Тема лекції 10:

Транзакції

- Основні поняття, пов’язані з транзакціями
- Основні властивості транзакцій
- Початок і завершення транзакції
- Точки відкату
- Конфлікти між транзакціями
- Блокування даних
- Рівні ізоляції транзакції
- Журнал транзакцій

Основні поняття

- Транзакція – це неподільна послідовність інструкцій маніпулювання даними, яка виконується як єдине ціле (все або нічого) і яка переводить базу даних з одного цілісного стану в інший цілісний стан
- Відкат транзакції – відміна дії усіх вже виконаних інструкцій транзакції в результаті невиконання якої-небудь інструкції у цій транзакції

Основні властивості транзакцій (ACID)

- Atomicity – атомарність
- Consistency – узгодженість
- Isolation – ізольованість
- Durability – довговічність,стійкість

Основні властивості транзакцій **Atomicity** (Атомарність)

- Транзакція виконується як атомарна операція: або виконується уся транзакція, або вона вся не виконується
- Тільки коли усі оператори транзакції виконуються без помилок, результати виконання усієї транзакції зберігаються в базі.

Основні властивості транзакцій **Consistency** (Узгодженість)

- Транзакція переводить базу даних з одного цілісного стану в інший цілісний стан; всередині транзакції узгодженість бази даних може порушуватись
- Стан цілісності бази даних підтримується на рівні обмежень різних типів
- Разом з поняттям цілісності бази даних існує поняття реакції системи на спробу порушення цілісності. Є два типи таких реакцій:
  - відмова виконати «незаконну операцію»; при цьому генерується повідомлення користувачу;
  - виконання компенсуючих дій, які полягають у виконанні додаткового коду у транзакції, який підтримує узгодженість та цілісність даних.

Основні властивості транзакцій **Isolation** (Ізольованість)

- Транзакції різних користувачів не повинні заважати одна одній
- Це означає, що в багатокористувацькому режимі транзакції повинні виконуватись одночасно, але так, щоб результат був таким, якби транзакції виконувались по черзі
- Дані, які змінюються транзакцією не повинні бути доступними для перегляду іншими користувачами, поки виконуються зміни
- Інші користувачі бази даних повинні бачити лише дані, отримані після завершення транзакції, щоб користувачі не приймали помилкових рішень, базуючись на проміжних даних.

Основні властивості транзакцій **Durability** (Довговічність, стійкість)

- Якщо транзакція виконана, то результати її роботи повинні зберегтись у базі даних, навіть якщо в наступний момент відбудеться збій системи.
- Проблема збою системи не повинна стати причиною лише часткового збереження або неповного оновлення даних
- У більшості СУБД ця властивість реалізована за допомогою журналу транзакцій, який зберігається окремо від бази даних.

Початок транзакції

- В **SQL:2003** не передбачено спеціального оператора початку транзакції (модель ANSI)
- **Transact-SQL** вимагає, щоб транзакція ініціювалась оператор

  BEGIN TRANSACTION

  (явний режим транзакції)

Завершення транзакції

- COMMIT – завершити виконання усіх інструкцій транзакції, яке відбувається послідовно в одному блоці. Застосовується тоді, коли усі SQL –оператори транзакції імовірно виконані (вони сприйняті СУБД і не викликали помилок) і вимагається підтвердити зміни, внесені у БД.
- ROLLBACK – відкат – відбувається відміна дії усіх інструкцій транзакції, і база даних повертається у вихідний стан. Застосовується тоді, коли необхідно відмінити зміни внесені транзакцією, і відновити БД у попередньому стані.

Конфлікти між транзакціями

- Транзакції не заважають одна одній, якщо вони звертаються до різних даних або виконуються в різні проміжки часу
- Транзакції називаються конкуруючими, якщо вони перетинаються у часі і звертаються до одних і тих самих даних
- В результаті конкуренції за дані між транзакціями виникають конфлікти доступу до даних

Типи конфліктів між транзакціями

- W-W (Write-Write) – Запис-Запис. Перша транзакція змінила об'єкт і не завершилась. Друга транзакція намагається змінити цей об'єкт. Результат – **втрата результатів оновлення** для транзакції, яка завершилась першою. Фіксуються зміни транзакції, яка завершилась пізніше.
- R-W (Read-Write) – Читання-Запис. Перша транзакція двічі читає об'єкт. Між цими читаннями вклинюється друга транзакція і намагається змінити цей об'єкт. Результат – **неповторне читання**. Перша транзакція працює з даними, які довільно міняються (**Non – repeatable read**)

Типи конфліктів між транзакціями

- W-R (Write-Read) – Запис-Читання. Перша транзакція змінила об'єкт і не завершилась. Друга транзакція прочитала цей об'єкт і працює з ним. Перша транзакція робить відкат і відновлює попередні дані. Результат – **фіктивне читання даних**. Друга транзакція працює з даними, яких нема і "не було" в БД. (**Dirty Read**)
- **Поява фантомів - Phantom record (read)**. Перша транзакція двічі виконує вибірку з фільтрацією. Між цими вибірками вклинюється друга транзакція, яка вставляє новий рядок, що задовольняє умову фільтрації. Результат – після другої вибірки з'явився фантом – лишній рядок, про який нічого не знає перша транзакція у випадку, якщо друга відкотилася після цього.

Стратегії вирішення проблем конкуруючих транзакцій

- Оптимістична: починаєм працювати без блокування, чи інших перевірок. Конфлікти, якщо виникнуть, вирішуються в момент завершення транзакції
- Песимістична: блокуємо всі необхідні ресурси перед початком транзакції (у випадку конфлікту транзакція не починається)
- Незалежно від стратегії, дані, які змінила транзакція, блокуються (для модифікації) до кінця завершення транзакції

Способи вирішення конфліктів між транзакціями

- Можна "тормозити" деякі транзакції настільки, наскільки це потрібно для забезпечення правильності виконання конкуруючих транзацій. Це означає виконання їх у різний момент часу. Цей метод реалізується шляхом використання **блокувань (Locking method).**
- Надавати конкуруючим транзакціям "різні" екземпляри даних, тобто забезпечити їх роботу з різними версіями даних шляхом використання даних з журналу транзакцій через механізм **виділення версій (Multi – version concurrency control)**.

Механізм виділення версій (Multi–version concurrency control)

- Надає транзакціям різні версії даних для роботи
- writers never block readers and readers never block writers
- Правило: Транзакції можуть бути довгими
- Блокування використовуються для неможливості одночасної модифікації одних і тих же даних кількома транзакціями

СУБД, які підтримують Multi–version concurrency control

- Oracle
- PostgreSQL
- MySQL
- MS SQL Server (from 2005 version)

Метод блокування
Locking method

- «притримує» транзакції, щоб вони виконувалися в різні періоди часу
- читаюча транзакція повинна або блокувати ту, яка модифікує, щоб забезпечити читання узгоджених даних, або прийняти неточні результати
- Правило: транзакції повинні бути короткими

Метод блокування
Locking method

- Якщо для виконання деякої транзакції необхідно, щоб певний об’єкт не змінювався і/або не читався без відома цієї транзакції, то цей об’єкт блокується.
- Доступ до цього об’єкта зі сторони інших транзакцій обмежується на час виконання транзакції, яка викликає блокування.
- Блокування не мають ніякого відношення до безпеки на рівні конкретних користувачів або їх груп.
- Блокування в транзакціях виконує СУБД.

СУБД, які підтримують метод блокувань (Locking method)

- SAP (Sybase) ASE
- SAP (Sybase) SQL Anywhere (Watcom SQL)
- MS SQL Server
- IBM DB2 / IBM Informix

Основні типи блокувань

- **Монопольні**, або виключні, блокування (**X-locks**, eXclusive Locks, X-блокування) – це блокування без взаємного доступу (блокування запису (write)). Якщо транзакція TRAN1 блокує об’єкт за допомогою X-блокування, то будь-який доступ до цього об’єкта з боку інших транзакцій відкидається;
- **Розділені** блокування (**S-locks**, Shared locks, S-блокування) – це блокування з взаємним доступом (блокування читання (read)). Якщо транзакція TRAN1 блокує об’єкт за допомогою S-блокування, то запити з боку інших транзакцій на X-блокування будуть відхилені, а запити з боку інших транзакцій на S-блокування цього об’єкта будуть прийняті.

Протокол доступу до даних з метою запобігання конфліктам

- перед тим як прочитати об’єкт, транзакція повинна накласти на цей об’єкт S-блокування;
- перед тим як оновити об’єкт, транзакція повинна накласти на цей об’єкт X-блокування. Якщо транзакція вже заблокувала об’єкт S-блокуванням (для читання), то перед оновленням об’єкта S-блокування необхідно замінити на X-блокування;
- якщо блокування об’єкта транзакцією TRAN2 відкидається, тому що об’єкт вже заблокований транзакцією TRAN1, то транзакція TRAN2 переходить у стан очікування до тих пір, поки транзакція TRAN1 не зніме блокування об’єкта;
- X-блокування, накладені транзакцією TRAN1, зберігаються до її завершення.

Типи блокувань в SQL Server

Типи блокувань	Застосування блокувань
Shared (S-блокування) або розділене блокування читання	Для операцій читання, наприклад, в операторі `SELECT`
Exclusive (X-блокування) або виключне блокування зміни	Для операцій модифікації даних, наприклад в операторах `INSERT`, `UPDATE`, `DELETE`.
Update (U-блокування)	Накладається на ресурси, які змінюються. Попереджає типові мертві блокування (тупики), коли багато з'єднань читають, блокують і можуть змінити ресурси пізніше
Intent (I- блокування) або блокування намірів	Використовується для встановлення ієрархії блокувань: INTent shared (IS), INTent (IX) і shared з INTent (SIX)
Schema	Використовується для операції зміни схеми даних: (Sch-M) або (Sch-S)
Bulk Update (BU)	Використовується для команди `BULK INSERT` або з хінтом `TABLOCK`

Рівні блокування

- блокування самої БД;
- блокування файлів БД;
- блокування таблиць БД;
- блокування сторінок;
- блокування окремих рядків таблиць (MS SQL Server, Oracle);
- блокування окремих полів (для цього необхідно використовувати індекси).

Рівні блокування

Загальним правилом, пов’язаним з блокуванням, є наступне:

- Чим більшим є об’єкт блокування, тим менше можливостей для паралельної роботи залишається після накладання блокування.
- З іншого боку, перевагою блокувань великих об’єктів є зменшення накладних витрат системи і усунення проблем, які не вирішуються через блокування менших об’єктів.

Параметри блокувань

- **Рівень блокувань**. СУБД сама визначає коли переходити на блокування об’єктів вищого рівня. Наприклад, від блокування рядків до блокування таблиці. Вручну змінити рівень блокування можна через підказку оптимізатору
- **Кількість блокувань**. У СУБД передбачені обмеження на кількість блокувань, доступних для одночасного виконання транзакцій.
- **Ескалація блокувань** – здатність СУБД об’єднати декілька блокувань одного рівня в одне блокування вищого рівня. Це автоматична здатність СУБД консолідувати велику кількість блокувань у малу.
- **Час очікування**. Більшість СУБД надають можливість адміністратору встановити параметри, що задають обмеження для часу очікування початку обробки транзакції. Якщо блокування не відбулось за вказаний час, то повертається повідомлення про помилку.

Взаємоблокування, або тупики (deadlocks)

- Тупикова ситуація характеризується тим, що дві чи більше транзакцій намагаються заблокувати одні і ті ж об’єкти і безкінечно довго очікують одна одну.
- Ситуація тупика може виникати при наявності не менше двох транзакцій, кожна з яких виконує не менше двох операцій.

Вихід з тупика

- Нормального виходу з тупикової ситуації не існує
- Тупик необхідно розпізнати та нейтралізувати.
- Методом вирішення тупикової ситуації є відкат однієї з транзакцій, яку називають транзакція-жертва, щоб інші транзакції продовжували свою роботу.
- Після усунення тупика транзакцію-жертву можна повторити.

Підхід 1 до виявлення тупиків і вибору транзакції-жертви

- СУБД **не слідкує** за виникненням тупиків. Транзакції самі вирішують, чи бути їм жертвою. Цей метод найпростіший і не потребує додаткових ресурсів.
  - Для транзакції задається час очікування (або кількість спроб), за який транзакція намагається встановити потрібне блокування.
  - Якщо за цей час (або за кількість спроб) блокування не завершається успішно, то транзакція відкочується (або генерується помилкова ситуація).
- За простоту цього методу доводиться платити тим, що транзакції-жертви вибираються випадково. В результаті цього через просту транзакцію може відкотитись досить дорога транзакція, на виконання якої затрачено багато часу і ресурсів.

Підхід 2 до виявлення тупиків і вибору транзакції-жертви

- СУБД **слідкує** за виникненням тупиків і вона ж приймає рішення, якою транзакцією пожертвувати. Цей спосіб є характерним для промислових СУБД.
- Основний механізм пошуку тупиків – це аналіз графа очікувань і виявлення в ньому циклів.
  - Граф очікувань транзакцій (Wait For Graph, WFG) – це орієнтовний дводольний граф, в якому існує два типи вершин: вершини, що відповідають транзакціям, і вершини, що відповідають об’єктам.
  - Ситуація тупика виникає, коли в графі очікування з’являється хоча б один цикл.
  - У цьому випадку для однієї з транзакцій, що попала в цикл, необхідно зробити відкат.
  - СУБД сама вибирає цю транзакцію через деякі вартісні характеристики, наприклад, найкоротшу, або з найменшим пріоритетом

Порівняння стратегій вирішення конфліктних ситуацій

- Блокування об’єктів необхідного рівня вирішують більшість проблем паралельного доступу до даних.
- При блокуванні на рівні рядків, якщо усі транзакції виконують протокол доступу, вирішуються усі конфліктні ситуації, крім фантомів.
- Блокування вирішують проблему конфліктів, якщо вони недовготривалі (мілісекунди, секунди)
- Якщо блокування тривають довго, вони блокують доступ до даних, тому доцільніше використовувати версії даних (Oracle).

Рівні ізоляції транзакції

- Рівень ізоляції транзакції – це рівень ізольованості між тразакціями різних користувачів БД і рівень їх взаємного впливу.
- З цією метою використовується оператор:

SET TRANSACTION
ISOLATION LEVEL <рівень ізоляції>;

Значення рівнів ізоляції транзакції

- SERIALIZABLE– послідовне виконання (значення за замовчуванням в ANSI SQL);
- REPEATABLE READ – повторюване читання;
- READ COMMITED – підтверджене читання;
- READ UNCOMMITED – непідтверджене читання.

Рівень ізоляції транзакції SERIALIZABLE

- Найбільший рівень ізоляції транзакцій – досягається найбільша безпека.
- Транзакції завжди виконуються не паралельно, а послідовно. Якщо одна з них вже почалась, то інша чекає її завершення.
- Апаратні і програмні збої можуть призвести до невиконання транзакції, але при такому рівні ізоляції можна бути впевненим, що результати роботи з БД будуть коректними.
- Однак, надаючи найбільшу надійність, такий рівень ізоляції знижує загальну продуктивність системи в багатокористувацькому режимі. Оскільки одні користувачі змушені чекати завершення транзакцій інших користувачів.
- У зв’язку з цим виникає задача пошуку компромісу між рівнем безпеки та продуктивністю системи в цілому.

Рівень ізоляції транзакції REPEATABLE READ

- Суть полягає в тому, що результати оновлення даних (зміна або видалення) іншими транзакціями не повинні відображатись у записах поточної транзакції.
- При цьому допускаються операції вставки
- Рівень виключає проблему неповторюваного читання.
- Однак, не усуває проблему фіктивного читання та фантомної вставки.

Рівень ізоляції транзакції READ COMMITED

- Гарантує, що поточна транзакція не зможе бачити незафіксовані результати оновлення інших транзакцій, поки ті не завершаться оператором COMMIT.
- Якщо поточна транзакція спробує оновити вже змінені і зафіксовані іншою транзакцією дані (під час виконання поточної транзакції), то це оновлення буде автоматично відкочене, щоб уникнути проблеми втрати результатів оновлення
- Рівень не звільняє від використання невірних (застарілих) даних. Це проблема неповторюваного читання. Однак, послідовні операції читання можуть виявляти зміну даних, виконані іншими транзакціями і зафіксованими між цими читанням.

Рівень ізоляції транзакції READ UNCOMMITED

- Найнижчий (слабкий) рівень ізоляції.
- При такому рівні зміни, внесені одним користувачем, можуть бути прочитані іншим користувачем, навіть якщо перший користувач не підтвердив їх за допомогою оператора COMMIT. Проблема виникає, якщо перший користувач виконав відкат своєї транзакції. Тоді усі наступні дії іншого користувача можуть базуватись на невірних (псевдооновлених) даних.
- Будь-які модифікації, що виконуються іншими транзакціми, впливають на поточну транзакцію, незалежно від їх фіксації. Виникає проблема незафіксованих даних, а також фантомів.
- Однак, СУБД не допустить виникнення проблеми втрачених результатів оновлення і використає відкат для змін поточної транзакції, якщо вони викличуть таку проблему.
- Рівень READ UNCOMMITED підходить при статистичній або аналітичній обробці даних, які змінюються досить рідко.

Конфлікти (аномалії) та рівні ізоляції транзакцій

Anomaly/ Isolation level	Dirty read	Non-repeatable read	Phantom reads
Read uncommitted	possible	possible	possible
Read committed		possible	possible
Repeatable read		Not possible	possible
Serializable	Not possible	Not possible	Not possible

Рівень ізоляції транзакцій в різних СУБД

- Встановлений по замовчуванню для бази даних
  - MS SQL – Read committed
  - Azure – repeatable read snapshot
  - Oracle – Read committed
  - MySQL – repeatable read
- Не можна міняти після старту транзакції

Журнал транзакцій

- Журнал транзакцій – це паралельне записування в окремий файл відомостей про будь-яку транзакцію, яка модифікує базу даних.
- Щоб відновити БД після перебою, необхідно, щоб кожна дія транзакції над БД була зафіксована в **журналі транзакцій**.

Журнал транзакцій

- У журналі записується інформація, необхідна для відновлення несуперечності системи після перебоїв. Ця інформація може містити:
  - ідентифікатор транзакції,
  - тип операції,
  - попереднє значення (стан) елементу даних,
  - нове значення (стан) елементу даних.

Дякую за увагу

Опрацювати: Д.Петковіч «Microsoft SQL Server 2012. Руководство для начинающих» ***ст.369-392***