Основні поняття архітектури клієнт-сервер.

При розміщенні бази даних (БД) на персональному комп'ютері, який не перебуває у мережі, БД завжди використовується в монопольному режимі. Навіть якщо БД використовують кілька користувачів, вони можуть працювати з нею тільки послідовно, і тому питань про підтримання коректної модифікації БД в цьому випадку тут не варто, вони вирішуються організаційними заходами - тобто визначенням необхідної послідовності роботи конкретних користувачів з відповідною БД. Однак навіть у деяких настільних БД потрібно враховувати послідовність зміни даних при обробці, щоб отримати коректний результат: так, наприклад, при запуску програми балансного бухгалтерського звіту всі бухгалтерські проводки - фінансові операції мають бути вирішені заздалегідь до запуску кінцевого додатки.

Однак робота на ізольованому комп'ютері з невеликою базою даних зараз стає вже нехарактерною для більшості додатків. БД відображає інформаційну модель реальної предметної області, вона росте за обсягом і різко збільшується кількість завдань, що вирішуються з її використанням, і відповідно з цим збільшується кількість додатків, що працюють з єдиною базою даних. Комп'ютери об'єднуються в локальні мережі, і необхідність розподілу додатків, що працюють з єдиною базою даних по мережі, є безсумнівною.

Дійсно, навіть коли ви будуєте БД для невеликої торгової фірми, у вас з'являється ряд специфічних пользователейБД, які мають свої бізнес-функції і територіально можуть перебувати в різних приміщеннях, але всі вони повинні працювати з єдиною інформаційною моделлю організації, тобто з єдиною базою даних .

Паралельний доступ до однієї БД декількох користувачів, у тому випадку якщо БД розташована на одній машині, відповідає режиму розподіленого доступу до централізованої БД. (Такі системи називаються системами розподіленої обробки даних.)

Якщо ж БД розподілена по декількох комп'ютерів, розташованим в мережі, і до неї можливий паралельний доступнесколькіх користувачів, то ми маємо справу з паралельним доступом до розподіленої БД. Подібні системи називаютсясістемамі розподілених баз даних. У загальному випадку режими використання БД можна представити в наступному вигляді (см.рис.1).



Рис. 1. Режими роботи з базою даних

Визначимо термінологію, яка нам буде потрібно для подальшої роботи. Частина термінів нам вже відома, але повторимо тут їх додатково.

Термінологія

Користувач БД - програма або людина, яка звертається до БД на ЯМД.

Запит - процес звернення користувача до БД з метою введення, отримання або зміни інформації в БД.

Транзакція - послідовність операцій модифікації даних в БД, яка переводить БД з одного несуперечливого стану в інший несуперечливе стан.

Логічна структура БД - визначення БД на фізично незалежному рівні, найближче відповідає концептуальної моделі БД.

Топологія БД = Структура розподіленої БД - схема розподілу фізичної БД по мережі.

Локальна автономність - означає, що інформація локальної БД і пов'язані з нею визначення даних належать локальному власнику і їм управляються.

Віддалений запит - запит, який виконується з використанням модемного зв'язку.

Можливість реалізації вилученої транзакції обробка однієї транзакції, що складається з безлічі SQL-запитів на одному віддаленому вузлі.

Підтримка розподіленої транзакції допускає обробку транзакції, що з кількох запитів SQL, які виконуються на декількох вузлах мережі (віддалених або локальних), але кожен запит в цьому випадку обробляється тільки на одному вузлі, тобто запити не є розподіленими. При обробці однієї розподіленої транзакції різні локальні запити можуть оброблятися в різних вузлах мережі.

Розподілений запит - запит, при обробці якого використовуються дані з БД, розташовані в різних вузлах мережі.

Системи розподіленої обробки даних в основному пов'язані з першим поколінням БД, які будувалися на мультипрограмних операційних системах і використовували централізоване зберігання БД на пристроях зовнішньої памятіцентральной ЕОМ і термінальний багатокористувацький режим доступу до неї. При цьому користувальницькі термінали не мали власних ресурсів - тобто процесорів і пам'яті, які могли б використовуватися для зберігання і обробки даних. Першою повністю реляційної системою, що працює в многопользовательском режимі, була СУБД SYSTEM R, розроблена фірмою IBM, саме в ній були реалізовані як мова маніпулювання даними SQL, так і основні принципи синхронізації, що застосовуються при розподіленої обробки даних, які досі є базисними практично у всіх комерційних СУБД.

Загальна тенденція руху від окремих mainframe-систем до відкритих розподілених систем, об'єднуючим комп'ютери середнього класу, отримала назву DownSizing. Цей процес зробив величезний вплив на розвиток архітектур СУБД і поставив перед їх розробниками ряд складних завдань. Головна проблема полягала в технологічної складності переходу від централізованого управління даними на одному комп'ютері і СУБД, що використала власні моделі, формати представлення даних і мови доступу до даних і т. Д., До розподіленої обробки даних в неоднорідній обчислювальної середовищі, що складається із з'єднаних в глобальну мережу комп'ютерів різних моделей та виробників.

У той же час відбувався зустрічний процес - UpSizing. Бурхливий розвиток персональних комп'ютерів, поява локальних мереж також зробили серйозний вплив на еволюцію СУБД. Високі темпи зростання продуктивності і функціональних можливостей PC привернули увагу розробників професійних СУБД, що призвело до їх активного поширення на платформі настільних систем.

Сьогодні взяла гору тенденція створення інформаційних систем на такій платформі, яка точно відповідала б її масштабами і завданням. Вона отримала назву RightSizing (приміщення рівно в той розмір, який необхідний).

Однак і в даний час великі ЕОМ зберігаються і співіснують із сучасними відкритими системами. Причина цього проста - свого часу в апаратне і програмне забезпечення великих ЕОМ були вкладені величезні кошти: в результаті багато хто продовжує їх використовувати, незважаючи на морально застарілу архітектуру. У той же час перенесення даних і програм з великих ЕОМ на комп'ютери нового покоління сам по собі представляє складну технічну проблему і вимагає значних витрат.

Моделі "клієнт-сервер" в технології баз даних

Обчислювальна модель "клієнт-сервер" початково пов'язана з парадигмою відкритих систем, яка з'явилася в 90-х роках і швидко еволюціонувала. Сам термін "клієнт-сервер" початково застосовувався до архітектури програмного забезпечення, яке описувало розподіл процесу виконання за принципом взаємодії двох програмних процесів, один з яких у цій моделі називався "клієнтом", а інший - "сервером". Клієнтський процес запитував деякі послуги, а серверний процес забезпечував їх виконання. При цьому передбачалося, що один серверний процес може обслужити безліч клієнтських процесів.

Раніше додаток (користувача програма) не розділялася на частини, воно виконувалося деяким монолітним блоком. Але виникла ідея більш раціонального використання ресурсів мережі. Дійсно, при монолітному виконанні використовуються ресурси тільки одного комп'ютера, а інші комп'ютери в мережі розглядаються як термінали. Але тепер, на відміну від епохи main-фреймів, всі комп'ютери в мережі володіють власними ресурсами, і розумно так розподілити навантаження на них, щоб максимальним чином використовувати їхні ресурси.

І як в промисловості, тут виникає древня як світ ідея розподілу обов'язків, поділу праці. Конвеєри Форда зробили свого часу прорив в автомобільній промисловості, показавши найвищу продуктивність праці саме через те, що весь процес складання був розбитий на дрібні і максимально прості операції і кожен робочий спеціалізувався на виконанні тільки однієї операції, але цю операцію він виконував максимально швидко і якісно.

Звичайно, в обчислювальній техніці не можна було безпосередньо використовувати технологію автомобільного чи іншого механічного виробництва, але ідею використовувати було можна. Однак для втілення ідеї необхідно було розробити модель розбиття єдиного монолітного додатки на окремі частини і визначити принципи взаємозв'язку між цими частинами.

Основний принцип технології "клієнт-сервер" стосовно до технології баз даних полягає в поділі функцій стандартного інтерактивного додатку на 5 груп, що мають різну природу:

1. функції введення і відображення даних (Presentation Logic);
2. прикладні функції, що визначають основні алгоритми вирішення завдань додатків (Business Logic);
3. функції обробки даних в середині програми (Database Logic);
4. функції управління інформаційними ресурсами (Database Manager System);
5. службові функції, які відіграють роль зв'язоків між функціями перших чотирьох груп.

Структура типового додатку, що працює з базою даних наведена на рисунку



Рис. 2. Структура типового інтерактивного додатка, що працює з базою даних

Презентаційна логіка (Presentation Logic) як частина програми визначається тим, що користувач бачить на своєму екрані, коли працює додаток. Сюди відносяться всі інтерфейсні екранні форми, які користувач бачить або заповнює в ході роботи програми, до цієї ж частини відноситься все те, що виводиться користувачеві на екран як результати вирішення деяких проміжних завдань або як довідкова інформація. Тому основними завданнями презентаційної логіки є:

• формування екранних зображень;

• читання і запис в екранні форми інформації;

• управління екраном;

• обробка рухів миші і натиснення клавіш клавіатури.

Деякі можливості для організації презентаційної логіки додатків надає знако-орієнтований інтерфейс користувача, що задається моделями CCIS (Customer Control Information System) і IMS / DC фірми IBM і моделлю TSO (Time Sharing Option) для централізованої main-фреймовой архітектури. Модель GUI - графічного інтерфейсу користувача, підтримується в операційних середовищах Microsoft's Windows, Windows NT, в OS / 2 Presentation Manager, X-Windows іOSF / Motif.

Бізнес-логіка, або логіка власне додатків (Business processing Logic), - це частина коду програми, яка визначає власне алгоритми вирішення конкретних завдань програми. Зазвичай цей код пишеться з використанням різних мов програмування, таких як C, C ++, Cobol, SmallTalk, Visual-Basic.

Логіка обробки даних (Data manipulation Logic) - це частина коду програми, яка пов'язана з обробкою даних всередині програми. Даними управляє власне СУБД (DBMS). Для забезпечення доступу до даних використовуються мова запитів і засоби маніпулювання даними стандартної мови SQL.

Зазвичай оператори мови SQL вбудовуються в мови 3-го або 4-го покоління (3GL, 4GL), які використовуються для написання коду програми.

Процесор управління даними (Database Manager System Processing) - це власне СУБД, яка забезпечує зберігання і управління базами даних. В ідеалі функції СУБД повинні бути приховані від бізнес-логіки додатка, однак для розгляду архітектури додатку нам треба їх виділити в окрему частину програми.

У централізованій архітектурі (Host-based processing) ці частини програми розташовуються в єдиному середовищі і комбінуються всередині однієї виконуваної програми.

У децентралізованій архітектурі ці завдання можуть бути по-різному розподілені між серверним і клієнтським процесами. Залежно від характеру розподілу можна виділити наступні моделі розподілів (див. Рис. 10.3):

• розподілена презентація (Distribution presentation, DP);

• віддалена презентація (Remote Presentation, RP);

• розподілена бізнес-логіка (Distributed Business Logic, DBL);

• розподілене управління даними (Distributed data management, DDM);

• віддалене управління даними (Remote data management, RDM).



Рис. 3. Розподіл функцій програми в моделях "клієнт-сервер"

Це умовна класифікація показивет, як можуть бути розподілені окремі завдання між серверним і кліенскімі процесами. У цій класифікації відсутня реалізація віддаленої бізнес-логіки. Дійсно, вважається, що вона не може бути видалена сама по собі повністю. Вважається, що вона може бути розподілена між різними процесами, які взагалі-то можуть виконуватися на різних платформах, але повинні коректно кооперуватися (взаємодіяти) один з одним