Міністерство освіти України
Львівський державний університет імені Івана Франка

В.Д. Вовк, Б.М. Голуб, А.В. Дубовик

Технологія "клієнт/сервер" ч. І. Основні ідеї та положення

Методичні вказівки для студентів факультету прикладної математики та інформатики

Львів ЛДУ 1999

Рекомендовано до друку науково-методичною радою факультету прикладної математики та інформатики. Протокол N 13 від 24.12.98

Уклали: Володимир Дмитрович Вовк, Богдан Михайлович Голуб, Андрій Владиславович Дубовик

Відповідальний за випуск Г.А.Шинкаренко

Редактор М.В.Ріпей

Видавничий центр Львівського державного університету ім. І.Франка.
290602 Львів, вул. Університетська, 1.

Зміст

Вступ

Практика свідчить, що в цілому бази даних (БД) швидше за інші види інформації на електронних носіях повертають з прибутком затрачені на їх розробку кошти. Тому саме вони дали поштовх масовому впровадженню персональної обчислювальної техніки у виробничу сферу. Навіть невеликі підприємства завдяки відносній дешевизні персональних комп'ютерів отримали змогу створювати власні БД. Разом із нагромадженням у БД інформації з'являється можливість узагальнення (генералізації) даних у показники діяльності підприємства, через що доступ до інформації стає потрібним керівництву та іншим підрозділам підприємства. Таким чином через деякий час після створення БД "виростають" з розряду персональних у багатокористувацькі (корпоративні). Водночас загострюються проблеми, пов'язані з їх зберіганням та обробкою, а саме:

- ускладнюється підтримка цілісності та узгодженості окремих залежних частин інформації;
- збільшується кількість користувачів інформацією, що сприяє зростанню імовірності конфліктних звертань до одного блоку даних з боку декількох клієнтів;
- збільшуються вимоги до захисту, копіювання та архівування великих обсягів інформації.

Результатом багаторічної боротьби із вказаними труднощами є поява технології "клієнт/сервер", що поєднує у собі передові методи обробки великих обсягів інформації з новітніми досягненнями в галузі комп'ютерних мереж. Опис окремих елементів цієї технології міститься здебільшого в розрізнених виданнях, товщина і нечисленність яких спонукає програмістів віддавати перевагу англомовним HELP-ам.

Цим виданням автори започатковують серію з трьох методичних вказівок : ч. I. "Основні ідеї та положення", ч. II. "Сервери БД" та ч. III. "Розробка клієнтських застосувань". У кожній з них розглянуто окремі аспекти технології "клієнт/сервер", а разом вони, на наш погляд, містять достатньо інформації для ефективної розробки великомасштабних БД.

1. Розвиток технологій корпоративної обробки інформації

Швидкість роботи ЕОМ відразу після їх появи збільшувалась так стрімко, що невдовзі одна машина змогла обслуговувати багатьох користувачів одночасно. Проте вартість такої обчислювальної установки була основною перепоною для масового впровадження комп'ютерних технологій. Тому на противагу

багатокористувацькій надпотужній, але дорогій ЕОМ був створений значно дешевший персональний комп'ютер, потужностей якого цілком вистарчало для задоволення потреб індивідуального користувача. Але ейфорія одноосібного володіння ресурсами ПЕОМ швидко минала разом із нагромадженням обсягів інформації і зростанням необхідності одночасного доступу до неї багатьох користувачів. Це стало однією з причин появи і стрімкого розвитку комп'ютерних мереж. Паралельно розробляли програмне забезпечення, яке контролювало потоки даних у мережі. Розглянемо основні етапи його розвитку.

Одним з головних завдань комп'ютерної мережі є організація колективного доступу багатьох одночасно працюючих користувачів до окремих ресурсів їх комп'ютерів, таких як пристрої, файли та ін. Зрозуміло, що для цього крім технічних засобів необхідне відповідне програмне забезпечення, яке контролює вирішення конфліктних ситуацій при одночасних звертаннях користувачів до одного і того ж ресурсу. Обслуговуюча спеціалізація такого програмного забезпечення визначила його узагальнену назву – **сервер**. Відтепер програми, що співпрацюють у мережі, не можуть звертатись безпосередньо до необхідного ресурсу, а змушені робити відповідний запит **серверу.** Залежно від ситуації **сервер** або ж надасть потрібний ресурс програмі-**клієнту** (користувачеві), або ж поставить його вимогу в чергу, поки ресурс не звільниться.

Історично склалось так, що відразу після появи комп'ютерної мережі з'явився **сервер друку**. Це програмне забезпечення дає змогу кільком ПЕОМ сумісно використовувати один друкуючий пристрій. Невдовзі було розроблено програмне забезпечення **диск-сервера**, який реалізував ідею сумісного використання простору жорсткого диска. Ера серверів даних почалась з появи **сервера файлів**. Розділеним між користувачами ресурсом у цьому випадку є файл даних. З будь-якого комп'ютера в мережі сервер дає змогу "бачити" файли на дисках інших ПЕОМ. При цьому після підключення до певного файла користувач може бути впевненим, що для інших клієнтів на цей час доступ до цього файла заблокований.

Ідея наявності спеціалізованого програмного забезпечення, яке виконує неконфліктний розподіл ресурсів між конкуруючими клієнтськими програмами, що одночасно працюють у мережі, лежить в основі технології **клієнт/сервер**. Проте вище згадані сервери опікуються універсальними (незалежними від виду задачі) ресурсами (пристроями, файлами). А тому здебільшого входять до складу програмного забезпечення мереж або операційних систем, і окремо не виділяються. Загальне ж збільшення обсягів комп'ютерної інформації супроводжується її поділом на окремі сфери застосування, де специфіка даних конкретної сфери може бути врахована сервером (наприклад сфера БД). Спеціалізація даних визначила два основні напрямки подальшого розвитку технологій клієнт/сервер. Перший з них пов'язаний з концепцією так званої загальної електронної пам'яті. Його програмне забезпечення реалізовано у таких відомих продуктах, як Lotus Notes, CompuServ, особливо популярному у наш час

WWW та ін. Для детального розгляду стратегії загальної пам'яті пропонуємо / 1 /.

Другий напрямок пов'язаний з обслуговуванням надвеликих БД. Технологія файл-сервера на час роботи клієнта з конкретним файлом передбачає блокування доступу до останнього інших клієнтів. Якщо ж таким файлом є таблиця БД, то її повне блокування невиправдане, оскільки клієнти переважно звертаються до різних записів, а значить, могли б одночасно працювати з таблицею, не заважаючи один одному. Ця ідея лежить в основі **сервера баз даних**. Оскільки це програмне забезпечення надалі буде основним об'єктом вивчення, попередньо розглянемо історію його виникнення.

2. Системи керування базами даних

Програмне забезпечення перших БД розробляли, використовуючи алгоритмічні мови високого рівня (здебільшого COBOL). Реалізацію корпоративного режиму виконували на цій же основі безпосередні розробники програми. Вона вимагала надзвичайної кваліфікації і трудозатрат. Проте специфіка БД, що полягає у зберіганні інформації у чітко структурованій формі (наприклад у вигляді таблиць), дала змогу розробити (і математично обґрунтувати) спеціалізовані високоефективні алгоритми обробки даних. Оскільки ці алгоритми є універсальними для всіх БД, то їх виділили у окремий вид програмного забезпечення – системи керування базами даних (СКБД), звільнивши при цьому розробника від рутинного програмування процесів доступу до інформації на фізичних носіях. Програміст, викликаючи стандартні процедури СКБД, або включає їх у свій продукт як складову частину, або ж використовує СКБД як системну оболонку для роботи своїх програм. Таким чином доступ до даних виконували за допомогою запитів до вибраної СКБД, що реалізували необхідний алгоритм обробки інформації. Найбільш відомі СКБД тих часів - MVS, ADABAS, DB2, JES, IDMS та ін. У наступні версії цих продуктів почали вводити засоби організації багатокористувацького режиму. Зауважимо, що на сьогодні більше половини всієї інформації, яка зберігається у вигляді БД, створено з їх допомогою.

Перехід від великих ЕОМ (мейнфреймів) до ПЕОМ з подальшим об'єднанням останніх у мережу, звичайно ж, знайшов своє відображення в засобах і методах роботи з БД. Ідея персонального комп'ютера трансформувалась в ідею персональної БД. Так з'явились СКБД CLIPPER, FOXPRO, DBASE, PARADOX, ACCESS та ін. Ці системи мають потужні засоби сортування, побудови запитів, формування звітів та ін. Крім того, поява і швидке розповсюдження об'єктно-орієнтованих технологій візуального програмування на фоні загального переходу персональної обчислювальної техніки на графічний інтерфейс дали змогу вводити у СКБД зручний і ефективний інструментарій швидкої розробки прикладних

програмних продуктів. Впровадження комп'ютерних технологій у виробництво набуло масового характеру. Проте БД навіть невеликих підприємств швидко наповнювались, "переростаючи" категорію персональних. Поява комп'ютерних мереж зробила технічно можливим спільне використання інформації БД на ПЕОМ багатьма клієнтами. Розробники СКБД у нові версії своїх продуктів почали вводити спеціальні засоби організації корпоративного режиму експлуатації БД. Ці засоби визначають і підтримують "правила гри" конкурентного використання інформації. Так система BDE доступу до БД фірми BORLAND (тепер INPRISE) у випадку одночасної роботи кількох клієнтів з однією db-таблицею даних у каталозі її розміщення створює службовий файл PARADOX.LCK. У цьому файлі працюючі програми фіксують блокування доступу інших клієнтів до тих записів, з якими працюють у певний момент часу, а після використання ресурсу оголошують про відновлення загального доступу до нього. Подібні засоби мають також ODBC драйвери фірми MICROSOFT та ін.

У процесі подальшого нагромадження обсягів інформації у БД виявлено низку істотних недоліків цього підходу, а саме :

- під час експлуатації БД з особливо важливою інформацією не можна покладатись на послідовне і точне дотримання "правил гри" всіма клієнтськими застосуваннями;
- збої в роботі апаратури чи програмного забезпечення можуть призвести до псування інформації;
- відсутність єдиного розпорядника інформацією не дає змоги організувати необхідний захист даних від несанкціонованого доступу.

Вказані недоліки привели до висновку, що користувачі інформацією не повинні мати беспосереднього виходу на файли даних. Між ними має бути спеціалізоване програмне забезпечення, яке одноосібно розпоряджається інформацією і видає її в дозволеному обсязі і вигляді на вимогу програм-клієнтів. Це і є вище згаданий сервер БД, який завдяки узурпації керування всією інформацією має змогу виконувати ще ряд корисних функцій копіювання, відновлення та обробки даних.

Отримані практичні висновки про доцільність використання технології клієнт/сервер мають глибоке теоретичне підґрунтя. Нижче наведемо основні його елементи.

3. Об'єктний підхід

Відомо, що чим більшу і складнішу систему необхідно створити, тим вигідніше для цього застосовувати об'єктний підхід. Парадигму цього підходу можна сформулювати у вигляді простого й інтуїтивно зрозумілого правила: довільна складна система повинна складатись із сукупності простіших самостійно

функціонуючих підсистем, у яких кожна зі складових частин в здатна існувати незалежно від інших. Більш строге визначення об’єктного підходу містить чотири основних принципи: абстрагування, модульності, обмеження доступу та ієрархії. Дотримання цих принципів забезпечує ефективність розробки, стійкість і надійність у роботі та простоту супроводження великих систем / 2 /. Зокрема використання об’єктного підходу у програмуванні трансформує згадані принципи у принципи інкапсуляції, спадковості та поліморфізму, що визначають суть об’єктно-орієнтованого програмування.

Розглянемо завдання створення великої інформаційної системи, оминаючи етап розробки моделі даних, а взявши за мету вирішення задачі оптимальної організації процесів введення, обробки та зберігання великих обсягів інформації. Для аналізу цих процесів застосуємо принципи об’єктного підходу.

Принцип **абстрагування** вимагає вибирання з постановки задачі найбільш важливої інформації, що визначає суть задачі, і нехтування маловажними деталями. Тому надалі будемо оперувати поняттями такого рівня абстракції: дані, обробка, комп’ютер, мережа, програмне забезпечення та ін.

Принцип **модульності** передбачає розділення предметної області задачі на окремі компоненти з чітко окресленими внутрішніми та зовнішніми зв’язками. Для інформаційних систем це означає, що введення, зберігання та обробка даних мають бути функціонально відокремлені, тобто виконуватись спеціалізованим для кожного випадку програмним забезпеченням (клієнтом чи сервером даних). Якщо ж для доступу до даних може бути використана комп’ютерна мережа, то вказані процеси вигідно розділити ще й просторово, тобто виконувати на різних комп’ютерах. Сервер даних несе відповідальність за цілісність та захист інформації від несанкціонованого використання чи збоїв апаратури. У багатьох випадках на нього додатково покладаються адміністративні обов’язки копіювання та архівування даних, які передані йому на зберігання. Найчастіше сервер даних та інформацію, якою він опікується, розміщують на одному і тому ж комп’ютері, через що останній часто також називають **сервером**. Зазвичай один сервер даних обслуговує декілька (можливо навіть десятків) програм-клієнтів.

Застосування принципу **обмеження доступу** приводить до висновку, що доступ до інформації з боку клієнта повинен відбуватись тільки за допомогою засобів, що надаються сервером даних, і ніяким чином в обхід. Тобто для того, щоб прочитати чи записати інформацію в місце її зберігання, необхідно попередньо зв’язатися із сервером, назвати один або декілька паролів для демонстрації прав доступу до даних, сформулювати вимогу (запит) на читання/запис блоку інформації і дочекатися відповіді сервера. Оскільки сервер даних шифрує всю надану йому інформацію, ніякими іншими програмними засобами прочитати чи скорегувати дані неможливо (хіба що знищити їх цілком або вивести з ладу пристрій пам’яті). Всі внутрішні процеси запису, пошуку та обробки даних заховані всередині сервера і запускаються ним на вимогу клієнтів. Сервер даних гарантує, що :

- жодна інформація, яка порушує узгодженість та цілісність даних від клієнтів на виконання прийнята не буде;
- жоден клієнт без достатніх на те прав не скористується і не змінить інформацію;
- у випадку одночасного доступу до одного блоку інформації з боку декількох клієнтів буде збережена коректна почерговість внесення змін;
- завдяки внутрішній буферизації збої у роботі апаратури чи системного програмного забезпечення не призведуть до пошкодження інформації.

Нарешті принцип **ієрархії** вимагає ранжування модулів, які виконують обробку інформації. У великих інформаційних системах дані можуть знаходитись на декількох (деколи десятках) комп'ютерах у віданні різних серверів. Зрозуміло, що шар серверів має виключний пріоритет доступу до інформації. Ранг клієнта дає право зробити запит до сервера на блок даних і чекати на виконання замовлення. З цього погляду ідеологія "клієнт/сервер" є не чим іншим, як одним з можливих варіантів співпраці комп'ютерів у мережі.

4. Технологія роботи в архітектурі клієнт/сервер

З'ясувавши передумови появи, історію розвитку та обґрунтування основних ідей технології клієнт/сервер розглянемо конкретні механізми реалізації цієї технології.

Підпорядкована серверу інформація зберігається у файлах специфічного для сервера формату даних і при потребі шифрується для виключення можливості несанкціонованого доступу. Тому довільна клієнтська програма для того, щоб скористуватись цією інформацією, повинна зв'язатись із сервером даних, повідомити йому свої права на інформацію і подати запит на необхідний їй блок даних. Отримані від сервера дані певним чином обробляються, при потребі відображаються у зручному для користувача вигляді чи, якщо це необхідно, передаються серверу на зберігання. Звідси маємо такі важливі наслідки:

- БД - це файли, фізичний доступ до яких мають лише програми ядра вибраної СКБД;
- програмне забезпечення сервера БД на момент запиту інформації має резидентно функціонувати і бути постійно доступне для зв'язку в мережі;
- всі клієнтські програми незалежно від типу і виду власного програмного забезпечення під час звертання до сервера БД повинні користуватись єдиною мовою, зрозумілою цьому серверу;
- довільну програмну систему можна умовно розділити на три логічні компоненти: відображення даних, прикладні функції та доступ до інформації.

Кожний з цих пунктів відображає окрему важливу ланку взаємодії сервер – клієнт, тому нижче розглянемо їх детальніше.

4.1. База даних

На початках БД розглядали як великі сховища інформації, що містять набори елементів однакової структури (записи, кортежі), і тому допускають уніфікацію основних алгоритмів їх обробки, таких як пошук, читання/запис, внесення змін та ін. Тому основним завданням перших СКБД було досягнення максимальної ефективності виконання і зручності використання цих процедур. Згодом виявили, що для описання об'єктів реального світу потрібно зберігати не тільки набір їх характеристик, а й динамічно відслідковувати зв'язки між ними. Тобто інформація у БД повинна задовольняти певні обмеження і вона є взаємозалежною. Наприклад, вік людини має бути невід'ємним і меншим, скажімо, 200 років, а температура води у рідкому стані належить інтервалу [0-100] градусів за Цельсієм. Це приклади найпростіших кількісних обмежень. Програмна система має автоматично відслідковувати коректність інформації, що вноситься в БД. Прикладом більш складних причинно-наслідкових зв'язків може бути прийом на роботу нового працівника, що потребує не тільки необхідності внесення відділом кадрів даних про нього, а й зміни штатного розпису планово-фінансовим відділом, внесення працівника у відомість про зарплату у бухгалтерії та ін. Як інший приклад, зазначимо, що обсяг постачання сировини на завод не може перевищувати обсягу вільного місця на складах із врахуванням потужностей її переробки, термінів прибуття і зберігання, можливостей оплати і відвантаження, наявних запасів та ін. Іншими словами, інформація, що вноситься у БД, не може суперечити вже існуючій, або, як ще кажуть, повинна відповідати умовам цілісності даних. Але і це ще не все. Навіть вже занесена в БД інформація може залежати від часу та інших параметрів. Наприклад, на вклади у банку мають автоматично нараховуватись проценти, а у випадку досягнення зрілості особою слід переоформити від опікуна на неї нарахування дивідендів. Отже, БД повинна бути не просто статичним набором записів, а має додатково містити інформацію про обмеження і зв'язки між даними та набір процедур їх обробки. Цей висновок відображає принцип інкапсуляції об'єктно-орієнтованого підходу : дані і процедури невіддільні одне від одного і разом складають інформаційну модель об'єктів реального світу. Отже зберігання в БД разом із даними також правил їх обробки переводить цю БД з рангу моделі даних у якісно вищий ранг інформаційної моделі вибраної предметної області. Такі БД називають активними.

4.2. Сервер баз даних

Сервер БД – це звичайна (хоч, звісно, і складна) програма, яку запускаєть на виконання під керівництвом операційної системи. Найбільш відомі сервери – ORACLE (виробник - фірма Oracle), SYBASE (фірма Sybase), Microsoft SQL Server (Microsoft), INTERBASE (INPRICE) та ін. Більшість з них написані у кількох варіантах, для роботи в різних операційних системах (наприклад Oracle може функціонувати під керівництвом UNIX, WINDOWS, Macintosh). Від можливостей і ефективності роботи кожного сервера БД залежить і його ціна $300 - $50000. Найліпше співвідношення ціна/можливості суттєво залежить від завдання, яке вирішує БД, і разом з багатьма іншими параметрами є критерієм вибору користувачем конкретного сервера.

Отже для використання БД хоча б одним клієнтом програма сервера БД має бути запущена і резидентно функціонувати на комп'ютері, який через це також називають сервером. У більшості випадків нижче з контексту буде зрозуміло, який сервер мається на увазі: програмний чи комп'ютер. При двозначностях наводимо уточнення.

Основні завдання, які вирішує сервер БД, детально розглянемо у ч. II "Сервери БД". З метою збереження цілісності викладення матеріалу нижче коротко їх прокоментуємо.

4.2.1. Керування даними в оперативній та зовнішній пам'яті

Організація безпосереднього доступу до даних на фізичних носіях була основним завданням перших СКБД. Максимально ефективний формат зберігання даних, найшвидші алгоритми пошуку, обміну та обробки інформації і для сучасних СКБД залишаються серед першочергових проблем. А під час корпоративного доступу до даних ці показники стають критично важливими, адже сервер має встигати виконувати запити багатьох клієнтів. Як і для локальних СКБД, для серверів БД традиційним розв'язком задачі пришвидшення пошуку інформації в БД є організація системи ключів та індексних файлів. Оптимізації процесів обміну даними між оперативною (ОП) та зовнішньою пам'яттю досягають завдяки їх буферизації, тобто зберігання найбільш актуальної чи часто використовуваної інформації безпосередньо в ОП. При цьому, крім системної буферизації низького рівня, сервери БД за власними алгоритмами буферизують найбільш активні блоки інформації (часто цілі таблиці), що завдяки специфіці роботи з БД при достатньо великому обсязі оперативної пам'яті майже мінімізує затрати на обміни із зовнішніми пристроями. А успіхи мікроелектроніки у збільшенні обсягів ОП навіть привели до появи нових напрямків у розробці СКБД, які розраховані на розміщення в ОП усієї інформації БД.

4.2.2. Керування доступом до даних

Постійні і тимчасові обмеження сервером БД доступу клієнтів до інформації викликані двома причинами - необхідністю захисту від несанкціонованого

втручання в БД та потребою вирішення можливих конфліктів у випадках одночасних звертань різних клієнтів до одних і тих же блоків інформації.

4.2.2.1. Парольний захист

Доступ до інформаціїї у БД надається тільки зареєстрованим користувачам, тому сервер БД зберігає список їх імен. На початку сеансу зв'язку з сервером кожний клієнт вказує своє ім'я і верифікує його відповідним паролем. Зрозуміло, що і серед зареєстрованих користувачаів не всі є рівноправними щодо можливостей доступу до даних. Одні мають право лише зчитувати інформацію, інші – вносити або корегувати. При цьому певні клієнти мають доступ до однієї частини даних, інші – до іншої. Повні права доступу має лише адміністратор БД. За допомогою визначених команд до СКБД він розподіляє привілеї між рештою клієнтів. Найчастіше використовують два підходи до розмежування прав на інформацію між користувачами.

У першому підході для кожного користувача визначають набір об'єктів (таблиці, поля ...), до яких він має доступ, а також вказують рівень доступу (читання/корегування/знищення). Для більшої гнучкості такої системи окремим клієнтам (здебільшого тимчасово) може бути приписана спеціальна роль (role), яка дає змогу у визначених ситуаціях або отримувати додаткові права доступу до окремих частини інформаціїї, або скористуватись викликом додаткових процедур, що такий доступ мають. Окремі СКБД підтримують поняття груп користувачів, що спрощує адміністрування БД у частині контролю за правами доступу до даних.

Другий підхід ґрунтується на присвоєнні кожній частині інформації, що потрапляє у базу даних, позначки про певний рівень безпеки. Відповідно і користувачі системи отримують права доступу не до конкретних об'єктів БД, а до довільної інформації, що позначена не більшим, ніж визначеним рівнем безпеки. Отже, захищатись можуть не тільки цілі таблиці, або їх поля, а й навіть окремі записи. Тобто для одного користувача видимими будуть одні записи, а для іншого – інші. Такий підхід багаторівневої безпеки через більшу простоту адміністрування все частіше застосовують у сучасних комерційних СКБД.

4.2.2.2. Транзакції

Взаємозв'язаність різних частин інформації створює певні обмеження щодо характеру і порядку внесення змін у БД. Наприклад, вилучення запису про особу зі списку акціонерів підприємства зумовлює необхідність вилучення з іншого списку всіх його уповноважених осіб, перегляд і вилучення з відповідної таблиці належних йому цінних паперів, анулювання всіх записів про операції з цими паперами, корекцію рахунків інших акціонерів, внесення змін до багатьох журналів та ін. Невиконання програмою хоча б одної із вказаних операцій з корекцією рахунків приведе до повної дезорганізації інформації в усій БД. Проте, навіть якщо програма адекватно і відслідковує всі необхідні зв'язки, порушення

цілісності даних все ж може відбутись унаслідок збоїв у роботі комп'ютера (апаратури чи живлення) або операційної системи. Таким чином окремі операції з інформацією в БД мають бути об'єднані в групи за принципом: або виконуються всі операції в групі, або ж ні одна. В літературі такі групи операцій називаються **транзакціями**. Зрозуміло, що кількість і зміст операцій у кожній транзакції визначається умовами вихідної задачі і задаються програмістом.

Для захисту БД від можливих збоїв сервер БД під час виконання транзакції запам'ятовує (журналізує) ще деяку додаткову інформацію, необхідну для відновлення стану БД на момент початку виконання першої операції транзакції. З її допомогою у випадку збою після рестарту роботи програмної системи сервер БД анулює наслідки роботи тих операцій транзакції, які встигли спрацювати до аварії, тобто, виконує відкочування (RallBack) транзакції. Не претендуючи на точність технічного терміну далі цей процес називатимемо **відкатом**. При успішному відновленні БД "збійна" і всі наступні транзакції можуть бути знову запущені на виконання.

4.2.2.3. Розмежування транзакцій

Внесення змін у БД на основі транзакцій, а не поодиноких операцій, є ефективним засобом боротьби зі збоями програмного чи технічного забезпечення. Проте для сервера БД поняття транзакції отримує додатковий зміст. Спроба одного клієнта коректувати інформацію, що якось пов'язана з даними, з якими в цей момент працює інший клієнт, також може привести до втрати цілісності даних у БД. Отже, транзакція у цьому випадку є деякою найменшою допустимою одиницею активності клієнта, у виконання якої не має права втручатись інший користувач. За дотриманням цього стежить сервер БД, блокуючи доступ решти клієнтів до вже задіяних блоків інформації. Проте зрозуміло, що в кожен момент виконання транзакції неможливо наперед передбачити весь обсяг взаємозв'язаної інформації, яка має бути заблокована. З іншої сторони, надто жорстке блокування даних може заблокувати цілий ряд "невинних" (з огляду на цілісність БД) операцій з інформацією. Тому сервер БД повинен підтримувати декілька рівнів розмежування транзакцій. Наприклад, СКБД ORACLE підтримує три таких рівні:

- транзакція "бачить" незафіксовані зміни інших транзакцій;
- тільки завершені зміни інших транзакцій видимі;
- прочитані один раз транзакцією дані залишаться для неї такими ж, навіть якщо інші транзакції за цей час встигнуть ці дані змінити.

Отже, кожна вимога клієнта на виконання транзакції має містити додаткові вказівки серверу БД про рівень її розмежування з транзакціями інших клієнтів. Зрозуміло, що перший з наведених рівнів найменш "строгий" з погляду сумісного доступу до даних, проте найбільш "небезпечний" з огляду на цілісність інформації у БД. Вибір цього рівня підвищує єфективність сумісного використання інформації з БД, але, разом з тим, збільшує ризик конфліктів між транзакціями.

4.2.3. Журналізація змін в БД та відновлення БД після збоїв

У процесі виконання транзакції сервер БД буферизує певну додаткову інформацію на випадок збою програмного чи технічного забезпечення. Він записує цю інформацію у недоступний для користувачів журнал змін БД (файл), який з огляду на особливу важливість часто зберігають у двох екземплярах (по можливості на різних носіях). Зрозуміло, що запис у журнал випереджує початок роботи відповідних операцій транзакції. Деякі СКБД підтримують локальний журнал для кожної транзакції, проте у більшості випадків усі транзакції архівують у єдиний загальносистемний журнал. Аварійну зупинку роботи програм, що може бути відновлена відкатом та повторним виконанням останньої транзакції, називають м'яким збоєм. Такі збої виникають найчастіше і майже завжди легко ліквідуються сервером БД разом з відновленням його роботи. Втрата з тих чи інших причин деякої частини інформації з БД (жорсткий збій) робить неможливим зворотній відкат транзакцій У цьому випадку використовують архівну копію БД, що фіксує стан БД на момент початку ведення журналу. Починаючи з цього стану, повторно (послідовно згідно з журналом) виконуються усі транзакції, зокрема, аварійні. Зрозуміло, що це тривалий (і, на жаль, не завжди успішний) процес.

4.2.4. Підтримка програмних мов для реалізації запитів клієнтів

Корпоративний режим роботи з даними передбачає можливість одночасного спілкування з сервером БД не тільки декількох копій однієї програми на різних комп'ютерах, а й різних програмних продуктів, що створені в різних системах програмування. Тому, зрозуміло, що у процесі їх звертання до сервера БД повинна використовуватись єдина мова, яку він розуміє. Більше того, якби різні програмні сервери підтримували єдиний синтакс такої мови, то кожна клієнтська програма могла б майже без змін перемикатись на кожен з них. Очевидна користь від такої уніфікації привела до того, що для більшості серверів, які підтримують реляційні БД, стандартом такої мови de facto стала SQL – мова структурних запитів. Звичайно ж, це не означає, що клієнтські застосування мають бути написані саме на цій мові. Для внутрішньої обробки даних, організації інтерфейсу з користувачем та інших функцій кожна система програмування надає власні засоби розробки, проте для зв'язку із сервером БД вона зобов'язана підтримувати стандартний синтакс SQL, який розглянемо у ч. II "Сервери БД" .

4.3. Клієнтське програмне забезпечення

Для використання інформації, що знаходиться під опікою сервера БД, мають бути створені відповідні програми. Наприклад, автоматизоване робоче місце портьє готелю, програма обліку руху товарів у мережі магазинів, реєстрації цінних паперів та ін. Для розробки таких клієнтських застосувань кожна СКБД надає власні засоби. Наприклад, ORACLE як окрему компоненту містить систему Oracle

Developer. А фірма Borland розширила ядро СКБД Paradox - BDE бібліотекою інтерфейсних функцій, викликом яких довільні застосування в системі WINDOWS можуть отримувати дані з БД. Крім цього до послуг користувачів низка вже готових програмних систем, таких як DataBase Desktop (прямий доступ до таблиць БД db-формату), Paradox8 (COREL) – середовище візуального створення програм (мова ObjectPAL) та ін. У популярну систему DELPHI введено набір готових компонент доступу через BDE до таблиць db-формату. Більше того, BDE має набір власних драйверів доступу до найбільш популярних серверів БД, таких як ORACLE, SYBASE, INTERBASE та ін. Для зв'язку з тими СКБД, для яких BDE не має власних драйверів, вона може використовувати зовнішні ODBC драйвери різних фірм розробників. Отже, одна програма, створена, скажімо, в DELPHI може одночасно використовувати інформацію з різних серверів БД. Це дає змогу об'єднувати початково різні БД у єдину велику інформаційну систему. Теорію і приклади розробки клієнтського програмного забезпечення, що функціонує у середовищі "клієнт/сервер", викладено в ч. III "Розробка клієнтських застосувань".

4.4. Прикладні програми

Сервер БД забезпечує зберігання й ефективний доступ до даних на фізичних носіях, а клієнтські програми дають змогу ці дані відображати у зручному для користувача вигляді. Проте інформацію у БД необхідно не тільки зберігати, а й регулярно обробляти. Наприклад, у визначені терміни нараховувати проценти за вкладами у банку, після завершення сесії відчисляти частину студентів за неуспішність, контролювати відпускання товарів зі складу та ін. Наскільки інформація в БД моделює стан вибраної предметної області, настільки ж ці процедури моделюють процеси, що у ній відбуваються. Зрозуміло, що ці процеси відбуваються за наперед визначеними правилами. Наприклад, нарахування пені за прострочення термінів оплати кожне підприємство виконує на основі власної стратегії, а виплати із соціального страхування регламентують поточним законодавством. Оскільки виробнича сфера стала основним споживачем БД, то в літературі ці правила називають "**бізнес-правилами**". Для позначення набору основних процедур обробки інформації у БД надалі цей термін використовуватимемо без лапок. Хто ж повинен забезпечувати виконання бізнес-правил: сервер чи клієнтські програми? Це питання буде предметом дослідження у наступних параграфах.

5. Архітектура застосувань у системі "клієнт/сервер"

Проектування систем спільного користування має дві складові: фізичну і логічну. У випадку фізичного проектування програмних систем розглядають

комп’ютери, диски, мережі та інші технічні засоби з метою створення цілісної надійної технічної бази з оптимальним співвідношенням ціна/продуктивність.

Логічне проектування призначене для розробки архітектури (структури) програмного забезпечення, яка дає змогу найефективніше розв’язати вихідну задачу при максимальному використанні технічних ресурсів. Основними критеріями ефективності такої структури є вартість розробки, надійність роботи та простота супроводу створеного програмного продукту. Все сказане у наступних параграфах стосуватиметься логічної архітектури застосувань.

5.1. Двошарова модель

Як ми уже зазначали, довільну програмну систему можна умовно розділити на три логічні компоненти: відображення даних, прикладні функції та доступ до інформації. Програмне забезпечення першої компоненти реалізує інтерфейс між людиною та системою, тобто надає користувачу засоби введення, корегування, пошуку та відображення інформації, що зберігається у системі. Це функції клієнтської частини загальної системи. Третя компонента складається з програм, які реалізують беспосередній доступ до інформації на фізичних носіях (баз даних, файлів та ін). Цю роботу виконує сервер даних. Відкритим ми залишили питання про приналежність прикладної компоненти, додатковою специфікою якої є те, що виконання деяких із прикладних процедур має ініціюватись не користувачем, а ситуацією, що склалась у БД. Прикладами такої ситуації може бути вичерпання запасу деталей на складі до критичного мінімуму або настання терміну нарахування процентів з кредитів. Програмна система має автоматично у потрібний момент запустити потрібні процедури на виконання. Залежно від розміщення прикладної копоненти у застосуванні та способу її взаємодії з іншими компонентами розрізняють декілька підходів до розробки загальної структури застосування.

Перший підхід передбачає, що БД міститься у файлах, доступ до яких розділяється файловим сервером операційної системи, через що його називають FS-моделлю (File Server). При цьому на кожному коп’ютері-клієнті виконується як клієнтська частина програми, так і копія ядра СКБД. Для одержання даних прикладна програма звертається до власної системи доступу до баз даних (СДБД), таких як BDE, ODBC та ін. СДБД ж по мережі зв’язується з ядром, що функціонує на комп’ютері-сервері і отримує потрібний блок даних у вигляді файла. Зобразимо це схематично:

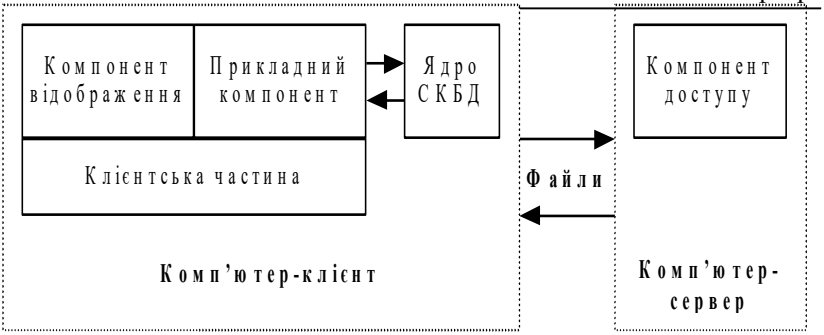


Схема взаємодії компонент програмної системи у FS-моделі

Зрозуміло, що обмін даними на рівні файлів приводить до значного завантаження мережі, допускає вузький спектр операцій з даними і забезпечує захист інформації не більше ніж на рівні файлової системи.

На відміну від FS-моделі в іншому підході спеціалізований сервер БД запускається лише на комп'ютері-сервері, де він (а не операційна система) обслуговує зв'язок з клієнтами по мережі (на рівні SQL запитів) і забезпечує захист інформації від несанкціонованого втручання, апаратних або програмних збоїв. При цьому для зв'язку з клієнтськими програмами може використовуватись або ядро СКБД (наприклад BDE), або відповідні драйвери (наприклад ODBC). Клієнтська частина може бути виконана за допомогою довільних програмних засобів, що підтримують мову SQL спілкування із сервером БД.

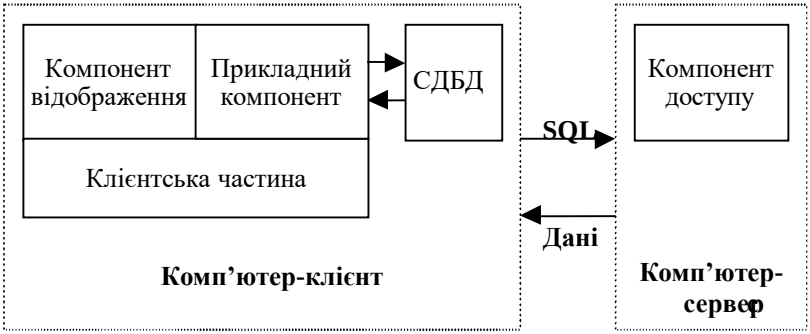


Схема взаємодії компонент програмної системи у RDA-моделі

Така модель організації програмних систем називається RDA – Remote Data Access (віддалений доступ до даних) і є найбільш пропулярною у наш час.

Оскільки в результаті виконання SQL-запиту клієнтові найчастіше повертається лише невеликий блок даних, зрозуміло, що завантаження мережі у порівнянні з FS-моделлю є істотно меншим. Проте виконання прикладної компоненти на комп’ютері-клієнті все ж ще вимагає активного обміну інформацією з сервером. Іншим недоліком розміщення прикладної компоненти у клієнтській частині застосування є можливість неоднозначного трактування бізнес-правил різними клієнтськими програмами. Більше того, для внесення змін у ці правила (наприклад, у зв’язку зі зміною законодавства) необхідно переробляти всі пов’язані з ними клієнтські програми.

Згадані недоліки RDA-моделі відсутні у DBS-моделі, у якій реалізація бізнес-правил виконується засобами сервера БД. Для цього він підтримує спеціальне розширення мови SQL, що називається механізмом збережених процедур. Визначений цим розширенням набір SQL-операторів дає змогу запрограмувати виконання SQL-сервером алгоритмів прикладної компоненти безпосередньо на комп’ютері-сервері. Таким чином сервер БД зберігає у БД не тільки інформацію, а й процедури її обробки (основа об’єктного підходу), залишаючи клієнтам лише представницькі функції введення і відображення даних. Порівняємо схему RDA-моделі з наведеною нижче DBS-схемою.

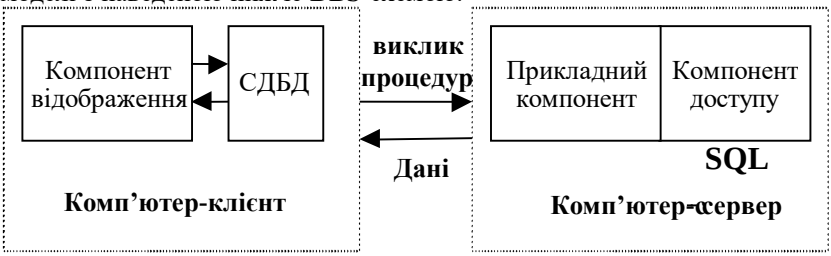


Схема взаємодії компонент програмної системи у DBS-моделі

Переваги:

- базові алгоритми, що визначають діяльність підприємства, однаково трактуються усіма користувачами і тому легко налаштовуються на поточну стратегію чи зміну законодавства;
- єдина для всіх клієнтів реалізація бізнес-правил дає змогу уникнути дублювання програмного коду прикладних процедур;
- виконання бізнес-правил відбувається на тому ж комп’ютері, де інформація зберігається, а по мережі передаються лише назви виконуваних процедур без переміщення значних обсягів даних.

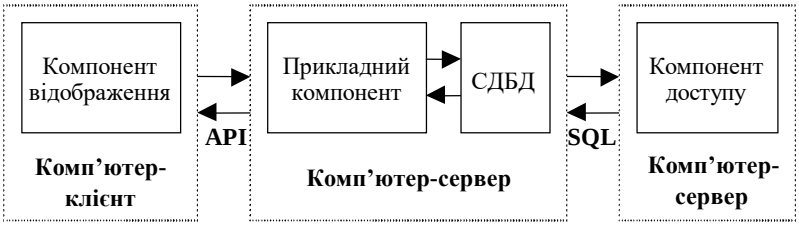
Недоліки:

- набір операторів SQL-розширення механізму збережених процедур далекий по можливостям і зручності від традиційних засобів програмування (C++, Object Pascal та ін.);
- реалізація прикладної компоненти сильно залежить від вибраної СКБД.

Зрозуміло, що на практиці реальні програмні системи найчастіше є деякими гібридами вище наведених моделей. Наприклад, частина найбільш загальних бізнес-правил реалізують на сервері, а часткові логічно складні правила виконують у клієнтських програмах. Пошук найбільш доцільних для цього завдання співвідношень між клієнтською і серверною частинами є серйозним випробуванням майстерності і досвіду розробників.

5.2. Тришарова модель

Двошарова модель застосування передбачає, що прикладна компонента реалізується або в клієнтському програмному забезпеченні, або ж інтегрується в роботу сервера БД. Проте є ще один варіант архітектури застосувань, схема якого зображена нижче.



AS - модель сервера застосувань

Тут прикладна компонента системи реалізована як окремий шар, який у літературі називається сервером застосувань (Application Server – AS). Клієнтські програми (Application Client – AC) звертаються до нього за виконанням окремих прикладних задач над БД. А AS з допомогою SQL отримує дані від сервера БД. AS - модель є основою для спеціального виду програмного забезпечення - моніторів транзакцій (Transaction Processing Monitor - TPM), які реалізують перехід від функціонування БД у локальних мережах до БД у розподілених системах глобальних мереж.

6. Розподілені системи

Перехід до архітектури “клієнт/сервер” забезпечив просте і відносно дешеве розв’язання проблеми колективного доступу до інформації в межах локальної мережі. Проте довільна система, рано чи пізно, виходить за свої межі і потребує внесення кардинальних змін в ідеологію підтримки свого функціонування. Вихід розподілу даних і їх обробки за межі локальної мережі переводить технологію “клієнт/сервер” у ранг лише першого наближення до більш загальних технологій обробки даних у глобальних мережах.

Наведемо гіпотетичний приклад розподіленої БД у дії. Ректор університету підписує наказ на купівлю у деякій торговій організації партії комп'ютерів. Відповідальний бухгалтер на автоматизованому робочому місці виконує процедуру перерахування необхідної суми грошей з рахунку університету на рахунок торгівельної організації. Сервер БД бухгалтерії університету посилає відповідний запит банківському серверу, який, перевіривши наявність необхідної суми, зв'язується із сервером БД іншого банку, де знаходиться рахунок торгівельної організації. Серверу бухгалтерії сповіщають про успішне виконання процедур оплати. Далі сервер торгової організації зв'язується з сервером її складу для подачі запиту на відпускання оплаченої партії комп'ютерів. Останній вносить у складську БД необхідні зміни і друкує відповідні документи. Додатково інформація про покупку передається у БД матеріального відділу бухгалтерії університету для виконання необхідних процедур прийому комп'ютерів на балансовий облік.

З цього прикладу видно, що в операції купівлі комп'ютерів задіяно кілька серверів (можливо навіть різних СКБД), кожен з яких підтримує власну БД. Зрозуміло, що збій у довільній ланці цього складного процесу відкатом транзакцій окремих серверів БД не ліквідується. Повинен бути додатковий механізм координації роботи серверів, транзакціями якого були б визначені пакети транзакцій в окремих БД. Серед шляхів розв'язання такої проблеми відомий протокол (набір угод з процедур і комунікацій) двофазної фіксації, який гарантує, що всі транзакції на всіх ділянках або успішно завершаться, або ніде не будуть застосовані до БД. Постачальники найбільш потужних СКБД (ORACLE, IBM, DEC та ін.) вводять у свої системи програмне забезпечення таких координаторів транзакцій. Проте останні, як правило, налаштовані на конкретну специфіку власної БД. Існують також універсальні за типом сервера БД координатори транзакцій високого рівня – Tuxedo (USL), Top End (NCR), Encina(Transarc) та ін.

Іншим підходом до реалізації розподілених систем є використання механізму віддаленого виклику процедур (Remote Procedure Call –RPC). Суть його полягає у можливості одного комп'ютера ініціалізувати виконання деякої підпрограми на іншому комп'ютері і отримати результати цього виконання. Отже, довільний сервер БД може керувати запуском потрібних процедур на інших серверах, тобто контролювати розподілену обробку даних. На сьогодні RPC–технологія є в основі стратегії побудови більшості розподілених систем. Проте розробка нових, більш ефективних і надійних координаторів транзакцій на основі двофазної фіксації швидко прогресує.

Список літератури

1. Васкевич Д. Стратегии клиент/сервер. Руководство по выживанию для специалистов по реорганизации бизнеса - Киев: Диалектика, 1996. - 384 с.

2. Буч Г. Объектно-ориентированное проектирование с примерами применения - М.: Конкорд, 1992. – 519 с.

3. Ладыженский Г. Системы управления базами данных – коротко о главном // Системи управления базами данных. – 1995. № 1-4.

4. Кузнецов С. Введение в системы управления базами данных // Системи управления базами данных. – № 1/1995 - 6/1996.