«ОРГАНІЗАЦІЯ ТА УПРАВЛІННЯ БАЗАМИ ДАНИХ»

РОЗПОДІЛЕНІ БД

Презентація супроводження лекційного курсу

проф. В.І.Кривенко

Основні питання.

1. ПОНЯТТЯ РОЗПОДІЛЕНОЇ БД
2. СТРАТЕГІЇ РОЗПОДІЛУ ДАНИХ
3. ТЕХНОЛОГІЇ РОБОТИ З РОЗПОДІЛЕНОЮ БАЗОЮ ДАНИХ
4. ТРАНСАКЦІЇ ТА МЕХАНІЗМ ЇХ ПІДТРИМКИ

ПОНЯТТЯ РОЗПОДІЛЕНОЇ БАЗИ ДАНИХ

Розподілена база даних (DDB — distributed database) — це сукупність взаємопов'язаних баз даних, розподілених у комп'ютерній мережі.

Система управління розподіленою базою даних визначається як програмна система, яка управляє базою даних у такий спосіб, щоб її розподіленість була прозорою для користувачів.

Прозорість — це досить поширене поняття незалежності даних у розподілених системах, яке передбачає, що користувач у цій системі працює з розподіленою базою даних як з логічно цілісною сукупністю даних, тобто на його роботу не повинно впливати те, як дані розподілені між вузлами мережі. Отже, в розподіленій системі користувачеві надається логічно цілісне подання фізично розподіленої бази даних.

Основною задачею розподіленої СУБД є забезпечення управління доступом до даних багатьох споживачів і цілісності й узгодженості даних в умовах використання мережі комп’ютерів.

Тобто основна функція таких СУБД – це координування спільної роботи багатьох користувачів з розподіленою інформацією.

Розв'язання проблеми автономності роботи користувачів розподіленої системи створює багато специфічних проблем в організації баз даних, оскільки різні користувачі можуть працювати паралельно з одними й тими самими даними, виконуючи з ними різні перетворення.

СТРАТЕГІЇ РОЗПОДІЛУ ДАНИХ

Можливі стратегії розподілення даних у РБД:

- централізована стратегія;
- розподілена стратегія без дублювання;
- розподілена стратегія з дублюванням;
- мішана, або комбінована стратегія.

Централізована стратегія характеризується тим, що всі дані розміщуються в одному вузлі мережі і є система управління доступу різних користувачів з інших вузлів до даних.

Переваги централізованої стратегії.

Якщо дані зберігаються в одному місці, то значно простіше реалізувати проблему забезпечення цілісності та захисту інформації.

При централізованій стратегії спрощується технологія створення та ведення файлів БД, оскільки можна скористатися єдиними стандартними процедурами та методами ведення і підтримування БД в актуальному стані.

Проектування такої розподіленої бази даних також досить просте порівняно з іншими стратегіями.

Недоліки.

За такої стратегії можуть виникати черги, що призводить до різкого збільшення часу реакції системи.

Крім того, витрачається певний час і на процедури пов'язані з передаванням інформації.

Обсяг бази даних обмежений пам'яттю однієї ЕОМ для зберігання даних.

Розподілена (децентралізована) стратегія без дублювання. За такої стратегії визначають дані, які потрібно зберігати в кожному вузлі мережі. При цьому розподілену базу даних проектують як неперетинні між собою підмножини даних, розподілені по вузлах мережі.

Ключовим фактором, який впливає на надійність і доступність бази даних, є так звана **локалізація посилань**.

Якщо база даних розподілена так, що дані, які розміщені в цьому вузлі, викликаються винятково його користувачем, то це свідчить про **високий ступінь локалізації посилань**.

Якщо подібне розчленування даних здійснити неможливо і для виконання запитів користувача потрібно звертатись за інформацією до інших вузлів, то це свідчить про **невисокий ступінь локалізації посилань**.

Розглянута стратегія підходить для тих предметних областей, в яких практично немає дублювання даних у різних вузлах мережі і потрібна мінімальна кількість логічних посилань для виконання інформаційних взаємозв'язків вузлів одного з одним. Тобто користувач кожного вузла працює зі своїми файлами і досить рідко використовує дані інших вузлів мережі.

Переваги **розподіленої (децентралізованої) стратегії без дублювання** полягають у тому, що зменшуються витрати на передавання інформації та вірогідність виникнення черг, коли кілька користувачів одночасно звертаються до одного і того самого файлу БД.

Недоліки:

Водночас, цю стратегію важко контролювати з точки зору дублювання даних, чим ускладнюється реалізація проблеми узгодженості та цілісностні даних.

Значно складнішими є проблеми адміністрування та підтримування БД даних в актуальному стані.

Розподілена (децентралізована) стратегія з дублюванням полягає в тому, що база даних проектується як за централізованого підходу, але фізично дублюється в кожному вузлі мережі. Кожний вузол має свою копію, продубльовану стільки разів, скільки вузлів у мережі.

Стратегія розподілу з дублюванням найбільш ефективно розв'язує проблеми доступу та вибірки даних з мінімальними витратами часу.

Система досить проста при проектуванні. Однак нарівні з перевагами цей підхід характеризується складністю адміністрування та розв'язання проблеми узгодженості файлів БД у різних вузлах мережі. Ця проблема узгодженості досить гостро може постати тоді, коли зв'язок у мережі порушується і в копії в різних вузлах виникають розбіжності. У цьому разі потрібно розробити спеціальний механізм для узгодження деяких копій бази даних.

Мішана стратегія розподілу даних поєднує два підходи, пов'язані з розподілом без дублювання та з дублюванням даних, з метою використання їх переваг.

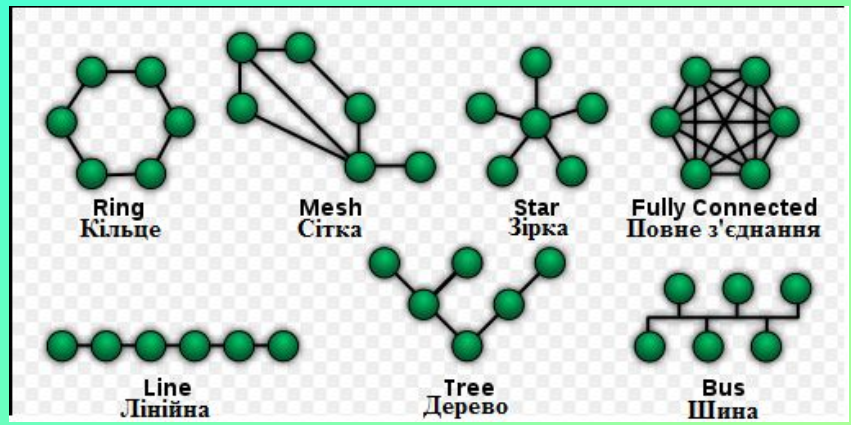
Ця стратегія поділяє базу даних на багато логічних фрагментів, як це зроблено в стратегії розподілу без дублювання. Крім того, вона повинна дозволяти мати довільну кількість фізичних копій кожного фрагмента.

Такий підхід до створення розподіленої бази даних дає змогу дублювати дані довільну кількість разів і в кожному вузлі; водночас у кожному вузлі може міститися потрібна частина бази даних. Система, побудована за цією стратегією, допускає досить просту реалізацію паралельної обробки даних, що скорочує час відгуку системи. Ця стратегія також забезпечує дуже велику надійність даних, за рахунок дублювання даних їх легко можна відновити при помилках чи збоях обладнання. Однак, як і при стратегії дублювання, виникає проблема узгодженості копій бази даних у всіх вузлах мережі.

ТЕХНОЛОГІЇ РОБОТИ З РОЗПОДІЛЕНОЮ БД

Розглянемо особливості технології роботи з розподіленою базою даних в умовах використання локальної обчислювальної мережі.

Локальна обчислювальна мережа (LAN – Local Area Network) — це кілька зв'язаних між собою комп'ютерів, абонентів мережі, які спільно використовують дані з файлів бази даних. За своєю роллю в мережі комп'ютери поділяються на дві групи. До першої групи відносять комп'ютери, які надають для колективного користування абонентам мережі свій вінчестер, на якому розміщена база даних. Ці комп'ютери називаються **файловими серверами**. Комп'ютери другої групи не допускають до своїх даних інших абонентів, але можуть використовувати файли, розміщені на файлових серверах. Ці комп'ютери називаються **робочими станціями**. Робоча станція може не мати свого вінчестера і навіть пристрою для дискет, тоді єдиним джерелом даних для неї є файлові сервери.



Комп'ютерна мережа може мати складну конфігурацію (**топологію**). Додатково в мережу можуть входити зовнішні пристрої, які використовуються колективно, наприклад сітковий принтер.

При роботі в мережі необхідна коректна робота з системою трасування трансакцій (Transaction Tracking System – TTS). TTS – це компонент операційної системи для роботи в мережі.

TTS виконує функції відстежування трансакцій, які полягають у тому, що в журналі трансакцій відмічається початок і кінець кожної трансакції, а також у деяких випадках виконується відкат незавершених трансакцій.

На початку трансакції її відмічають у журналі - якщо трансакція пов'язана із внесенням змін у БД, то оригінал запам'ятовується в цьому журналі.

При успішному завершенні трансакції журнал очищується; якщо спостерігається аварійне завершення, то виконується відкат, і оригінал БД відновлюється.

Відкат може виконуватись у таких випадках:

- якщо вимкнеться робоча станція, то TTS втратить зв'язок зі станцією і виконає автоматично відкат;
- якщо вимкнеться файл-сервер, це фіксується і при наступному його ввімкненні виконується відкат незавершених трансакцій.

Основна проблема при роботі в мережі полягає в узгодженості роботи з файлами одразу кількох абонентів.

Наприклад, не можна допускати, щоб один абонент вносив зміни до файлу, тоді як інший абонент використовує його для виконання певних розрахунків.

У мережі дозволяється так званий режим розподілу, коли кілька користувачів можуть відкрити один і той самий файл для роботи з ним. Але цей режим роботи з файлами не завжди прийнятний, тому що можуть виникнути конфлікти при використанні одних і тих самих даних різними прикладними програмами.

Тому для цих випадків передбачено **режим монопольного використання**. Якщо файл відкритий в режимі монопольного доступу, то він закритий для доступу інших програм. Цей режим називається **режимом повного блокування**. Блокувати можна або весь файл, або окремі його записи. При роботі з розподіленими даними, аби запобігти виникненню конфліктів при доступі до ресурсів і гарантувати повне виконання трансакцій, необхідно користуватись такими правилами:

- блокувати одразу всі ресурси, необхідні для виконання трансакцій;
- якщо деякий ресурс не можна захопити, то необхідно відмовитись від виконання трансакції.

Існує багато альтернатив обробки даних у мережі. Є дві технології, які використовуються при обробці в мережі: **клієнт–сервер** і **файл–сервер**. В обох цих технологіях розподілена база даних зберігається на сервері, а з клієнтських машин здійснюють доступ до даних, які зберігаються на сервері.

Проте ці технології мають суттєву відмінність, яка полягає в тому, що у **файл–серверній технології** на сервері розміщується лише програмне забезпечення, яке підтримує роботу мережі, та файли бази даних, а все прикладне програмне забезпечення знаходиться на комп'ютерах користувачів, тобто на робочих станціях.

Особливості **технології клієнт–сервер** полягають у тому, що прикладне програмне забезпечення зберігається не лише на робочій станції, певні його компоненти можуть зберігатися на сервері.

Найбільш популярною зараз є архітектура клієнт–сервер, і більшість сучасних розподілених СУБД орієнтовані саме на цю технологію розподіленої обробки даних.

Розподілені системи, які підтримують **архітектуру клієнт–сервер**, можна ще визначити як системи типів:

- *багато-клієнтів/один-сервер;*
- *багато-клієнтів/багато-серверів.*

У системах першого типу (*багато-клієнтів/один-сервер*) управління базою даних централізоване і виконується досить просто, оскільки вся база зберігається на одному сервері. У таких системах управління доступом до даних клієнтів зводиться до управління процесами блокування та управління буферами клієнтів.

У системах другого типу (*багато-клієнтів/багато-серверів*) база даних може розміщуватися на кількох серверах і для виконання запитів користувачів потрібна взаємодія серверів одного з одним. Кожна клієнтська машина має свій сервер і на нього направляє запити користувача – нібито працює з централізованою базою. Взаємодії серверів при виконанні запиту прозорі для користувача. У більшості існуючих СУБД реалізовано один із двох типів архітектури **клієнт–сервер**.

Розподілені СУБД не повинні розрізняти клієнтські та серверні машини. В ідеальному варіанті кожний вузол мережі може виступати залежно від ситуації як клієнт або як сервер.

Така архітектура визначається як *рівний-до-рівного.*

Але поки що немає СУБД, які б підтримували цю архітектуру, бо досить складно на програмному рівні реалізувати розподіл даних по багатьох вузлах, а це призводить до ускладнення протоколів управління даними.

УПРАВЛІННЯ ОДНОЧАСНИМ ДОСТУПОМ ДО ДАНИХ

Робота з БД у мережі створює проблему простежування за зміною значень атрибутів, яке виконується одним користувачем у процесі здійснення операцій іншим користувачем. Якщо виконуються складні запити, пов'язані з пошуком по багатьох файлах БД, користувач може почати операції пошуку при одному стані бази даних, а закінчити зовсім при іншому. Це може виникнути в результаті внесення змін у файли бази даних іншими користувачами.

Якщо не виконувати певних дій при досить динамічних з змінах БД кількома користувачами, то виявиться неможливим формування коректних звітів на такій базі даних через неузгоджений зміст її файлів.

Наприклад, при паралельному внесенні змін в один і той самий запис один користувач може стерти зміни іншого користувача.

Для вирішення таких колізій використовується **блокування** або інакше – **накладання замків (захватів)**. Блокування виконуються за правилами сумісності блокувань, що виключає виникнення конфліктів типу читання–запис, запис–читання або запис–запис. Існують три методи блокування: *централізоване блокування*, *блокування первинної копії і розподілене блокування* .

При ***централізованому блокуванні*** для всієї розподіленої бази даних підтримується єдина таблиця блокувань, яка розмішується на одному з вузлів під управлінням єдиного менеджера блокувань.

Менеджер блокувань відповідає за встановлення та зняття захватів від імені всіх трансакцій. Оскільки управління блокуванням зосереджено на одному з вузлів, то воно аналогічно централізованому управлінню одночасним доступом до даних.

Недоліки:

По-перше, недостатня надійність, яка при відмові чи недоступності центрального вузла блокувань може призвести до виходу з ладу всієї системи.

По-друге, центральний вузол може стати вузьким місцем у системі через великі обсяги обробки даних на ньому і через генерованість навколо нього інтенсивного трафіку мережі.

Блокування первинної копії – це управління одночасним доступом, що використовується для баз даних з **реплікаціями**, в яких копії одних і тих самих даних можуть зберігатися на багатьох вузлах.

Одна з таких копій виділяється як *первинна*, і для доступу до будь-якого елемента даних необхідно встановити блокування на його первинну копію.

Множина первинних копій відома всім вузлам розподіленої системи, і запити на блокування надходять на ті вузли, де зберігаються ці копії.

Якщо в розподіленій базі даних не використовуються реплікації, то механізм блокування первинної копії зводиться до розподіленого блокування.

Розподілене (децентралізоване) блокування пропонує розподілити обов'язки з управління блокуванням між усіма вузлами системи.

Під час блокування згідно з цим механізмом необхідна координація взаємодій менеджерів блокувань кількох вузлів.

Цей підхід до блокування усуває недоліки централізованого блокування, пов'язані з перевантаженням централізованого вузла, проте він складніший і характеризується більш високими комунікаційними витратами.

Блокувати можна весь файл, окремий його запис чи групу записів. При блокуванні система повинна інформувати інших користувачів по мережі про те, що в даний момент виконано блокування.

Проте наявність замка не повинна заважати виконанню операцій пошуку у файлах, їх перегляду або доступу до отриманих результатів. Замок має захищати лише від паралельного внесення змін різкими користувачами.

Загальний побічний ефект усіх розглянутих алгоритмів управління паралельним доступом за допомогою блокувань – це можливість виникнення тупікових ситуацій. ***Тупік, глухий кут*** – це така ситуація, коло багато запитів створюють цикл, очікуючи зняття захватів іншими запитами з цієї самої множини. Виявлення та усунення тупиків – це одна з найскладніших задач у розподілених системах.

ТРАНСАКЦІЇ ТА МЕХАНІЗМ ЇХ ПІДТРИМКИ

Для забезпечення логічної узгодженості даних у процесі внесення змін, а також для підвищення стійкості системи до різного роду збоїв система повинна підтримувати механізм обслуговування трансакцій.

Трансакція – це певна сукупність операцій над файлами БД, яка поєднує певний набір команд, що змінюють зміст файлів бази даних, але при цьому ці зміни вносять не одразу, а відкладають до закриття трансакції.

Неможливою є проміжна ситуація, щоб при виконанні трансакції частину змін заносили до бази даних, а частину – відміняли.

Для кожної трансакції при її закритті **всі зміни** заносять до бази даних або відміняють і здійснюють ***відкат*** трансакції, тобто повинна виконуватись вимога до ***атомарності трансакцій*** – виконання всіх дій або ж невиконання жодної з них.

Другою важливою ознакою трансакції є її ***довговічність***, яка полягає в тому, що коли трансакція зафіксована, то її результат не повинен втратитися навіть в разі зруйнування системи.

Важливою вимогою до трансакцій є так звана їх ***серіалізація,*** яка полягає в тому, щоб різні трансакції не мали випадкового впливу одна на одну.

Трансакції також повинні мати властивості ***ізольованості***, тобто результат трансакції не повинен залежати (тобто бути ізольованим) від інших трансакцій, виконуваних паралельно.

Кожна трансакція має починатися при цілісному стані БД, такий самий стан повинен залишалися після її завершення.

Невиконання цієї умови призводить до того, що замість фіксації результатів трансакції в БД виконується відкат, тобто база даних повертається в той початковий стан, при якому починалась виконуватися трансакція.

Проілюструємо дію цього механізму на прикладі.

Нехай необхідно перевести деяку суму одного розрахункового рахунку в банку на інший. Ця операція складається з двох кроків:

1) зняти суму з одного розрахункового рахунку (модифікація запису 1);

2) добавити суму на другий рахунок (модифікація запису 2).

Зазначимо, що після виконання першого кроку файли бази даних перебуватимуть у неузгодженому стані (рахунки будуть балансуватись на величину суми, знятої з першого рахунку).

При паралельній роботі кількох користувачів такий стан файлу буде помилковим, оскільки інші користувачі бачать файли з неузгодженою інформацією.

Крім того, другий крок операції може завершитися тупиком (наприклад, другий запис 6уде заблокований іншим користувачем) або аварійним збоєм системи.

Апаратний збій між виконанням першого і другого кроків взагалі може призвести до втрати контролю над цілісністю та узгодженістю бази даних.

Спроба розв’язати задачу блокуванням файлів на період виконання операції дуже уповільнить роботу інших користувачів мережі і не зможе забезпечити відновлення файлів у результаті апаратних збоїв системи.

Для того щоб усунути подібні ускладнення, використовують механізм обслуговування трансакцій.

Перед виконанням трансакції їй присвоюють ідентифікатор початкової трансакції і виконують усі необхідні дії (кроки 1 і 2). При цьому інші користувачі не обмежені у виконанні операцій пошуку у файлах, які використовуються в активній трансакції, і вбачають її у початковому стані. Операції редагування цих файлів під час виконання трансакції заборонені.

Якщо трансакція завершилась успішно, то після відповідного сигналу всі зміни, внесені у файли бази даних, стають видимими для всіх користувачів, а файли – доступними для внесення змін.

При виникненні ускладнень під час виконання наступного кроку (наприклад, ресурс недоступний для виконання) трансакція завершується аварійно.

При аварійному завершенні трансакції система має виконати так званий відкат. При цьому всі виконані з моменту початку трансакції перетворення повинні бути знищені, а файли повернені в початковий стан. Аналогічний відкат повинен виконуватись і після збою системи.

ЗБОЇ. У розподілених системах виділяють чотири типи збоїв: збій трансакції, збій вузла, збій носія (диска), збій комунікаційної лінії.

Збій трансакції може бути спричинений помилками у вхідних даних і виявленням тупика. У всіх перелічених ситуаціях збою трансакцію необхідно перервати і виконати відкат бази даних. Для попередження «зависання» системи необхідний контроль за часом виконання трансакції, Якщо ліміт часу, відведений для виконання трансакції, перевищено її потрібно відмітити як таку, що підлягає аварійному завершенню незалежно від результату початкових кроків, і всі файли, задіяні при її виконанні, перевести в початковий стан.

Для підтримання цілісності бази даних при різного роду збоях використовують ***протоколи журналізації***, які вміщують інформацію про всі зміни, які вносяться до бази даних під час виконання трансакції.

В одну трансакцію доцільно об'єднувати операції, які виконують логічно зв'язані між собою зміни файлів бази даних.

Наявність механізму підтримки трансакцій – це показник рівня розвиненості СУБД. Але, на жаль, не всі СУБД мають механізм підтримки трансакцій; його мають лише ті системи, які проектувались і розроблялись як розподілені СУБД для роботи в багатокористувацькому середовищі. **Підтримка трансакцій – це основа забезпечення цілісності БД.**

У сучасних розподілених СУБД підтримку трансакцій можна виконувати за допомогою двох методів: ***двофазною фіксацією трансакцій*** та ***дублюванням даних***.

Ці два методи відрізняються тим, що в першому випадку багато трансакцій працюють одночасно з однією розподіленою базою даних.

У другому випадку кожна трансакція має змогу копіювати необхідні для її виконання дані з розподіленої бази даних і працювати з нею як зі своєю особистою базою даних, модифікувати ці дані, а потім після виконання трансакції повертати їх у розподілену базу даних.

Підтримка трансакцій через механізм *двофазної фіксації трансакції*.

Двофазна фіксація трансакцій спирається на такі правила:

1. Якщо хоча б один вузол не може з будь-яких причин зафіксувати трансакцію, то розподілена трансакція припиняється на всіх, інших вузлах, які беруть участь у її виконанні.
2. Якщо всі вузли погоджуються з фіксацією трансакції, то вона фіксується на всіх вузлах, які беруть участь в її реалізації.

При використанні цього методу трансакцію виконують у два етапи (фази).

Перший етап, (перша фаза), полягає в тому, що на вузлі, де виконуватимуть трансакцію, створюють ***процес-координатор***, а на всіх інших вузлах – ***процеси-учасники***.

Спочатку процес-координатор розсилає повідомлення учасникам про початок трансакції, і кожний з них самостійно вирішує чи може трансакція бути завершеною на даному вузлі.

Учасники, готові завершити трансакцію, надсилають повідомлення на вузол–координатор про свою готовність до завершення трансакції. Учасники, які не можуть зафіксувати свою частину трансакції, надсилають повідомлення про те, що трансакцію буде перервано.

Координатор, зібравши повідомлення учасників, вирішує долю трансакції згідно з правилами.

Якщо всі учасники надіслали згоду, координатор надсилає повідомлення про глобальну фіксацію трансакції, згідно з яким вносяться зміни на всіх серверах.

При відмові хоча б одного з учасників процес-координатор приймає рішення про припинення виконання трансакції і надсилає повідомлення про глобальне припинення тим учасникам, які надіслали згоду на виконання трансакції.

Тим учасникам, які надіслали відмову на виконання трансакції, повідомлення можна не надсилати, оскільки вони самі в змозі прийняти рішення згідно з правилами двофазної фіксації трансакцій.

Ця ситуація називається **одностороннім припиненням виконання трансакції**.

В результаті на всіх серверах трансакція завершується однаково (фіксується або переривається).

Якщо всі вузли мають змогу зафіксувати трансакцію, то всі перетворення виконують на всіх вузлах; якщо хоча б один вузол не в змозі зафіксувати результати трансакції, то її припиняють на всіх вузлах і виконують відкат.

Протоколом обслуговування трансакцій передбачається обмін повідомленнями між координатором і учасниками двічі, тому цей механізм отримав назву двофазної фіксації трансакцій.

Недоліки двофазної трансакції:

- одночасний захват всіх ресурсів може надовго заблокувати доступ до даних;
- вихід з ладу під час виконання трансакції того вузла мережі, який заблокував ресурси, призведе до блокування деяких даних усієї мережі доти, поки не буде відновлено його роботу.

Проблеми:

- забезпечення коректної поведінки системи в разі виходу з ладу сервера або пошкодження ліній зв’язку. Тому організація розподіленої системи в даному варіанті потребує надійних і швидких ліній зв'язку.
- вирішення конфліктів, коли одна трансакція вступає в конфлікт з іншою трансакцією з приводу доступу до певних даних. Тому системи, що базуються на двофазній фіксації трансакцій, повинні мати деякий механізм вирішення конфліктів, пов'язаних з одночасним доступом.

Підтримка трансакцій через механізм *дублювання (тиражування) даних*.

Суть механізму - необхідні для виконання трансакції дані копіюють на той сервер, де їх оброблятимуть.

Усі зміни, внесені іншими користувачами протягом виконання запиту, не впливають на його виконання, оскільки вони фіксуються в основних файлах і не відображуються в їх копіях.

Такий механізм дає змогу завершити трансакцію з ланцюжком пошукових запитів будь-якої довжини, не порушивши логічної цілісності даних, а також є засобом уникання конфліктів під час роботи з базою даних. Наприклад, кілька користувачів можуть одночасно модифікувати одні й ті самі дині, не очікуючи один одного.

Але коли змінені дані будуть повернені в основну базу даних, то вона може мати кілька версій, тому для таких системі потрібен механізм управління версіями.

Після успішного завершення трансакції зміни вносять на всі сервери у відповідні файли, які брали участь у виконанні трансакції.

При цьому виникає потреба забезпечити синхронізацію процесу тиражування змін на різних серверах, щоб база даних не втратила своєї цілісності

Можливі два режими тиражування змін: *синхронний* і *асинхронний.*

Синхронний режим тиражування змін – це процес, при якому зміни поширюються одразу по всіх вузлам після їх виконання.

Синхронний варіант внесення змін слід використовувати в інформаційних системах, які повинні функціонувати в режимі реального часу.

Прикладом таких систем може бути, автоматизована банківська система, в якій зміни залишку рахунків повинні відображатись одразу, щоб не було можливості виконати з ними будь-які несанкціоновані дії.

Цей варіант виконання трансакцій порівняно з їх двофазною фіксацією має деякі **переваги**.

По-перше, запит виконують на одному сервері, при цьому не потрібно передавати дані і розподіляти виконання запиту між серверами.

По-друге, хоч і потрібно тиражування змін даних, які виконала трансакція, то ці зміни вносять асинхронно щодо самої трансакції; отже, користувач отримує результат виконання запиту швидше, оскільки йому не потрібно очікувати закінчення обміну даними між серверами по внесенню змін.

Іще одна важлива перевага синхронного тиражування – підтримування «дзеркальної» бази даних на резервному сервері, причому обидва сервери (основний і резервний) в цьому випадку можуть працювати паралельно.

При ***асинхронному режимі*** зміни запам'ятовуються, а момент, їх передачі вибирають за деяким певним правилом.

При такому способі підтримки логічної цілісності розподіленої бази даних спостерігається деяка розсинхронізація стану локальних баз даних, яка полягає в тому, що зміни, внесені до деяких локальних баз, відстають за часом їх внесення одна від одної.

Інтервал часу, через який поновлюються дані в інших вузлах, має узгоджуватися з інтервалом розв'язання задачі в цих вузлах.

Асинхронне дублювання має **недолік**, оскільки не забезпечує виникнення процедур поновлення даних у масштабі реального часу.

Цей режим можна використовувати в таких інформаційних системах, у яких складно підтримувати постійний зв'язок між серверами, і термін тимчасової неузгодженості стану бази даних не конфліктує з періодичністю розв'язання задач у цій системі.

ПОРІВНЯЛЬНА ТАБЛИЦЯ ДВОХ МЕТОДІВ ПІДТРИМКИ ТРАНСАКЦІЙ В РОЗПОДІЛЕНІЙ БАЗІ ДАНИХ

Методи	Двофазна фіксація трансакцій	Дублювання (тиражування) даних
Характеристики	•Однорангові сервери •Зміни вносять у реальному масштабі часу •Синхронні зміни •Блокування нових трансакцій на час внесення змін	•Сервери ведучий/підлеглий •Зміни вносять у режимі часу, який близький до реального •Асинхронні зміни •Безперервна робота сервера
Результати	Інформаційна система має обмежені можливості управління внесенням змін	Інформаційна система має можливості управління синхронізацією і трафіком мережі

У розподілених системах важливою є проблема проектування розподіленої бази даних та її адміністрування.

Насамперед потрібно обрати архітектуру розподіленої бази даних і визначити правила доступу даних і правила її адміністрування.

Встановлення правил адміністрування гарантує коректність роботи системи.

Найпростіший варіант архітектури, який виключає можливість виникнення конфліктів, полягає в тому, що серед усіх вузлів, які зберігають одну і ту саму копію деяких даних, ***вибирають один, що має право на внесення змін***, інші вузли лише мають доступ до цієї копії без права внесення змін. Цей варіант можуть вибрати банк і його філії, коли останнім дозволяється переглянути певні дані головної контори без права внесення змін.

Другим варіантом архітектури, який також гарантує, що конфлікти не виникнуть, є динамічна передача права модифікації від сервера до сервера.

При цій архітектурі кожний елемент даних має спеціальний додатковий атрибут, у якому можна вказати дозвіл на внесення змін при передаванні даних між серверами.

Цей варіант можна проілюструвати на прикладі бази даних торговельної організації. **Наприклад**, замовлення на товар було виконано, і він надійшов на склад; інформацію про це можна передати на сервер відділу продажу з правом внесення змін у ці дані.

Обидва механізми підтримки трансакцій є коректними, кожний з них має певні переваги й недоліки і може використовуватися залежно від специфіки задач та специфіки предметної області. Тому в СУБД обидва варіанти повинні підтримуватись.