

Obsługa transakcji rozproszonych w języku Java

Marek Wojciechowski, Maciej Zakrzewicz
Instytut Informatyki, Politechnika Poznańska



Plan prezentacji

- Transakcje i ich własności
- Proste transakcje w JDBC
- Transakcje rozproszone w JDBC 2.0
- Transakcje na platformie J2EE – Standard JTA
- JTA w aplikacjach Java (EJB)
- Podsumowanie



Czym jest transakcja?

- Transakcja to podstawowa logiczna jednostka interakcji użytkownika z systemem
- Transakcja to sekwencja operacji, która musi zakończyć się sukcesem w całości – w przeciwnym wypadku przywrócony musi być stan początkowy
- Przykład transakcji:

Operacja przelewu kwoty x z konta A na konto B:

- 1) Pobranie kwoty x z konta A
- 2) Dodanie kwoty x do konta B



Własności transakcji (ACID)

- Atomowość (ang. atomicity)
 - zbiór operacji składowych transakcji musi być wykonany w całości lub wcale
- Spójność (ang. consistency)
 - transakcja przeprowadza bazę danych z jednego stanu spójnego do innego stanu spójnego
- Izolacja (ang. isolation)
 - polega na logicznym odseparowaniu od siebie transakcji współbieżnie wykonywanych w systemie (możliwe różne poziomy izolacji)
- Trwałość (ang. durability)
 - Odporność wyników zatwierdzonej transakcji na awarię systemu



Transakcje lokalne i globalne

- Transakcje lokalne:
 - Dotyczą jednego serwera bazy danych
 - Obsługiwane przez dany serwer bazy danych
- Transakcje globalne (rozproszone):
 - Dotyczą więcej niż jednego serwera bazy danych
 - Serwery mogą pochodzić od różnych dostawców (heterogeniczne transakcje globalne)
 - Obsługiwane przez zarządcę transakcji (ang. transaction manager):
 - Zewnętrzny (np. dostępny w ramach serwera aplikacji)
 - Wbudowany w jeden z serwerów biorących udział w transakcji



2-Phase Commit

- Dla zapewnienia własności atomowości transakcji rozproszonych opracowano protokół zatwierdzania dwufazowego (ang 2-Phase Commit)
- Faza 1 – **Prepare**: węzły (serwery) biorące udział w transakcji przygotowują się do jej zatwierdzenia (na żądanie koordynatora transakcji)
- Faza 2 – **Commit**: gdy wszystkie węzły są gotowe do zatwierdzenia transakcji następuje jej rzeczywiste zatwierdzenie



Standardy dostępu do baz danych z aplikacji języka Java

- JDBC - Opracowany przez Sun
 - Specyfikuje interfejsy implementowane w postaci sterowników JDBC dla poszczególnych SZBD
 - JDBC 1.0 – transakcje lokalne
 - JDBC 2.0 – transakcje rozproszone, efektywność
 - JDBC 3.0 – „drobne usprawnienia” (np. SAVEPOINT)
- SQLJ - Standard ISO/ANSI
 - Naturalne i mniej podatne na błędy zagnieżdżanie SQL w kodzie Java
 - Powszechnie implementowany poprzez prekompilację do wywołań JDBC (w zakresie transakcji funkcjonalność zależy od sterowników JDBC)



Transakcje lokalne w JDBC (1/2)

- Obiekt Connection uzyskany poprzez DriverManager
 - Rozwiązanie dostępne od JDBC 1.0

```
// rejestracja sterownika Oracle JDBC
DriverManager.registerDriver(new oracle.jdbc.OracleDriver());

// otwarcie połączenia z bazą poprzez DriverManager
Connection conn = DriverManager.getConnection(
    "jdbc:oracle:thin:@host1:1521:orcl", "scott", "tiger");

// wyłączenie trybu automatycznego zatwierdzania
conn.setAutoCommit(false);
// utworzenie i wykonanie instrukcji SQL
Statement stmt = conn.createStatement();
stmt.executeUpdate("DELETE FROM emp WHERE empno = 3981");
// zatwierdzenie transakcji
conn.commit();
```



Transakcje lokalne w JDBC (2/2)

- Obiekt Connection uzyskany poprzez DataSource
 - Rozwiązanie dostępne od JDBC 2.0
 - Źródła danych mogą oferować dodatkową funkcjonalność:
 - Connection pooling
 - Wsparcie dla transakcji rozproszonych
 - Typowo źródła danych dostępne przez JNDI (np. udostępnione aplikacji przez serwer aplikacji w oparciu o plik konfiguracyjny)

```
...  
// uzyskanie obiektu reprezentującego kontekst JNDI  
Context ctx = new InitialContext();  
// wyszukanie źródła danych przez nazwę logiczną  
OracleDataSource ds  
    = (OracleDataSource) ctx.lookup("jdbc/FinanceDB");  
// uzyskanie obiektu Connection  
Connection conn = ds.getConnection();  
...
```



Transakcje rozproszone w JDBC 2.0

- Mechanizmy obsługi transakcji rozproszonych w JDBC 2.0 stanowią implementację standardu XA
- XA jest ogólnym standardem dla transakcji rozproszonych stanowiącym część standardu X/Open
- Funkcjonalność XA zapewnia zarządcy transakcji koordynację poszczególnych gałęzi transakcji (ang. transaction branches) reprezentowanych przez zasoby XA (XA resource) i zatwierdzanie lub wycofywanie gałęzi transakcji



Elementy standardu XA

- **Źródła danych XA** (ang. XA data sources) - jedno źródło danych dla jednej bazy danych biorącej udział w transakcji
- **Połączenia XA** (ang. XA connections) - uzyskane ze źródeł danych XA, służące do uzyskania instancji XA resource i połączeń JDBC
- **Zasoby XA** (ang. XA resources) - wykorzystywane przez zarządcę transakcji do koordynacji gałęzi transakcji rozproszonej
 - Metody: start, end, prepare, commit, i rollback
- **Identyfikatory transakcji** (ang. transaction IDs)
 - służące do identyfikacji poszczególnych gałęzi transakcji rozproszonych



Przebieg transakcji rozproszonej XA

- 1) Przygotowanie źródeł danych, połączeń i identyfikatorów gałęzi transakcji
 - 2) Rozpoczęcie gałęzi transakcji
 - 3) Wykonanie operacji SQL w poszczególnych gałęziach transakcji
 - 4) Zakończenie gałęzi transakcji
 - 5) Przeprowadzenie zatwierdzania dwufazowego
- W praktyce funkcjonalność XA jest ukryta przed aplikacjami użytkowymi i implementowana na poziomie serwera aplikacji i zarządcy transakcji



Transakcje na platformie J2EE

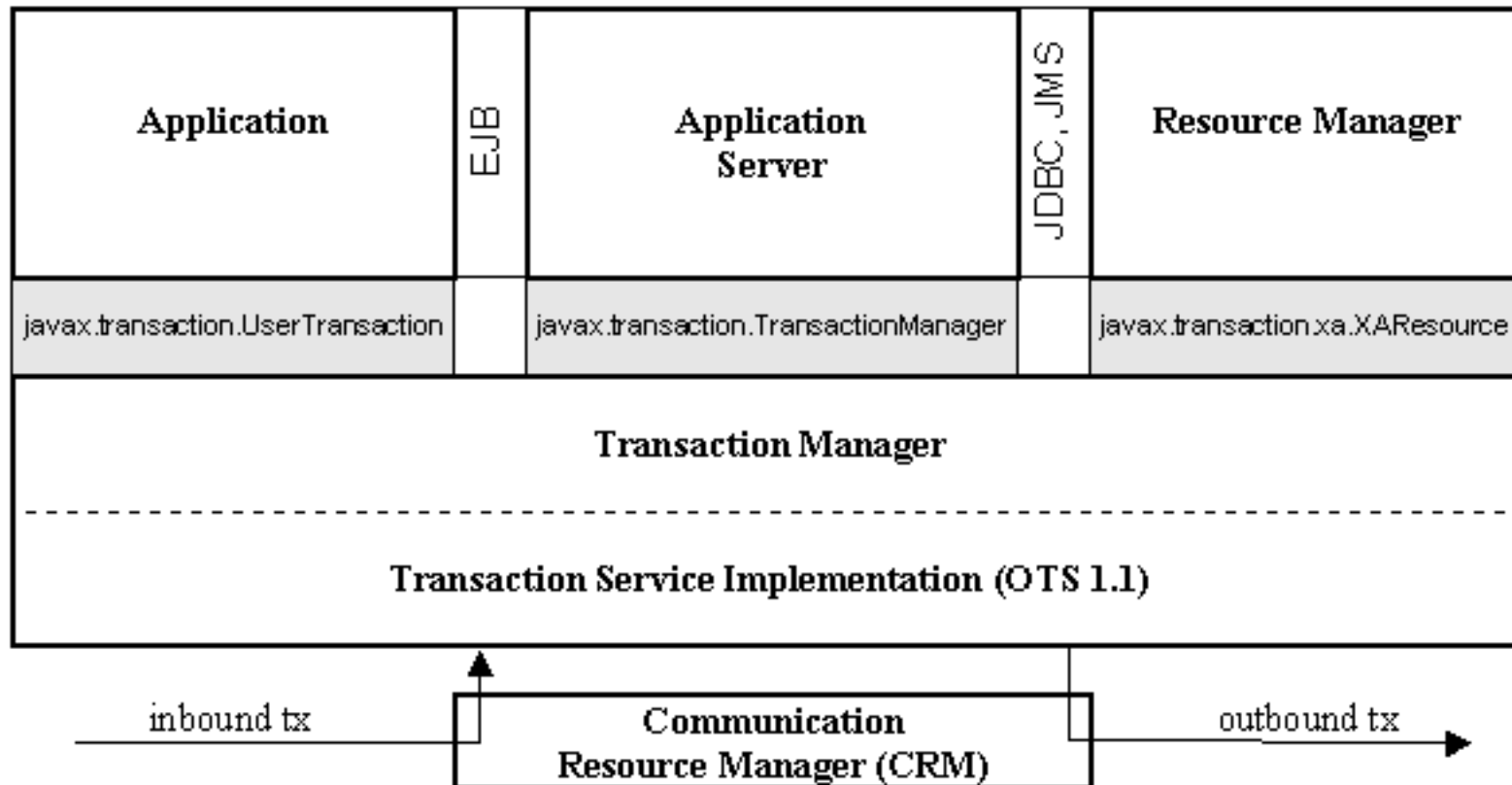
- Realizacja transakcji rozproszonych na platformie J2EE opiera się na współpracy oprogramowania różnego typu:
 - **Zarządca transakcji** (ang. *transaction manager*)
 - **Serwer aplikacji** (ang. *application server*)
 - np. serwer EJB
 - **Zarządca zasobów** (ang. *resource manager*)
 - Dostępny dla aplikacji poprzez bibliotekę programową
 - Najczęściej serwer bazy danych (dostępny poprzez JDBC)
 - **Transakcyjna aplikacja**
 - np. komponentowa aplikacja EJB
 - **Zarządca komunikacji** (ang. *communication resource manager*)



Standard JTA (Java Transaction API)

- Specyfikuje interfejsy między zarządcą transakcji (zgodnym z JTS – Java Transaction Service) a innymi jej „uczestnikami”
- Oparty na specyfikacji JDBC 2.0 i standardzie XA
- Obejmuje trzy główne części:
 - Interfejs wysokiego poziomu umożliwiający aplikacji transakcyjnej wytyczanie granic transakcji (*javax.transaction.UserTransaction*)
 - Interfejs zarządcy transakcji umożliwiający serwerowi aplikacji obsługę wyznaczania granic transakcji dla aplikacji zarządzanych przez dany serwer aplikacji (*javax.transaction.TransactionManager*)
 - Mapowanie do języka Java interfejsu XA umożliwiającego transakcyjnemu zarządcy zasobów udział w globalnej (rozproszonej) transakcji zarządzanej przez zewnętrznego zarządcę transakcji (*javax.transaction.xa.XAResource*)

Funkcjonalność JTA - Schemat





JTA w aplikacjach J2EE

- Wykorzystywany przede wszystkim w transakcjach realizowanych przez komponenty EJB
 - Również dostępny w serwletach/JSP
 - Niektóre typy EJB (sesyjne, komunikatowe) mogą korzystać z transakcji JDBC (dla transakcji lokalnych)
- Realizacja transakcji JTA w aplikacji J2EE wymaga:
 - Rejestracji zasobów
 - Rejestracja zasobów wiąże się z konfiguracją źródeł danych w jednym z plików konfiguracyjnych aplikacji
 - Od liczby wykorzystanych zasobów w transakcji zależy czy użyty będzie mechanizm 2-Phase Commit
 - Wyznaczenia granic transakcji - 2 możliwe podejścia:
 - Przez komponent (ang. bean-managed transaction)
 - Przez kontener (ang. container-managed transaction)



Transakcje zarządzane przez komponent

- Aplikacja jawnie rozpoczyna i kończy transakcję przez interfejs UserTransaction
- Dozwolone dla sesyjnych i komunikatowych EJB, niedozwolone dla encyjnych EJB, jedyna możliwość wykorzystania transakcji JTA w serwletach / JSP

```
...
Context ctx = new InitialContext();
UserTransaction ut
    = (UserTransaction) ctx.lookup("java:comp/env/UserTransaction");
// rozpoczęcie transakcji
ut.begin();
DataSource ds
    = (DataSource) ctx.lookup("java:comp/env/jdbc/OracleDS");
// uzyskanie obiektu Connection po (!) rozpoczęciu transakcji
Connection conn = ds.getConnection();
...
// zatwierdzenie transakcji
ut.commit();
...
```



Transakcje zarządzane przez kontener

- Dostępne tylko dla komponentów EJB (dla encyjnych jedyna możliwość)
- Transakcją steruje kontener, na podstawie informacji podanych w sposób deklaratywny w pliku deployment descriptor (tam również specyfikacja czy transakcje zarządzane przez kontener czy komponent)
- Deklaracje mają postać atrybutu transakcyjnego (Required, RequiresNew, Supports, NotSupported, Mandatory, Never) podanego dla całego komponentu lub poszczególnych jego metod



Podsumowanie

- Specyfikacje dotyczące obsługi transakcji rozproszonych w języku Java są implementacjami ogólnych standardów
- Dla skomplikowanych operacji transakcyjnych (w tym transakcji rozproszonych) dobrym rozwiązaniem jest technologia EJB
- Technologia EJB i standard JTA pozwalają na realizację transakcji lokalnych i rozproszonych na tej samej zasadzie (z punktu widzenia aplikacji)