Лекція 16. Моделі серверів для роботи з базами даних.

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

• функції введення і відображення даних (Presentation Logic);

• прикладні функції, що визначають основні алгоритми вирішення завдань додатків (Business Logic);

• функції обробки даних в середині програми (Database Logic);

• функції управління інформаційними ресурсами (Database Manager System);

• службові функції, які відіграють роль зв'язок між функціями перших чотирьох груп.

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



Рис. 1. Структура типового інтерактивного додатка, що працює з базою даних

**Дворівневі моделі**

Дворівнева модель фактично є результатом розподілу п'яти зазначених функцій між двома процесами, які виконуються на двох платформах: на клієнті і на сервері. У чистому вигляді майже ніяка модель не існує, проте розглянемо найбільш характерні особливості кожної дворівневої моделі.

Модель віддаленого управління даними. Модель файлового серверу

Модель віддаленого управління даними також називається моделлю файлового серверу (File Server, FS). У цій моделі презентаційна логіка і бізнес-логіка розташовуються на клієнті. На сервері розташовуються файли з даними і підтримується доступ до файлів. Функції управління інформаційними ресурсами в цій моделі знаходяться на клієнті.

Розподіл функцій в цій моделі представлено на рис. 2.

У цій моделі файли бази даних зберігаються на сервері, клієнт звертається до сервера з файловими командами, а механізм управління всіма інформаційними ресурсами, власне база мета-даних, знаходиться на клієнті.



Рис. 2. Модель файлового серверу

Переваги цієї моделі в тому, що ми вже маємо поділ монопольного програмного процесу на два взаємодіючих процеси. При цьому сервер (серверний процес) може обслуговувати безліч клієнтів, які звертаються до нього із запитами. Власне СУБД повинна знаходитися в цій моделі на клієнті.

Який алгоритм виконання запиту клієнта?

Запит клієнта формулюється в командах ЯМД. СУБД переводить цей запит в послідовність файлових команд. Кожна файлова команда викликає перекачування блоку інформації на клієнта, далі на клієнті СУБД аналізує отриману інформацію, і якщо в отриманому блоці не міститься відповідь на запит, то приймається рішення про перекачування наступного блоку інформації і т. д.

Перекачування інформації з сервера на клієнт проводиться до тих пір, поки не буде отримана відповідь на запит клієнта.

Недоліки:

• високий мережевий трафік, який пов'язаний з передачею по мережі безлічі блоків і файлів, необхідних додатком;

• вузький спектр операцій маніпулювання з даними, який визначається тільки файловими командами;

• відсутність адекватних засобів безпеки доступу до даних (захист тільки на рівні файлової системи).

**Модель віддаленого доступу до даних**

У моделі віддаленого доступу (Remote Data Access, RDA) база даних зберігається на сервері. На сервері же знаходиться ядро ​​СУБД. На клієнті розташовується презентаційна логіка і бізнес-логіка програми. Клієнт звертається до сервера із запитами на мові SQL. Структура моделі віддаленого доступу наведена на наступному рисунку.



Рис. 3. Модель віддаленого доступу до даних

Переваги даної моделі:

• перенесення компонента уявлення та прикладного компонента на клієнтський комп'ютер істотно розвантажив сервер БД, зводячи до мінімуму загальне число процесів в операційній системі;

• сервер БД звільняється від невластивих йому функцій; процесор або процесори сервера цілком завантажуються операціями обробки даних, запитів і транзакцій. (Це стає можливим, якщо відмовитися від терміналів, що не надто ресурсами, і замінити їх комп'ютерами, які виконують роль клієнтських станцій, які володіють власними локальними обчислювальними ресурсами);

• різко зменшується завантаження мережі, оскільки по ній від клієнтів до сервера передаються не запити на введення-виведення в файлової термінології, а запити на SQL, і їх обсяг істотно менше. У відповідь на запити клієнт отримує тільки дані, релевантні запиту, а не блоки файлів, як в FS-моделі.

Основна перевага RDA-моделі - уніфікація інтерфейсу "клієнт-сервер", стандартом при спілкуванні програми-клієнта і сервера стає мова SQL.

Недоліки:

• все-таки запити на мові SQL при інтенсивній роботі клієнтських додатків можуть істотно завантажити мережу;

• так як в цій моделі на клієнті розташовується і презентаційна логіка, і бізнес-логіка програми, то при повторенні аналогічних функцій у різних додатках код відповідної бізнес-логіки повинен бути повторений для кожного клієнтського додатку. Це викликає зайве дублювання коду додатків;

• сервер в цій моделі грає пасивну роль, тому функції управління інформаційними ресурсами повинні виконуватися на клієнті. Дійсно, наприклад, якщо нам необхідно виконувати контроль страхових запасів товарів на складі, то кожен додаток, яке пов'язане зі зміною стану складу, після виконання операцій модифікації даних, що імітують продаж або видалення товару зі складу, повинна виконувати перевірку на обсяг залишку, і в разі , якщо він менше страхового запасу, формувати відповідну заявку на поставку необхідного товару. Це ускладнює клієнтський додаток, з одного боку, а з іншого - може викликати необгрунтоване замовлення додаткових товарів декількома додатками.

**Модель сервера баз даних**

Для того щоб позбутися недоліків моделі віддаленого доступу, повинні бути дотримані наступні умови:

1. Необхідно, щоб БД в кожен момент відображала поточний стан предметної області, який визначається не тільки даними, але і зв'язками між об'єктами даних. Тобто дані, які зберігаються в БД, в кожен момент часу повинні бути несуперечливими.

2. БД повинна відображати деякі правила предметної області, закони, за якими вона функціонує (business rules). Наприклад, завод може нормально працювати тільки в тому випадку, якщо на складі є деякий достатній запас (страховий запас) деталей певної номенклатури, деталь може бути запущена у виробництво тільки в тому випадку, якщо на складі є в наявності достатньо матеріалу для її виготовлення, і т. д.

3. Необхідний постійний контроль за станом БД, відстеження всіх змін і адекватна реакція на них: наприклад, при досягненні деяким вимірюваним параметром критичного значення має відбутися відключення певної апаратури, при зменшенні товарного запасу нижче допустимої норми має бути сформована заявка конкретного постачальника на поставку відповідного товару .

4. Необхідно, щоб виникнення деякої ситуації в БД чітко і оперативно впливало на хід виконання прикладної задачі.

5. Однією з найважливіших проблем СУБД є контроль типів даних. На даний момент СУБД контролює синтаксично тільки стандартно-допустимі типи даних, тобто такі, які визначені в DDL (data definition language) - мові опису даних, який є частиною SQL. Однак у реальних предметних областях у нас діють дані, які несуть в собі ще й семантичну складову, наприклад, це координати об'єктів або одиниці різних метрик, наприклад робочий тиждень на відміну від реальної має відразу після п'ятниці понеділок.

Дану модель підтримують більшість сучасних СУБД: Informix, Ingres, Sybase, Oracle, MS SQL Server. Основу даної моделі становить механізм збережених процедур як засіб програмування SQL-сервера, механізм тригерів як механізм відстеження поточного стану інформаційного сховища і механізм обмежень на користувача типи даних, який іноді називається механізмом підтримки доменної структури. Модель сервера баз даних представлена ​​на рис. 4.



Рис. 4. Модель активного сервера БД

У цій моделі бізнес-логіка розділена між клієнтом і сервером. На сервері бізнес-логіка реалізована у вигляді збережених процедур - спеціальних програмних модулів, які зберігаються в БД і управляються безпосередньо СУБД. Клієнтський додаток звертається до сервера з командою запуску збереженої процедури, а сервер виконує цю процедуру і реєструє всі зміни в БД, які в ній передбачені. Сервер повертає клієнту дані, релевантні його запиту, які потрібні клієнтові або для виводу на екран, або для виконання частини бізнес-логіки, яка розташована на клієнті. Трафік обміну інформацією між клієнтом і сервером різко зменшується.

Централізований контроль в моделі серверу баз даних виконується з використанням механізму тригерів. Тригери також є частиною БД.

Термін "тригер" взято з електроніки і семантично дуже точно характеризує механізм відстеження спеціальних подій, які пов'язані зі станом БД. Тригер в БД є як би деяким тумблером, який спрацьовує при виникненні певної події в БД. Ядро СУБД проводить моніторинг всіх подій, які викликають створені і описані тригери в БД, і при виникненні відповідної події сервер запускає відповідний тригер. Кожен тригер являє собою також деяку програму, яка виконується над базою даних. Тригери можуть викликати збережені процедури.

Механізм використання тригерів припускає, що при спрацьовуванні одного тригера можуть виникнути події, які викличуть спрацьовування інших тригерів. Цей потужний інструмент вимагає тонкого та узгодженого застосування, щоб не вийшов нескінченний цикл спрацьовування тригерів.

У даній моделі сервер є активним, тому що не тільки клієнт, але і сам сервер, використовуючи механізм тригерів, може бути ініціатором обробки даних в БД.

І збережені процедури, і тригери зберігаються в словнику БД, вони можуть бути використані декількома клієнтами, що істотно зменшує дублювання алгоритмів обробки даних у різних клієнтських додатках.

Для написання збережених процедур і тригерів використовується розширення стандартної мови SQL, так званий вбудований SQL. Вбудований SQL ми розглянемо в "Вбудований SQL".

Недоліком даної моделі є дуже велике завантаження сервера. Дійсно, сервер обслуговує безліч клієнтів і виконує такі функції:

• здійснює моніторинг подій, пов'язаних з описаними тригерами;

• забезпечує автоматичне спрацьовування тригерів при виникненні пов'язаних з ними подій;

• забезпечує виконання внутрішньої програми кожного тригера;

• запускає збережені процедури по запитах користувачів;

• запускає збережені процедури з тригерів;

• повертає необхідні дані клієнта;

• забезпечує всі функції СУБД: доступ до даних, контроль і підтримку цілісності даних в БД, контроль доступу, забезпечення коректної паралельної роботи всіх користувачів з єдиною БД.

Якщо ми переклали на сервер більшу частину бізнес-логіки додатків, то вимоги до клієнтів у цій моделі різко зменшуються. Іноді таку модель називають моделлю з "тонким клієнтом", на відміну від попередніх моделей, де на клієнта покладалися набагато більш серйозні завдання. Ці моделі називаються моделями з "товстим клієнтом".

Для розвантаження сервера була запропонована **трирівнева** модель.

Модель сервера додатків

Ця модель є розширенням дворівневої моделі і в ній вводиться додатковий проміжний рівень між клієнтом і сервером. Архітектура трирівневої моделі наведена на рис. 5. Цей проміжний рівень містить один або кілька серверів додатків.



Рис. 5. Модель сервера додатків

У цій моделі компоненти програми діляться між трьома виконавцями:

• Клієнт забезпечує логіку уявлення, включаючи графічний користувальницький інтерфейс, локальні редактори; клієнт може запускати локальний код додатку клієнта, який може містити звернення до локальної БД, розташованої на комп'ютері-клієнті. Клієнт виконує комунікаційні функції front-end частини програми, які забезпечують доступ клієнту в локальну або глобальну мережу. Додатково реалізація взаємодії між клієнтом і сервером може включати в себе управління розподіленими транзакціями, що відповідає тим випадкам, коли клієнт також є клієнтом менеджера розподілених транзакцій.

• Сервери додатків складають новий проміжний рівень архітектури. Вони спроектовані як виконання загальних незагружаемую функцій для клієнтів. Сервери додатків підтримують функції клієнтів як частин взаємодіючих робочих груп, підтримують мережеву доменну операційну середу, зберігають і виконують найбільш загальні правила бізнес-логіки, підтримують каталоги з даними, забезпечують обмін повідомленнями і підтримку запитів, особливо в розподілених транзакціях.

• Сервери баз даних у цій моделі займаються виключно функціями СУБД: забезпечують функції створення та ведення БД, підтримують цілісність реляційної БД, забезпечують функції сховищ даних (warehouse services). Крім того, на них покладаються функції створення резервних копій БД і відновлення БД після збоїв, управління виконанням транзакцій і підтримки застарілих (успадкованих) додатків (legacy application).

Відзначимо, що ця модель має більшу гнучкість, ніж дворівневі моделі. Найбільш помітні переваги моделі сервера додатків в тих випадках, коли клієнти виконують складні аналітичні розрахунки над базою даних, які відносяться до області OLAP-додатків. (On-line analytical processing.) У цій моделі велика частина бізнес-логіки клієнта ізольована від можливостей вбудованого SQL, реалізованого в конкретній СУБД, і може бути виконана на стандартних мовах програмування, таких як C, C ++, SmallTalk, Cobol. Це підвищує переносимість системи, її масштабованість.

Функції проміжних серверів можуть бути в цій моделі розподілені в рамках глобальних транзакцій шляхом підтримки XA-протоколу (X / Open transaction interface protocol), який підтримується більшістю постачальників СУБД.