*Лекція № **8***
*ТЕЗИ*

# Архітектура клієнт-сервер. Сховища даних. Реляційні бази даних як основа сховищ даних

## Моделі організації обчислень та обробки інформації

Сьогодні використовуються фактично дві *основні* моделі огранізації обчислень та обробки інформації:
- модель ***централізованих обчислень***
- модель ***розподілених обчислень***

***Модель централізованих обчислень*** виникла і використовується з самого початку комерційного використання комп'ютерів з 1950-тих років.
Цю модель можна уявити собі як аплікацію, що працює на одному (або кількох не з'єднаних між собою) комп'ютерах.

***Модель розподілених обчислень*** виникає з широким використанням мережевих з'єднань — тобто у 1980-тих роках.
Цю модель можна уявити собі як аплікацію, що працює на кількох з'єднаних між собою комп'ютерах одночасно.

Якщо описати обидві моделі з точки зору *теорії систем*, то різниця між ними є в наступному:
- модель централізованих обчислень має лише один пристрій обробки і один пристрій збереження даних, хоча пристроїв вводу-виводу (терміналів) може мати і декілька.
- Модель розподілених обчислень х-зується вже як мінімум двома пристроями обробки і мін. двома пристроями збереження даних.

Хоча історично централізовані обчислення виконувалися набагато раніше за розподілені, на сьогоднішній день можна сказати, що модель розподілених обчислень має зараз більше поширення, ніж модель ц. обч.

Однією з найбільш поширених версій (варіантів) моделі розподілених обчислень є ***архітектура клієнт-сервер.***

## Що розуміють під архітектурою клієнт-сервер?
(далі — трохи матеріалу з Вікіпедії)

Архітектура клієнт-сервер є одним із ***архітектурних шаблонів програмного забезпечення*** .
Архітектура клієнт-сервер є домінуючую концепцією у створенні розподілених мережних застосувань і передбачає взаємодію та обмін даними між ними.

Вона передбачає такі основні компоненти:
- набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них (в найпростішому випадку — сервер один);

- набір клієнтів, які використовують сервіси, що надаються серверами;
- мережа, яка забезпечує взаємодію між клієнтами та серверами.

Сервери є незалежними один від одного.
Клієнти також функціонують паралельно і незалежно один від одного.
Немає жорсткої прив'язки клієнтів до серверів.
Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів;
з іншого боку, клієнт може звертатися то до одного сервера, то до іншого.
Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів.

**Клієнт та сервер**

Дуже важливо ясно уявляти, хто або що розглядається як «клієнт».
Можна говорити про ***клієнтський комп'ютер***, з якого відбувається звернення до інших комп'ютерів.
Можна говорити про ***клієнтське*** та серверне ***програмне забезпечення***.
Нарешті, можна говорити про ***людей***, які бажають за допомогою відповідного програмного та апаратного забезпечення отримати доступ до тієї чи іншої інформації.

Загальноприйнятим є положення, що клієнти та сервери – це перш за все програмні модулі.
Найчастіше вони знаходяться на різних комп'ютерах, але бувають ситуації, коли обидві програми – і клієнтська, і серверна, фізично розміщуються на одній машині; в такій ситуації сервер часто називається ***локальним сервером***.

**Обов'язки та взаємодія**

Центральним поняттям в архітектурі к-с. є поняття взаємодії .
Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером.

Логічно можна виокремити три рівні операцій:
- рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;
- прикладний рівень, який реалізує основну логіку застосування і на якому здійснюється необхідна обробка інформації;
- рівень управління даними, який забезпечує зберігання даних та доступ до них.

***Дворівнева клієнт-серверна архітектура*** передбачає взаємодію двох програмних модулів – клієнтського та серверного. В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

- ***модель тонкого клієнта***, в рамках якої вся логіка застосування та управління даними зосереджена на сервері.
  Клієнтська програма забезпечує тільки функції рівня представлення;
- ***модель товстого клієнта***, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта.

Тонкими клієнтами часто також називають пристрої з обмеженою потужністю: кишенькові комп'ютери, мобільні телефони та ін.

**Приклади**

Типовим прикладом клієнт-серверної взаємодії є WWW.
Існує величезна кількість веб-серверів, на яких розміщується та чи інша інформація. У найпростішому випадку ця інформація являє собою набір веб-сторінок, які можуть зберігатися на сервері у вигляді файлів, розмічених за допомогою мови розмітки HTML.

Для того, щоб людина, яка працює в Інтернет, могла переглянути ту чи іншу сторінку, на її комп'ютері повинно бути встановлено відповідне клієнтське програмне забезпечення. Програми для перегляду веб-сторінок називаються браузерами (веб-оглядачами).

Але, крім браузерів, до серверів можуть звертатися і інші клієнти, а саме – автономні програми. Вони можуть передбачати взаємодію з людиною, а можуть працювати в цілком автоматичному режимі. Типовим класом таких програм є роботи, призначені для автоматичного перегляду веб-ресурсів. Зокрема, роботи є важливим елементом пошукових систем і використовуються ними для перегляду сторінок і збору інформації про них.

Для запиту до веб-сервера клієнтська програма повинна задати місцезнаходження комп'ютера, на якому розміщується серверна програма, назву потрібного документа і, можливо, інші дані, які специфікують запит. Мережа забезпечує знаходження сервера і передачу йому клієнтського запиту. Серверні програми обробляють цей запит, відповідь пересилається по мережі клієнтові.

Інші приклади — сервіси електронної пошти, синхронізації часу, сервіси аплікацій (напр мережева версія 1с-бухгалтерії), складські програми і т.д.

**Переваги і недоліки тнкого/товстого клієнта**

Важливим фактором в к-с архітектурі є поняття ***часу доступу***, оскільки обмін даними (запитами) між пристроями збереження і обробки даних здійснюється постійно.
В залежності ***від характеру даних і роботи з ними*** може бути вигідним як тонкий так і товстий клієнт:
якщо даних небагато а робота з ними (обчислення) є досить ресурсоємкою, то краще використовувати товстого клієнта.
Наприклад обробка графіки з загального файлового сервера є недоцільною (час доступу може бути дуже великим в порівнянні з локальним)

Якщо ж дивитися з точки зору підтримки клієнтського ПЗ, то очевидно, що підтримувати один чи кілька серверів є вигідніше, ніж багато клієнтів. (напр. обробка документів по типу Google docs — серверне ПЗ — вигідніша, ніж підтримувати Word на кожній робочій станції).

Знов таки з міркувань ціни для малої кількості клієнтів серверне ПЗ може бути задорогим.

**Трирівнева клієнт-серверна архітектура**

***Трирівнева клієнт-серверна архітектура***, яка почала розвиватися з середини 90-х років, ***передбачає відділення прикладного рівня від управління даними***.

Виокремлюється окремий програмний рівень, на якому зосереджується прикладна логіка застосування.
Програми проміжного рівня можуть функціонувати під управлінням спеціальних серверів застосувань, але запуск таких програм може здійснюватися і під управлінням звичайного веб-сервера.
Нарешті, управління даними здійснюється сервером даних.

Наприклад
Клієнтський рівень:
Для роботи з системою користувач використовує стандартне програмне забезпечення – звичайний браузер. Це позбавляє його необхідності завантажувати та інсталювати спеціальні програми (хоча інколи така необхідність все-таки виникає).
Браузер може формувати запит та пересилає його до сервера.

Середній рівень (middleware) — рівень обробки запиту:
Сервер викликає серверні програмні модулі, які забезпечують обробку запиту і в разі потреби звертаються до сервера даних.

Рівень даних:
Сервер даних здійснює операції з даними, що зберігаються в системі та складають її інформаційну основу. Зокрема, він може здійснити вибірку з інформаційної бази відповідно до запиту та передати її модулю проміжного рівня для подальшої обробки.
Дані, з якими працює сервер даних, найчастіше організовані як реляційна база даних.

Найчастіше сервер даних і серверні модулі проміжного рівня розміщуються на одному комп'ютері, хоч і являють собою окремі і логічно незалежні програмні модулі.

НАМАЛЮВАТИ МАЛЮНОК

**Приклади трирівневої архітектури клієнт-сервер**

Найхарактерніший приклад знов таки з вебу — сучасна веб-аплікація не складається з статичних файлів, які нам (браузеру) повертає httpd-сервер. Сучасний веб-сайт має трирівневу архітектуру:
- клієнтський рівень (браузер)
- прикладний рівень (прогрмане забезпечення, модулі, які викликає httpd-сервер: PHP, Java, Perl і т.п.) відділено від
- рівня даних, які акумулюються як правило в базі даних.

Слід відмітити, що на сьогоднішній день технології розподіленої обробки даних розвиваються дуже і дуже активно.

**Сховища даних**

Завдяки глобальній комп'ютеризації сьогодні стало можливим накопичувати неймовірні за розмірами масиви даних. Порівняно низькою вартість зберігання цих даних, але виникає інша проблема:
самі по собі масиви даних не несуть корисної інформації, не маю вартості для особи, яка приймає рішення.

Дані

- часто зберігаються в різних форматах
- в різних місцях
- не є агреговані (напр. не об'єднані в часові ряди — відомо інфо про кожнен продаж але не відомо скільки продано товару за місяць)

Наприклад, який сенс в зберіганні банком даних по міліонах транзакцій, якщо неможливо відповісти напр. на питання: яка середня сума транзакції за певний період і як вона змінилася в порівнянні з попереднім таким же періодом.

***Для того, щоб дані мали вартість*** в сенсі основи для прийняття рішень, вони повинні пройти ***якісний і кількісний аналіз,*** тоді з них отримується ***інформація і знання.***

Створення сховища даних і є шляхом до перетворення даних в інформацію і знання.

Під ***сховищем даних розуміють централізований архів реорганізованих даних: предметно-орієнтованих, інтегрованих, розміщених із збереженням хронології і незмінних, які є основою для підтримки процесу прийняття рішень.***

Сховище даних являє собою багатовимірну базу даних, окрему від операційних баз (від баз тразакцій тобто).
Багатовимірність означає, що користувач має справу з "зрізами" даних, наприклад по продуктах, по районах, по продажах, по часу і по комбінаціях цих даних.

Отже сховища даних є:
- предметно-орієнтовані: опереують в термінах предметів, входів. Напр. по поставщиках, по продуктах, по клієнтах, по точках продажу і т.п
- інтегровані — тобто уніфіковані не зважаючи на різні джерела цих даних. Напр. стать задається символами "Ч" і "Ж" а не "чол", "male" і т.п. Це необхідно для наступної автоматизованої обробки даних, їх агрегації.
- Підтримують хронологію — організовані в часові ряди, дають можливість отримання часових агрегацій, НЕ орієнтовані на отримання тільки поточного статусу
- незмінні — дані в сховищі не коригуються, доступні тільки для читання.

***Основою сховищ даних як правило служать реляційні бази даних.***

Для відображення ж даних і роботи з ними використовуються маса різних технологій і програм. Дуже часно для кожної області використання вони специфічні.
Технології сховищ даних включають напр. ODBC, запити до баз даних, системи інтелектуального аналізу даних, оперативну аналітичну обробку (OLAP), web-технології і т.п.

Сховище даних необхідне, як правило, навіть для невеликих підприємств чи організацій.
***Сховище даних служить основою для прийняття бізнес-рішень.***

Слід зауважити, що проектування і побудова сховищ даних — складний і досить довготривалий процес.
На сьогодні технологічно фактично нереально зберігати всі дані без агрегування, тому ДУЖЕ важливо визначитися з максимальним набором можливих запитів і звітів по даних. ПОЯСНИТИ.

Також існує поняття про вітрини даних — сегменти даних разом з їх представленнями із сховища, які призанчені до використання певній групі людей, яка не використовує решту даних.

***Управління і експлуатація сучасних сховища даних передбачає:***

- доступ до будь-якої внутрішньої операційної системи даних, а також, напиклад через ODBC, до інших можливих джерел даних (напр. до даних фондової біржі, демографічних, даних агенції новин і т.п.)
- інтелектуальний аналіз даних — найважливіше для підтримки прийняття рішень, дані обробляються та агрегуються за різними алгоритмами і оцінюються згідно різних критеріїв оцінки
- візуалізація даних — технології OLAP (оперативна аналітична обробка даних), ГІС (гео-інформаційні системи), різноманітні діаграми, деревовидні представлення і т.п.
- Запити до баз даних, звіти і моніторинг
- веб-інтерфейс — найчастіше на сьогодні використовується як інтерфейсне рішення.

***Майбутні тенденції розвитку сховищ даних:***

- нові типи даних, що зберігаються і є об'єктами маніпулювання (напр. індексування, пошук по мультимедіа)
- міжкорпоративні сховища даних
- роль веб-технологій повинна тільки зростати і не лише для доступу але і для збору інформації
- об'єми даних будуть зростати

**Реляційні бази даних — основа сховищ даних**

Вище зазначалося, що як правило на сьогодні власне дані сховища зберігаються в реляційних базах даних.
Оскільки вони відіграють важливу роль в трирівневій архітектурі клієнт-сервер, то насправді роль реляційних баз даних на сьогодні важко переоцінити.

Концептуально теорія систем управління реляційними базами даних була розроблена в 1970-тих роках Едгаром Коддом, британським вченим, який працював в той час в дослідницьких лабораторіях IBM.

Значення реляційної моделі є сьогодні величезним. Теорія реляційних баз даних Кодда відноситься до найваливіших іновацій останніх 85-ти років (за версією журналу Forbes від 2002 року). Відомими є “12 првил Кодда”, які характеризують реляційну систему.

Основна ідея є дуже простою (все геніальне є простим, але не все просте є геніальним):

- дані зберігаються в таблицях (двомірних структурах), які складаються із стовбців (полів) та рядків (записів).
- Кожне поле може містити тільки один тип даних.
- Записи містять значення, що відповідають кожному із стовбців — це т.зв. кортеж, сукупність взаємозв'язаних даних.
- Один із стовбців (або комбінація з кількох стовбців) — первинний ключ (primary key) - однозначно ідентифікує кожен рядок з таблиці, забезпечує можливість відрізнити один рядок від іншого
- певним чином можна зв'язувати запис однієї таблиці із записом (записами) з інших.