними методами обробки даних і складною організацією даних (що характерно при автоматизації складних бізнес процесів) набули поширення так звані NoSQL бази даних та сховища (MongoDB, CouchDB, Redis, Memcached). Вони значно молодше реляційних баз даних, істотно відрізняються від них і застосовуються частіше не для зберігання всіх даних системи, а лише для вирішення специфічних завдань (журналювання, кешування, черги завдань), і тому менш поширені в простих проектах.

MongoDB – це NoSQL сховище даних, вкрай зручне для зберігання інформації, яка не може бути у простий спосіб нормально структурована в рамках реляційних баз даних.

MongoDB – СУБД з відкритим вихідним кодом, що не вимагає опису схеми

таблиць і написана на мові C++. Завдяки спеціалізації бази даних вдалося відійти від принципу «один розмір під все», а за рахунок мінімізації методів роботи з транзакціями з'явилася можливість вирішення цілого ряду проблем, пов'язаних з нестачею продуктивності, що також сприяло поліпшенню горизонтального масштабування.

Документи (дані) в MongoDB зберігаються в JSON або BSON форматах (рис 1.5), робота з такою моделлю простіше кодується і простіше управляється, а внутрішня угруповання релевантних даних забезпечує додатковий виграш у швидкодії. MongoDB, за даними розробників, повинна заповнити розрив між найпростішими NoSQL-СУБД, що зберігають дані у вигляді «ключ – значення» (простими і легко масштабованими, але володіють мінімальними функціональними можливостями, такими як Memcached, наприклад) і великими реляційними СУБД (зі складною організацій даних (структурними схемами даних) і системами генерації складних запитам з вбудованими оптимізаторами).

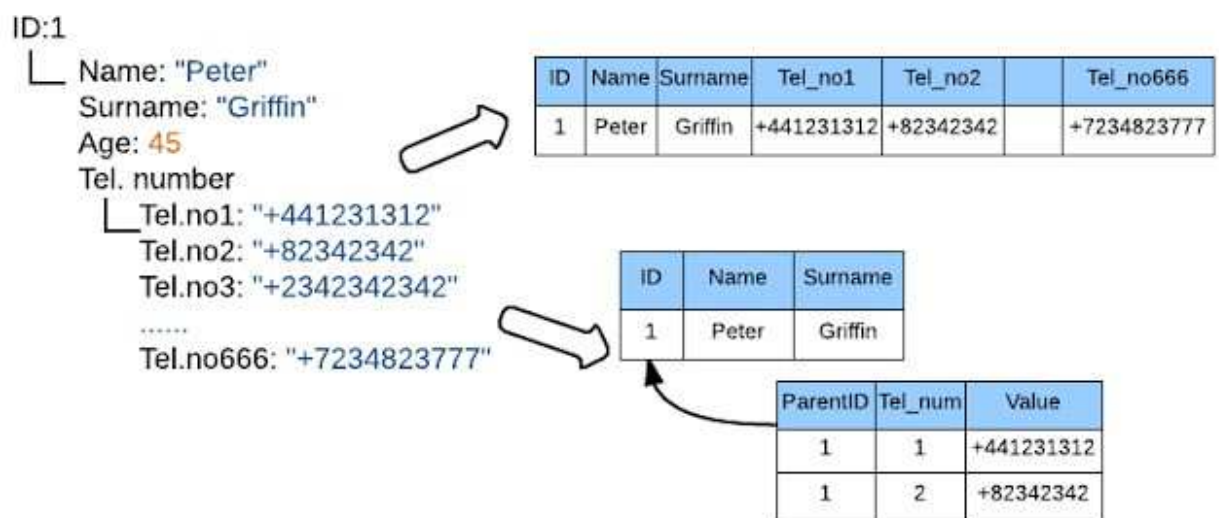


Рис. 1.4. Відображення предметної області в реалізації інформаційної системи для NoSQL СУБД

Переваги NoSQL:

- гнучка мова для формування запитів;
- динамічні запити;
- індексація колекцій;
- профілювання запитів;
- журналювання операцій запису;
- відмовостійкість і масштабованість;
- асинхронна реплікація і шардінг;
- повнотекстовий пошук з підтримкою морфології.

Робота з MongoDB реалізована в мовах програмування Java, C++, C#, PHP, Python, Perl, Ruby та інших, а відповідні компоненти є в багатьох фреймворках (наприклад, в Ruby on Rails або в Yii).

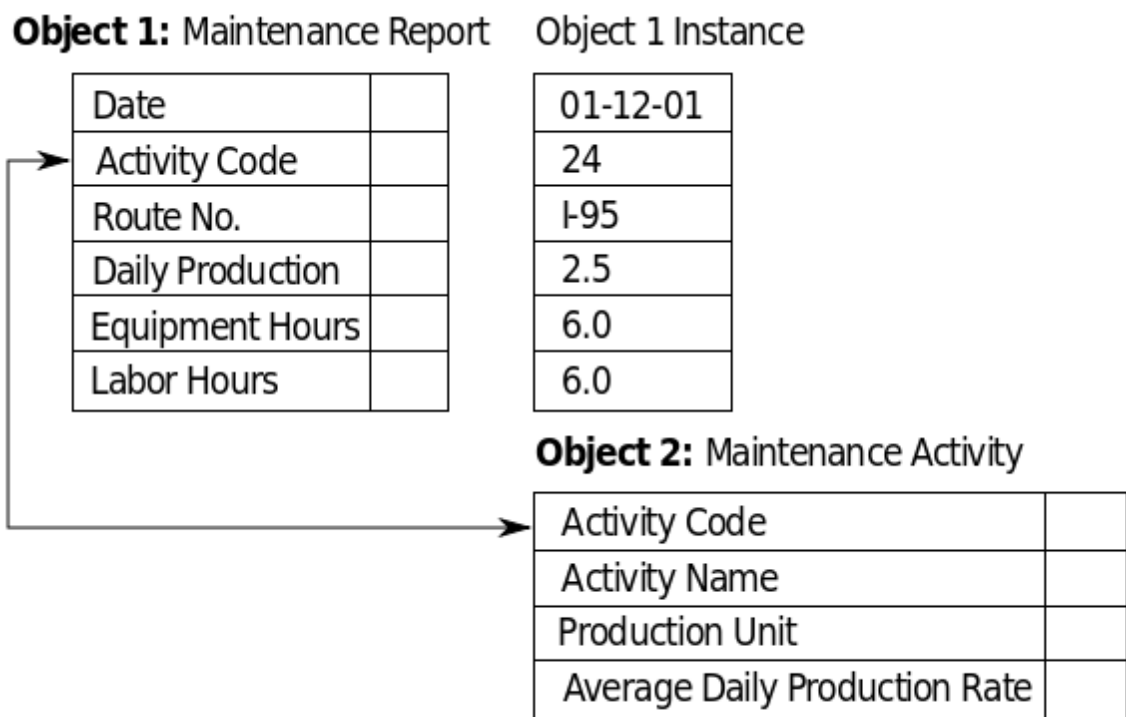


Рис. 1.5. Відображення даних в ОО СУБД

1.1.4. ОО СУБД

Зазначена проблема могла б бути подолана за рахунок використання

об'єктно-орієнтованих систем управління базами даних (СУБД), що дозволяють повною мірою задовольнити потреби в організації даних з боку об'єктів програмного забезпечення, але, на жаль, об'єктно-орієнтовані СУБД на сьогодні не набули широкого поширення, насамперед – через свою складності і надмірної вартості. Перші публікації були викликані проблема РСУБД при високій складності відношень (рис. 1.2).

Об'єктно-орієнтована база даних зазвичай рекомендовані для тих випадків, коли потрібна високопродуктивна обробка даних, що мають складну структуру.

Обов'язкові характеристики, яким має відповідати будь-яка ООБД наступні:

- Підтримка складних об'єктів. В системі має бути передбачена можливість створення складових об'єктів за рахунок застосування конструкторів. Необхідно, щоб будь-який з конструкторів можна було використовувати з будь-яким об'єктом.
- Підтримка індивідуальності об'єктів. Всі об'єкти мають мати унікальний ідентифікатор, котрий не залежить від значень їх атрибутів.
- Підтримка інкапсуляції. Інкапсуляція досягається за рахунок того, що програмісти мають право доступу тільки до специфікації інтерфейсу методів, а дані і реалізація методів приховані всередині об'єктів.
- Підтримка типів і класів. Вимагається, щоб ООБД підтримувалась хоча б одна концепція відмінності між типами і класами.
- Підтримка наслідування типів і класів від їх предків. Підтип або підклас має наслідувати атрибути і методи від супертипу або суперкласу.
- Перезавантаження у поєднанні з повним пов'язуванням. Реалізація методу має залежати від типу об'єктів, для якого даний метод викликається.
- Обчислювальна повнота. Мова операцій над даними має бути мовою програмування загального призначення.
- Набір типів даних має бути розширюваним. Користувач повинен мати

засоби створення нових типів даних на основі набору зумовлених системних типів. Більш того, між способами використання системних і користувацьких типів даних не повинно бути жодних відмінностей.

З метою втілення зазначених концепцій та подолання наявних в РСУБД проблем останнім часом широко використовують об'єктно-орієнтовані надбудови над реляційними базами даних (ORM, Object Relational Mapping), які забезпечують підтримку відповідності організації даних на рівні додатку і на рівні структур даних бази даних. Таким чином, використання ORM надає розробнику можливість відносно вільно маніпулювати даними на рівні об'єктів і абстрагуватися від особливостей реалізації структур даних на конкретному сервері баз даних. Це також дозволяє відносно простими засобами отримувати зрозумілий і якісний код, спрощує його тестування, подальшу модифікацію і супровід.

Деякі програмні пакети вирішують цю проблему, надаючи бібліотеки класів, здатних виконувати перетворення даних автоматично. Маючи список таблиць в базі даних і об'єктів в програмі, вони автоматично перетворюють об'єктні запити в реляційні та забезпечують конвертацію даних з внутрішнього подання бази даних в об'єктне уявлення, придатний для використання в об'єктно-орієнтованій програмі. При цьому в результаті виконання ORM-запиту генеруються і виконуються відповідні запити до таблиць сервера баз даних, а результати перетворюються в об'єкти відповідного типу.

ORM також відмінно підходить для реалізації і втілення такої сучасної парадигми ОО-програмування, як MVC (Model-View-Controller, рис 1.6). Ця парадигма дозволяє відокремити інтерфейси програми від даних і бізнес логіки, а також загальних операцій над даними (рис. 1.7).

Застосування ORM-технологій має і свої недоліки, насамперед, через деяке зменшення пропускну здатності сервера баз даних. Більше того, шар транзакцій при цьому виявляється повільнішим, а часто ще й неефективним (особливо в термінах згенерованого запиту, записаного мовою запитів – SQL).

Таким чином, використання ORM може призвести до того, що програми, які його використовують, будуть працювати повільніше і використовувати більше оперативної пам'яті, ніж програми, що містять «вручну» написані запити. Тим не менше, ORM позбавляє розробника від необхідності написання великої кількості коду, часто одноманітного і схильного до помилок, і, головне, який тісно пов'язаний зі структурами даних, що унеможливорює оперативну модифікацію цих структур. Тим самим ORM значно підвищуючи швидкість розробки. На додаток до можливостей генерації запитів на основі об'єктних моделей більшість сучасних реалізацій ORM дозволяють також жорстко визначати код SQL запитів, який буде використовуватися при тих чи інших непередбачених розробниками ORM діях (збереження в базу даних, завантаження, пошук тощо) з постійним об'єктом.

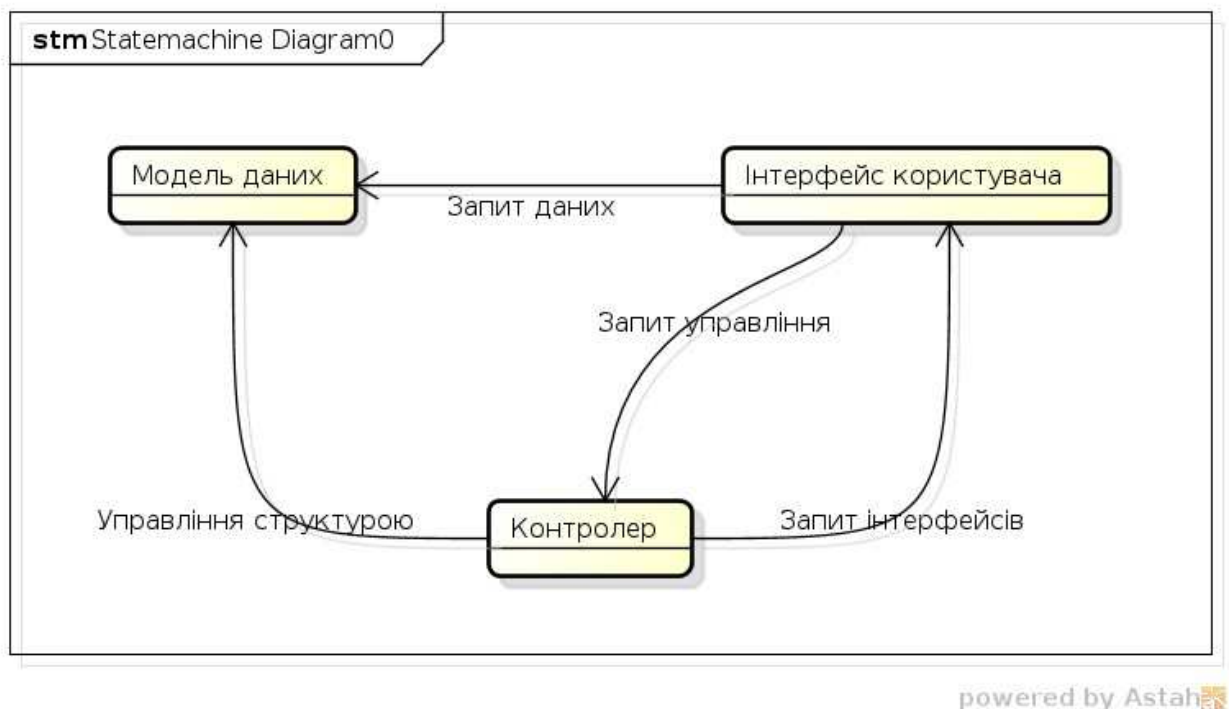


Рис. 1.6. Структура додатку для моделі Model-View-Contriller

PureMVC – це фреймворк, що базується на шаблонах проектування. Він виник з нагальної необхідності проектування високопродуктивних RIA-клієнтів. Зараз він вже портований іншими мовами та платформами, включаючи

серверні середовища[хоча інтерпретація та реалізація мають свої особливості для кожної підтримуваної PureMVC мовних платформ.

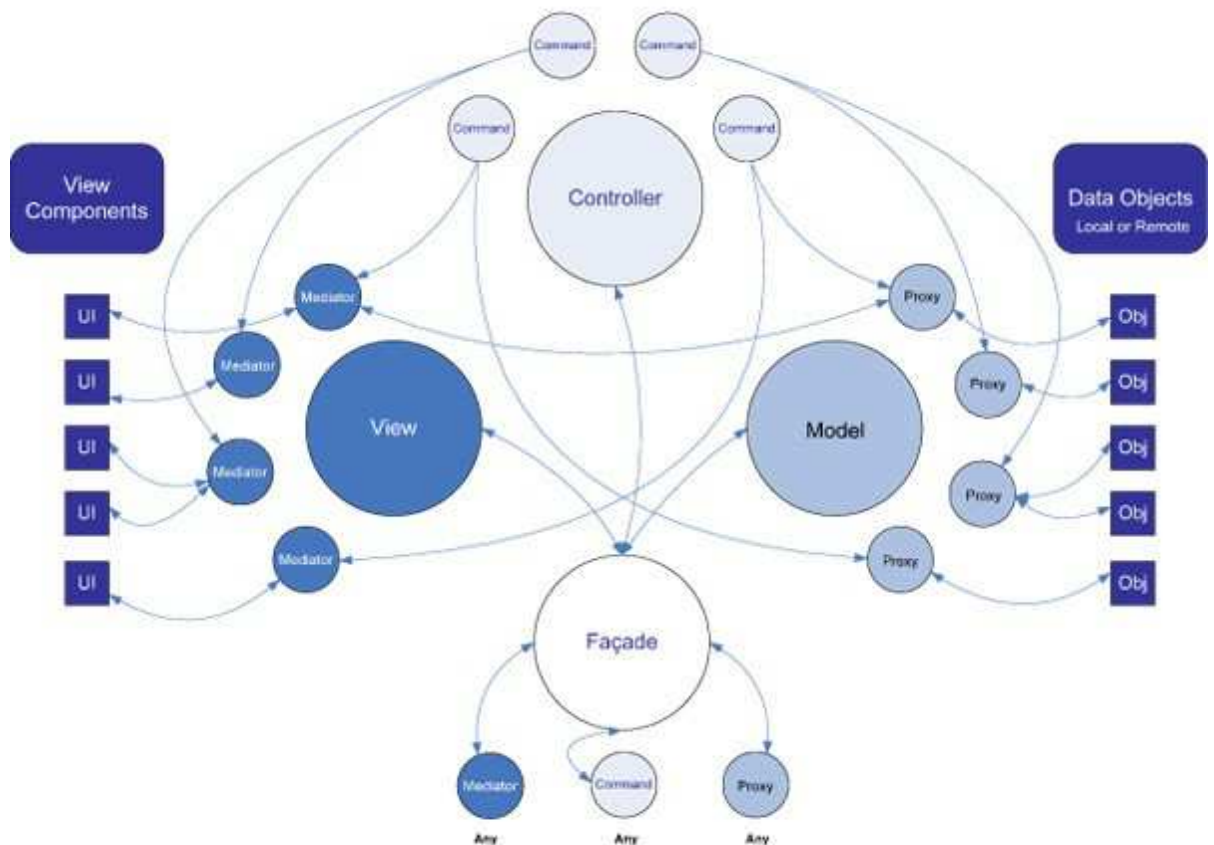


Рис. 1.7. Деталізована структура додатку в моделі Model-View-Contriller для платформ PureMVC

Взагалі, для проектів, які характеризуються високим навантаженням на сховища даних, питання отримання необхідної продуктивності гостро стоїть і без застосування ORM. Але для таких систем також мають велике значення та можливості отримання відносно простих інструментів розширення системи, а саме такі можливості і забезпечуються використанням ORM.

Впровадження ORM зазвичай призводить до деградації продуктивності системи на близько 30%. Маючи на увазі швидкий прогрес у підвищенні продуктивності обчислювальних систем (близько 30% – 50% на рік) і тенденції до зменшення вартості обладнання, можна розраховувати на те, що протягом

року недоліки системи, пов'язані з деградацією обчислень внаслідок використання ORM будуть повністю нівельовані за рахунок підвищення продуктивності апаратного забезпечення. При цьому переваги від впровадження ORM, пов'язані з можливостями більш простого розширення структур даних, повною мірою будуть збережені і надалі.

1.1.5. Документо-орієнтовані бази даних

Документо-орієнтовані бази даних — спеціально призначені для зберігання ієрархічних структур даних (документів) і зазвичай реалізовані за допомогою підходу NoSQL. В основі документо-орієнтованих СУБД лежать документні сховища, які мають структуру дерева або лісу. Структура дерева починається з кореневого вузла і може містити кілька внутрішніх і листових вузлів. Листові вузли містять дані, які при додаванні документа заносяться в індекси, що дозволяє навіть при досить складній структурі знаходити місце (шлях) шуканих даних.

API для пошуку дозволяє знаходити в документо-орієнтованих базах даних за запитом документи або окремі частини документів. На відміну від сховищ типу ключ-значення, вибірка за запитом до документного сховища може містити частини великої кількості документів без повного завантаження цих документів на клієнтські машини. Такий підхід виявляється цінним для систем, в яких обсяг трафіку між клієнтами та сервером є критичним.

1.1.6. Розподілені бази даних

Розподілені БД складають ще один напрям в просторі досліджень і розробок систем керування базами даних. У цих системах доводиться вирішувати всі завдання, властиві централізованим СКБД, але, як правило, в більш складних постановках. У розподілених системах виникають і специфічні проблеми, від вирішення яких багато в чому залежить ефективність, надійність і доступність систем БД. В даний час більшість розподілених СКБД базується

на реляційній моделі даних і розрахована на використання в локальних обчислювальних мережах. Багато проблем поширюються і на розподілені СКБД в територіально рознесених мережах, і майже всі проблеми зберігаються для розподілених СКБД, заснованих на інших моделях даних.

Розподілена база даних (РБД) - це множина логічно взаємозалежних баз даних, розподілених у комп'ютерній мережі.

Серед особливостей використання таких баз даних можна відзначити:

- великий обсяг обмінів даними (напружений мережевий трафік);
- низьку надійність системи через відмову каналів зв'язку та центральної СКБД;
- низьку загальну продуктивність (вузькі місця - мережа і центральний сервер);
- великі витрати на розробку і експлуатацію.

1.1.7. Висновки щодо методів та засобів збереження даних

Для звичайних проектів простіше використовувати MySQL або PostgreSQL, оскільки на простих операціях не надто помітна різниця між різними базами даних. Однак, якщо проект передбачає складну логіку обробки даних, то вибір бази варто проводити виходячи з технічних характеристик та області використання кожної окремої системи та її сховища даних (Таблиця 1.1, рис 1.8).

1.2. Огляд інтернет-сервісів та протоколів доступу до даних корпоративних інформаційних систем

1.2.1. Механізми доступу та сервіси

Окрім структури організації даних бази даних також істотно відрізняються механізмами доступу, які можуть бути реалізовані на різних рівнях реалізації інформаційної системи: на рівні файлової системи ОС, СУБД або навіть на рівні додатку.

Залежно від рівня реалізації механізмів доступу також істотно відрізняються можливості щодо побудови складних запитів (чим нижче рівень – тим складніше) та швидкості їх виконання (чим вище рівень – тим повільніше).

Таблиця 1.1. Типи сховищ баз даних за областями призначення

<i>№</i>	<i>Тип сховища</i>	<i>Область використання сховища</i>
1	Реляційні СУБД	Класичні проекти з відносно простою бізнес логікою
2	NoSQL СУБД	Проекти зі складною бізнес логікою та відносно невеликим обсягом даних
3	ОО СУБД	Проекти зі складною бізнес логікою та відносно великим обсягом даних
4	Документо-орієнтовані СУБД	Проекти з ієрархічно-впорядкованими даними
5	Розподілені СУБД	Проекти систем обробки даних з низькими вимогами до надійності
6	Файлові сховища	Проекти з масовим доступом до окремих файлів

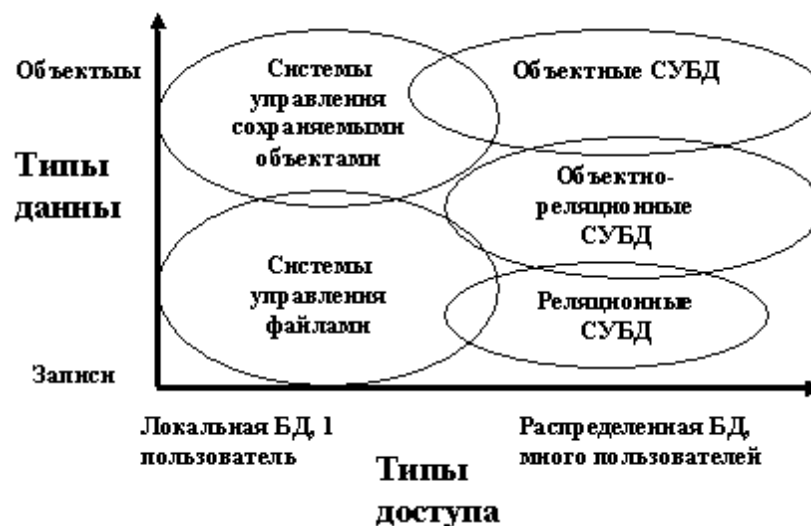


Рис. 1.8. Области використання СУБД з різною організацією

1.2.2. Інтернет-сервери HTTP та HTTPS

Найбільш популярний серед інтернет-розробників вважається комплект LAMP (Linux, Apache, MySQL, PHP). Цей інтернет-сервер дозволяє підключати зовнішні модулі для роботи з інтерпретуються мовами програмування (наприклад, з PHP, Ruby, Python) і для надання даних, може використовувати СУБД для аутентифікації користувачів, можлива модифікація повідомлень про помилки і багато іншого. Apache є кросплатформним ПЗ, підтримує операційні системи Linux, BSD, Mac OS, Microsoft Windows, Novell NetWare, BeOS, хоча найбільш часто використовується тільки на серверних дистрибутивах Linux і BSD.

Сам по собі інтернет-сервер вельми потужний і надійний, але споживання ресурсів при його роботі зазвичай досить високо. Часто Apache використовують як бекенд-сервера, встановлюючи позаду інтернет-сервера nginx.

HTTPS – захищений протокол передачі гіпертексту. HTTPS – розширення протоколу HTTP, що підтримує шифрування. Передані по протоколу HTTPS дані зашифровані, що забезпечує захист від прослуховування. HTTPS широко використовується в світі і підтримується всіма популярними браузерами.

HTTPS (HyperText Transfer Protocol Secure) – це не окремий протокол, а

розширення звичайного HTTP, що працює через шифровані транспортні механізми SSL або TLS. HTTPS забезпечує захист від атак, заснованих на прослуховуванні мережевого з'єднання, тобто від сніферських атак і атак типу man-in-the-middle.

На відміну від HTTP, який використовує TCP-порт 80, для HTTPS за замовчуванням використовується TCP-порт 443. Щоб підготувати інтернет-сервер для обробки https-з'єднань, адміністратор повинен отримати і встановити в систему SSL-сертифікат для цього інтернет-сервера.

SSL-сертифікат – це свого роду «електронний паспорт» сайту, в якому він підтверджує, що він дійсно той сайт, за який видає себе. Сертифікат сайту покликаний додатково підтверджувати, що цей сайт – дійсно сайт банку, платіжної системи і т.д. SSL-сертифікат, як і всякий паспорт, зберігає різні дані: найменування організації та підрозділи, країну, регіон, місто і ім'я сервера, для якого даний сертифікат був створений.

Також, для нормальної роботи HTTPS, SSL-сертифікат повинен бути підписаний організацією, якій довіряють все. У разі SSL-сертифікатів такими організаціями є центри сертифікації. І якщо сертифікат підписаний надійним центром сертифікації, то вважається, що він містить правильну і відповідну дійсності інформацію.

1.2.3. Особливості використання файлових сховищ в задачах бізнес-аналітики

Файлові сховища або файловий хостинг – сервіс, що надає користувачу місце під його файли і цілодобовий доступ до них через інтернет-інтерфейс, як правило за протоколом HTTP. Такий сервіс дозволяє зручно «обмінюватися» файлами. На спеціальній сторінці файлового сховища (найчастіше на головній) користувач завантажує файл на сервер сховища і отримує постійне посилання, яке він може розсилати по e-mail, публікувати в блогах, на форумах тощо. Перейшовши за таким посиланням інший користувач може викачати

розміщений файл.

Інформація в файлових сховищах поділена, зазвичай, на три базові категорії:

- Захищена інформація. Режим доступу визначається власниками і надається за спеціальною угодою. До цього виду ресурсів відносяться комерційні архіви (наприклад, комерційні версії програм в архівах <ftp.microsoft.com>), закриті національні і міжнародні некомерційні ресурси (наприклад, роботи за міжнародними проектами CES або IAEA), приватна некомерційна інформація із спеціальними режимами доступу (наприклад, приватні добродійні фонди).
- Інформаційні ресурси обмеженого використання. До них відносяться ресурси обмеженого часу використання або обмеженого часу дії, наприклад, програми класу shareware.
- Відкриті інформаційні ресурси. До цих ресурсів відноситься все, що можна вільно отримати по мережі без спеціальної реєстрації (freeware). Це може бути документація, безкоштовне програмне забезпечення, фотографії, звукові файли тощо. Зазвичай, вільно поширюване програмне забезпечення не має сертифікату якості, але його розробники є відкритими для обміну досвідом.

Найбільшої популярності файлові хостинги набули, коли зростання користувачів Інтернету, можливості розміщення чи завантаження файлів великого розміру почали спричиняти величезного навантаження на багато ресурсів.

Завдяки файловим хостингам вдалося вирішити проблему перевантаження для ряду ресурсів, а також надати для звичайного користувача можливість передавання файлу, наприклад, розміром 100 мегабайт абсолютно безкоштовно. Природно, що за безкоштовність доводиться платити (банери на сторінці завантаження, відтерміноване викачування користувачем, примусовий перегляд інтернет-сторінок), бо витрати власників файлових хостингів

необхідно якимось окупати.

Перш ніж завантажити на певний сервіс файли корпоративної системи, слід ознайомитися з правилами роботи служби та файлового сховища. Також необхідно зважати, що одна з головних вимог публічних сховищ – обмеження щодо нелегального контенту (зокрема, неліцензійних програм, піратських копій медійного забезпечення тощо), використання якого порушує авторські права чи може вплинути на працездатність сервера. Якщо подібний матеріал і з'являється, він оперативно видаляється адміністрацією, а осіб, що його завантажили можуть притягнути до юридичної відповідальності. Ці обмеження діють навіть у випадках, якщо збереження таких копій (наприклад – комп'ютерних вірусів) є аспектом легальної діяльності підприємства, яке завантажує файли на сервер.

Більшість файлових сховищ дозволяють викачувати файли лише одним потоком, по одному за раз, і часто з паузою в годину/дві перед завантаженням наступного файлу. Пам'ятати, що вже вивантажено, а що ні, скільки часу лишилося до можливості нового викачування, отримати пряме посилання на файли, щоб поставити їх в чергу менеджера закачування – досить складно та непродуктивно. Кращим вирішенням цього обмеження є застосування спеціальних програм для викачування з файлових сховищ, наприклад, розробка Universal Share Downloader.

Для доступу до файлових сховищ здебільшого використовують спеціалізовані протоколи передачі даних. SFTP – протокол передачі файлів прикладного рівня, призначений для копіювання та виконання інших операцій з файлами поверх надійного і безпечного з'єднання. SSH використовується для безпечної передачі, що приховує (шифрувальної) логін і пароль, а також шифрувальної вміст.

SFTP не пов'язаний з FTP, за винятком того, що він теж передає файли і має аналогічний набір команд для користувачів. SFTP, або безпечний FTP, – це програма, що використовує SSH (Secure Shell) для передачі файлів. На відміну

від стандартного FTP він шифрує і команди, і дані, оберігаючи паролі та конфіденційну інформацію від відкритої передачі через мережу. За функціональністю SFTP схожий на FTP, але так як він використовує інший протокол, клієнти стандартного FTP не можуть зв'язатися з SFTP-сервером і навпаки.

Використання таких протоколів, як і файлових сховищ доцільне лише у випадку розподілених сховищ даних, коли дані зберігаються на одних вузлах розподіленої системи, а бізнес-аналітика виконується на інших, як правило спеціалізованих, вузлах системи.

1.2.4. Задачі бізнес-аналітики і технологія OLAP

OLAP – зручний інструмент аналізу, перегляду і візуалізації інформації. Традиційні звіти, навіть побудовані на основі єдиного сховища, позбавлені одного – гнучкості. Їх не можна "прокрутити", "розгорнути" або "згорнути", щоб отримати бажане представлення даних, тому отримання даних, їх представлення є доволі тривалою задачею. Як наслідок, бізнес-аналітик звичайно може перевірити за день декілька ідей, що не відповідає вимогам вирішення задач бізнес-аналітики, які мають вирішуватись онлайн протягом одиниць секунд.

Оперативні для OLAP дані збираються з різних джерел, очищаються, інтегруються та складаються у сховище (звичайно - реляційне). При цьому вони вже доступні для аналізу за допомогою різноманітних засобів побудови звітів. Далі дані (всі повністю або частково) готуються для OLAP-аналізу. Вони можуть бути завантажені у спеціальну БД OLAP або залишені у основному сховищі. Найважливішим його елементом є метадані, тобто інформація про структуру, розміщення та трансформацію даних. Завдяки їм забезпечується ефективна взаємодія різних компонентів сховища. Підсумовуючи, можна визначити OLAP як сукупність засобів багатовимірного аналізу даних, накопичених у сховищі. Теоретично засоби OLAP можна застосовувати

безпосередньо до оперативних даних або їх точним копіям (щоб не заважати оперативним користувачам). Але ми цим ризикуємо наступити вже описані вище граблі, т. е. почати аналізувати оперативні дані, які безпосередньо для аналізу непридатні.

Користувач OLAP отримує природну, інтуїтивно зрозумілу модель даних, організуючи їх як багатомірні куби (Cubes). Куб OLAP надає зручні швидкі засоби доступу, перегляду та аналізу ділової інформації. Осями багатовимірної системи координат є основні атрибути аналізованого бізнес-процесу. Наприклад, для продажу це може бути товар, регіон, тип покупця. Як один з вимірів використовується час. На перетинах осей – вимірювань (Dimensions) – знаходяться дані, що кількісно характеризують процес – заходи (Measures). Це можуть бути обсяги продажу в штуках або в грошовому вираженні, залишки на складі, витрати і т.п. по тижнях) відомості та здійснювати інші маніпуляції, які йому спадають на думку в процесі аналізу. Як заходи у тривимірному кубі, використані суми продажів, а як вимірювання - час, товар і магазин. Вимірювання представлені певних рівнях угруповання: товари групуються за категоріями, магазини - країнами, а дані час проведення операцій - за місяцями. Трохи згодом ми розглянемо рівні угруповання (ієрархії) докладніше.

1.2.5. Висновки щодо методів та засобів передачі даних

Широкий спектр технологій доступу до даних породжує проблеми вибору цих технологій в окремих проектах (Таблиця 1.2).

Вибір конкретних технологій здійснюється, як правило, за результатами вибору бази даних для проекту.

Таблиця 1.2. Типи технологій доступу залежно від баз даних

№	Тип доступу	Область призначення сховища
1	HTTP-доступ	Файлові сховища, реляційні СУБД
2	FTP-доступ	Файлові сховища з розподіленим

		зберіганням
3	P2P-доступ	Файлові сховища з розподіленим зберіганням

1.3. Проблеми доступу до даних і аналізу даних в проектах корпоративних інформаційних систем

1.3.1. Проблеми захисту даних та каналів передачі даних

Очевидно, що засоби захисту даних від несанкціонованого доступу дуже важливі і будуть продовжувати вдосконалюватися. Будуть ускладнюватися алгоритми кодування даних на всіх рівнях (в БД, архівах, при передачі по мережі), способи авторизації і аутентифікації і т.д. Останнім часом з'явилася тенденція виносу механізму управління користувачами (обліковими записами) з окремих СУБД і додатків в єдину централізовану систему організації (Identity & Access management). Що спрощує управління користувачами в масштабах організації та дозволяє керувати ними з урахуванням вимог бізнесу, в автоматичному режимі. Удосконалюються методи аудиту, з'являтиметься інтегрована середовище збору та аналізу аудит-інформації всієї організації (зі безлічі додатків і БД).

Крім того, інтенсивно вдосконалюються засоби захисту даних усередині БД. Наприклад, швидко розвивається механізм завдання політик перевизначення запитів на льоту, тобто запит автоматично модифікується залежно від самих різних параметрів (назва програми, час запуску на виконання, ім'я користувача, місце надходження запиту – Інтернет / локальна мережа, номер терміналу і т.д.).

Наступним аспектом, що визначає захищеність даних, є їх обсяг і застосовуються для зменшення обсягу механізми стиснення даних. По-перше, для різних даних повинні автоматично застосовуватися різні алгоритми стиснення і лише тоді, коли стискання дає вигоду. По-друге, адміністратор повинен мати можливість вибирати різні рівні стиснення, розуміючи, що за

економію місця доведеться платити процесорним часом. По-третє, стиснення має бути прозоре для програм. При цьому СУБД повинна працювати зі стиснутими даними так само ефективно, як і з не стиснутими даними, не витрачаючи час на перебудову таблиць, на розтискання непотрібних блоків.

Традиційні алгоритми стиснення дозволяють стиснути дані в 2-3 рази. Деякі специфічні види стиснення дозволяють збільшити цю ступінь стиснення на порядок, але при цьому сильно сповільнюється виконання операцій DML з цими даними. Наприклад, відомо, що стиснення за полями збережених і заздалегідь відсортованих даних забезпечить дуже високий рівень стиснення (іноді в десятки разів). Проте спроби зміни таких таблиць в Oracle 11.2 показали, що час оновлення даних значно збільшується. Але для сховищ даних та історичних даних такий підхід цілком прийнятний.

Адміністратор повинен мати можливість для різних даних, різних таблиць і навіть частин таблиць вибирати ті варіанти стискання та зберігання (за строками, по стовпцях), які найбільш підходять з точки зору бізнес-використання цих даних. Причому, оскільки для додатків спосіб зберігання і стиснення прозорий, DBA може періодично міняти способи зберігання і стиснення. Цей підхід буде дуже корисний користувачам великих БД (наприклад, SAP-користувачам), які сьогодні змушені постійно збільшувати свої багатотерабайтні системи зберігання. Ну, і розумний механізм стиснення дозволяє залишати найсвіжіші та найбільш активно використовувані дані деякий час не стиснутими. Пізніше вони стискаються автоматично у фоновому режимі.

Ще один перспективний напрям розвитку – ізоляція DBA від даних. Сьогодні адміністратор може бачити і змінювати всі дані в БД. Нові засоби захисту (такі як Oracle Data Vault option) дозволять DBA виконувати всі операції по адмініструванню БД, не дозволяючи йому бачити і змінювати дані. Також адміністратору можна буде обмежити набір дозволених до виконання операцій. Обмеження може бути прив'язане до імені DBA, часу, точці входу в систему.

1.3.2. Проблеми, викликані модифікацією структур даних

Як показано на рисунку 1.2 структура баз даних може істотно відрізнятись. Це значить, що перехід з однієї структури на іншу не може бути здійснений шляхом заміни операційної системи або сервера СУБД – такий перехід потребуватиме повного перепроєктування бази даних та переробки алгоритмів доступу до даних в програмному забезпеченні додатків.

Отже, фактично, перехід на нову структуру бази даних вимагає повної переробки системи. Причому, витрати на таку переробку можуть бути навіть більшими, ніж на первинну розробку, оскільки ця переробка має здійснюватись в умовах, коли в первинній системі вже були накопичені дані, а їх перенесення і забезпечення повноти та внутрішньої узгодженості може стати складною і відповідальною задачею.

1.3.3. Проблеми синхронізації даних в корпоративних сховищах даних

Проблеми синхронізації даних розглянемо на прикладі мережі біт-торент, яка покликана забезпечувати таку синхронізацію.

BitTorrent Sync синхронізує файли використовуючи однорангову мережу P2P, цей протокол може ефективно використовуватися для передачі великих файлів між безліччю пристроїв, цей принцип передачі даних використовується в таких додатках як Torrent, Transmission і BitTorrent. BitTorrent Sync відрізняється від хмарних сервісів тим, що користувацькі дані знаходяться на локальному носії, а синхронізація вимагає як мінімум одного пристрою підключеного до мережі.

BitTorrent Sync використовує шифрування даних за допомогою алгоритму AES з довжиною ключа в 128 біт, який може бути як створеним випадково так і вибраним користувачем. Ключ є похідною від секрету, який може бути поширений серед користувачів, які обмінюються даними. Дані надсилаються безпосередньо від пристрою до пристрою, якщо приймаючий пристрій

недоступний (наприклад, перебуває за фаєрволом або виключено), то дані будуть спрямовані іншому вузлу мережі. Але дані будуть прийняті вузлом тільки в тому випадку, якщо йому відомий секретний ключ для розшифровки і перегляду даних. Вузол мережі може бути доданий до синхронізації, якщо йому буде передано відповідний секретний ключ, що дозволяє йому переглядати (RO-доступ) або змінювати (RW-доступ) дані.

Синхронізація робочих файлів між пристроями – найбільш очевидний спосіб використання, можна просто синхронізувати дані між декількома ПК або серверами, забезпечуючи наявність на кожному з пристроїв ідентичною інформації. Даний спосіб корисний в організації зберігання особистих і корпоративних даних.

Резервне копіювання – BitTorrent Sync може використовуватися в якості транспортного механізму між пристроєм, дані з якого вимагають резервування, і місця зберігання резервних копій. По суті, на резервується пристрої можна створити дзеркала потрібних директорій і реалізувати на їх базі інкрементне резервне копіювання або версійність.

Віддзеркалення і реплікація не самих критичних даних в хмарних або кластерних інтернет-додатках забезпечує зберігання ідентичного набору даних на декількох серверах, що дозволяє розподілити між ними навантаження. При цьому треба враховувати, що синхронізація потребує часу та обчислювальних ресурсів, тому в реальній роботі не варто спиратися на те, що всі вузли володіють актуальним і ідентичним набором даних. Також варто врахувати те, що репліки не усувають конфлікти інтелектуально – якщо в наборі даних є файли з ідентичними ім'ям і шляхом, то буде збережена тільки найбільш пізня версія. Таким чином, якщо файли створюються програмним забезпеченням відразу на декількох вузлах (master-master реплікація), то використовують механізми іменування, які дозволяють уникнути конфліктів імен.

1.3.4. Проблеми, що визначаються предметною областю проектів

В якості прикладу, розглянемо декілька предметних областей застосування корпоративних інформаційних систем в інтернет-проектах: область освітніх технологій, SOHO.

Серед інтернет-проектів в області освітніх технологій розглянемо проекти, що базуються на системі MOODLE. MOODLE дозволяє викладачам в простий спосіб організувати подання навчальних матеріалів для студентів. Ці матеріали можуть бути подані у вигляді окремих файлів, таких як текстові документи, презентації зображення і т.п. Для розміщення файлів у системі MOODLE можна використовувати файлові сховища (репозитарії).

Використовуючи стандартні сховища системи MOODLE викладачі можуть завантажувати до курсу розроблені навчальні матеріали які можуть зберігатися на персональному комп'ютері або на хмаро-орієнтованих сховищах, наприклад Google Drive, Microsoft OneDrive, Dropbox, Youtube і т.п.

Використання хмаро-орієнтованих сховищ, зокрема Google Drive, Microsoft OneDrive, Dropbox, дозволяє викладачам зберігати файли поза системою MOODLE, які потім можна додавати до курсу використовуючи стандартні сховища або вбудовувати вміст цих файлів в деякі ресурси та види діяльності курсу.

За допомогою служби Google Диск користувачі можуть створювати документи, таблиці, форми, презентації та малюнки, співпрацюючи з іншими користувачами в реальному часі безпосередньо у вікні web-браузера. Всі ресурси (документи, таблиці, форми, презентації та малюнки) розміщені в службі Google Диск можна опублікувати в системі MOODLE використовуючи різні види діяльності та ресурси, які можна додавати в системі. Також в системі MOODLE можна використовувати репозиторії, які може налаштувати адміністратор з метою вбудовування в систему контенту, що зберігається поза нею, зокрема даних (файлів), які розміщені на Google Диску.

Використовуючи службу OneDrive користувачі, аналогічно до роботи з Google Диск, можуть створювати документи, таблиці, форми, презентації,

співпрацюючи з іншими користувачами в реальному часі безпосередньо у вікні web-браузера. Викладачі можуть надавати спільний доступ до документів для користувачів системи MOODLE використовуючи різні види діяльності та ресурси, які розміщені на OneDrive.

Крім стандартних сховищ у системі MOODLE можна використовувати сховища, які можуть бути додані адміністратором сайту. Одним із таких сховищ може бути Open Source Physics. Використання цього сховища дозволяє викладачам додавати до курсу розроблені Java-аплети, як правило з фізики. Використання Java-аплету забезпечує можливість використовувати на інтернет-сторінках, у нашому випадку у курсі, «виконуваний вміст» шляхом виконання в гіпертекстовому документі міні-програм. Особливістю є формат готового до виконання коду – апаратно-незалежний формат, за допомогою якого транслюються Java-аплети, тобто програма виконується у вікні браузера і не залежить від операційної системи. Для того, щоб ці аплети відображалися у курсі необхідно адміністратору сайту додати модуль діяльності EJSApp (Easy Java Simulations Applets), а також на комп'ютері користувача має бути встановлена Java.

В SOHO-проектах важливим з точки зору архітектури та організації системи також є вибір інструментального забезпечення інтеграції підсистем автоматизованої системи в єдину інформаційну систему. У відносно невеликих проектах така інтеграція, як правило, забезпечується безпосередньо на рівні програмного коду системи таким чином, що зв'язки між підсистемами визначаються в результаті безпосереднього звернення до керуючих об'єктам та об'єктам даних однієї підсистеми і викликом їх функцій обробки з інших підсистем. Наявність таких зв'язків спрощує розробку системи на початкових етапах реалізації, але, на жаль, не дозволяє простими засобами нарощувати систему надалі – у завершальних стадіях її розробки і на етапі експлуатації, коли будь-яка модифікація системи на рівні одного модуля неминуче призводить до зміни встановлених зв'язків між цим модулем та іншими

модулями.

Для подолання цієї проблеми в складних системах реалізують більш гнучкі технології взаємодії підсистем шляхом реалізації керованих механізмів створення зв'язків між модулями, у тому числі – користувачем системи на етапі її експлуатації. Так, в системах, подібних MS Office або 1С, і в більшості інших складних систем реалізовані і надаються користувачам цих систем механізми, що дозволяють встановлювати нові правила ведення розрахунків і змінювати структуру бази даних шляхом простого створення нових зв'язків між модулями, модифікації існуючих або створення нових таблиць даних і т.д.. Реалізація таких механізмів вимагає забезпечення специфічних технологій доступу до даних об'єктів, а в окремих випадках – ще й використання спеціалізованих мов програмування з підтримкою механізмів метапрограмування.

В проектах надання масового доступу до файлів. Реалізація і використання розподілених систем мають не лише плюси, але і мінуси, що пов'язані з особливостями забезпечення безпеки. Отримати контроль над розгалуженою і великою мережею P2P, або використати дірки в реалізації протоколів для власних потреб є бажаною метою для хакерів. До того ж захистити розподілену структуру складніше, ніж централізований сервер.

Величезну кількість ресурсів в мережах P2P, важко шифрувати/дешифрувати, тому більша частина інформації про IP-адреси і ресурси учасників зберігається і пересилається в незашифрованому вигляді, що робить її доступною для перехоплення. При перехопленні зловмисник не лише отримує власне інформацію, але і дізнається про вузли, на яких вона зберігається.

В клієнтах більшості крупних мереж ця проблема вирішується шляхом шифрування заголовків пакетів і ідентифікаційної інформації. З'являються клієнти з підтримкою технології SSL, впроваджуються спеціальні засоби захисту інформації про місце знаходження ресурсів.

Серйозною проблемою є поширення вірусів і підробка ID ресурсів з

метою їх фальсифікації. Наприклад, в клієнті Kazaa використовується хеш-функція UUHash, яка дозволяє швидко знаходити ID для великих файлів навіть на малопотужних комп'ютерах, але при цьому залишається можливість для підробки файлів і запису зіпсованого файлу, що має такий же ID.

Щоб подолати таку проблему, клієнти повинні користуватися надійними хеш-функціями («деревами» хеш-функцій, якщо файл копіюється по частинах), такими, як SHA-1, Whirlpool, Tiger, і лише для вирішення некритичних завдань – контрольними сумами CRC. Для зменшення переданих об'ємів даних і полегшення їх шифрування можна застосувати компресію. Для захисту від вірусів потрібно мати можливість зберігати ідентифікаційну метаінформацію про них, як це, наприклад, зроблено в мережі Gnutella2.

Іншою проблемою є можливість підробки ID серверів і вузлів. За відсутності механізму перевірки достовірності переданих службових повідомлень, наприклад за допомогою сертифікатів, існує можливість фальсифікації сервера або вузла (багатьох вузлів). Оскільки вузли обмінюються інформацією, підробка деяких з них приведе до компрометації всієї мережі або її частини. Закрите ПЗ клієнтів і серверів не є вирішенням проблеми, оскільки є можливість для реінжинірингу протоколів і програм (reverse engineering).

Виділені сервери і вузли періодично обмінюються між собою службовою інформацією і за необхідності додають підроблені сервери/вузли до чорного списку для блокування доступу.

Ще однією проблемою є нечесна поведінка частини клієнтів, які лише копіюють чужі файли, але нічого не пропонують іншим для копіювання (leechers).

Для боротьби з цим використовуються різні методи. В Bittorrent реалізовано схему «скільки блоків файлу отримав, стільки віддав». В eMule застосовано метод кредитів: скопіював файл – кредит зменшився, дозволив скопіювати свій файл – кредит збільшився (xMule – кредитна система із заохоченням поширювання рідкісних файлів).

1.3.5. Проблеми, що визначаються інструментальним забезпеченням

За окремими виключеннями будь який проект може з початкового рівня перерости в значний за обсягом. Таким чином, з точки зору предметної області проекти вимоги до проектів не можуть бути досить просто визначені відповідно до особливостей предметної області, але ці особливості напряду впливають на вибір інструментального забезпечення, що використовується при реалізації проекту. Отже, при цьому виникають істотні проблеми, пов'язані з специфікою стосовно предметної області та рівнем стосовно операційної системи задач, що мають бути вирішені в проекті.

Для написання програм на низькому рівні абстракції, докладного і детального опису операцій, що враховують архітектура комп'ютера та операційної системи використовують низькорівневі мови програмування, такі як асемблер та С. У веб програмуванні асемблер не використовується, а мова С звичайно застосовується для розробки самого інструментального забезпечення, а не користувацьких додатків. В порівнянні з більш високорівневими мовами програмування програми мовою С звичайно вимагають більших витрат часу та високого рівня програмістів, отже збільшується і ймовірність появи помилок. Це можна вважати недоліком програмування на мовах низького рівня.

Мови програмування високого рівня дозволяють подолати недоліки низькорівневого програмування – використовувати більш розвинуті операції обробки агрегатних даних (списки, вектори, масиви, черги, стеки, сервіси операційної системи, комплексні числа тощо), не зважаючи на деталі їх реалізації на конкретній архітектурі. Програми при цьому виявляються більш короткими, надійними і універсальними, а процес їх складання скорочується. Програми на мовах високого рівня також легше читати та супроводжувати, в них простіше розібратися. Мови високого рівня бувають чотирьох видів: імперативні (процедурні), функціональні, логічні, об'єктно-орієнтовані. Основними об'єктами в імперативних мовах є змінні, оператори присвоювання,

стандартні алгоритмічні конструкції.

Об'єктно-орієнтовані мови спрощують програмування з використанням технології об'єктно-орієнтованого програмування (Python, Java тощо), що, з точки зору можливого документально-орієнтованого представлення інформації, є корисним.

Враховуючи вищезазначене визначимо, що однією з основних проблем вибору інструментального забезпечення для реалізації активних сховищ корпоративних інформаційних систем слід вважати специфіку задач кожного конкретного проекту з точки зору вимог особливостей методів обробки даних, що визначаються бізнес-процесами та реалізуються інструментальним засобами розробки.

1.3.6. Проблеми обсягу даних корпоративних інформаційних систем

При зростанні обсягу БД вузьким місцем стають канали передачі даних між сервером БД і пристроями зберігання. Можливість збільшити кількість і пропускну здатність таких каналів, а також знизити обсяг переданих даних за рахунок виконання частини операцій з перегляду та вибірці даних на рівні осередків зберігання, дозволяє зняти проблему каналів передачі. Така зв'язка обладнання та спеціального ПЗ, звана машина баз даних, давно описана в літературі. Приклади – Netezza, Datallegro, Teradata.

1.4. Постановка задачі дослідження

На основі зазначених чотирьох критеріїв та прикладів їх застосування для класифікації проектів можна зробити висновок, що будь які проекти можуть бути класифіковані в системі з чотирьох критеріїв. Це значить, що за наявності двійкової оцінки за кожним з критерієм існує всього 16 типів проектів

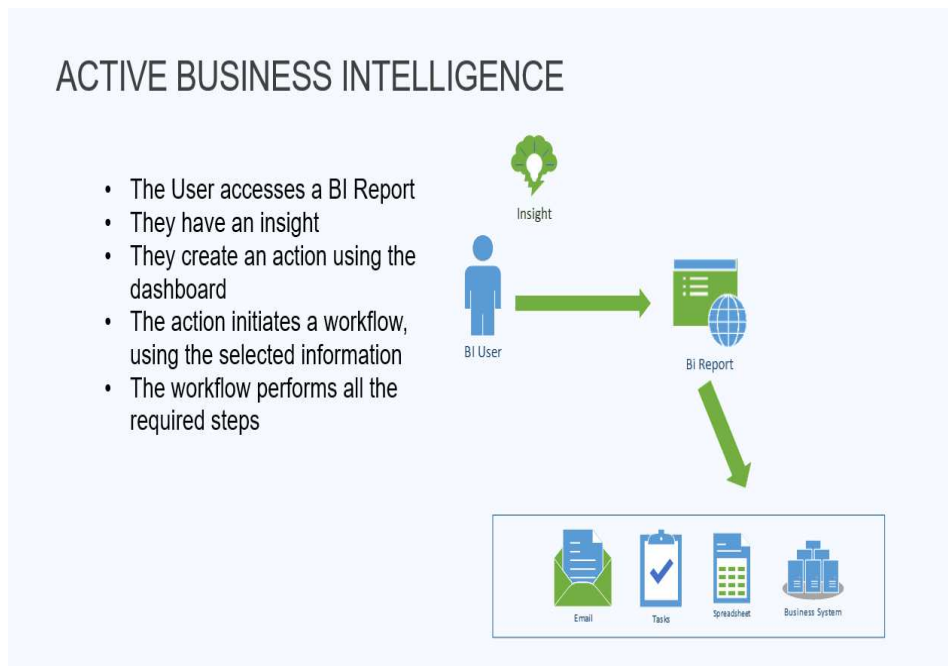


Рис. 1.9. Модель доступу до даних активного сховища даних АІС компанії

Виходячи з результатів аналізу виконаємо постановку задачі дослідження у наступному формулюванні: визначити можливості науково-обґрунтованого підходу до вибору технологій організації і доступу до даних, та можливостей їх застосування в проектах аналізу даних корпоративних інформаційних систем, що дозволять з наявного спектру інструментального забезпечення, методів та засобів збереження, транспортування, захисту та обробки даних виконувати розробку корпоративних інформаційних систем та, зокрема, інтернет-проектів з найкращими результатами.

1.5. Висновки до розділу 1

В розділі були розглянуті технології організації і доступу до даних у корпоративних інформаційних системах та, зокрема, інтернет-проектах, що визначаються інструментальним забезпеченням, методами та засобами збереження, транспортування, захисту та обробки даних корпоративних інформаційних систем та інтернет-проектів.

Визначено, що широке представлення та популярність технологій

збереження даних у файлових сховищах та інтеграція корпоративних систем у загальний інформаційний простір вказують на те, що при збереженні значних обсягів інформації корпоративні системи збереження даних можуть використовувати такі сховища та надавані ними інструменти доступу до даних і аналізу даних як одну з технологій збереження, доступу і обробки даних.

Виконано постановку задачі дослідження.

РОЗДІЛ 2. РОЗРОБКА МОДЕЛЕЙ ТА МЕХАНІЗМІВ АКТИВНИХ СХОВИЩ ДАНИХ КОРПОРАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

2.1. Визначення критеріїв оцінки методів та засобів розробки проектів сховищ даних корпоративних інформаційних систем

2.1.1. Визначення критеріїв оцінки методів та засобів збереження і аналізу даних в проектах корпоративних інформаційних систем

Враховуючи проблеми, пов'язані з бізнес-логікою і обробкою і аналізом даних, обсягом даних та обсягом запитів до СУБД, активні сховища даних можуть розглядатись як навантажені СУБД. Такі СУБД можуть розглядатись як:

- системи з великими обсягами даних;
- системи зі складною організацією даних;
- системи з великим обсягом запитів (як правило — зі значним числом користувачів).

Візуально різновиди навантажених СУБД можуть бути представлені у вигляді моделі (рис 2.1).

На основі результатів аналізу сучасних систем управління базами даних, їх переваг та недоліків можна визначити декілька критеріїв щодо можливостей застосування СУБД кожного з типів для вирішення задач збереження даних корпоративних інформаційних систем. Ці критерії наступні:

- Відповідності - сервіси та протоколи за своїми функціональними можливостями повинні відповідати призначенню проекту, наприклад, в комерційних проектах повинні застосовуватися захищені канали передачі даних, сертифікація, стиснення даних тощо;
- Критерій розвитку активного сховища даних (в даному випадку стабільні краще);
- Критерій підтримки СУБД інструментальними засобами (бажана підтримка сучасних мов програмування);
- Критерій вартості входження та супроводу активного сховища даних (прості і безкоштовні краще у зв'язку з коротким часом життя проектів);

- Критерій підтримки інструментів з сучасними технологіями програмування (кращі ті інструменти, що відносно простими засобами дозволяють реалізувати більш складну бізнес логіку, наприклад з підтримкою технологій узагальненого або мета- програмування);
- Критерій простоти реорганізації модифікації структури бази даних (так, щоб кожен окремий запит звертався тільки до тих таблиць, які містять дані, що відносяться до цього запиту. Як наслідок великі таблиці активних сховищ даних повинні бути розбиті на кілька, може однотипних, але малих).

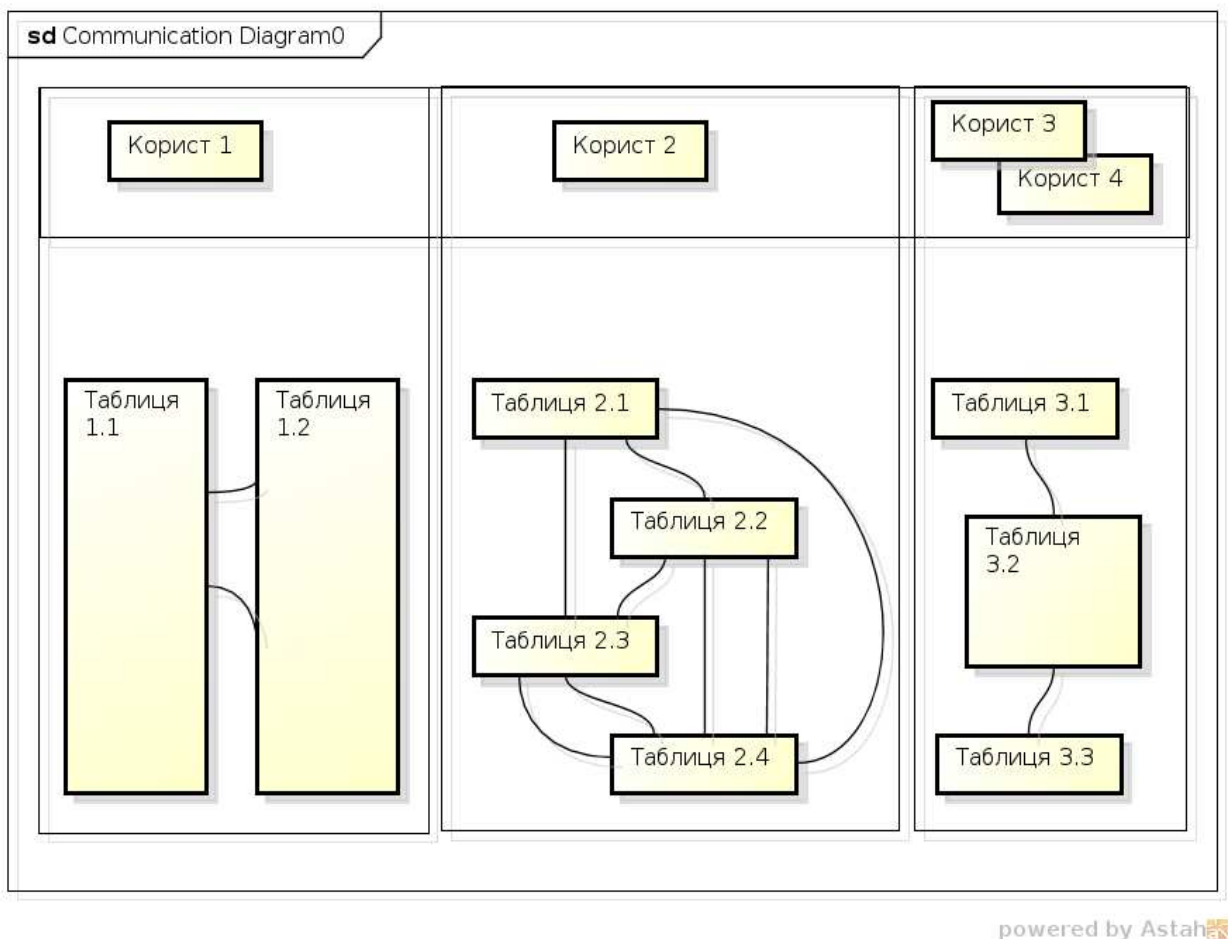


Рис. 2.1. Різновиди навантажених СУБД: з великим обсягом даних (ліворуч), зі складною структурою (у центрі), зі значним числом користувачів (праворуч)

2.1.2. Визначення критеріїв оцінки методів та засобів розробки проектів активних сховищ даних корпоративних інформаційних систем

Враховуючи, що однією з основних проблем вибору інструментального забезпечення для реалізації проектів сховищ даних корпоративних інформаційних систем слід вважати специфіку задач кожного конкретного проекту з точки зору вимог особливостей методів обробки даних, що визначаються бізнес-процесами і використовують дані безпосередньо з запитів (транзакційний пакет даних, рисунок 2.1).

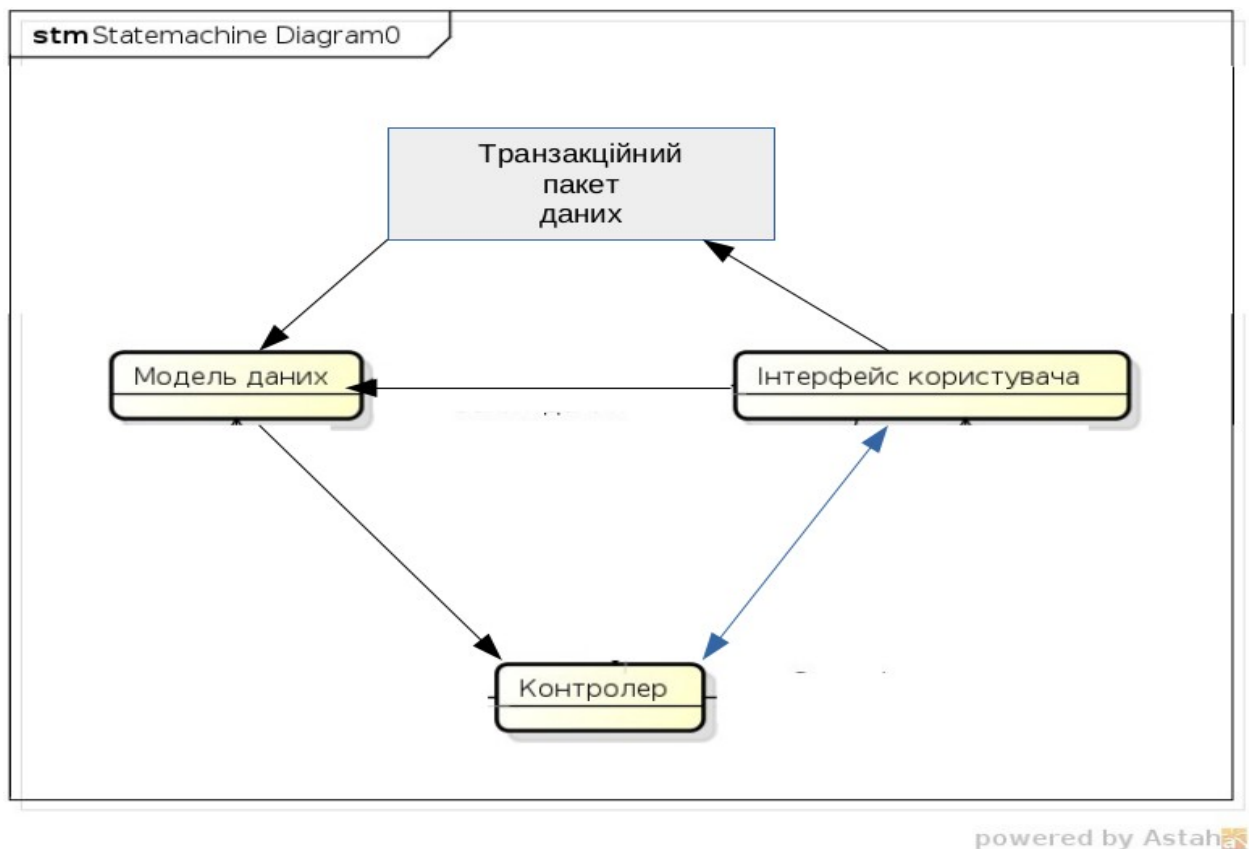


Рис. 2.1. Базова MVC-модель додатка з пасивним сховищем даних

Зважаючи на вищенаведене, визначимо наступні ключові критерії щодо застосування інструментальних засобів активних сховищ даних в корпоративних і інтернет-проектах (Таблиця 1.3).

Активне сховище даних обумовлює високе навантаження на сервер баз даних в частині отримання великих масивів даних з безперервним поповненням бази даних, що визначає високі вимоги до потужності і надійності СКБД.

Активне сховище даних обумовлює високе навантаження на сервер додатків в частині безперервної обробки великих масивів даних для вирішення задач бізнес-аналітики.

Обробка даних повинна відбуватись в режимі ізоляції транзакцій для забезпечення відповідності результатів аналізу даних поточному запиту і наявним даним у базах даних.

Залежно від вимог щодо забезпечення кожної з цих областей наявні

корпоративні та інтернет-проекти можуть бути поділені на категорії з різними вимогами. Нижче наведені приклади (таблиця 2.1).

Таблиця 2.1. Ключові критерії щодо застосування інструментальних засобів в корпоративних та інтернет-проектах

№	Критерій	Примітка
1	Збереження та надання інформаційних ресурсів	
2	Комунікації та обмін інформацією	
3	Масові обчислення	
4	Управління соціальними та технічними системами	

Наведений перелік звичайно не є повним, але він дає можливість виконати оцінку адекватності зазначених критеріїв стосовно предметних областей зберігання і аналізу даних. Як наслідок, для забезпечення цих критеріїв можуть застосовуватись різні інструментальні засоби, кожен з яких тим чи іншим чином відповідає вимогам щодо критеріїв.

Очевидно, що для забезпечення відповідності властивостей систем зазначеним вище критеріям якості інформаційних систем знадобляться і відповідні інструментальні засоби, методи проектування, механізми збереження, обробки та передачі даних.

2.2. Розробка моделей оцінки якості механізмів збереження та доступу до активних сховищ даних

2.2.1. Критерії оцінки якості технологій збереження та доступу до активних сховищ даних корпоративних інформаційних систем

Враховуючи визначені в попередньому розділі (пп. 1.4) критерії, що визначають якість механізмів збереження та доступу до даних, та особливості

MVC-моделі додатків корпоративних систем, визначимо додаток з активним сховищем даних як такий, що використовує при обробці даних поточного запиту не лише дані, що зберігаються у сховищі даних, але і дані із самого запиту (рисунок 2.2).

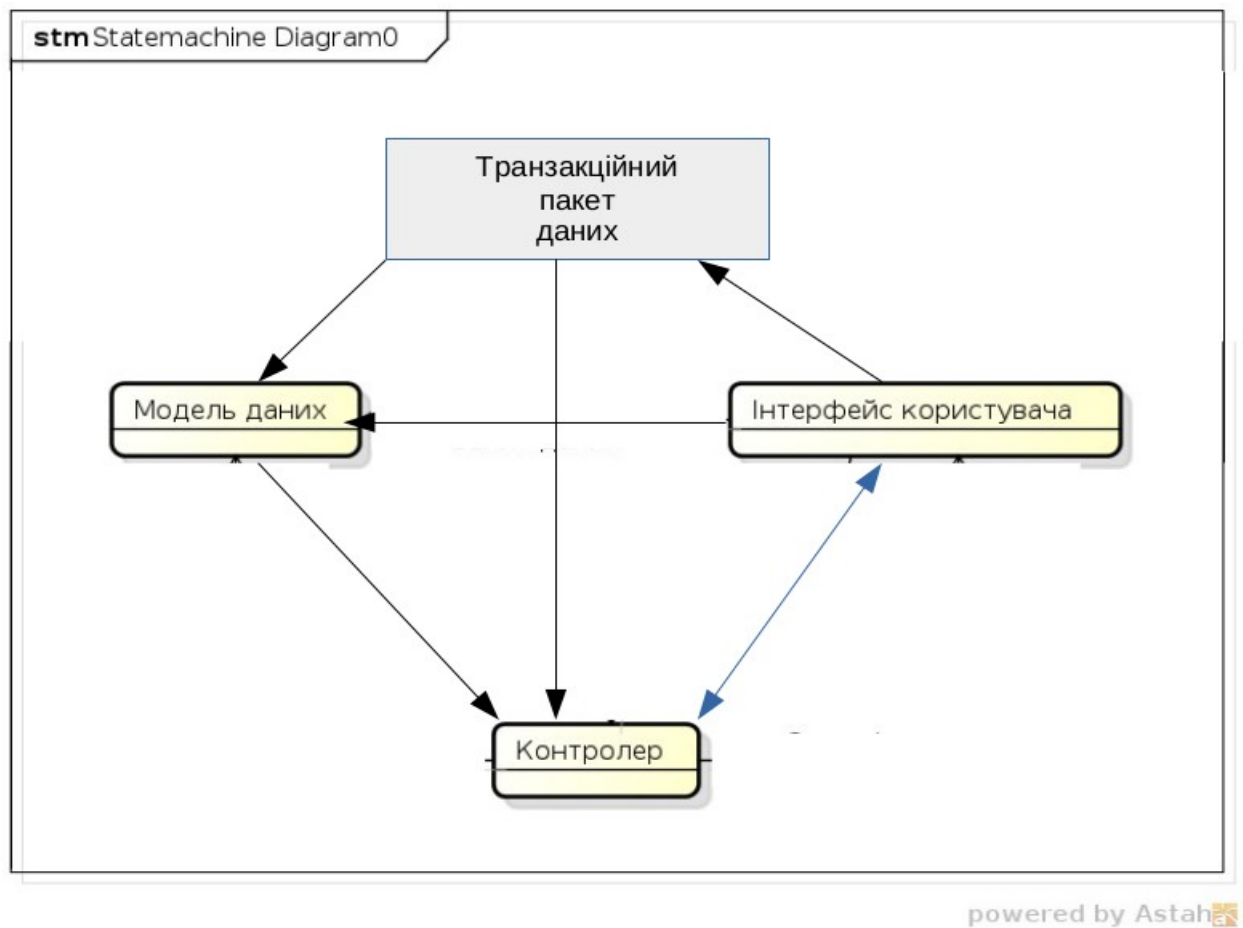


Рис. 2.2. Базова MVC-модель додатка з активним сховищем даних

На основі зазначеного визначимо підходи до розробки моделей оцінки якості механізмів збереження та доступу до даних.

Таблиця 2.2. Приклади ранжування проектів корпоративних систем за категоріями вимог

№	Предметна область	Значення для критерію			
		Збереження та надання ресурсів	Комунікації, захист та обмін інформацією	Масові обчислення	Управління іншими системами
1	Електронні ЗМІ	високе	низьке	низьке	низьке

2	Електронна торгівля	високе	середнє	низьке	низьке
3	Навчання	середнє	середнє	середнє	середнє
4	Інженерні розрахунки	середнє	середнє	високе	низьке
5	Соціальні мережі	низьке	високе	низьке	низьке
6	Електронний уряд	високе	середнє	низьке	високе
7	Системи моніторингу та управління	високе	високе	високе	високе

Дано розгорнуте трактування різновидів навантажених СУБД корпоративних інформаційних систем (рисунки 2.3) та окремих із зазначених критеріїв якості проектів активних сховищ даних.

Критерій відповідності полягає в тому, що сервіси та протоколи за своїми функціональними можливостями повинні відповідати призначенню корпоративного або інтернет-проекту, наприклад, в комерційних проектах повинні застосовуватися захищені канали передачі даних, сертифікація, стиснення даних тощо.

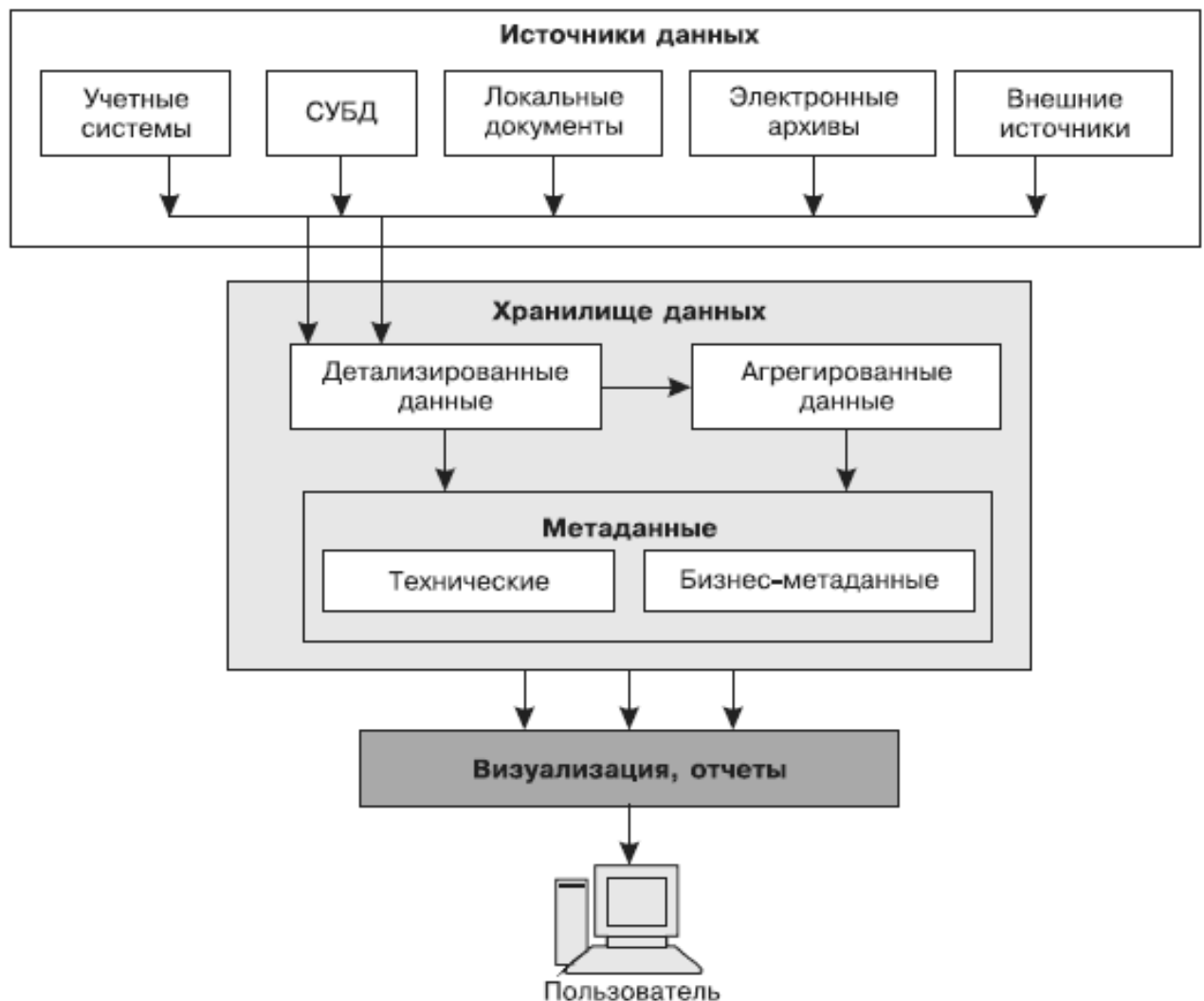


Рис. 2.3. Разновиды навантажених СУБД корпоративних інформаційних систем

Критерій підтримки проекту інструментальними засобами полягає в тому, що окрім поточної розробки будь яку систему необхідно підтримувати і надалі. При цьому непопулярність інструментальних засобів в майбутньому може призвести до проблем з пошуком та залученням відповідних фахівців. Як наслідок, в проекті бажана орієнтація на сучасні мови веб-програмування і опису даних.

Критерій простоти і вартості СУБД полягає в тому, що прості і безкоштовні СУБД є кращим вибором з погляду на відносно короткий термін життя окремих інтернет-проектів.

Критерій простоти реорганізації бази даних полягає в тому, щоб кожен окремий запит звертався тільки до тих таблиць, які містять дані, що відносяться до цього запиту. Як наслідок великі таблиці повинні бути розбиті на кілька, може однотипних, але маленьких. Приклад: інтернет форум, в якому повідомлення форуму краще зберігати не в одній великій таблиці по всіх темах форуму, а в декількох невеликих таблицях, але містять повідомлення тільки однієї теми.

Критерій простоти модифікації структури БД визначається тим, що у зв'язку з високим навантаженням і коротким часом життя інтернет проектів структура баз даних часто зазнає змін через відносно спрощені технології проектування ПЗ корпоративних та інтернет-проектів.

Визначимо, що оцінка якості проектів має враховувати два основні аспекти:

- якість використання програмної системи, тобто система має задовольняти експлуатаційним вимогам;
- якість розробки програмної системи, тобто система має задовольняти технологічним вимогам.

Виходячи з того, що система, яка не може використовуватись, має нульову якість так само як і система, що не може бути реалізована, визначаємо, що ці аспекти можуть бути враховані в загальній оцінці якості як середнє геометричне за кожним з аспектів:

$$c = \sqrt{c_u * c_d * c_a}, \quad (2.1)$$

де: c – оцінка, c_u – відповідність вимогам бізнес аналітики; c_d – відповідність вимогам збереження; c_a – відповідність технологіям.

В свою чергу, відповідність вимогам бізнес аналітики визначається відповідністю механізмів доступу до даних та механізмів збереження даних:

$$c_u = 1/n \sum_{i=1}^n f(m_i * t_i, m_i * s_i), \quad (2.2)$$

де $f(t_i, s_i)$ – функція обчислення критерію якості окремої транзакції залежно від обсягу даних і часу обробки.

$$c_d = 1/n \sum_{i=1}^n g(m_i * t_i, m_i * s_i), \quad (2.3)$$

де $g(t_i, s_i)$ – функція обчислення критерію якості окремої транзакції залежно від обсягу даних і часу доступу до даних.

$$\sum_{i=1}^n m_i = 1.0,$$

де m_i – відносна ймовірність припустимого варіанта конфігурації даних СКБД і транзакційного пакету.

Зважаючи на результати аналізу, визначаємо, що відповідність вимогам розробника забезпечується відповідністю механізмів збереження та доступу до даних залежно від предметної області задачі, а відповідність вимогам розробника – відповідністю інструментальних засобів (в тому числі мови програмування) щодо можливостей забезпечення збереження та доступу до даних.

2.2.2. Механізми забезпечення якості активних сховищ даних корпоративних інформаційних систем

Відповідно до визначених вище критеріїв якості, згідно до таблиці 2.1 та (2.2) визначимо значення критерію c_s як сумму:

$$c_a = \frac{k_e * c_e + k_q * c_q + k_v * c_v + k_w * c_w + k_r * c_r}{k_e + k_q + k_v + k_w + k_r}, \quad (2.4)$$

де: e - захищеність; q – швидкість розробки; v – наочність коду; w – швидкість виконання базових операцій; r – простота тестування.

Коефіцієнти k, c будуть визначатись на основі результатів експертного оцінювання.

Серед основних механізмів забезпечення якості в активних системах збереження даних найбільш цікавим, з точки зору корпоративних технологій, є

масштабування баз даних – партиціонування, реплікація і шардинг.

СУБД – це дуже часто «вузьке місце» в продуктивності інтернет-додатків. У момент, коли сервер баз даних не може впоратися з навантаженнями, проводиться масштабування. Розглянемо основні способи збільшення продуктивності СУБД.

Розглянуті механізми масштабування застосовні як для реляційних баз даних, так і для NoSQL-сховищ. Зрозуміло, що у всіх баз даних і сховищ є своя специфіка, тому ми розглянемо тільки основні напрямки і в деталі реалізації вдаватися не будемо.

Партиціонування (partitioning) – це розбиття таблиць, що містять велику кількість записів, на логічні частини за деякими обраним адміністратором критеріям. Партиціонування таблиць ділить весь обсяг операцій з обробки даних на кілька незалежних і паралельно виконуються потоків, що істотно прискорює роботу СУБД. Для правильного конфігурування параметрів партиціонування необхідно, щоб в кожному потоці було приблизно однакову кількість записів.

Наприклад, на новинних сайтах має сенс партиціонувати записи за датою публікації, так як свіжі новини на кілька порядків більш витребувані і частіше потрібна робота саме з ними, а не з усіх архівом за роки існування новинного ресурсу.

Реплікація (replication) – це синхронне або асинхронне копіювання даних між декількома серверами. Провідні сервера називають майстрами (master), а ведені сервера – слейвами (slave). Майстри використовуються для зміни даних, а слейви – для зчитування. У класичній схемі реплікації зазвичай жоден майстер і кілька слейвів, оскільки в більшій частині корпоративних та інтернет-проектів операцій читання на кілька порядків більше, ніж операцій запису. Однак у більш складній схемі реплікації може бути і кілька майстрів.

Наприклад, створення декількох додаткових slave-серверів дозволяє зняти з основного сервера навантаження і підвищити загальну продуктивність

системи, а також можна організувати слейви під конкретні ресурсомісткі завдання і таким чином спростити складання серйозних аналітичних звітів – використовуваний для цих цілей slave може бути навантажений на 100%, але на роботу інших користувачів додатку це не вплине.

Шардінг (sharding) – це прийом, який дозволяє розподіляти дані між різними фізичними серверами. Процес шардінга припускає рознесення даних між окремими вузлами на основі якогось ключа шардінга. Пов'язані однаковим значенням ключа шардінга сутності групуються в набір даних по заданому ключу, а цей набір зберігається в межах одного фізичного вузла. Це істотно полегшує обробку даних. Наприклад, в системах типу соціальних мереж ключем для шардінга може бути ID користувача, таким чином всі дані користувача будуть зберігатися та оброблятися на одному сервері, а не збиратися по частинах з декількох джерел.

Крім цього, істотним з точки зору задоволення критеріям оцінки якості, є механізми зберігання та індексування даних в БД.

Серед популярних механізмів зберігання даних найбільш цікавим для реалізації вузькоспеціалізованих і високонавантажених серверів БД, до яких відносяться БД корпоративних та інтернет-проектів, представляються In-memory СУБД реального часу як кеш для комерційних СУБД.

Практично всі універсальні комерційні СУБД сьогодні – це СУБД, орієнтовані на роботу з дисками. Така архітектура, навіть при дуже гарній налаштування додатки, може дати час відгуку порядку декількох мілісекунд. Однак для багатьох додатків (наприклад, для систем, що працюють в реальному часі) потрібно швидкодію на порядок вище (час відгуку – мікросекунди). Для отримання такої швидкодії використовуються спеціальні СУБД, що працюють в оперативній пам'яті (In memory DB) і мають спеціальну архітектуру, спеціальні способи зберігання, адресації і індексації даних, спеціальні механізми оптимізації і буферизації і т.д. Дані на диск вони не скидають або роблять це у фоновому асинхронному режимі. Надійність роботи та обробка збоїв

реалізується спеціальними методами (наприклад, за рахунок реплікації пам'ять – пам'ять). Багато хто не дуже важливі функції таких СУБД можна відключати (наприклад, журналювання) на догоду швидкодії.

2.2.3. Механізми забезпечення якості доступу до активних сховищ даних корпоративних інформаційних систем

Відповідно до визначених вище критеріїв якості, згідно до пункту 1.4.2 та (2.2) визначимо значення критерію c_a як сумму:

$$c_{us} = k_b * c_{ub} + k_p * c_{up} , \quad (2.5)$$

де: $k_b * c_{ub}$ – ваговий коефіцієнт та значення критерію безпеки передачі даних;
 $k_p * c_{up}$ – ваговий коефіцієнт та значення критерію доступності протоколів передачі даних.

З точки зору оцінки якості доступу до даних в складних корпоративних системах перевагу слід віддавати тим рішенням, які, насамперед, дозволяють реалізувати необхідну бізнес логіку (так як в іншому випадку вже не важливі такі характеристики, як швидкість, захищеність і т.д.). Виходячи зі складності таких систем, для їх розробки застосовують, як правило, об'єктно-орієнтовані технології програмування та проектування. При цьому сховища даних також повинні бути об'єктними. Враховуючи, що реляційні СУБД все ще популярні і широко використовуються рішенням задачі об'єктного доступу до даних в реляційному сховищі є ORM-системи.

Для забезпечення якості програміста ORM-система повинна виглядати як постійне сховище об'єктів. На практиці все ORM-системи, як правило, виявляють себе саме таким чином, забезпечуючи можливість ігнорування реляційної природи бази даних як набору таблиць і відносин між ними (рис. 2.4).

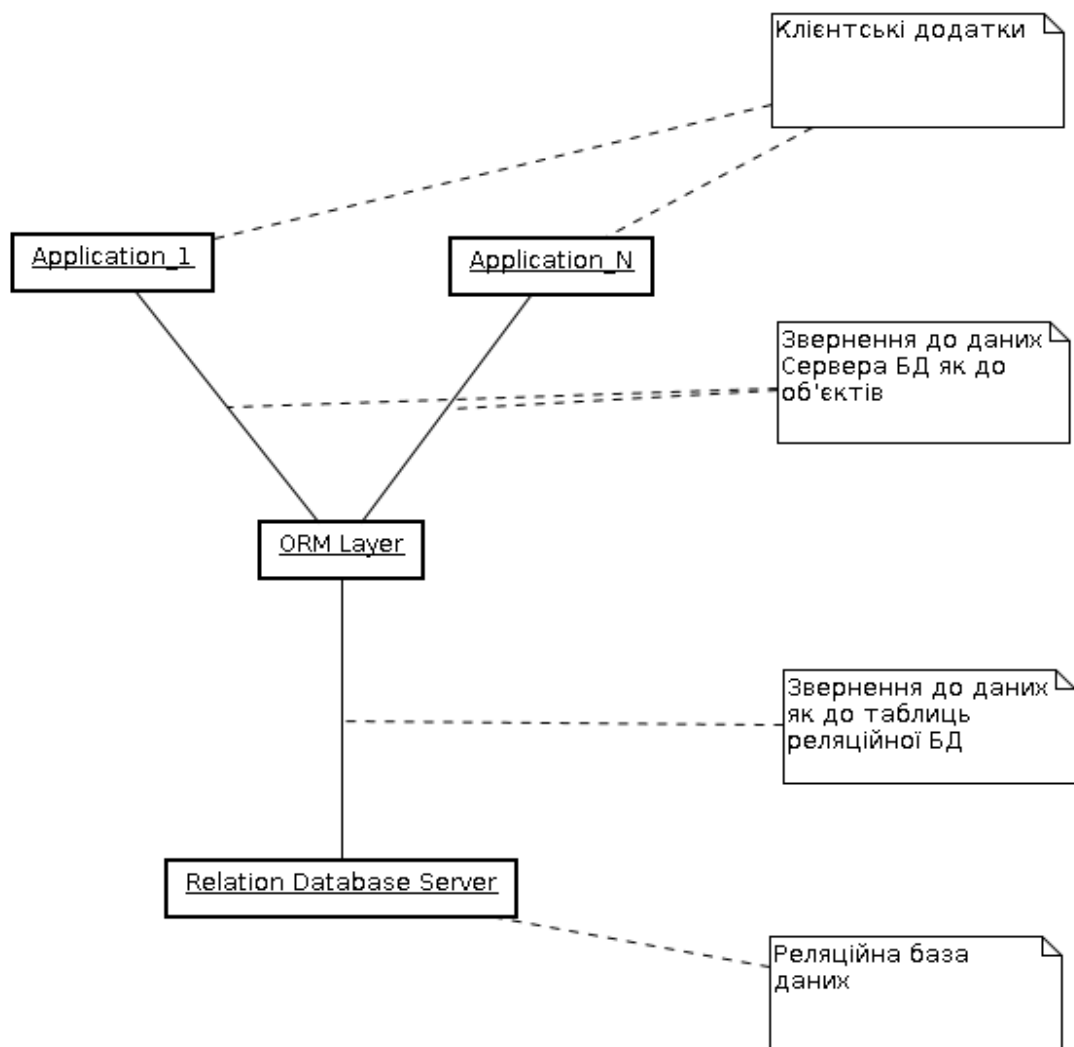


Рис. 2.4. Компонентна модель доступу до реляційної бази даних з Object Relational Mapping (ORM)

Крім цього, враховуючи, що програмне забезпечення для складних систем зазвичай розробляється відповідно до принципів об'єктно-орієнтованого програмування, необхідність використання ORM-систем так само виникає як результат невідповідності способів організації структурованих даних систем управління реляційними базами даних способам організації об'єктів як структурованих даних і механізмів їх обробки в програмах, які використовують такі бази даних (рис. 2.5, 2.6). Ця невідповідність полягає в тому, що об'єкти програми можуть перебувати у відносинах агрегації, асоціації та використання, а реляційні бази даних підтримують тільки відносини асоціації, причому,

відносини асоціації "багато-до-багатьох" більшістю реляційних баз даних взагалі не підтримується.

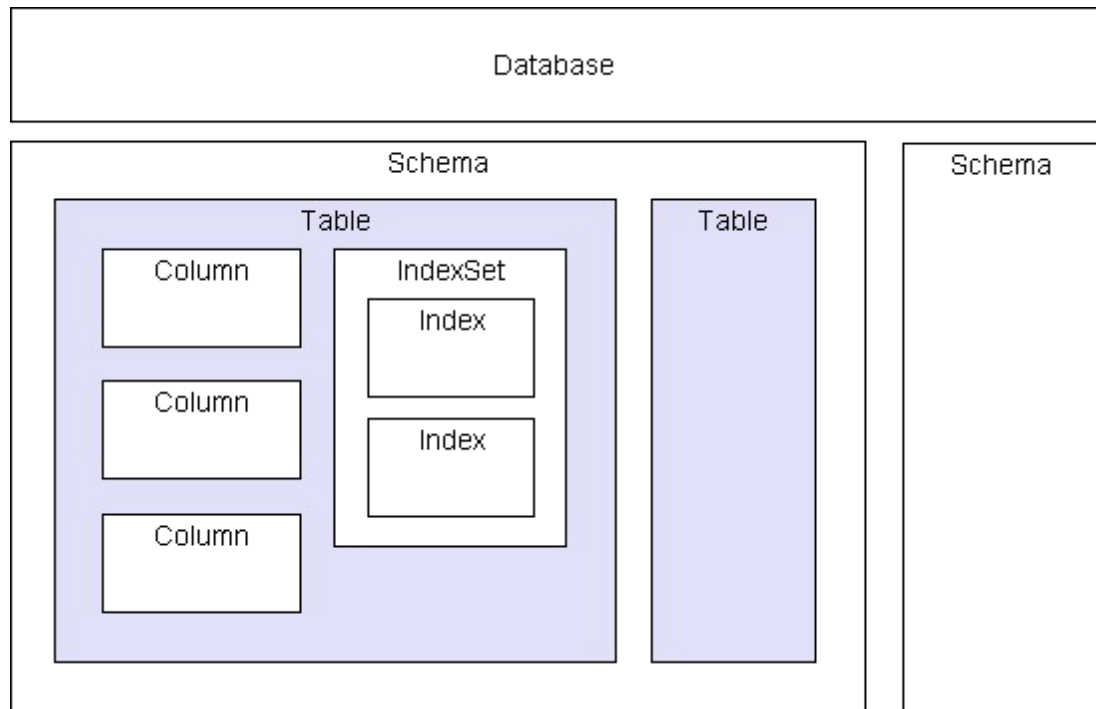


Рис. 2.5. Логічна структура даних в ORM-системі

Захист клієнта від можливості отримання недостовірних даних або втрати власних базуються на наступних механізмах:

- захисті каналів передачі даних та захисті від потрапляння на клієнтську машину недостовірних даних, що має місце при підміні сервера, коли за тією самою адресою працює інший, недостовірний сервер (який може видавати себе за достовірний шляхом надання очікуваного контенту) або в результаті автоматичного перенаправлення запиту;
- впровадження в сторінку шкідливого коду, який буде виконаний на комп'ютері користувача при відкритті ним цієї сторінки (XSS-атака).

Найбільш часто експлуатуються такі уразливості на сайтах, де користувачі можуть публікувати свій контент, роблячи його доступним для

інших. Зловмисник, при відсутності належної фільтрації, може або безпосередньо використовувати уразливість, або заражати системи користувачів. Розробники веб проєктів для запобігання атак цього типу змушені або просто обмежувати можливості користувачів публікувати на сайті що-небудь, або ретельно фільтрувати надходять від користувачів дані.

2.2.4. Механізми доступу та збереження даних клієнта активних сховищ даних корпоративних інформаційних систем

Браузерні або клієнтське кешування. Кешування статичних ресурсів (картинок, скриптів, стилів) і незмінних сторінок на стороні браузера може заощадити час завантаження сторінок, якщо користувач відвідує сайт багаторазово або при відвідуванні переглядає кілька сторінок, які використовують однакові ресурси (таблиця 2.3).

Таблиця 2.3. Типи та періодичність кешування ресурсів інтернет-сторінок на стороні клієнта

№	Типи ресурсів	Призначення	Період
1	Файли зображень	Медійна інформація	1 місяць
2	Скрипти JS	Анімація, обробка даних на стороні клієнта	1 місяць
3	CSS файли	Стилі представлення, оформлення	1 місяць
4	Статичні сторінки	Інформація, що змінюється розробником контенту	1 день – 1 тиждень
5	Бінарні файли	Інформація, що змінюється сервером за запитом	до 1 дня

Найчастіше на стороні браузера кешуються файли зображень, JS і CSS файли. Трохи рідше має сенс кешувати сторінки і бінарні файли (медіа-файли, PDF, завантажувані документи, архіви і т. д.).

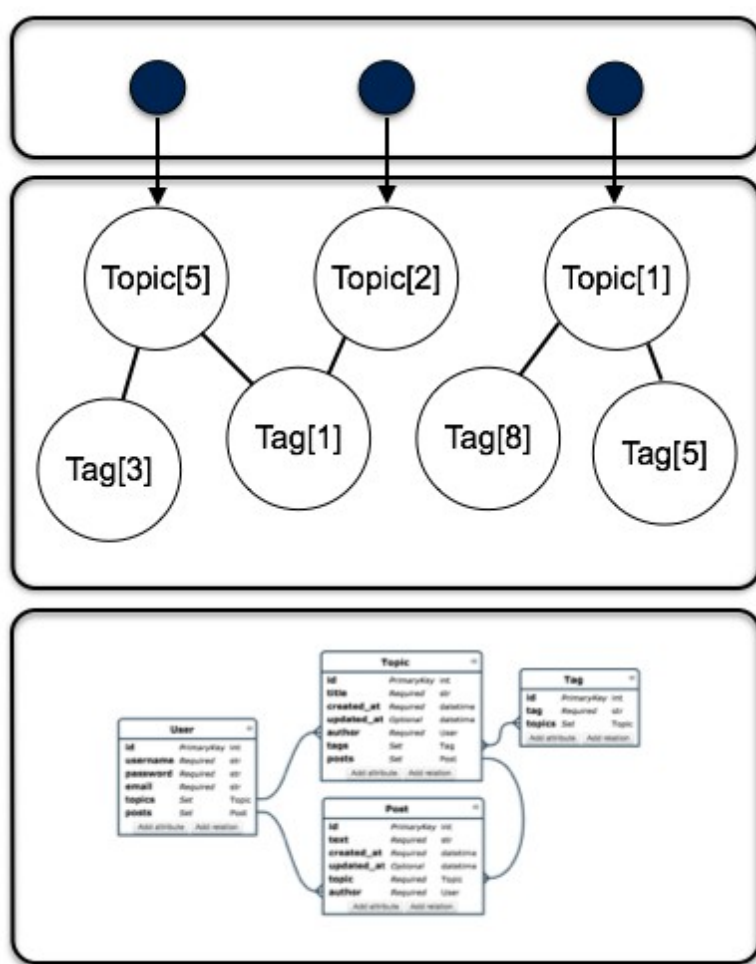


Рис. 2.6. Фізична структура даних в ORM-системі (зверху-донизу: рівень доступу; рівень організації, рівень збереження)

Статичні ресурси повинні мати хоча б тижневе час життя кешу, а краще кешувати їх відразу на місяць – з врахуванням сезонного характеру окремої інформації. Таким чином, ці ресурси будуть тільки один раз отримують з сервера, а потім браузер буде або відразу використовувати локальну копію (якщо вказано заголовок Expires або Cache-Control), або після перевірки на незмінність (якщо вказано заголовок Last-Modified або Etag).

Заголовки Expires і Cache-Control: max-age. Як значення у цих заголовків використовується дата. До тих пір, поки вона не настала, браузер буде без будь-яких додаткових перевірок використовувати закешовану версію ресурсу. Expires підтримується трохи ширше, ніж Cache-Control: max-age, тому краще використовувати саме його.

Заголовки Expires і Cache-Control: max-age. Як значення у цих заголовків використовується дата. До тих пір, поки вона не настала, браузер буде без будь-яких додаткових перевірок використовувати закешовану версію ресурсу. Expires підтримується трохи ширше, ніж Cache-Control: max-age, тому краще використовувати саме його.

Заголовки Last-Modified і Etag. Як значення заголовка Last-Modified використовується дата, а для заголовка ETag – довільна рядок. Якщо використовуються ці заголовки, то браузер, перш ніж використовувати закешований ресурс, отримає з сервера поточне значення заголовка і порівняє його з заголовком закешованої версії – якщо дані співпадуть, то буде використовуватися локальна версія, а якщо ні – відбудеться повторне завантаження. Запити перевірки відбуваються швидше, ніж повне завантаження ресурсів, що дає приріст продуктивності сайту при повторному використанні ресурсів (таблиця 2.4).

Таблиця 2.4. Елементи сторінки, що впливають на періодичність кешування ресурсів інтернет-сторінок на стороні клієнта

<i>№</i>	<i>Типи заголовку</i>	<i>Тип</i>	<i>Примітки</i>
1	Expires	дата	
2	Cache-Control: max-age	дата	Обмежена підтримка
3	Last-Modified	дата	
4	Etag	рядок	

Оптимальна стратегія клієнтського кешування. Для різко мінливих файлів

варто поставити Expires на дату, яка настане через місяць, а в момент зміни цих ресурсів використовувати метод «відбитків пальців» (fingerprinting) – додавати до імені файлу невелику частину хешу, яка утворюється на основі його змісту. Таким чином, вся статика завжди буде закешована на клієнті, а при зміні цієї самої статички буду змінені імена файлів і вони автоматично будуть перезавантажуватися браузером.

Власне техніка fingerprinting активно використовується для статичних ресурсів (стилів і скриптів) у багатьох фреймворках (зокрема, в Ruby on Rails). А масову прив'язку для всіх типів статичних файлів заголовка Expires можна реалізувати на стороні інтернет-сервера (в nginx, наприклад).

2.3. Розробка моделей вибору методів та засобів збереження та доступу і аналізу даних в проектах корпоративних інформаційних систем

Вище були визначені критерії оцінки методів та засобів збереження інформації, які, при бінарному значенні («потрібні», «непотрібні») дозволяють визначити вимоги до будь якого сайту з 16 (або 2^{**4}) можливих типів сайтів. Очевидно, що необхідні механізми визначення наслідків порушення цих критеріїв для визначення напрямків та заходів, що мають бути застосовані при покращенні роботи кожного сайту.

Найпростіший спосіб введення таких механізмів полягає у використанні системи вагових коефіцієнтів з можливістю приведення оцінок за критеріями до скаляра для подальшого співставлення з інструментами, які дозволять найкращим чином досягти визначених критеріїв.

Очевидно, що для кожної з предметних областей значення цих вагових коефіцієнтів будуть відрізнятися (що було показано в п. 1.4.2). В якості прикладу розглянемо проекти з двох предметних областей (табл. 2.3).

Також визначимо оцінки для інструментальних засобів розробки корпоративних систем і сховищ даних різними мовами програмування щодо можливостей забезпечення зазначених вище критеріїв (табл. 2.4, для прикладу

для мов C/C++ та Python).

Таблиця 2.5. Основні параметри активних сховищ корпоративних ІС

Критерій	Навчання	Соціальні мережі
Збереження та надання інформаційних ресурсів	0.5	0.2
Комунікації та обмін інформацією	0.6	0.8
Масові обчислення	0.4	0.2
Управління соціальними та технічними системами	0.4	0.2

Також визначимо якості кожного з інструментальних засобів розробки стосовно розробників системи (табл. 2.5, для прикладу для мов C/C++ та Python).

Таблиця 2.6. Основні параметри інструментальних засобів щодо забезпечення властивостей системи

Критерій	C/C++	Python
Збереження та надання інформаційних ресурсів	0.5	0.5
Комунікації та обмін інформацією	0.6	0.6
Масові обчислення	0.9	0.2
Управління соціальними та технічними системами	0.4	0.2

Таблиця 2.7. Основні параметри інструментальних засобів щодо складності розробки

Критерій	C/C++	Python
Захищеність програмного забезпечення	0.9	0.1
Швидкість розробки ПЗ проекту	0.3	0.8
Наочність коду	0.4	0.9
Швидкість обробки запитів	0.8	0.5
Валідність коду	0.6	0.7

2.4. Вибір інструментального забезпечення розробки проектів активних сховищ даних корпоративних інформаційних систем

Враховуючи результати аналізу, наведені в розділі 1, можна стверджувати, що розробку проектів бізнес аналізу даних в корпоративних інформаційних системах та інтернет-проектах на поточний момент доцільно виконувати на одній з наступних мов: PHP, Python, Perl, Ruby та Java. Кожна з цих мов має розвинуті інструментальні засоби (фреймворки), написані цими ж мовами. Хоча кожна з зазначених мов програмування має свої особливості, їх спільні переваги з погляду технологій програмування корпоративних інформаційних систем та інтернет-проектів наступні:

- Висока швидкість розробки. Це обумовлено як технічними особливостями фреймворку (наприклад, продумані угоди спрощують конфігурацію), так і інструментами розробки (консольні утиліти і генератори, готові бібліотеки, розширення і модулі).
- Простота реалізації складної бізнес-логіки – конвенції написання програмного коду на базі цих мов дозволяють створювати зрозумілий програмний код, який згодом може бути без проблем модифікований в адекватні терміни, а дотримання цих конвенцій робить програмний код супроводжуваним не тільки початковим розробником, але і будь-яким іншим спеціалістом або будь-якою іншою командою. Відсутність прив'язки до основного розробника є вкрай важливим у складних проектах.
- Висока надійність і супроводжаємость рішень – в цих мовах зазвичай використовується TDD-підхід (TDD – робити речі правильно, на відміну від BDD – робити правильні речі), що робить створювані продукти більш стабільними і супроводжуваними. Функціональність фреймворків також звичайно покрита автоматичними тестами, що робить їх використання дійсно надійним – є впевненість, що нічого не зламається. Для бізнес-проектів ця складова вкрай важлива – від стабільності роботи додатка

часто залежить ефективність роботи бізнесу в цілому.

- Масштабованість, продуктивність і високі навантаження – фреймворки звичайно «заточені» під розробку додатків, до яких пред'являються високі вимоги щодо доступності: окремі фреймворки відмінно розгортаються і працюють в кластерах серверів або в «хмарах». Для інтернет-сервісів це дуже важливий критерій, оскільки для власної ефективності і для привабливості в очах користувача вони повинні мати високі швидкості розгортання і роботи.
- Компетентні розробники. Частка хороших програмістів на цих мовах набагато вища, ніж на інших мовах. Це є наслідком їх високого рівня, значного поширення та великої кількості корпоративних проектів та інтернет-проектів. Чим компетентніше розробники, тим швидше (дешевше) розробляється проект і тим він більш продуманий, масштабується і надійний.

Будемо вважати, що інтегральна оцінка кожного з інструментів може бути обрахована як середнє геометричне якостей кожного продукту (таблиця 2.5):

$$c_d = k_e * c_e + k_q * c_q + k_v * c_v + k_w * c_w + k_r * c_r, \quad (2.5)$$

де: $k_e * c_e$ – ваговий коефіцієнт та коефіцієнт якості щодо захищеності;
 $k_q * c_q$ – ваговий коефіцієнт та коефіцієнт якості щодо швидкості; $k_v * c_v$ – ваговий коефіцієнт та коефіцієнт якості щодо наочності; $k_w * c_w$ – ваговий коефіцієнт та коефіцієнт якості щодо швидкості обробки запитів; $k_r * c_r$ – ваговий коефіцієнт та коефіцієнт якості щодо валідності.

В результаті здійснення підстановок (2.5), (2.4), (2.3), (2.2), в (2.1) отримаємо формулу розрахунку значення оцінки якості проекту.

Для визначення вагових коефіцієнтів можуть бути взяті дані з таблиць 2.3, 2.4, 2.5 або такі коефіцієнти можуть бути сформовані шляхом проведення експертизи, наприклад, за популярністю кожної з мов програмування (табл. 2.8) або обсягами інвестицій (рис. 2.5).

Таблиця 2.8. Розподіл проектів за мовами програмування в Україні (дані на початок 2018 року)

№	Мова	Доля ринку, % зараз	Зміна, %	Робочий зараз	Перспект. доля ринку	Планують зараз	Не комерц. проекти	Задоволення, %
1	Java	24.50	-1.60	719	24.25	828	731	79
2	C#	21.34	0.00	626	21.38	730	652	87
3	PHP	15.23	0.70	447	9.70	331	603	61
4	C++	8.62	-1.80	253	7.50	256	410	70
5	Python	6.65	0.30	351	10.28	351	448	86
6	JavaScript	5.45	1,75	160	5,71	195	804	60
7	Ruby	3.13	-0.69	92	5.53	189	204	79
8	Perl			23		17	55	60

2.5. Визначення властивостей мов веб-програмування корпоративних інформаційних систем з веб-доступом

Для уточнення прийнятих вище коефіцієнтів також можуть бути розглянуті особливості кожної з поширених мов веб-програмування.

PHP (Hypertext PreProcessor, препроцесор гіпертексту) – мова програмування, що виконується на стороні інтернет-сервера як інструмент створення динамічних та інтерактивних інтернет-сайтів. Мова виявився досить гнучким і потужним інструментом, тому отримала велику популярність і використовується в проектах будь-якого масштабу: від простого блогу до найбільших інтернет-додатків в Інтернеті (рис. 2.6).

Переваги PHP як інструмента розробки:

- є вільним програмним забезпеченням, поширюваним під особливою ліцензією (PHP license);
- легкий в освоєнні на всіх етапах;
- підтримується великою спільнотою користувачів і розробників;
- має розвинену підтримку баз даних;

- є величезна кількість бібліотек і розширень мови;
- може використовуватися в ізольованому середовищі;
- пропонує нативні засоби організації інтернет-сесій, програмний інтерфейс розширень;
- є досить повною заміною проприетарного середовища ASP (Active Server Pages) від Microsoft;
- може бути розгорнутий майже на будь-якому сервері;
- портований під велику кількість апаратних платформ і операційних систем.

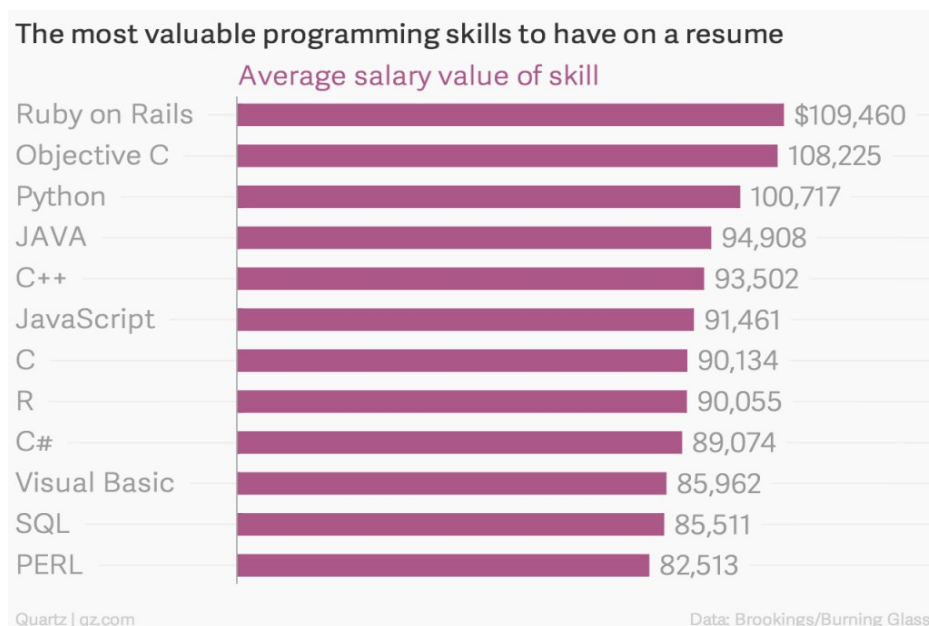


Рис. 2.5. Розподіл пропозицій щодо заробітної платні за мовами програмування (тис. \$ США на рік, за даними Burning Glass и Brookings Institution).

Недоліки PHP як інструмента розробки:

- не підходить для створення десктопних програм або системних компонентів;
- має слабкі засоби для роботи з винятками;
- глобальні параметри конфігурації впливають на базовий синтаксис мови, що ускладнює налаштування сервера і розгортання додатків;
- об'єкти передаються за значенням, що бентежить багатьох програмістів,

звиклих до передачі об'єктів по посиланню, як це робиться в більшості інших мов;

- веб-додатки, написані на PHP, часто мають проблеми з безпекою.

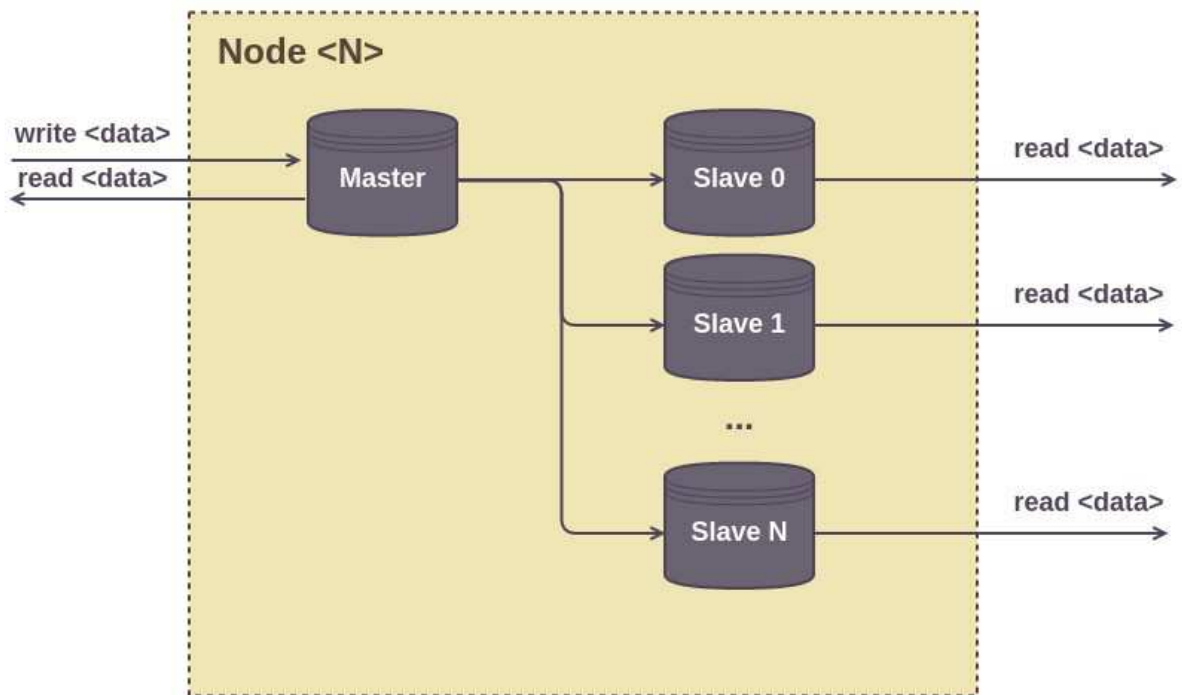


Рис. 2.6. Кластеризація даних в розподілений СУБД

Ruby – динамічна імперативна об'єктно-орієнтована мова програмування, створена під впливом таких мов, як Perl, Eiffel і Smalltalk. Характеризується динамічною типізацією і автоматичним управлінням пам'яттю. Мова Ruby використовується у інтернет-розробці у складі відкритого інтернет-фреймворку Rails, частіше званого Ruby on Rails (RoR):

Переваги Ruby як інструмента розробки:

- відкрита розробка;
- працює на багатьох платформах;
- може впроваджуватися в HTML-розмітку;
- відноситься до мов програмування надвисокого рівня (VHLL), тобто володіє високим рівнем абстракції і предметним підходом у реалізації алгоритмів;
- реалізує концептуально чисту об'єктно-орієнтовану парадигму;

- надає просунуті методи маніпуляції рядками і текстом;
- легко інтегрує в свої програми високопродуктивні сервери баз даних (DB2, MySQL, Oracle і Sybase);
- завдяки VHLL програми на Ruby добре масштабуються і легко супроводжуються;
- простий і чистий синтаксис значно полегшує програмістам перші кроки в навчанні цій мові;
- є простий програмний інтерфейс для створення багатопоточних додатків;
- має засоби для роботи з масивами;
- можливості мови можна розширити за допомогою бібліотек, написаних на C або Ruby;
- зарезервовані слова можуть бути ідентифікаторами, якщо це не створює неоднозначності для парсера;
- додаткові можливості для забезпечення безпеки;
- вбудований відладчик.

Недоліки Ruby як інструмента розробки:

- навчання мові вище початкового рівня може виявитися непростим;
- інформаційних ресурсів, присвячених Ruby, явно недостатньо;
- Ruby менш продуктивний порівняно з багатьма іншими мовами, застосовуваними у інтернет-розробці;
- Ruby відносно повільно розробляється і розвивається.

Python широко застосовується як інтерпретована мова для скриптів різного призначення (хоча існують і транслятори мови Python). Як і Ruby, Python має на меті наблизити синтаксис реальної програми, написаної на ньому, до опису задачі псевдокодом, що дозволяє програмісту зменшити обсяг програми. Мову відрізняють елегантний дизайн і ефективний, дисциплінуючий синтаксис, що полегшують програмістам спільну роботу над кодом. Python – мультипарадигмальна мова програмування: вона дозволяє поєднувати процедурний підхід до написання коду з об'єктно-орієнтованим і

функціональним. Переваги Python як інструмента розробки:

- відкрита розробка;
- досить простий у вивченні, особливо на початковому етапі;
- особливості синтаксису стимулюють програміста писати добре читається код;
- надає кошти швидкого прототипування та динамічної семантики;
- має велике співтовариство, позитивно налаштоване по відношенню до новачків;
- безліч корисних бібліотек і розширень мови можна легко використовувати в своїх проектах завдяки гранично уніфікованому механізму імпорту та програмним інтерфейсами;
- механізми модульності добре продумані і можуть бути легко використані;
- абсолютно все в Python є об'єктами в сенсі ООП, але при цьому об'єктний підхід не нав'язується програмісту.

Недоліки Python як інструмента розробки:

- не надто вдала підтримка багатопоточності;
- відсутність комерційної підтримки засобів розробки (хоча ця ситуація з часом змінюється);
- споконвічна обмеженість коштів для роботи з базами даних;
- бенчмарки показують меншу продуктивність Python в порівнянні з основними Java VM, що створює цій мові репутацію повільної.

2.6. Висновки до розділу 2

В розділі були розглянуті технології організації і доступу до даних у інтернет-проектах, що визначаються інструментальним забезпеченням, методами та засобами збереження, транспортування.

Виконано розробку математичної моделі обчислення відповідності технологій проекту щодо можливостей його реалізації в даній предметній області та можливостей забезпечення необхідних користувацьких якостей.

Розроблено спрощену систему визначення вагових коефіцієнтів відповідно до властивостей інструментального забезпечення проекту.

РОЗДІЛ 3. ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕНЬ АКТИВНИХ СХОВИЩ ДАНИХ

3.1. Розробка адаптивної моделі навантаженого сервера активного сховища даних корпоративної інформаційної системи

Враховуючи результати досліджень та з метою перевірки адекватності отриманих рішень розробимо адаптивну модель організації даних для навантаженого сервера даних корпоративної інформаційної системи. В якості основних механізмів забезпечення адаптивності будемо використовувати наступні:

- нормалізація даних в БД (генерація додаткових індексів);
- денормалізація даних в БД (зміна структури БД);
- збереження проміжних результатів (створення тимчасових таблиць).

Для вибору механізмів адаптивного перетворення будемо використовувати результати профілювання СУБД та оригінальні утиліти збору даних, які базуються на часі виконання запитів.

Алгоритми процедур наведені на рис. 3.1 — 3.4.

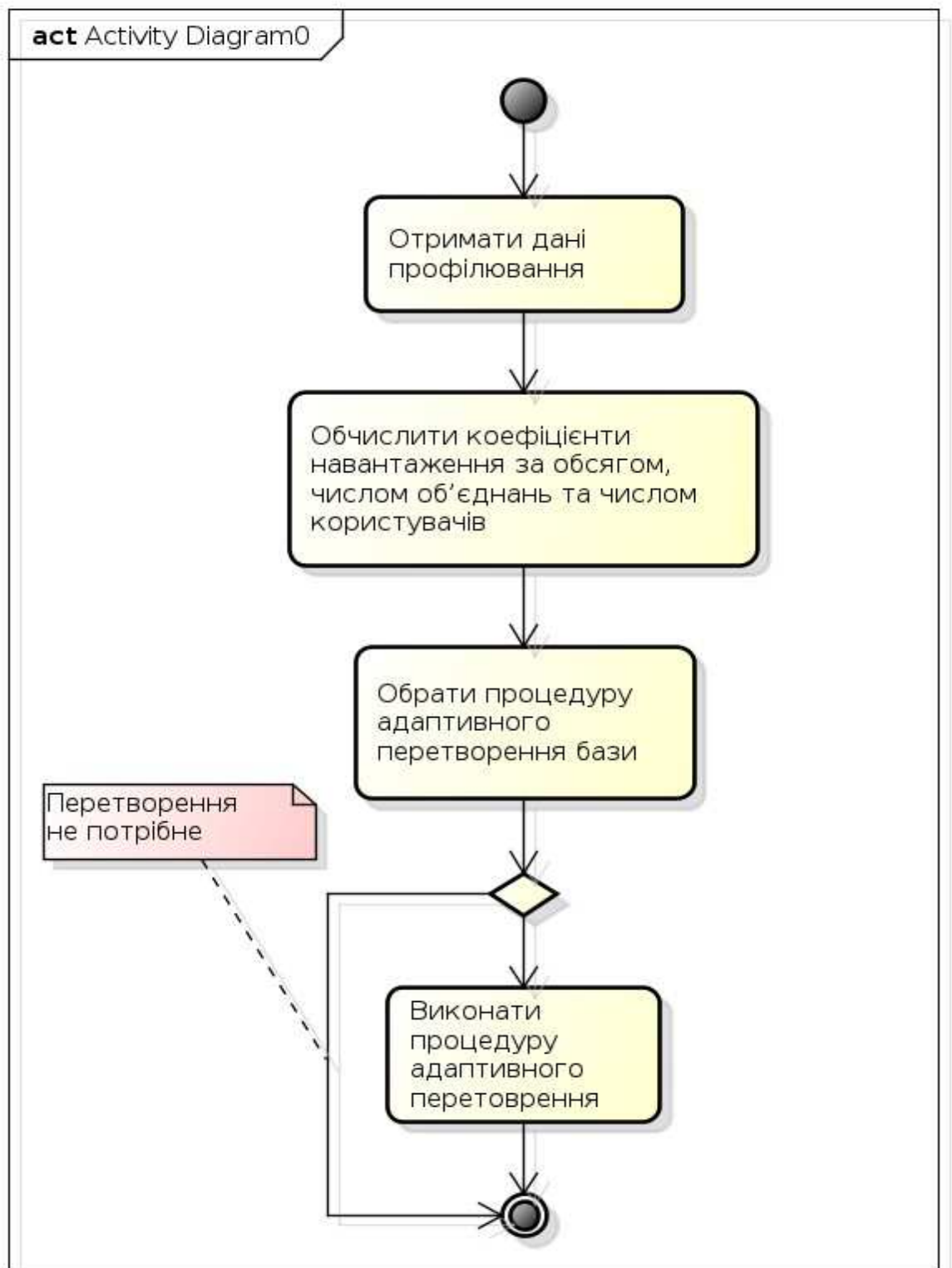
З метою проведення оцінки якості інструментального забезпечення корпоративних інформаційних систем та інтернет-проектів був реалізований спрощений алгоритм обчислення значення цільової функції, який виконується методом перебору (Рис. 3.1).

Реалізація виконана мовою програмування Python (Додаток В).

В більш складному випадку (при значному числі інструментальних засобів, обмеженнях у часі тощо) можуть бути застосовані методи вирішення оптимізаційних задач (лінійного програмування, сімплекс методи тощо).

На основі розрахунків, проведених на основі розробленого алгоритму (рис. 3.1) визначено, що у навчальних системах доцільно використовувати хмаро-орієнтовані сховища, зокрема Google Drive, Microsoft OneDrive, Dropbox, які дозволять викладачам зберігати файли поза системою MOODLE, з можливістю додавати до курсів посилання на ці файли, використовуючи

стандартні сховища або вбудовувати вміст цих файлів в деякі ресурси та види діяльності курсу.



powered by Astah

Рис. 3.1. Алгоритм процедури забезпечення адаптивності

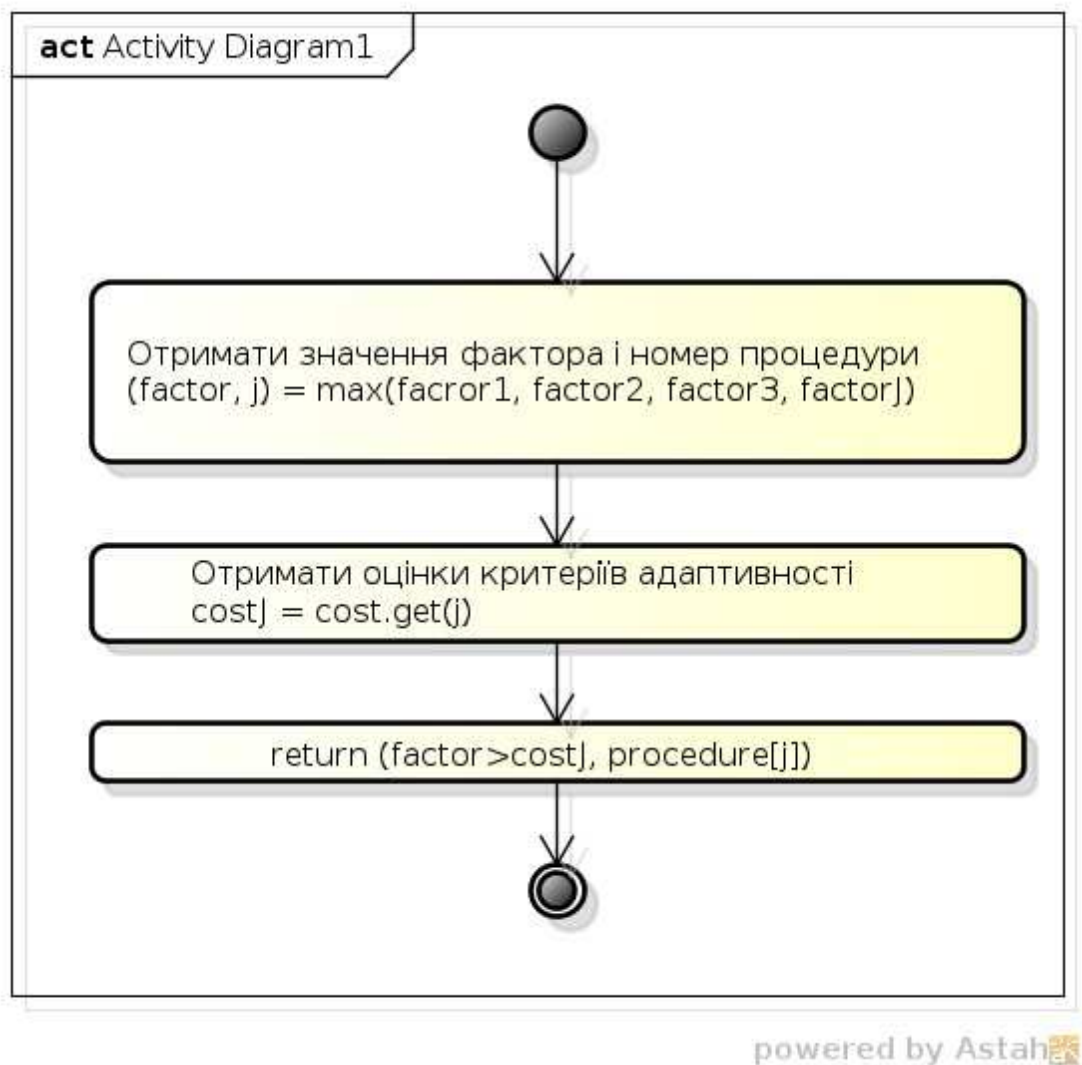
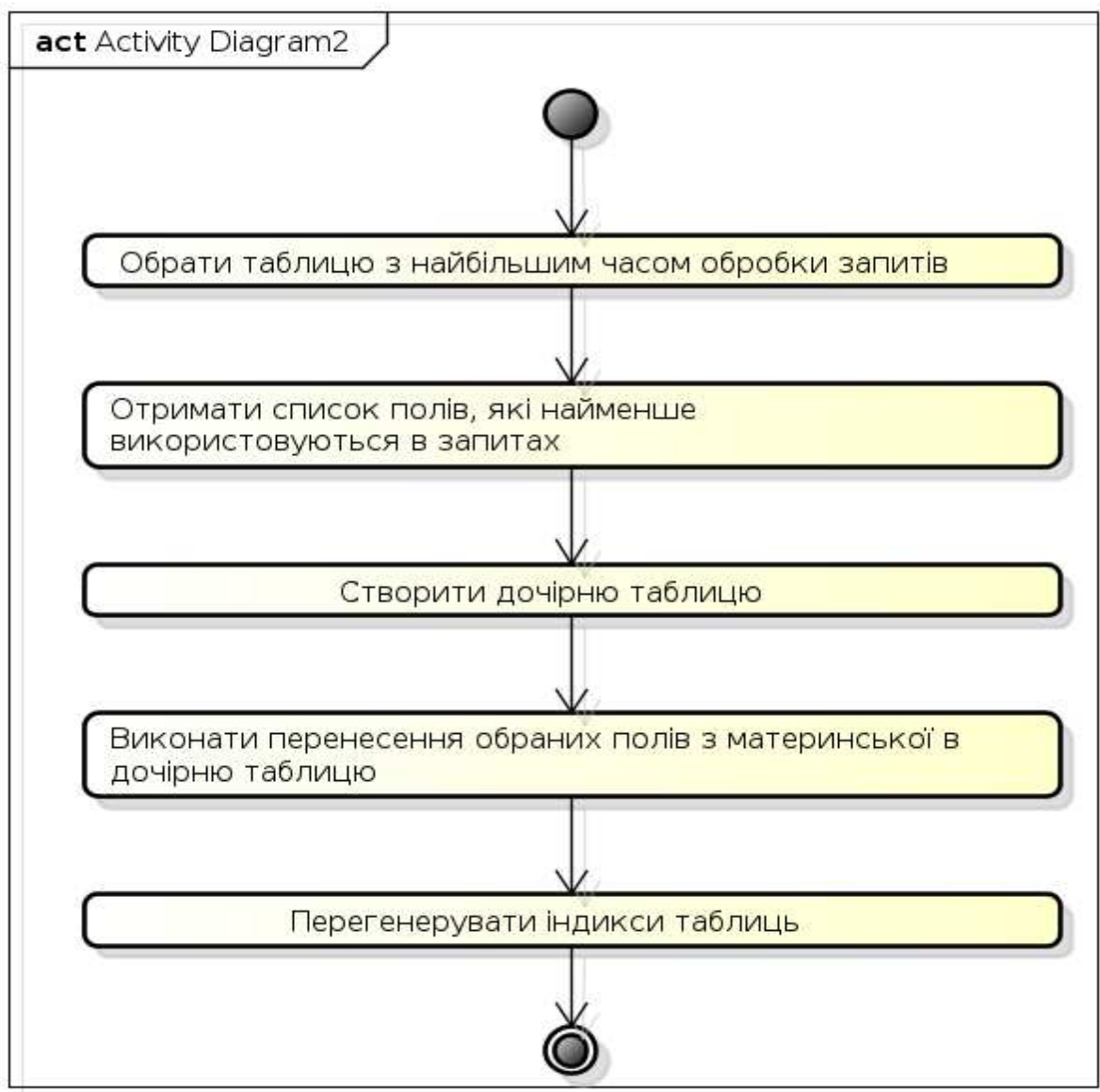


Рис. 3.2. Алгоритм процедури забезпечення адаптивності

Результати досліджень були застосовані в окремих модулях проекту автоматизованої системи управління комунальним господарством.

З метою забезпечення можливостей горизонтальної інтеграції підсистем проєктованої автоматизованої системи між собою і вертикальної інтеграції системи в інформаційну систему управління проєктами розвитку технічних систем комунального господарства міста, у проєкті автоматизованої системи повинні бути передбачені відповідні складності і особливостям вирішуваних завдань механізми та засоби інтеграції підсистем інформаційної системи в єдину систему.



powered by Astah

Рис. 3.3. Алгоритм адаптивної нормалізації за атрибутами

Забезпечення можливостей горизонтальної інтеграції підсистем між собою повинна передбачати такі інструменти реалізації системи, які дозволять досить просто нарощувати функціональність системи шляхом розширення інформаційної бази системи і введення в систему нових модулів. Вертикальна інтеграція системи в інформаційну корпоративну систему повинна передбачати забезпечення таких можливостей доступу до системи з боку зацікавлених осіб і

не повинні вимагати істотної модифікації їх інформаційних систем. Це завдання може бути вирішена шляхом надання віддаленого доступу до ресурсів системи для інших клієнтів системи через локальну мережу або Інтернет із застосуванням поширених і загальнодоступних комунікаційних протоколів для передачі даних і управління між підсистемами корпоративних інформаційних систем.



Рис. 3.4. Алгоритм адаптивної кластеризації за серверами

Локальний доступ може забезпечуватися через автоматизовані робочі місця (АРМ), встановлені на клієнтських робочих місцях персоналу підприємства.

Для цього на клієнтські робочі місця повинно бути встановлено відповідне функціям клієнтів клієнтське програмне забезпечення, а клієнтам повинні бути надані права і повноваження по доступу до інформаційних ресурсів сервера інформаційної системи відповідно до функціональних обов'язків кожного клієнта системи.

Враховуючи, що в кожній інформаційній системі накопичується інформація про параметри інформаційної системи, яка може використовуватися як в поточній, так і в інших системах, необхідно розглянути механізми доступу до інформаційних ресурсів системи з боку інших систем. Для цього може використовуватися глобальний (віддалений) доступ через мережу Інтернет, у тому числі – із застосуванням протоколів TCP / IP і HTTP.

У результаті забезпечення зазначених механізмів доступу, доступ до інформаційних ресурсів системи може бути наданий для наступних видів користувачів системи:

- авторизованих користувачів – менеджерів підприємства, команд управління проектами (надання прав і повноважень доступу до інформаційних ресурсів системи повинне буде забезпечуватися як для режиму локального доступу);
- авторизованих користувачів – співробітників розрахункових центрів комунальних служб водопостачання (авторизація виконується на основі відомої користувачам приватної облікової інформації: адреса, номер розрахункової книжки тощо);
- інших користувачів – інформація надається без авторизації (оголошення, таблиці-графіки ремонтних робіт, параметри води і т.п.).

3.2. Результати тестування активного сховища даних для різних “engines” РСУБД MySQL

На основі результатів дослідження розробимо рекомендації з вибору конфігурації системи (рис. 3.2):

- Визначити можливість використання монітору транзакцій.
- Визначити можливість використовувати систему в конфігурації клієнт-сервер.
- Визначити число одночасно активних користувачів.
- Визначити стратегії індексації даних.
- Визначити чистий розмір бази даних.
- Визначити ємність та швидкість виконання операцій читання-запису для дискової пам'яті необроблених даних, індексів, тимчасових табличних просторів.
- Узгодити передбачувану стратегія резервного копіювання.

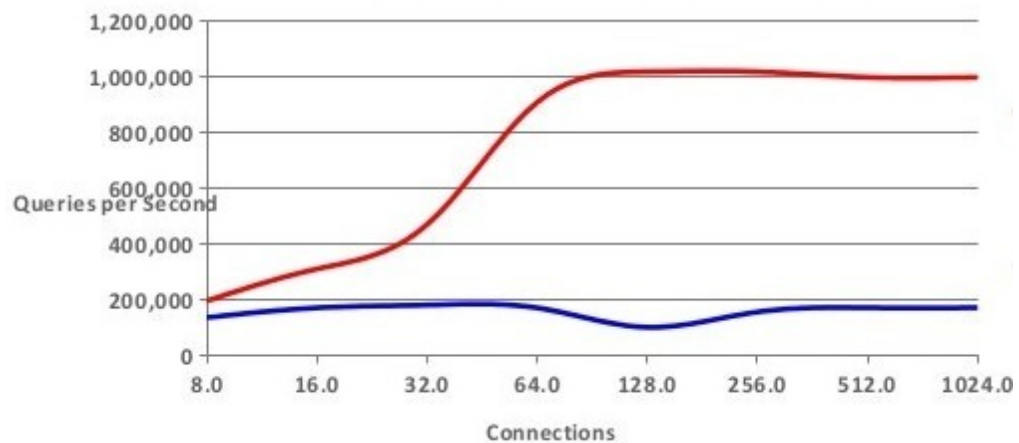


Рис. 3.2. Тести навантаження сервера (внизу з фіксованою структурою, зверху з вибором структури збереження активного сховища даних)

3.3. Висновки до розділу 3

В розділі було запропоновано спрощений алгоритм пошуку рішення на основі цільової функції оцінки якості, розглянуті можливості її застосування щодо проектів з декількох предметних областей.

Розроблені пропозиції щодо використання інструментальних засобів в різних корпоративних інформаційних системах та інтернет-проектах, а також забезпечення адаптивності структури БД.

РОЗДІЛ 4. РОЗРАХУНОК ЕКОНОМІЧНОГО ЕФЕКТУ ВІД ВПРОВАДЖЕННЯ АКТИВНОГО СХОВИЩА АІС

4.1. Загальні положення

На даний момент сучасні обчислювальні засоби широко використовуються в повсякденній діяльності людини. Сьогодні обчислювальна техніка дозволяє вирішувати найрізноманітніші задачі: від звільнення людини від монотонної ручної праці до розробки складних технічних розрахунків, які не під силу виконати одній людині. Сучасні засоби програмування дозволяють вирішувати в домашніх умовах задачі, що стояли в недавньому минулому перед цілими колективами наукових співробітників.

Розвиток нових технологій і зростання застосування обчислювальної техніки привели до широкого застосування ЕОМ для вирішення проблем, пов'язаних з обробкою інформації на виробництві, її контролем і аналізом.

Інформаційна система «Project control» дозволяє автоматизувати процеси проектування і розробки проекту програмного забезпечення, яке виконує аутсорсингова компанія. Інформаційна система надає можливість створення, перегляду і корегування проектної документації, перегляду і внесення змін до джерельного коду програми, тестування програми, перегляду стану проекту в цілому і визначенню ступеня готовності продукту. Також інформаційна система дозволяє обмінюватися повідомленнями між програмістами і проводити конференції з керівниками проекту. Керівники мають можливість давати завдання програмістам і аналізувати ефективність роботи програмістів, створювати і редагувати графік робіт, які треба виконати, аналізувати ризики зриву графіку виконання проекту

4.2. Розрахунок витрат на створення і впровадження ІС управління програмними проектами компаній з розробки програмного забезпечення

Витрати на розробку системи складаються з витрат на зарплату розробника, на амортизацію ЕОМ, на якій виконується розробка, на

експлуатацію цієї ЕОМ, на засоби розробки та витрат на матеріали і комплектуючі.

Виходячи з того, що основна заробітна плата розробників програмного забезпечення складає 4000 грн./міс., вартість сучасної ПЕОМ складає 10000 грн. і вартість кіловат-години електроенергії складає 0,9467 грн., розрахуємо вартість розробки системи:

- Розрахунок заробітної плати зробимо по формулі 4.1:

$$З_{зп} = ЗП_{роз} * T_{роз} \quad (4.1)$$

де $ЗП_{роз}$ – зарплата розробника, за місяць;

$T_{роз}$ – тривалість розробки (дослідження, створення, налагодження і впровадження).

Для розробки даної системи необхідно 6 місяців (за експертною оцінкою часу на розробку аналогічних систем). З цього випливає, що загальна сума витрат на заробітну плату складе:

$$З_{зп} = ЗП_{роз} * T_{роз} = 4000 * 6 = 24000 \text{ (грн);}$$

Додаткова заробітна плата складає 20% від основної заробітної плати:

$$З_{дод} = З_{зп} * 0,2 = 4800 \text{ (грн.)}$$

- Відрахування на соціальні заходи складають 38% від основної і додаткової заробітної плати, розраховуються по формулі 4.2:

$$В_{сз} = (З_{зп} + З_{дод}) * 38\% \quad (4.2)$$

$$В_{сз} = (24000 + 4800) * 0,38 = 10944 \text{ (грн)}$$

- Витрати на амортизацію ЕОМ, на якій виконується розробка, розраховується по формулі 4.3:

$$Z_{\text{амор}} = C_{\text{варт}} * A * T_{\text{роз}} \quad (4.3)$$

де $C_{\text{варт}} = 10000$ грн. – балансова вартість ЕОМ;

$A = 40\%$ – амортизація за рік;

$T_{\text{роз}} = 0,5$ року – час, необхідний для розробки системи.

$$Z_{\text{амор}} = 10000 * 0,4 * 0,5 = 2000 \text{ (грн)};$$

- Витрати на експлуатацію ЕОМ, на якій виконується розробка, полягають в оплаті споживаної нею електричної енергії і розраховуються по формулі 4.4:

$$Z_{\text{елек}} = P_{\text{еом}} * T_{\text{розр}} * N_{\text{рд}} * N_{\text{рч}} * E_{\text{ел}} \quad (4.4)$$

де $P_{\text{еом}} = 0,2$ квт./год. – норма споживання електроенергії ЕОМ;

$T_{\text{розр}} = 6$ міс. – тривалість розробки;

$N_{\text{рд}} = 21$ день – число робочих днів у місяці;

$N_{\text{рч}} = 8$ годин – число годин у робочому дні;

$E_{\text{ел}} = 0,9467$ грн. – вартість 1 квт. електроенергії.

$$Z_{\text{елек}} = 0,2 * 6 * 21 * 8 * 0,9467 = 190,85 \text{ (грн)}$$

- Загальновиробничі і адміністративні затрати складають 50% від основної

та додаткової заробітної плати, розраховуються за формулою 4.5:

$$З_{зв} = (З_{зп} + З_{дод}) * 50\% \quad (4.5)$$

$$З_{зв} = (24000 + 4800) * 0,5 = 14400 \text{ грн}$$

- Під час розробки інформаційної системи будуть інші поточні затрати, які включають у себе канцелярські товари, флеш накопичувачі та інші витратні матеріали, які складають 10% від основної заробітної плати:

$$В_{пот} = 24000 * 0,1 = 2400 \text{ грн}$$

- Для розробки системи необхідні наступні засоби розробки (таблиця 4.2).

Таблиця 4.2. Програмні засоби розробки

№	Найменування	Вартість, грн.
1	MySQL Server +J Connector	0
2	Eclipse	0
Разом		0

- Загальний кошторис витрат на створення системи, в урахуванням п.1) – п.7) приведений в таблиці 4.3.

Таблиця 4.3. Загальний кошторис витрат на створення системи

Найменування витрат	Сума, грн.
Заробітна плата основна	24000,00
Заробітна плата додаткова (20% від п.1)	4800,00
Витрати на соціальні заходи	10944,00
Загальновиробничі витрати	14400,00
Витрати на амортизацію ЕОМ	2000,00
Витрати на експлуатацію ЕОМ	190,85
Інші витрати	2400,00
Витрати на програмні засоби розробки	0,00
Разом на створення системи	58734,85

4.3. Витрати на впровадження системи з урахуванням вартості навчання кадрів і витрати на монтаж і налаштування

- 1) Впровадження інформаційної системи включає в себе встановлення серверу з

високошвидкісним доступом до глобальної мережі Інтернет і джерелом безперервного живлення. Витрати на обладнання наведені у таблиці 4.4. Доступ до глобальної мережі Інтернет можна вважати вже існуючим і не потребує витрат через специфіку галузі, в якій існують софтверні компанії.

2) Також необхідно налаштувати операційну систему і серверну частину інформаційної системи.

Таблиця 4.4. Витрати на обладнання

Обладнання	Сума, грн
Сервер	16000.00
Джерело безперервного живлення	3000.00
Монітор	3000.00
Клавіатура Logitech	300.00
Маніпулятор миша	250.00
Разом:	22650.00

Таблиця 4.5. Витрати на налаштування

Найменування робіт/програмного забезпечення	Сума, грн.
ОС "Cent OS"	0,00
Встановлення і налаштування операційної системи	2000,00
Монтаж і налаштування локальної мережі	500,00
ІС "Project Control"	10000,00
Встановлення серверної частини інформаційної системи	1000,00
Разом:	13500,00

3) Витрати на технічну підтримку і підготовку кадрів наведена в таблиці 4.6

Таблиця 4.6. Витрати на технічну підтримку і підготовку кадрів

Найменування	Сума, грн.
Підготовка кадрів	1000,00
Технічна підтримка за рік користування інформаційною системою	3500,00
Разом:	4500,00

Разом загальна сума витрат на створення і впровадження системи складає

40650 грн..

4.4. Економічна ефективність розробки і впровадження ІС управління програмними проектами компаній з розробки програмного забезпечення

Основним показником економічної ефективності функціонування системи є підвищення ефективності керування інформацією у вигляді зниження витрат на керування при одночасному збільшенні швидкості і якості одержання потрібного результату.

Крім багатьох інших негативних ефектів, ручна обробка інформації спричиняє наступні негативні економічні ефекти:

- а) високі витрати на складування паперових документів (сейфи, шафи й ін.);
- б) підвищення витрат на канцтовари;
- в) витрати, пов'язані з роботою виявлення раніше допущених помилок (людський фактор).

До числа основних факторів, що визначають приріст прибутку в зв'язку з упровадженням програми, відносяться:

- а) підвищення продуктивності праці;
- б) вивільнення робочого часу.

Крім того, не піддається прямій грошовій оцінці підвищення оперативності керування, якість одержуваних результатів, поліпшення організації праці і т.д.

Обов'язковою умовою визначення економічної ефективності програми є оцінка усіх показників у часі, за цінами й іншими нормами, використовуваними для визначення показників, за змістом і колом елементів витрат.

Використання даної системи дозволяє вивільнити 20% робочого часу керівника проекту а також 10% робочого часу програмістів. Таким чином ми можемо сказати, що для реалізації проекту нам необхідно буде на 10% менше програмістів і на 20% менше керівника проекту.

Визначаємо пряму економічну ефективність ґрунтуючись на тому, що на

проекті працює 10 програмістів і по 1 керівнику проекту з боку аутсорсера і замовника. При розрахунку можна вважати, що при виконанні проекту можна звільнити 1 програміста.

Зарплата 0,2 керівника проекту у рік складає

$$\Delta C_k = 25000 * 12 * 0,2 = 60000(\text{грн.})$$

Зарплата 1 програміста у рік складає

$$\Delta C_n = 4000 * 12 = 48000(\text{грн.})$$

Річний економічний ефект розраховується по формулі 4.4:

$$E_{\text{рік}} = \Delta C_n + 2 * \Delta C_k - C_{\text{супр}} - E_n * k \quad (4.4)$$

де E_n – нормативний коефіцієнт економічної ефективності капітальних вкладень

у галузь – для обчислювальної техніки приймається = 0,6;

$C_{\text{супр}}$ – вартість супроводження системи;

k – додаткові капітальні вкладення з урахуванням витрат на проектування, створення і функціонування системи.

$$\begin{aligned} E_{\text{рік}} &= 48000 + 2 * 60000 - 350 - 0,6 * (58734,85 + 20314) \\ &= 120220,69 (\text{грн.}) \end{aligned}$$

Строк окупності системи розраховується по формулі 4.5:

$$I = \frac{k}{E_{\text{рік}}} = \frac{73048,85}{120220,69} = 0,61 (\text{року}) = 8 \text{ місяців} \quad (4.5)$$

Тобто, система окупиться через 8 місяців.

РОЗДІЛ 5. ОХОРОНА ПРАЦІ

5.1. Аналіз шкідливих та небезпечних факторів на робочому місці програміста

Однією із характерних особливостей сучасного розвитку суспільства є зростання сфер діяльності людини, в яких використовуються інформаційні технології. Широке розповсюдження отримали персональні комп'ютери. Однак їх використання загостило проблеми збереження власного та суспільного здоров'я, вимагає вдосконалення наявних та розробки нових підходів до організації робочих місць, проведення профілактичних заходів для запобігання розвитку, негативних наслідків впливу ПК на здоров'я користувачів.

Шкідливі та небезпечні фактори діляться на:

- а) Фізичні;
- б) Хімічні;
- в) Біологічні;
- г) Психофізіологічні.

Для програмістів можна виділити наступні фактори, що негативно впливають на здоров'я людини:

- а) небезпечний фізичний фактор – ураження електричним струмом при взаємодії з струмо провідними частинами обладнання;
- б) шкідливі фізичні фактори:
 - 1) недостатня освітленість робочого місця;
 - 2) температура зовнішньої середовища;
 - 3) шум;
- в) шкідливі психофізіологічні фактори.
 - 1) постійне напруження зору;
 - 2) постійне статичне навантаження;
 - 3) нервово-психічне перевантаження;
 - 4) напружена зорова робота;

Освітлення – отримання, розподіл та використання світлової енергії для

забезпечення нормальних умов праці. Таким чином, освітлення повинне бути достатнім, тобто відповідати вимогам санітарних норм, рівномірним, не повинно осліплювати очі та не створювати відблиски на робочій поверхні, за спектральним складом має наближатись до сонячного світла. Оптимальним вважається таке освітлення, при якому втома зору найменша. Недопустимою величиною освітлення вважається така, нижче якої відбувається порушення зорової функції.

Дія світла на очі людини викликає трансформацію енергії електромагнітного випромінювання у нервові імпульси, які по зоровому нерву передаються в головний мозок – до зорової зони кори великих півкуль.

Слід зазначити, що світло має надзвичайно важливе значення для життєдіяльності людини. Воно впливає на стан вищих психічних функцій та на фізіологічні процеси в організмі людини. Відомо, що нормальне освітлення діє тонізуюче, викликає гарний настрій, покращує діяльність вищої нервової системи людини.

Раціональне влаштоване освітлення виробничих приміщень позитивно впливає на працівників, підвищує ефективність та безпеку праці, знижує втому та травматизм, забезпечує високу працездатність.

Санітарні та гігієнічні норми щодо виробничого освітлення, котрі базуються на психологічних особливостях сприйняття світла та його вплив на організм людини, можуть бути зведені до наступного:

- а) Спектральний склад світла, який забезпечується електричним джерелом, повинен бути близьким до сонячного;
- б) Рівень освітлення повинен бути достатнім і відповідати гігієнічним нормам, які враховують зорові роботи;
- в) Освітлення повинно забезпечувати рівномірність та стабільність світла у приміщенні, щоб запобігти частоті переадаптації та втоми зору.

Гігієнічними нормами вимагається максимально використовувати природне освітлення, оскільки денне світло краще сприймається органами зору.

Штучне освітлення у виробничих та адміністративно-громадських приміщеннях має здійснюватись системою загального рівномірного освітлення, допускають застосування системи комбінованого освітлення.

Значення освітленості на поверхні робочого столу в зоні розміщення документів має становити 300-500лк. Як джерела світла для штучного освітлення мають застосовувати переважно люмінесцентні лампи типу ЛБ.

Шум, як фізичне явище, являє собою хвильові коливання матеріальних тіл – твердих, газоподібних або рідких. Виникнення звукових відчуттів людини пов'язане з коливаннями повітря.

Шум – один з несприятливих факторів на підприємствах, що знижує працездатність робітників, їх уважність і створює передумови для виробничого травматизму та професійних захворювань.

Шум, як фізіологічне явище, являє собою несприятливий фактор для людини, викликає гіпертонію, глухоту, розлад центральної нервової системи, злоякісні пухлини тощо.

Інтенсивність шуму та вібрації є тими шкідливими факторами виробничого середовища, що негативно впливають не лише на слух, але і на нервову, серцево-судинну та інші системи організму. Судинні розлади та захворювання нервової системи в більшій мірі обмежують працездатність людини, ніж зниження слуху.

Виходячи з гігієнічних нормативів, витікає, що знижувати треба не будь-яку віброакустичну активність машин, а тільки ту, яка шкідливо впливає на людський організм, тобто за рівнем вище за гранично допустимий норматив. Таким нормативом є гранично допустимий спектр шуму та вібрації. Саме граничний спектр може бути взятий за «умовний віброакустичний нуль» (УВАН). УВАН – такий рівень віброакустичної активності машини і процесу, який забезпечує віброакустичну нешкідливість і безпеку.

Усунення промислового шуму може бути досягнуто лише в результаті проведення комплексу заходів, особливо в галузі вдосконалення технологічних

процесів і обладнання. При цьому, в першу чергу, повинні бути усунені найсильніші джерела шуму. Знешумлення окремої одиниці обладнання в загальному технологічному процесі при великій кількості комп'ютерів, що продовжують випромінювати шум, недоцільно. Необхідне знешумлення тих комп'ютерів, шумові характеристики яких виходять за рамки вимог санітарних норм.

Допустимі рівні звуку, еквівалентні рівні звуку і рівні звукового тиску октавних смугах частот мають відповідати вимогам СН 3222-85, ГОСТ 12.1.003-85, ГР 2411-81. Вони наведені в таблиці 5.1.

Таблиця 5.1. Допустимі рівні звуку

Вид трудової діяльності	Рівні звукового тиску в дБ октавних смугах із середньогеометричними частотами, ГЦ									
	31,5	63	125	250	500	1000	2000	4000	8000	Рівні звуку, дБа
Програмісти ЕОМ	86	71	61	54	49	45	42	40	38	50
Оператори в залах обробки інформації на ЕОМ	96	83	74	68	63	60	57	55	54	65
Оператори в приміщеннях з шумними агрегатами ЕОМ	103	91	83	77	73	70	68	66	64	75

Для нормальної життєдіяльності людини важливим є забезпечення

нормальних метеорологічних умов у виробничих приміщеннях, котрі суттєво впливають на самопочуття людини.

В поняття «метеорологічні умови», або «мікроклімат», виробничих приміщень входять ті фізичні фактори виробничого середовища, які впливають на тепловий стан організму і котрі необхідно постійно контролювати. Це температура, вологість, швидкість руху повітря, барометричний тиск і теплове випромінювання. Від стану виробничого приміщення залежить самопочуття і здоров'я людини.

Якщо кількість кисню в повітрі зменшиться до 12% то утруднюється дихання. В таких умовах людина напружує дихальний апарат, дихає частіше; такий стан людина витримує до 0,5 години.

Перегрівання організму відбувається за умов надлишкового конвективного випромінювання тепла нагрітих поверхонь. Розрізняють 2 фази перегрівання. Перша – фізіологічна, при котрій вступають в активну реакцію пристосувальні функції організму. При цьому активізується робота серцево-судинної та дихальної систем, відбувається інтенсивне потовиділення, котре сягає 5л за зміну. З потом втрачається велика кількість мінеральних солей та вітамінів.

Охолодження організму можливе найчастіше в зимовий і перехідний періоди року при виконанні робіт на відкритому повітрі, або робіт, які проводяться в неопалюваних виробничих і складських приміщеннях.

Тривала дія низької температури може викликати різні небажані зміни в організмі людини. Охолодження пов'язане зі значними тепловиділеннями через кінцівки рук і ніг. На охолодження впливає вологість і швидкість руху повітря.

Вологість повітря визначається ступенем його насиченості водяною парою.

Вологість повітря суттєво впливає на терморегуляцію людського організму. Підвищення відносної вологості повітря у виробничому приміщенні ускладнює терморегуляцію, зменшення тепловиділення організмом. Фізіологічно

оптимальною є відносна вологість в межах 40..60%. Норми мікроклімату для приміщень з ЕОМ представлені в таблиці 5.2

Праця працівників невиробничої сфери характеризується тривалою багатогодинною працею в одноманітному напруженому положенні, малою руховою активністю при значних локальних динамічних навантаженнях.

Робоче положення «сидячи» супроводжується статичним навантаженням значної кількості м'язів ніг, плечей, шиї та рук, що дуже втомлює. М'язи перебувають довгий час у скороченому стані і не розслаблюються, що погіршує кровообіг. В результаті, виникають больові відчуття в руках, шиї, верхній частині ніг, спині та плечових суглобах. Внаслідок динамічного навантаження на кістково-м'язовий апарат кистей рук виникають больові відчуття різної сили в суглобах та м'язах кистей рук; оніміння та уповільнена рухливість пальців; судоми м'язів кисті; ниючий біль в ділянці зап'ястя.

Таблиця 5.2. Норми мікроклімату для приміщень з ЕОМ

Пора року	Категорія робіт	Температура повітря, С, не більше	Відносна вологість повітря, %	Швидкість руху повітря, м/с
Холодна	Легка – 1а	22-24	40-60	0,1
	Легка – 1б	21-23	40-60	0,1
Тепла	Легка – 1а	23-25	40-60	0,1
	Легка – 1б	22-24	40-60	0,2

У результаті, виникають локальні м'язові перенапруження, хронічні розтягнення м'язів травматичного характеру, що можуть викликати професійні захворювання: дисоціативні моторні розлади, захворювання периферійної нервової та кістково-м'язової систем.

Крім того, робота «сидячи» призводить до зниження м'язової активності – гіподинамії. За браком рухів відбувається зниження споживаного кисню тканинами організму, сповільнюється обмін речовин. Це сприяє розвитку

атеросклерозу, ожиріння, може стати причиною дистрофії ' міокарда, хронічного головного болю, запаморочення, безсоння, роздратування.

Трудова діяльність працівників невиробничої сфери належить до категорії робіт, які пов'язані з використанням великих обсягів інформації, із застосуванням комп'ютерних робочих місць, із частим прийняттям відповідних рішень в умовах дефіциту часу, безпосереднім контактом із людьми різних типів темпераменту тощо. Це зумовлює високий рівень нервово-психічного перевантаження, знижує функціональну активність центральної нервової системи, призводить до розладів в її діяльності, розвитку втоми, перевтоми, стресу.

Тривала робота на комп'ютеризованому робочому місці призводить до значного навантаження на всі елементи зорової системи і зумовлює втому та перевтому зорового аналізатора. Напружена зорова робота викликає «очні» (біль, печія та різь в очах почервоніння повік та очей, ломота у надбрівній частині тощо) та «зорові» (пелена перед очима, подвоєння предметів, мерехтіння, швидка втома під час зорової роботи) порушення органів зору, що може викликати головний біль, посилення нервово-психічного напруження, зниження працездатності.

5.2. Розрахунок системи штучного освітлення приміщення

Освітлення відіграє важливу роль у житті людини. Біля 90% інформації сприймається через зоровий канал, тому правильно виконане раціональне освітлення має важливе значення для виконання всіх видів робіт. Раціональне освітлення повинно відповідати таким умовам: бути достатнім (відповідним нормі і: рівномірним; не утворювати тіней на робочій поверхні; не засліплювати працюючого; напрямок світлового потоку повинен відповідати зручному виконанню робіт.

За своєю природою світло – це видиме випромінювання електромагнітних хвиль довжиною від 380 до 780нм. Випромінювання вище 780нм називають

інфрачервоним, нижче – ультрафіолетовим. Залежно від джерела світла виробниче освітлення може бути 3-х видів:

а) природне – пряме або відбите світло сонця (небосхила), що освітлює приміщення через світлові прорізи в зовнішніх відгороджувачих конструкціях.

б) штучне – здійснюється штучними джерелами світла і призначене для освітлення приміщень у темні години доби, або таких приміщень, які не мають природного освітлення.

в) сполучене – одночасне поєднання природного і штучного освітлення.

Для розрахунку системи штучного освітлення приміщення застосовуються методи: питомої потужності і коефіцієнт використання світлового потоку.

Для розрахунку використаємо такі формули:

$H_0 = H - h_p$ – відстань від стелі до робочої поверхні

$h_c = 0,2 * H_0$ – відстань від стелі до світильника

$h = H_0 - h_c$ – висота підвіса світильника над освітлювальною поверхнею

$H_n = h + h_p$ – висота підвіса світильника над підлогою

$L = 1,5 * h$ – відстань між центрами світильників

$N = S / L^2$ – необхідна кількість світильників

$i = (A * B) / (h * (A + B))$ – індекс приміщення

$\Phi_p = (E_{min} * S * K_z * Z) / (N * \eta)$ – світловий потік однієї лампи

$E_f = E_{min} * (\Phi_l / \Phi_p)$ – фактична освітленість

$P_{общ} = P_l * N$ – загальна потужність

Розрахуємо освітленість в приміщенні офісу софтверної компанії. Приміщення має такі параметри, довжина $A = 5$ м, ширина $B = 3$ м, висота $H = 2,7$ м, висота робочої поверхні $h_p = 1$ м, світильник типа УМП-15. Мінімальна освітленість для лампи розжарювання за нормами $E_{min} = 100$ лк. Коефіцієнт відображення стелі $S_n = 50\%$, стін $S_c = 30\%$, робочої поверхні $S_p = 10\%$. Напруга в мережі 220В.

$H_0 = H - h_p = 2,7 - 1 = 1,7$ м.

$$h_c = 0,2 * H_0 = 0,34 \text{ м.}$$

$$h = H_0 - h_c = 1,36 \text{ м.}$$

$$H_{п} = h + h_p = 2,36 \text{ м.}$$

Для досягнення рівномірності освітлені приймаємо співвідношення:

$$L/h = 1,5$$

$$L = 1,5 * 1,36 = 2,04 \text{ м.}$$

$$N = S / L^2 = (5*3)/2,04^2 = 3,6$$

Приймаємо 4 світильники (два ряди по два світильники)

$$i = (A * B) / (h * (A + B)) = (5*3) / (1,36 * (5+3)) = 1,38$$

З таблиці коефіцієнтів використання світлового потоку при $i = 1,38$, $S_{п} = 50\%$, $S_{с} = 30\%$, $S_{р} = 10\%$ для світильника типа УНМ-15 коефіцієнт використання світлового потоку $\eta = 0,49$.

$$\Phi_p = (E_{min} * S * K_3 * Z) / (N * \eta) = (100 * 15 * 1,3 * 1,15) / (3,6 * 0,49) = 1271 \text{ лм}$$

Обираємо відповідну лампу – Б 215-225-100.

$$E_{ф} = E_{min} * (\Phi_{л} / \Phi_p) = 100 * (1350 / 1271) = 106 \text{ лк}$$

$$P_{общ} = P_{л} * N = 100 * 4 = 400 \text{ Вт}$$

З наведених вище результатів, отримали, що не треба розробляти заходи щодо зміни освітленості приміщення. Схема розташування освітлювальних приладів представлена на рисунку 5.1.

5.3 Основні положення техніки безпеки під час експлуатації ЕОМ

До роботи з персональними ЕОМ і зовнішніми пристроями ЕОМ допускаються особи, які пройшли медичний огляд, вступний інструктаж і навчання на робочому місці, перевірку знання по охорони праці.

Оператори і користувачі ЕОМ зобов'язані:

- 1) Дотримуватись правил внутрішнього розпорядку
- 2) Знати і дотримуватися правил з охорони праці в обсязі виконуваних вимог.
- 3) Виконувати тільки ту роботу, яка визначена інструкцією: з експлуатації устаткування та посадовими інструкціями, затвердженими адміністрацією

підприємства, і за умови, що безпечні способи її виконання добре відомі

4) Знати й уміти надавати першу медичну допомогу при ураження електричним струмом і при інших нещасних випадках

5) Дотримуватися інструкції про заходи пожежної безпеки.

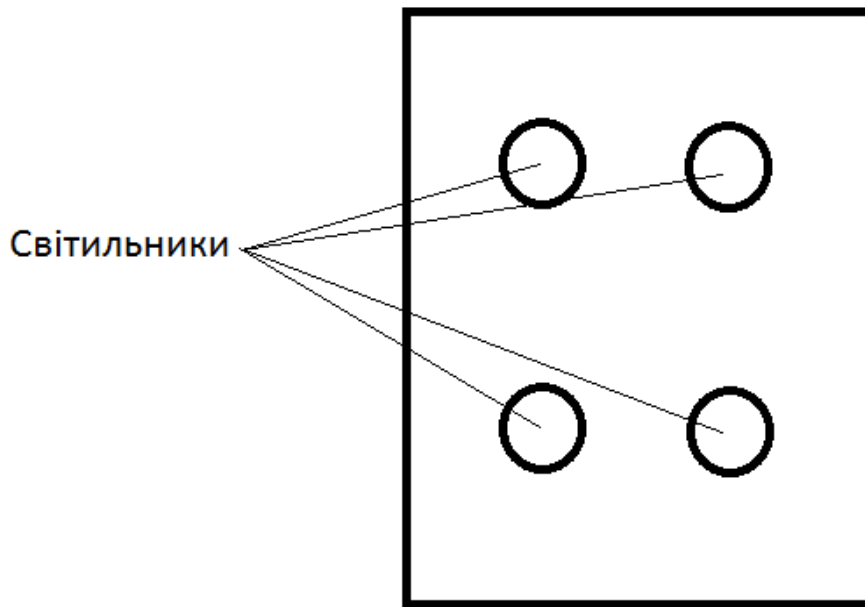


Рис. 5.1. Схема розташування освітлювальних приладів

Якщо не виконувати профілактичні заходи і не дотримуватися режиму роботи, робота з ЕОМ, як правило, супроводжується зоровим і загальним перевтомою.

Організація робочого місця:

- 1) Приміщення повинно мати природне і штучне освітлення.
- 2) Площа на одне робоче місце з ЕОМ для дорослих користувачів повинна бути не менше 6,0 кв.м, обсяг – не менш 20,0 куб.м.
- 3) У виробничих та адміністративно-громадських приміщеннях у випадках переважної роботи з документами допускається застосування системи комбінованого освітлення.
- 4) Освітленість на поверхні столу в зоні розміщення робочого та документа

повинна бути 300-500 лк.

5) Робочі місця по відношенню до світлових отворів повинні розташовуватися так, щоб природне світло падало з боку переважно сторінки. З метою профілактики перевтоми і перенапруження при роботі з дисплеями необхідно виконувати під час регламентованих перерв комплекси спеціальних вправ (вони наведені нижче).

6) Схеми розміщення робочих місць з ЕОМ повинні враховувати відстані між робочими столами з відеомоніторами (в напрямі тилу поверхні одного відеомонітора і екраном іншого), яка повинна бути не менше 2,0 м відстань між бічними поверхнями відеомонітора не менше 1,2 м. 7) Робочий стіл повинен регулюватися по висоті в межах 680-760 мм, за відсутності такої можливості його висота має складати 1600х900 мм. Під стільницею робочого столу має бути вільний простір для ніг з розмірами по висоті не менше 600 мм, по ширині 500 мм, по глибині 650 мм.

7) Розмір екрана повинен бути не менше 31 см (14 дюймів) по діагоналі, при цьому відстань від очей до екрану повинна бути в межах 40-80 см.

8) Робочий стіл (крісло) повинен бути підйомно-поворотним і регульованим по висоті та кутах нахилу сидіння і спинки, а також – відстані спинки від переднього краю сидіння. Конструкція його повинна забезпечувати:

а) ширину і глибину поверхні сидіння не менше 400 мм;

б) поверхню сидіння з заокругленим верхнім краєм;

в) регулювання висоти поверхні сидіння у межах 400-550 мм і кутом нахилу вперед на 15 град., і тому до 5 град.;

г) висоту опорної поверхні спинки 300 плюс-мінус 20 мм, ширину не менше 380 мм та радіус кривизни горизонтальної площині – 400 мм;

д) кут нахилу спинки у вертикальній площині в межах 0 + / – 30 град.;

е) врегулювання відстані спинки від переднього краю сидіння в межах 250-400 мм;

є) стаціонарні або знімні підлокітники довжиною не менше 250 мм: і шириною

50-70 мм; регулювання підлокітників по висоті над; сидінням у межах 230 + / – 30 мм і внутрішнього відстані: між підлокітниками в межах 350-500 мм.

Робоче місце має бути обладнане підставкою для ніг, що має ширину не менше 300 мм, глибину не менше 400 мм, регулювання по висоті в межах до 150 мм і кутом нахилу опорної поверхні підставки до 20 град. Поверхня підставки має бути рифленою і мати по передньому краю бортик висотою 10 мм.

Робоче місце з ЕОМ повинне бути оснащене легко переміщуваним пюпітром для документів.

Клавіатуру слід розташовувати на поверхні столу на відстані 100-300 мм від краю, зверненого до користувача, або на спеціальною регульованою по висоті робочої поверхні, відокремленої від основний стільниці. Про кожний нещасний випадок на виробництві потерпілий або очевидець негайно сповіщає безпосереднього керівника.

За невиконання даної інструкції винні притягуються до відповідальності згідно з правилами внутрішнього трудового розпорядку або стягненням, визначених Кодексом законів про працю України.

РОЗДІЛ 6. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

6.1. Вступ

Важко уявити сьогоденне життя без персонального комп'ютера (ПК), електроніки, мобільного телефону. ПК всюди – вдома, на роботі, школі, вузі, аптеці, банку, на вокзалі, магазині, лікарні і т.д. Інформатизація суспільства в повному обсязі. Бази даних, величезні потоки різної інформації, Internet – звичні слова і поняття. Кількість персональних комп'ютерів зростає з кожним роком.

На сьогоднішній день (за різними оцінками фахівців), обсяг випущених в світі ПК становить 1 мільярд одиниць. Але будь-яка техніка стрімко застаріває, їй на зміну приходять нові, більш потужні, більш сучасні ПК та оргтехніка. Людство, хоче воно того чи ні, втягнуте в постійний процес модернізації та заміни електронної техніки. Ми радіємо новим моделям персональної техніки з новими можливостями. Поступово виникає проблема: а що робити зі старою технікою, морально застарілою або з тих чи інших причин, що вийшла з ладу, яка захащує підсобні приміщення та склади.

6.2. Проблема екології

За період з 1991 року по теперішній час в Україну завезено різними постачальниками близько 10 млн. одиниць (близько 400000 т.) персональної й оргтехніки (це за найскромнішими підрахунками), мобільних телефонів – 37-40 млн. шт. (близько 4800 т.). І це приблизні дані. Точного підрахунку ніхто не проводить.

Стурбованість громадськості проблемами екології, а також нові, більш жорсткі закони по захисту навколишнього середовища змушують великих виробників устаткування створювати мережі по збору вийшла з обігу техніки і заводи з її утилізації. Крім того, в конструкції обладнання максимально збільшується частка матеріалів, придатних для переробки. Розміри мережі з утилізації "електронного брухту" залежать від регіону і місцевого

законодавства. Так, наприклад, у Західній Європі, де екологічні закони дуже суворі, компанія Hewlett-Packard створила вельми значну інфраструктуру зі збору та переробки застарілих комп'ютерів та оргтехніки. Всього в Європі продукцію HP переробляють 30 заводів, один з яких знаходиться в Росії. Справедливості заради потрібно відзначити, що ці заводи не є власністю HP. Вони належать партнерським компаніям американського вендора, які беруть участь у програмі утилізації списаної техніки.

Вся оргтехніка включає в свій склад як органічні складові (пластик різних видів, матеріали на основі полівінілхлориду, фенолформальдегіда), так і майже повний набір металів.

Всі ці компоненти не є небезпечними в процесі експлуатації виробу. Однак ситуація докорінно змінюється, коли виріб потрапляє на звалище. Такі метали, як свинець, сурма, ртуть, кадмій, миш'як входять до складу електронних компонентів переходять під впливом зовнішніх умов у органічні і розчинні сполуки і стають найсильнішими отрутами. Утилізація пластиків, що містять ароматичні вуглеводні, органічні хлорпохідні з'єднання є насуцною проблемою екології. Тому вся оргтехніка повинна утилізуватися за методикою затвердженою Державним комітетом України по телекомунікаціям. Завдяки комплексній системі утилізації оргтехніки зводяться до мінімуму не перероблювані відходи, а основні матеріали (пластмаси, кольорові і чорні метали) і цінні компоненти (рідкісні метали, люмінофор, ферити і інші) повертаються у виробництво. Дорогоцінні метали, що містяться в електронних компонентах оргтехніки концентруються і після переробки на афінажних заводах здаються в Держфонд.

6.3. Облік дорогоцінних металів у виробках

Майже у всіх комп'ютерах, електронній техніці вітчизняного або імпортного виробництва є деяка кількість золота, срібла та інших дорогоцінних металів. Це загальновідомий факт. А ось про те, що всі без винятку фірми

повинні вести їх облік знають небагато. Тим не менш, за відсутність такого обліку можна поплатитися штрафом. Будь-яка організація зобов'язана документально оформляти надходження, рух, інвентаризацію та вибуття дорогоцінних металів, що містяться в складових частинах офісної техніки (комп'ютерів, телевізорів і т.д.).

Незважаючи на те, що ці нормативні акти ніхто не відміняв, вони якось випали з поля зору фірм, діяльність яких безпосередньо не пов'язана з дорогоцінними металами. Чим же обернеться фірмі відсутність обліку «дорогоцінних» запчастин? Відповідають фахівці Центральної державної інспекції пробірного нагляду Російської державної пробірної палати (саме цей держорган перевіряє, чи правильно фірми ведуть облік дорогоцінних металів). «Вести облік дорогоцінних металів, у тому числі що містяться в комп'ютерній техніці, зобов'язані всі фірми, – відповіли нам у інспекції. – Порушникам загрожує штраф за статтею 19.14. КоАП України.»

Комп'ютерами та оргтехнікою в нашій країні володіють фізичні та юридичні особи. Утилізація оргтехніки фізичними особами відбувається дуже прозаїчно – викинув на смітник у гіршому випадку, в кращому – розбирання на частини для подальшого застосування, що в кінцевому рахунку закінчується тією ж звалищем. Іноді просунуті керівники, щоб не зв'язуватися з утилізацією, просто дарують стару техніку своїм підшефним організаціям (дитячі будинки, школи, заклади культури). Але сама проблема залишається. Цей процес в даний час не хвилює ні більшу частину населення, ні держава. Утилізацією в цьому випадку займаються шукачі кольорових і чорних металів. З юридичними особами все набагато складніше. Згідно із законодавством, персональні комп'ютери відносяться до основних засобів і підлягають бухгалтерському обліку на підприємстві із зазначенням кількості дорогоцінних металів, які в них містяться. Мало того, на цю техніку поширюється правило про амортизацію протягом 10 років.

Іншими словами, списувати і утилізувати дане обладнання можна тільки

через 10 років. Однак ми знаємо, що персональна техніка морально застаріває набагато раніше. Це пов'язано зі стрімким розвитком електроніки і програмного забезпечення. У такому випадку юридична особа повинна звернутися до спеціалізованим підприємствам, які займаються ремонтом і обслуговуванням персональної техніки для проведення технічної експертизи обладнання з отриманням висновку про те, що персональна техніка морально застаріла, знята з виробництва, ремонтна база відсутня, підлягає списанню та утилізації в установленому порядку . Тільки після цього необхідно укласти договір з підприємством, яке займається роботою з відходами.

Як нам відомо, персональний комп'ютер містить усі види відходів. В даному випадку обладнання, що містить дорогоцінні метали, кольорові, чорні метали, полімери, здаються на підприємства, які мають ліцензію на роботу з дорогоцінними металами. Ці підприємства – переробники дорогоцінних металів, в свою чергу, укладають договори з підприємствами, які мають інші ліцензії і таким чином персональна техніка розбирається і утилізується згідно із законодавством.

6.4. Процес переробки

В принципі, будь-який комп'ютер або телефон можна переробити і пустити у вторинне використання. При грамотній утилізації близько 95% відходів техніки здатні повернутися до нас в тому чи іншому вигляді, і приблизно 5% відправляються на звалища або федеральні заводи з переробки твердих побутових відходів.

Співвідношення ручного та автоматизованого праці на фабриках по переробці комп'ютерної техніки залежить від її типу. Для монітора це співвідношення приблизно 50 на 50 – розбирання старих кінескопів є досить трудомістким заняттям. Для системних блоків та оргтехніки частка автоматичних операцій вище.

НР вперше запропонувала переробку відслужили свій термін продукції

ще в 1981 році. Сьогодні НР володіє інфраструктурою по збору й переробки використаних ПК та оргтехніки в 50 країнах світу. У рік утилізації піддається близько 2,5 млн. одиниць продукції. В одному тільки 2007 році НР переробив близько 100 тис. тонн списаного обладнання та витратних матеріалів, – майже в півтора рази більше, ніж роком раніше.

Перший етап завжди проводиться вручну. Це – видалення всіх небезпечних компонентів. У сучасних настільних ПК і принтерах таких компонентів практично немає. Але переробці піддаються, як правило, комп'ютери і техніка, випущені в кінці 90-х – початку 2000-х років, коли плоских рідкокристалічних моніторів просто не існувало. А в кінескопних моніторах міститься чимало сполук свинцю. Інша категорія продукції, що містить небезпечні елементи, – ноутбуки. В акумуляторах і екранах застарілих моделей є певна кількість ртуті, яка також дуже небезпечна для організму. Важливо відзначити, що в нових моделях ноутбуків від цих шкідливих компонентів позбулися.

Потім видаляються всі великі пластикові частини. У більшості випадків ця операція також здійснюється вручну. Пластик сортується в залежності від типу і подрібнюється для того, щоб надалі його можна було використовувати повторно. Що залишилися після розбирання частини відправляють у великий подрібнювач-шредер, і всі подальші операції автоматизовані. Багато в чому технології переробки запозичені з гірничої справи – приблизно таким же способом витягують цінні метали з породи.

Подрібнені в гранули залишки комп'ютерів піддаються сортуванню. Спочатку за допомогою магнітів витягуються всі залізні частини. Потім приступають до виділення кольорових металів, яких в ПК значно більше. Алюміній добувають з брухту за допомогою електролізу. У сухому залишку виходить суміш пластику і міді. Мідь виділяють способом флотації – гранули поміщають в спеціальну рідину, пластик спливає, а мідь залишається на дні. Сама ця рідина не отруйна, проте, робочі на заводі використовують захист

органів дихання – щоб не вдихати пил.

6.5. Утилізація комп'ютерів у інших країнах

У США, ЄС та Японії протягом декілька останніх років активізували свою діяльність компанії з утилізації старих комп'ютерів. Так, у США нарощуючи зусилля по організації переробки ПК, один з найбільших виробників персональної техніки компанія Dell надає гранти 12 американських міст на організацію центрів переробки застарілої техніки. Dell почала переробку комп'ютерної техніки восени 2002 року. Спочатку компанія брала на себе всі витрати на переробку, за винятком витрат на перевезення. З весни 2003 року компанія стала приймати старі комп'ютери в переробку по ціні 15 доларів США за одиницю. Щороку в Німеччині викидається 2,2 млн. комп'ютерів, принтерів і моніторів. До них додається інша побутова техніка. Кожні шість місяців на ринок надходять більш досконалі моделі, а середньостатистичний німець повністю або частково оновлює свій комп'ютер кожні 2-3 роки. До останнього часу рядовий німець був змушений викладати за утилізацію свого комп'ютера залежно від моделі від 15 до 30 євро. Ситуація повинна кардинально змінитися з прийняттям Європейським союзом нових правил утилізації електроніки. З 2005 року всі виробники зобов'язані безкоштовно приймати для утилізації свою продукцію. Зараз так відбувається в Німеччині з батареями: викидати їх у загальний сміття заборонено, але можна здати в будь-який магазин з продажу електроніки. За даними експертів на кожного жителя Євросоюзу доводиться щорічно по 14 кг. «Електронного сміття», а в Німеччині цей показник досягає 16 кг. Німецький концерн Siemens і японська компанія Fujitsu мають намір створити спільне підприємство з утилізації персональних комп'ютерів, утилізації оргтехніки. СП планує побудувати заводи в 5-ти містах Німеччини. Намічено створення аналогічних виробництв у Великій Британії, Франції та скандинавських країнах. На думку Siemens і Fujitsu організація утилізації оргтехніки і персональних комп'ютерів поліпшить екологічну

ситуацію і сприятиме збільшенню попиту на нові моделі комп'ютерів.

У Японії компанія Sony організувала службу по збору та утилізації застарілих персональних комп'ютерів своєї марки. Створена спеціальна мережу по збору та доставки ПК із залученням перевізників. Вартість переробки одного комп'ютера 4-5 тисяч ієн (40-50 доларів США).

6.6. Висновки

У даний момент можна з упевненістю сказати про те, що закон про утилізацію комп'ютерів необхідний, так як в дійсності ми не повинні негативно впливати на навколишнє нас середовище. Але в нашій країні все набагато простіше, існує дуже багато компанії, які із задоволенням приймуть замовлення на утилізацію старого устаткування, вам не доведеться тривалий час проводити за документами, які необхідні для утилізації, ваші комп'ютери зможуть забрати компанії, що займаються утилізацією. Все що від вас буде потрібно – це правильно оформлене списання, пам'ятайте, що списання комп'ютерів можна без особливих складнощів здійснити за допомогою спеціальних компаній, а також складання всього одного документа на утилізацію оргтехніки. Вартість таких послуг не велика, і ви безумовно зможете дозволити собі зробити навколишній світ чистішим.

ЗАГАЛЬНІ ВИСНОВКИ ПО РОБОТІ

В роботі в цілому вирішені задачі дослідження, що дозволило досягти основної мети дослідження: підвищення ефективності процесів прийняття рішень щодо вибору технологій організації і доступу до активних сховищ даних на основі наукового обґрунтованих моделей та механізмів.

Якість вибору технологій забезпечується якістю експертних оцінок.

Запропоновані рішення можуть бути застосовані при вирішенні задач проектування навантажених сховищ даних в автоматизованих системах бізнес-аналітики.

Запропоновані рішення можуть бути застосовані при вирішенні подібних задач для інших програмних проектів. Для цього необхідно лише визначити значення вагових коефіцієнтів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Агальцов, В.П. Базы данных. В 2-х т. Т. 1. Локальные базы данных: Учебник / В.П. Агальцов. – М.: ИД ФОРУМ, НИЦ ИНФРА-М, 2013. – 352 с.
2. Агальцов, В.П. Базы данных. В 2-х т. Т. 2. Распределенные и удаленные базы данных: Учебник / В.П. Агальцов. – М.: ИД ФОРУМ, НИЦ ИНФРА-М, 2013. – 272 с.
3. Мазуркевич, А. МВ PHP: настольная книга программиста / А. Мазуркевич, Д. Еловой. – Мн.: Новое знание, 2003. – 480 с.: ил
4. Баррет Д. JavaScript. Web-профессионалам / Д. Баррет. – Киев: БХВ – Киев, 2001. – 240 с.
5. Бранденбау Д. JavaScript: сборник рецептов / Д. Бранденбау – СПб.: Питер, 2000. – 416 с.
6. Будилов В. JavaScript, XML и объектная модель документа / В. Будилов – СПб.: НиТ, 2001. – 352 с.
7. Вагнер, Р. JavaScript. Энциклопедия пользователя / Р. Вагнер, В. Аллен. – Киев: ДиаСофт, 2001. – 480 с.
8. Вайк, А. JavaScript в примерах / А. Вайк. – Киев: ДиаСофт, 2000. – 232 с.
9. Вандер, В. Э. JavaScript для "чайников" / Вер Э. Вандер. – Диалектика, 2001. – 304 с.
10. Вейнер П. Языки программирования JAVA и JavaScript / П. Вейнер. – М: ЛОРИ, 2000. – 242 с.
11. Гарнаев А.Ю. Web-программирование на Java и JavaScript / А.Ю. Гарнаев, С.Ю. Гарнаев. – СПб.: БХВ Санкт-Петербург, 2005. – 1019 с.
12. Голицына, О.Л. Базы данных / О.Л. Голицына, Н.В. Максимов, И.И. Попов. – М.: Форум, 2004. – 352 с.
13. Голицына, О.Л. Базы данных: Учебное пособие / О.Л. Голицына, Н.В. Максимов, И.И. Попов. – М.: Форум, 2012. – 400 с.

14. Гутманс, Э. PHP 5. Профессиональное программирование / Э. Гутманс, С. Баккен, Д. Ретанс. Пер. с англ. СПб.: Символ-Плюс – 2006. – 704 с., ил.
15. Дарнел Р. JavaScript. Справочник / Р. Дарнел. – СПб.: Питер, 2000. – 192 с.
16. Вандик, Д.К. Pro Drupal 7 Development: Third Edition / Джон К. Вандик, М. Вестгейт. – Apress, 2010. – 320 с.
17. Дмитриева М.В. Самоучитель JavaScript / М.В. Дмитриева. – СПб.: БХВ Санкт-Петербург, 2001. – 512 с.
18. Дмитриева М.В. JavaScript. Быстрый старт / М.В. Дмитриева. – СПб.: БХВ Санкт-Петербург, 2002. – 336 с.
19. Дронов В. JavaScript в Web-дизайне / В. Дронов. – СПб.: БХВ Санкт-Петербург, 2002. – 879 с.
20. Дубаков, М.А. Интернет-мастеринг / М.А. Дубаков. – СПб.: ВHV, 2002. – 532 с.
21. Карпова, И.П. Базы данных: Учебное пособие / И.П. Карпова. – СПб.: Питер, 2013. – 240 с.
22. Кингсли-Хью Э. JavaScript 1.5: учебный курс / Э. Кингсли-Хью. – СПб.: Питер, 2001. – 272 с.
23. Кириллов, В.В. Введение в реляционные базы данных / В.В. Кириллов, Г.Ю. Громов. – СПб.: БХВ-Петербург, 2012. – 464 с.
24. Кошелев, В.Е. Базы данных в ACCESS 2007: Эффективное использование / В.Е. Кошелев. – М.: Бином-Пресс, 2009. – 592 с.
25. Кузин, А.В. Базы данных: Учебное пособие для студ. высш. учеб. заведений / А.В. Кузин, С.В. Левонисова. – М.: ИЦ Академия, 2012. – 320 с.
26. Ульман, Л. Основы программирования на PHP: / Ларри Ульман. Пер. с англ. – М.: ДМК Пресс, 2001. – 288 с.: ил.
27. Ливена, С.В. Практика увольнений за прогул. По материалам базы данных "Пакет кадровика" / С.В. Ливена. – М.: ИНФРА-М, 2008. – 51 с.
28. Пирогов, В.Ю. Информационные системы и базы данных: организация и

- проектирование: Учебное пособие / В.Ю. Пирогов. – СПб.: БХВ–Петербург, 2009. – 528 с.
29. Мак–Федрис, П. Использование JavaScript. Специальное издание / П. Мак–Федрис. – Диалектика, 2002. – 896 с.
30. Матросов, А.В. HTML 4.0. / А.В. Матросов, А.О. Сергеев, М.П. Чаунин. – СПб.: БХВ Санкт–Перебург, 1999. – 672 с.
31. Монкур, М. JavaScript 1.3 / М. Монкур. – М.: Вильямс, 2000. – 304 с.
32. Мэрдок, К. JavaScript: наглядный курс создания динамических Web–страниц / К. Мэрдок. – Диалектика, 2001. – 288 с.
33. Николенко, Д.В. Практические занятия по JavaScript (+дискета) / Д.В. Николенко. – СПб.: НиТ, 2000. – 129 с.
34. Советов, Б.Я. Базы данных: теория и практика: Учебник для бакалавров / Б.Я. Советов, В.В. Цехановский, В.Д. Чертовской. – М.: Юрайт, 2013. – 463 с.
35. Хольцнер, С. PHP в примерах / Стивен Хольцнер. М.: ООО «Бином–Пресс», 2007 г. – 352 с
36. Томсон, Л. Разработка Web–приложений на PHP и MySQL: Пер. с англ. /Лаура Томсон, Люк Веллинг. – 2–е изд., испр. – СПб: ООО «ДиаСофтЮП», 2003. – 672 с.
37. Фуфаев, Э.В. Базы данных: Учебное пособие для студентов учреждений среднего профессионального образования / Э.В. Фуфаев, Д.Э. Фуфаев. – М.: ИЦ Академия, 2012. – 320 с.
38. Хоумер А. Dynamic HTML. Справочник / А. Хоумер, К. Улмен. – СПб.: Питер, 2000. – 512 с.; ил.

ДОДАТОК А.

**ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ПРОГРАМНОГО
ЗАБЕЗПЕЧЕННЯ ПІДСИСТЕМИ АКТИВНОГО СХОВИЩА ДАНИХ**

Вступ

Назва розробки “Підсистема активного сховища даних”.

Призначенням розробки – автоматизація бізнес-процесів аналізу даних корпоративної автоматизованої інформаційної системи.

Вимоги до проекту

Вимоги до функціональних характеристик програмного забезпечення:

- Автоматичний розрахунок коефіцієнтів навантаження між окремими серверами баз даних за таблицями, атрибутами таблиць і клієнтами.
- Формування і надання звітів про розподіл навантаження між окремими серверами баз даних за таблицями, атрибутами таблиць і клієнтами.
- Виконання ручного і автоматичного розподілення даних між серверами СКБД.
- Виконання ручного злиття даних з серверів СКБД на вказаний сервер.

Вимоги до надійності і якості

Надійність і якість функціонування проекту повинні забезпечуватися:

- оптимізованою структурою проекту;
- системою виводу повідомлень про можливі загрози;
- наявністю дублікатів проекту і зберіганням програмного забезпечення на магнітних носіях.

Умови експлуатації

Програмне забезпечення розрахована на користувача з кваліфікацією роботи з СКБД на рівні адміністратора баз даних.

Вимоги до складу і параметрів апаратного забезпечення проекту

Проект розрахований на роботу в локальній чи Інтернет мережі.
Мінімальні вимоги до технічного забезпечення проекту:

- Процесор із частотою 3,2 ГГц
- Оперативна пам'ять 4 Гб
- Відео пам'ять 32 МБ
- Мережева карта не нижче 1 Гб
- Жорсткий диск 2 ТБ

Вимоги до інформаційної та програмної сумісності

Проект повинен задовольняти вимогам використовуваної операційної системи, бути сумісними із системними програмними засобами і СКБД, мати можливість переносу на інші ЕОМ аналогічного класу.

Вимоги до випробування

Випробування має виконуватись на запропонованому замовником контрольному прикладі. Контрольний приклад повинен включати тестові завдання на фрагментацію і дефрагментацію бази даних і забезпечувати перевірку правильності й працездатності функціонального наповнення проекту і його програмної реалізації, забезпечувати перевірку працездатності програм у різних ситуаціях.

Контрольний приклад має охопити всі варіанти рішення, передбачені функціональним описом і алгоритмами, відбивати всі етапи рішення, демонструвати можливості проекту.

Вимоги до програмної документації

Вимоги до складу, утримування й оформлення програмної документації визначаються стандартами ЕСПД. Склад програмної документації:

- Опис програми.
- Опис застосування.
- Керівництво системного програміста.
- Функціональний опис.
- Зміст програми на носії даних.

Стадії та етапи розробки

Стадії та етапи розробки підсистеми адаптивного сховища даних, зміст робіт і терміни виконання наведені в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки

Стадії	Етапи роботи	Початок робіт	Закінчення робіт
1. Аналіз	Виконання аналізу і постановка задачі	12.09.2021	26.09.2021
2. Дослідження та розробка моделей	Дослідження методів забезпечення комунікацій в активних сховищах	27.09.2021	12.10.2021
	Затвердження концепції та технічного завдання	13.10.2021	17.10.2021
	Розробка проектних рішень по видам забезпечення системи	18.10.2021	31.10.2021
	Затвердження проектних рішень	01.11.2021	06.11.2021
3. Робочий проект	Розробка програмного забезпечення системи, розгортання системи	07.11.2021	30.11.2021
	Розробка документації по системі	01.12.2021	08.12.2021
	Випробування програми	09.12.2021	10.12.2021

ДОДАТОК Б.

СТРУКТУРА ТАБЛИЦЬ БАЗИ ДАНИХ КЛІЄНТІВ АКТИВНОГО СХОВИЩА ДАНИХ КОРПОРАТИВНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ІТ-КОМПАНІЇ (ФРАГМЕНТ)

```
--
-- Table structure for table `it_client`
--

DROP TABLE IF EXISTS `it_client`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `it_client` (
  `enable` varbinary(1) DEFAULT 'y',
  `description` varbinary(255) DEFAULT NULL,
  `idcode` varbinary(40) DEFAULT '#####',
  `typ` varbinary(255) DEFAULT 'phys',
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `icon` varbinary(60) DEFAULT NULL,
  `abonentcategory` bigint(20) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `typ_idx` (`typ`)
) ENGINE=InnoDB AUTO_INCREMENT=16708 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `it_clientaccount`
--

DROP TABLE IF EXISTS `it_clientaccount`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `it_clientaccount` (
  `enable` varbinary(1) DEFAULT 'y',
  `description` varbinary(255) DEFAULT NULL,
  `service` mediumint(9) DEFAULT NULL,
  `peoplecount` mediumint(9) DEFAULT NULL,
```

```

`abonent` mediumint(9) DEFAULT NULL,
`id` mediumint(9) NOT NULL AUTO_INCREMENT,
`num` varbinary(255) DEFAULT NULL,
`percentage` double DEFAULT NULL,
`balance` double DEFAULT NULL,
`benefitarea` double DEFAULT NULL,
`benefitcount` mediumint(9) DEFAULT NULL,
`icon` varbinary(60) DEFAULT '',
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `it_clientaccount`
--

LOCK TABLES `it_clientaccount` WRITE;
/*!40000 ALTER TABLE `it_clientaccount` DISABLE KEYS */;
/*!40000 ALTER TABLE `it_clientaccount` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `it_clientcategory`
--

DROP TABLE IF EXISTS `it_clientcategory`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `it_clientcategory` (
  `enable` varbinary(1) DEFAULT 'y',
  `description` varbinary(255) DEFAULT NULL,
  `shortname` varbinary(100) DEFAULT NULL,
  `icon` varbinary(60) DEFAULT NULL,
  `typ` varbinary(255) DEFAULT 'phys',
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `name` blob,
  PRIMARY KEY (`id`),
  UNIQUE KEY `shortname` (`shortname`)
) ENGINE=InnoDB AUTO_INCREMENT=12 DEFAULT CHARSET=utf8;

```

```

/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `it_address`
--

DROP TABLE IF EXISTS `it_address`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `it_address` (
  `id` mediumint(9) NOT NULL AUTO_INCREMENT,
  `enable` varbinary(1) DEFAULT 'y',
  `description` varbinary(255) DEFAULT NULL,
  `icon` varbinary(60) DEFAULT '',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `it_address`
--

LOCK TABLES `it_address` WRITE;
/*!40000 ALTER TABLE `it_address` DISABLE KEYS */;
/*!40000 ALTER TABLE `it_address` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `it_apartment`
--

DROP TABLE IF EXISTS `it_apartment`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `it_apartment` (
  `building` bigint(20) DEFAULT NULL,
  `enable` varbinary(1) DEFAULT 'y',
  `description` varbinary(255) DEFAULT NULL,
  `floor` varbinary(255) DEFAULT NULL,

```



```
`apartmenttype` bigint(20) DEFAULT NULL,  
`roomcount` bigint(20) DEFAULT '0',  
`num` varbinary(50) DEFAULT NULL,  
`id` bigint(20) NOT NULL AUTO_INCREMENT,  
`icon` varbinary(60) DEFAULT NULL,  
`letter` varbinary(10) DEFAULT NULL,  
PRIMARY KEY (`id`),  
UNIQUE KEY `num_building_idx` (`num`,`building`)  
) ENGINE=InnoDB AUTO_INCREMENT=15834 DEFAULT CHARSET=utf8;  
/*!40101 SET character_set_client = @saved_cs_client */;
```

**ПРИКЛАД РЕАЛІЗАЦІЇ МОДЕЛЕЙ ДАНИХ З МЕХАНІЗМАМИ
АКТИВНОГО СХОВИЩА ДАНИХ ІНФОРМАЦІЙНОЇ СИСТЕМИ
(ФРАГМЕНТ)**

```
class ProxyBase(Proxy):
    def __init__(self, adopt=True):
        Proxy.__init__(self)
        self._adopt = adopt

    def _has_adopt(self, kb, cub, kp, cup):
        if not self._adopt:
            return 0.0
        return kb * cub + kp * cup

    def adopt(self, modelProxynome, kb, cub, kp, cup, limit=0.1):
        adopt_weight = self._has_adopt(kb, cub, kp, cup)
        if adopt_weight < limit:
            return
        modelProxyCls = getProxyByProxynome(modelProxynome)
        tmpModel = createTmpModel(modelProxynome)
        adoptModel(modelProxyCls, tmpModel, adopt_weight)
        restoreData(modelProxyCls, tmpModel)

class PhysPlacementProxy(ProxyBase):
    def __init__(self):
        ProxyBase.__init__(self, \
            Cluster((dsab.Placement & dsab.Abonent ^ "physabonent") & \
                dsab.PhysAbonent & dsab.Apartment & dsab.Building & \
                dsab.Street & dsab.Location, \
                None, lambda p, a, pa, apart, b, s, l: a.typ=="phys"))

class JurPlacementProxy(ProxyBase):
    def __init__(self):
        ProxyBase.__init__(self, \
            Cluster((dsab.Placement & dsab.Abonent ^ "jurabonent") & \
                dsab.JurAbonent & (dsab.JurAbonentApartment & \
                dsab.Apartment) & dsab.Building & dsab.Street & \
                dsab.Location, None, \
                lambda p, a, ja, jaborapart, apart, b, s, l: a.typ=="jur"))

    def save(self, modelname, vals):
        apartmentids = vals.pop('JurAbonentApartment.apartment')
```

```

        items = self._cluster.distribute_values(vals)[modelname]
        placement = self._models[modelname](**items).save()
        apartments = self.save_jurabonentapartments(placement.id, \
            apartmentids)
        if placement.is_saved():
            self.facade.sendNotification('%sIsUpdate' % modelname)
        else:
            self.facade.sendNotification('%sUpdateError' % modelname)
        return placement.id

def save_jurabonentapartments(self, placementid, apartmentids):
    apartments = []
    dsab.JurAbonentApartment.remove(placement=placementid)
    for apartmentid in apartmentids:
        apartments.append( \
            dsab.JurAbonentApartment(apartment=apartmentid, \
                placement=placementid).save())
    return apartments

def delete(self, modelname, id):
    dsab.JurAbonentApartment.remove(placement=id)
    super(JurPlacementProxy, self).delete(modelname, id)

def getVO(self, fields, filt=None, direction='top', order=None):
    return ProxyBase.getVO(self, fields, filt=filt, \
        direction=direction, order=order, distinct=True)

class PhysAbonentProxy(ProxyBase):
    def __init__(self):
        ProxyBase.__init__(self, \
            Cluster(dsab.Abonent & dsab.PhysAbonent, None))

    def save(self, modelname, vals):
        items = self._cluster.distribute_values(vals)[modelname]
        abon = self._models[modelname](**items).save()
        phys = self.save_phisabonent(abon.id, vals)
        if abon.is_saved() and phys.is_saved():
            self.facade.sendNotification('%sIsUpdate' % modelname)
        else:
            self.facade.sendNotification('%sUpdateError' % modelname)
        return abon.id

    def save_phisabonent(self, abonid, vals):
        vals['PhysAbonent.abonent'] = abonid
        items = self._cluster.distribute_values(vals)["PhysAbonent"]
        return self._models["PhysAbonent"](**items).save()

```

```

def delete(self, modelname, id):
    '''удаляет объект модели @modelname по @id'''
    o = self._models[modelname].get(id)
    try:
        o._get_reference().delete()
        o.delete()
        self.facade.sendNotification('%sIsDelete' % modelname)
    except:
        self.facade.sendNotification('%sDeleteError' % modelname, \
            o.require_related())

class JurAbonentProxy(ProxyBase):
    def __init__(self):
        ProxyBase.__init__(self, Cluster(dsab.Abonent & dsab.JurAbonent, \
            None))

    def save(self, modelname, vals):
        items = self._cluster.distribute_values(vals)[modelname]
        abon = self._models[modelname](**items).save()
        jur = self.save_jurabonent(abon.id, vals)
        if abon.is_saved() and jur.is_saved():
            self.facade.sendNotification('%sIsUpdate' % modelname)
        else:
            self.facade.sendNotification('%sUpdateError' % modelname)
        return abon.id

    def save_jurabonent(self, abonid, vals):
        vals['JurAbonent.abonent'] = abonid
        items = self._cluster.distribute_values(vals) ["JurAbonent"]
        return self._models["JurAbonent"](**items).save()

    def delete(self, modelname, id):
        '''удаляет объект модели @modelname по @id'''
        o = self._models[modelname].get(id)
        try:
            o._get_reference().delete()
            o.delete()
            self.facade.sendNotification('%sIsDelete' % modelname)
        except:
            self.facade.sendNotification('%sDeleteError' % modelname, \
                o.require_related())

```

**РЕАЛІЗАЦІЯ СПРОЩЕНОГО АЛГОРИТМУ ВИБОРУ
ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ВІДПОВІДНО ДО КРИТЕРІЇВ
ОЦІНКИ ВИМОГ ДО ІС**

```
def main(argv):  
    k = [0] * n # создание массива с n количеством элементов  
    c = [0] * n # создание массива с n количеством элементов  
    result = 0.0  
    choices = None  
    # заполнение массива ...  
    for i in len(n):  
        print "Enter c"  
        c_i = sys.readline()  
        c_i = float(c_i)  
        c[i] = c_i # значениями от 0 до n-1  
    # заполнение массива ...  
    for i in len(n):  
        print "Enter k"  
        k_i = sys.readline()  
        k_i = float(c_i)  
        k[i] = k_i # значениями от 0 до n-1  
  
    for c_i in c: # перебор всех элементов до заданного числа  
        for k_i in k:  
            f = target(k_i, c_i)  
            if f < result:  
                choices = (k_i, c_i)  
    print "k: ", choices[0]  
    print "c: ", choices[1]  
  
#import sys
```

```
if __name__ == "main":  
    main(sys.argv)
```