MO809

Transações em Java

Felipe Leme
RA 930886 – MSC 2006

Agenda

- Introdução
- Transações em Banco de Dados
  - JDBC
- Transações Distribuídas
  - DTP/XA
  - JTA
- Links Úteis
- Dúvidas

Introdução

- Linguagem Java não oferece suporte à transações
  - ... mas APIs sim
- Java SE: transações em bancos de dados
  - JDBC
- Java EE: transações distribuídas
  - JTA (DTP/XA)
  - EJB (Componentes)
  - JMS (Mensagens)
  - JTS (Corba)
  - JCA (Sistemas Legados)

Exemplo de caso de uso

- Caso clássico: transferência bancária
- Interface SistemaBancario:
  - transfere(Conta conta1, Conta conta2,
    double valor);
- Pseudo-implementacao:
  - inicia transação
  - retira X de conta1
  - deposita X na conta2
  - commit transação

JDBC

- Java Database Connectivity
  - API Java para acesso a banco de dados
- Padronização no acesso a diferentes SGBDs
  - Aplicação: interfaces JDBC
  - SGBD: implementação (*driver JDBC*)
- Uso típico:
  - obtém conexão (*java.sql.Connection*)
  - envia comando (*java.sql.Statement*)
  - iterage com resultado (*java.sql.ResultSet*)
  - fecha comando
  - fecha conexão

JDBC - exemplo

```
public double getSaldo( Conta conta ) throws
  Exception { // tratamento de exceções omitido
   Connection conn = getConnection(); //abstração
   PreparedStatement stmt = conn.prepareStatement(
     "select saldo from conta where id = ?" );
   stmt.setInteger( 1, conta.getId() );
   ResultSet rs = stmt.executeQuery();
   rs.next();
   double saldo = rs.getDouble(1);
   stmt.close();
   conn.close();
   return saldo;
}
```

Transações JDBC

- Comportamento default: conexão em modo *auto-commit*
  - cada comando em uma transação
- Pode ser alterado – interface Connection
  - boolean getAutoCommit();
  - void setAutoCommit(boolean autoCommit);
- Controle manual de transação
  - Início automático
  - void commit();
  - void rollback();
  - void rollback(Savepoint savepoint);

Transações JDBC - exemplo

```
public void Transfere(Conta conta1, Conta conta2,
  double valor ) throws Exception {
  Connection conn = getConnection(); //abstracao
  conn.setAutoCommit( false );
  PreparedStatement stmt = conn.prepareStatement(
    "update conta set saldo = saldo + ?
     where id   = ?" );
  stmt.setDouble( 1, -valor );
  stmt.setInteger( 2, conta1.getId() );
  stmt.executeUpdate(); //retira X de conta1
  stmt.setDouble( 1, valor );
  stmt.setInteger( 2, conta2.getId() );
  stmt.executeUpdate(); // deposita X na conta2
  stmt.close();
  conn.commit(); conn.setAutoCommit( true );
  conn.close();
}
```

Obtendo a conexão

- Modo clássico:
  - carrega classe do driver
  - obtém conexão passando URL de conexão
- Através de DataSource (JDBC 2.0 stdext)
  - obtém dataSource (tipicamente via JNDI)
    - pode ser mantido para uso posterior
  - chama dataSource.getConnection()
- Vantagens DataSource:
  - permite uso de pool de conexões
  - facilita manutenção
  - automatiza participação em transações distribuídas

Exemplo DriverManager

```
static String DRIVER = “com.mysql.jdbc.Driver”;
static String URL = “jdbc:mysql://localhost/mydb”
static String USER = “sa”;
static String PASS = “”;

public Connection getConnection() throws
  Exception {
    Class.forName( DRIVER );
    return java.sql.DriverManager.getConnection(
      URL, USER, PASS );
}
```

Exemplo DataSource/Java EE

```
static String DS_JNDI = “jdbc/myDb”;
static javax.sql.DataSource ds;

public Connection getConnection() throws
  Exception {
    if ( ds == null )
      javax.naming.Context ctx =
        new javax.naming.InitialContext();
      ds = (javax.sql.DataSource) ctx.lookup(
        “java:comp/env/” + DS_JNDI );
    }
    return ds.getConnection();
}
```

Transações distribuídas

- Problema: e se conta1 e conta2 pertencerem a 2 bancos diferentes?
- Solução: transações distribuídas
  - transação T1 no BD do banco1 para sacar conta1
  - transação T2 no BD do banco2 para depositar conta2
  - coordenador de transações
    - garante que ambas falham ou ambas sucedem
- Ou ainda:
  - 1 transação global (TG)
  - 2 transações locais (TL1 e TL2)

X/Open – The Open Group

- X/Open Company (1984-1996)
  - consórcio de empresas para definição de padrões abertos
- Em 1996, fundiu-se com a OSF (Open Software Foundation)
  - resultado: The Open Group
- The Open Group criou/possui copyright de uma série de padrões e tecnologias:
  - Unix, LDAP, POSIX, CDE, Motif, DCE, X etc...
  - DTP (Distributed Transaction Processing)

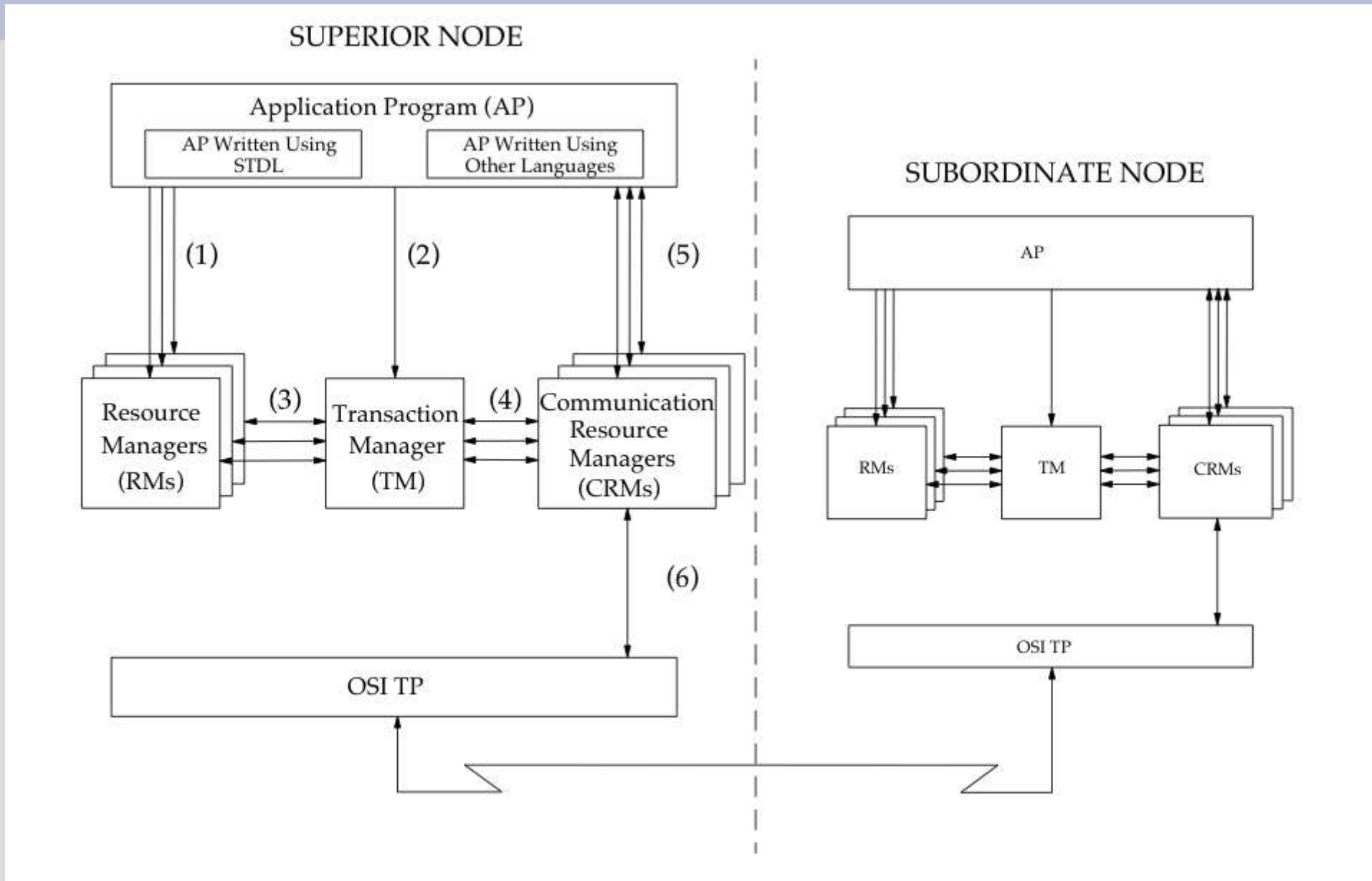
DTP – Distributed Transaction Processing

- Série de especificações
- Padrão da indústria para transações distribuídas
- Definições baseadas em sistemas UNIX e linguagem C
  - mas se aplicam a demais sistemas e linguagens
    - por exemplo, Java e COM/DCOM
- 2 especificações principais:
  - Distributed Transaction Processing: Reference Model
  - Distributed TP: The XA Specification

Arquitetura DTP

- Modelo DTP define 5 componentes
  - AP (User Application)
  - RM (Resource Manager)
  - TM (Transaction Manager)
  - CRM (Communication Resource Manager)
  - Communication Protocol (OSI-TP)
- Cada componente pode estar presente em 1 ou + nós
  - nó que inicia a transação é chamado nó superior
  - demais são nós subordinados

Diagrama DTP



Interfaces entre Componentes

(1) AP-RM
- Específica de cada RM

(2) AP-TM
- TX (Transaction Demarcation) Specification

(3) TM-XM
- XA Specification

(4) TM-CRM
- XA+ Specification

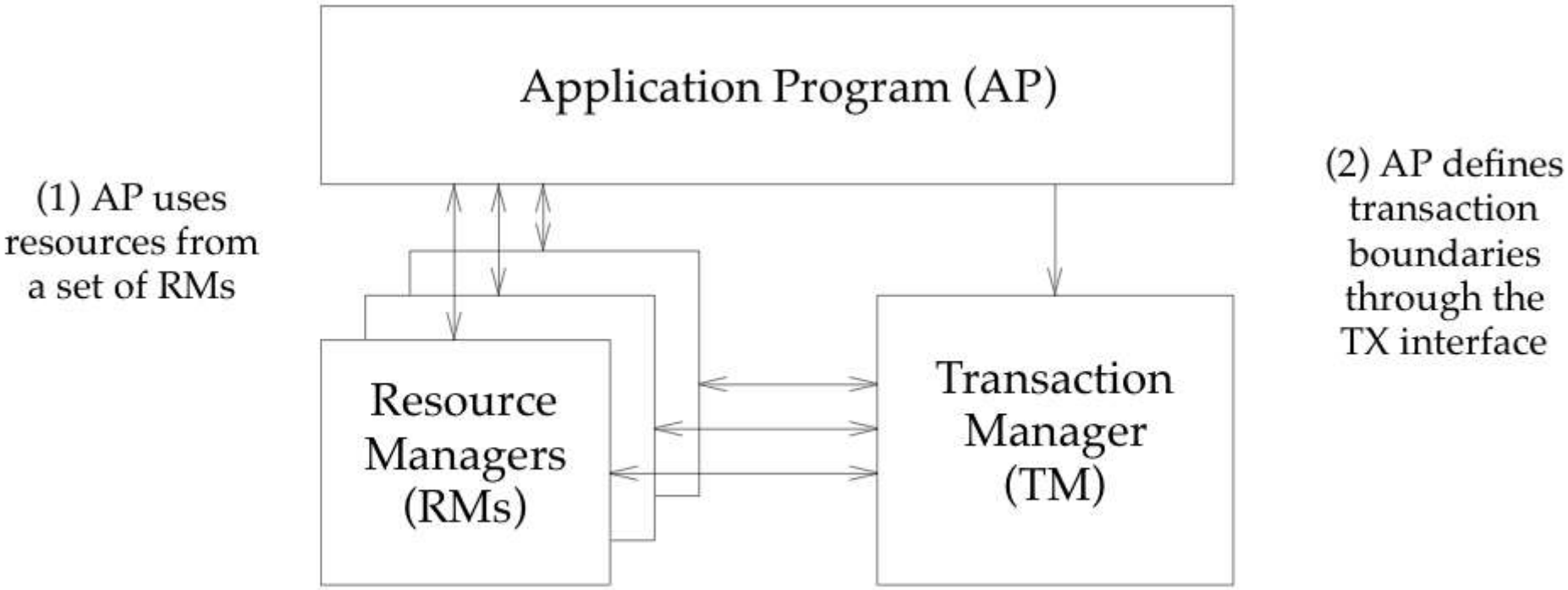
(5) AP-CRM
- XCPI-C Specification
- TxRPC Specification
- XATMI Specification

(6) CRM-OSI TP

Descrição dos Componentes

- AP (User Application)
  - aplicação que faz uso das transações e recursos
  - comunica tanto com o TM e RMs
- TM (Transaction Manager)
  - “orquestrador” da transação distribuída
  - comunica com os RMs
- RM (Resource Manager)
  - gerencia transações locais
  - exemplos: SGBD, Sistema de Arquivos, ERP, Servidor de Impressão, Fila de Mensagens
- CRM e Communication Protocol
  - uso implícito

Diagrama XA


Application Program (AP)
(1) AP uses resources from a set of RMs
(2) AP defines transaction boundaries through the TX interface
Resource Managers (RMs)
Transaction Manager (TM)
(3) TM and RMs exchange transaction information

Conceitos

- Global Transaction
  - transação global/distribuída gerenciada pelo TM
  - iniciada pelo AP
- Transaction Branch / Local Transaction
  - sub-transação que envolve (apenas) um RM
  - múltiplas branches em uma global transaction
  - usa TPC (two-phase commit)
- Thread of control
  - contexto da AP que define a transação
- XID
  - identificador da transação global

Exemplo transferência

- AP inicia transação global (TX: AP->TM)
  - xid = tx_begin()
- AP inicia transação SGBD1 (AP->RM)
  - SGBD1 registra-se no TM (XA: RM->TM)
    - ax_reg(id_sgbd1, xid)
  - TM avisa SGBD1 que transação iniciou (XA)
    - xa_start(id_sgbd1, xid);
- AP inicia transação SGBD2 (AP->RM)
  - SGBD2 registra-se no TM (XA: RM->TM)
    - ax_reg(id_sgbd2, xid)
  - TM avisa SGBD2 que transação iniciou (XA)
    - xa_start(id_sgbd2, xid);

Exemplo transferência (cont.)

- AP faz commit transação global (AP->TM)
  - tx_commit(xid)
  - TM executa fase 1 (prepare) do 2PC
    - xa_prepare(id_sgbd1, xid)
    - xa_prepare(id_sgbd2, xid)
  - TM executa fase 2 (commit) do 2PC
    - xa_commit(id_sgbd1, xid)
    - xa_commit(id_sgb2, xid)
  - SGBDs se liberam do TM
    - ax_unreg(id_sgbd1, xid)
    - ax_unreg(id_sgbd2, xid)

Recuperação de falhas

- Cada RM é responsável por garantir o seu estado
- TM deve garantir consistência após o término da fase 1
- Branches que causam rollback podem se liberar do TM antes da fase 2

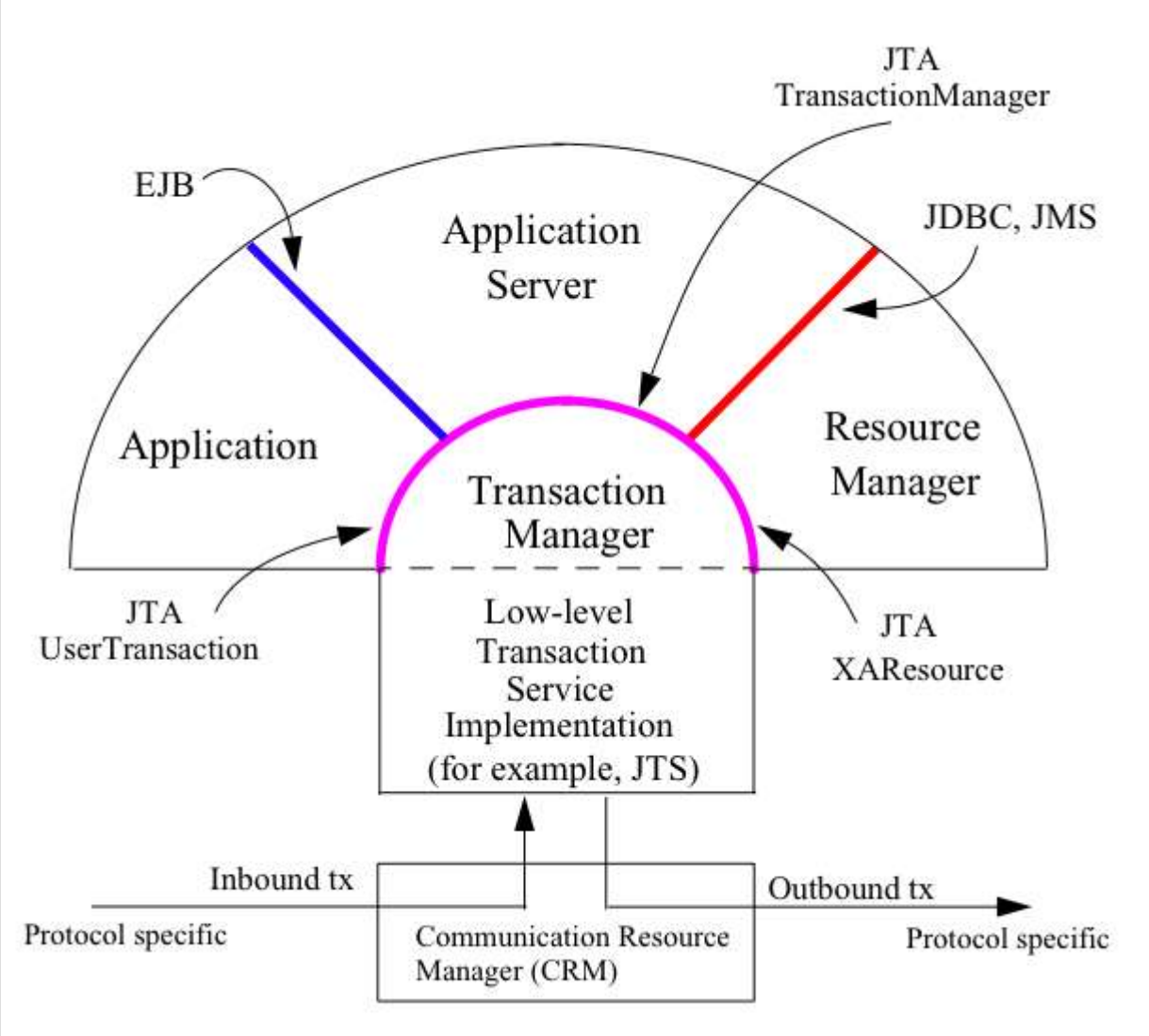
JTA – Java Transaction API

- Implementação dos conceitos DTP em Java
  - especialmente a especificação XA
- “Substitui” o communications protocol por um Application Server
  - faz o papel de TP monitor
  - automatiza vários passos do processo
- API opcional (extension) Java SE
  - apenas algumas interfaces e exceções presentes
  - para usar tudo, necessário uso JAR separado
- ...mas obrigatória em Java EE
- Define 2 pacotes:
  - Interface AP->TM (TX): javax.transaction
  - Interface RM->TM (XA): javax.transaction.xa

Principal Interface: javax.transaction.UserTransaction

```
interface UserTransaction {
  void begin();
  void commit();
  int getStatus();
  void rollback();
  void setRollbackOnly();
  void setTransactionTimeout( int
 seconds );
}
```

JTA - Diagrama



AP (Aplicação Java EE)

- Acessa transações globais através da interface javax.transaction.UserTransaction
  - obtida através de JNDI lookup: *"java:comp/UserTransaction"*
- Transação global associada ao contexto da thread
- Pode ser usada por qualquer componente Java EE (Servlet, EJB, JMS Listener, etc...)
  - EJBs oferecem gerenciamento automático de transações
    - CMT – Container Managed Transaction

TM (Transaction Manager)

- Tipicamente fornecido pelo próprio Application Server:
  - deve fornecer instâncias de javax.transaction.UserTransaction para APs
    - ... e opcionalmente de javax.transaction.TransactionManager
  - responsável também por retornar DataSources que implementem XADataSource
- Possível implementação: JTS
  - Java Transaction Service
  - interoperável com transações Corba

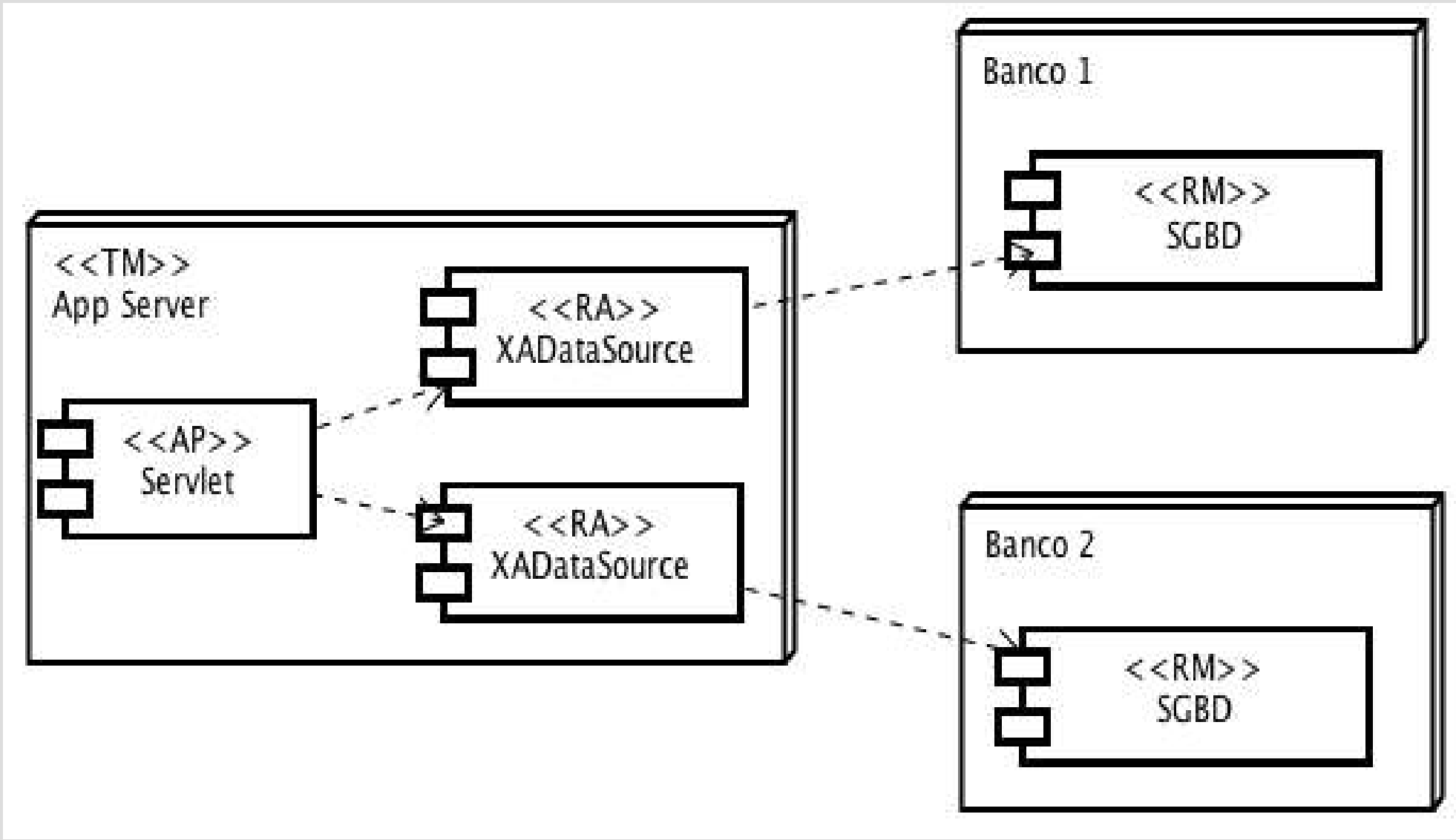
RM (Resources Manager)

- RM é acessado via RA (Resource Adapter)
- Exemplo:
  - SGBD -> RM
  - JDBC Driver -> RA
- No caso de SGBD, deve implementar interfaces definidas no JDBC 2.0 Standard Extension (ou JDBC 3.0) API:
  - javax.sql.XAResource
  - javax.sql.XADataSource
- Comunicação RM/TM é feita via implementação do JTA/XA

Exemplos

1. Servlet acessando 2 SGBDs
   - Uso explicíto de UserTransaction
   - Transação distribuída nos RMs apenas
2. EJBs encapsulando processos
   - Gerenciamento automático de transações
   - Transação distribuídas em componentes
3. Uso de mensagens (JMS)
   - Gerenciamento automático de transações
   - Transação distribuídas em componentes
   - Assincronismo

Exemplo 1 - Arquitetura



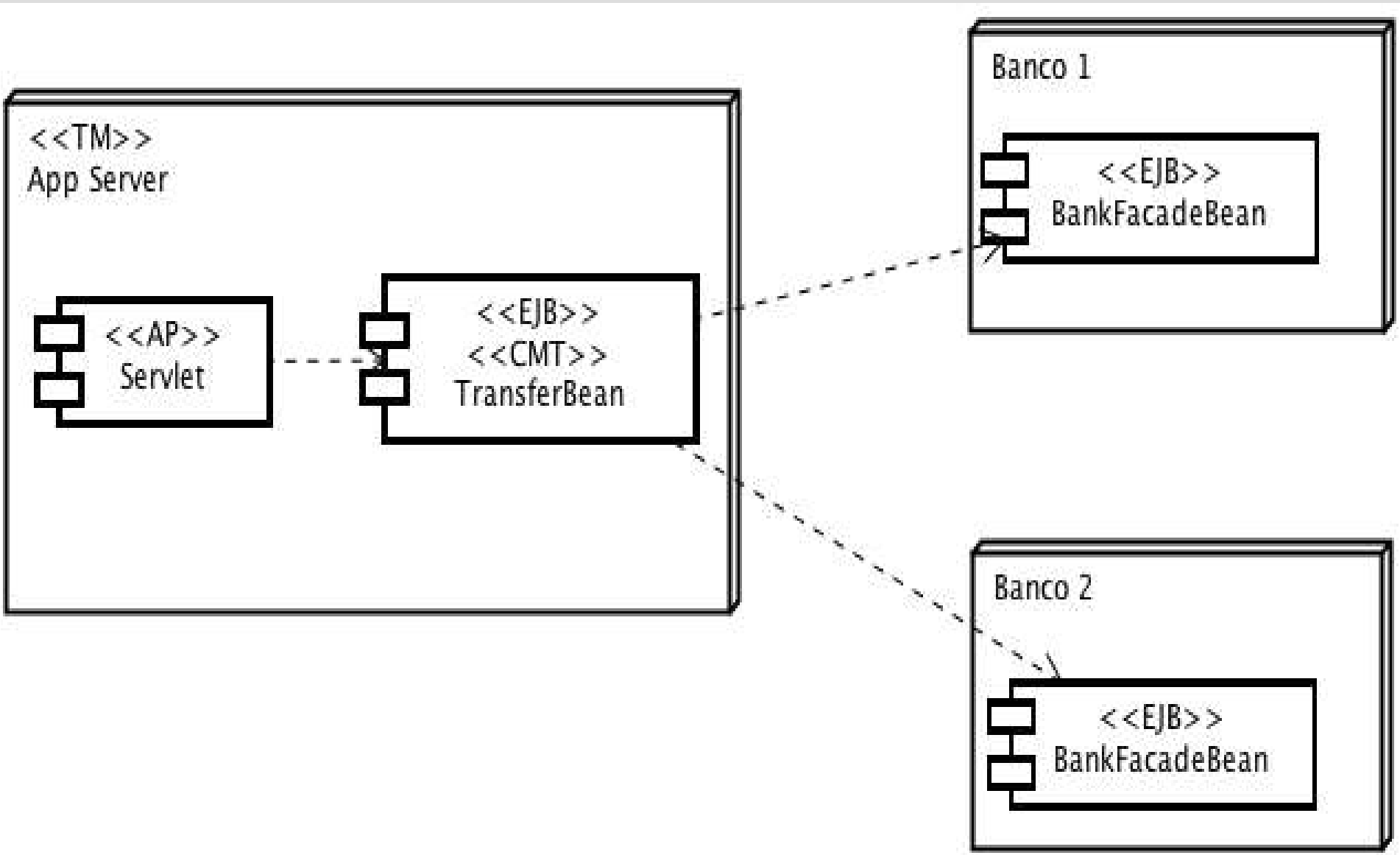
Exemplo 1 - Código

```
void doGet( ... ) {
  double valor = ...;
  int conta1 = ..., conta2 = ...;
  Context ctx = new InitialContext();
  UserTransaction tx = (UserTransaction)
    ctx.lookup("java:comp/UserTransaction");
  tx.begin();
  adiciona( "jdbc/banco1", conta1, -valor );
  adiciona( "jdbc/banco2", conta2, valor );
  tx.commit();
}
```

Exemplo 1 – Código (cont.)

```
void adiciona( String nome, int conta,
  double valor ) throws Exception {
   Context ctx = new InitialContext();
   DataSource ds = (DataSource)
     ctx.lookup("java:comp/env/" + nome);
   Connection conn = ds.getConnection();
// conexão já está enlisted na TX global!
   // executa e fecha statement
// é feito commit na transação local
   conn.commit();
   conn.close();
}
```

Exemplo 2 - Arquitetura



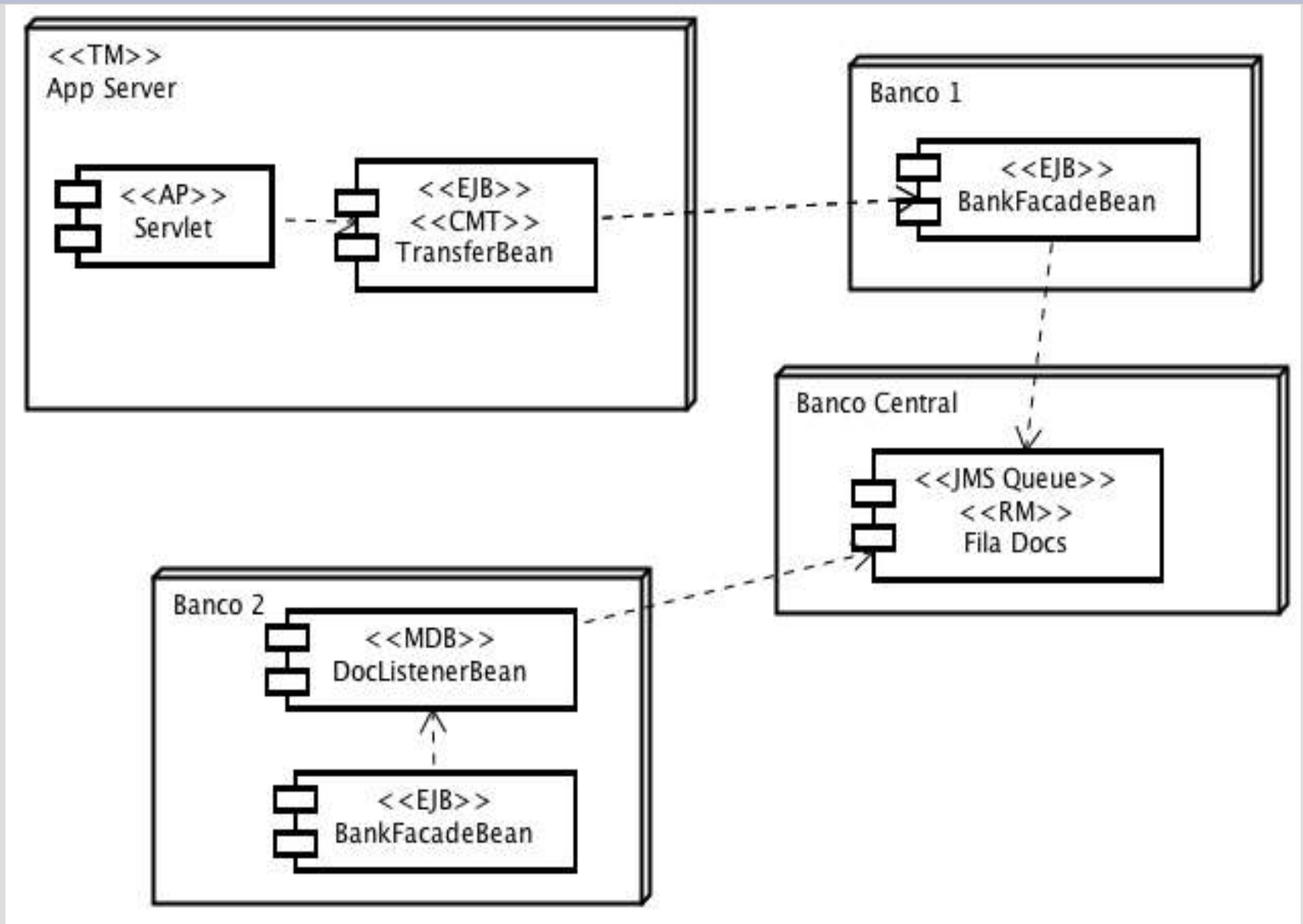
Exemplo 2 – Código (Servlet)

```
void doGet( ... ) {
  double valor = ...;
  int conta1 = ..., conta2 = ...;
  Context ctx = new InitialContext();
  TransferBeanHome home = (TransferBeanHome)
    ctx.lookup
  (“java:comp/env/ejb/TransferBean”);
  TransferBean bean = home.create();
// Servlet não usa transações
  bean.transfere( conta1, conta2, valor );
}
```

Exemplo 2 – Código (EJB)

```
void transfere( int conta1, int conta2,
  double valor ) {
// TX begin (implicitamente)
   Context ctx = new InitialContext();
   BankFacadeHome home1 = (BankFacadeHome)
     ctx.lookup("java:comp/env/ejb/Bank1");
   BankFacadeBean bank1 = home1.create();
   // obtém bank2 similarmente
// TX propagada para bank1
   bank1.retira( conta1, valor );
// TX propagada para bank2 automaticamente
   bank2.deposita( conta2, valor );
// TX commits (implicitamente)
}
```

Exemplo 3 - Arquitetura



Exemplo 3 – Código (Servlet)

```
void doGet( ... ) {
  double valor = ...;
  int conta1 = ..., conta2 = ...;
  Context ctx = new InitialContext();
  TransferBeanHome home = (TransferBeanHome)
    ctx.lookup
  (“java:comp/env/ejb/TransferBean”);
  TransferBean bean = home.create();
// Servlet não usa transações
  bean.fazDoc( conta1, conta2, valor );
}
```

Exemplo 3 – Código (TransferBean)

```
void transfere( int conta1, int conta2,
  double valor ) {
// TX1 begin (implicitamente)
   Context ctx = new InitialContext();
   BankFacadeHome home = (BankFacadeHome)
     ctx.lookup("java:comp/env/ejb/Bank1");
   BankFacadeBean bank = home.create();
// TX1 propagada automaticamente para bank1
   bank.fazDoc( conta1, conta2, valor );
// TX1 commit (implicitamente)
}
```

Exemplo 3 – Código (BankFacadeBean)

```
void fazDoc( int conta1, int conta2, double
  valor ) {
// thread participa de TX1 (implicitamente)
 // faz a retirada usando próprio EJB
 retira( conta1, valor );
 // faz DOC assincronamente
 DocMessage mensagem =
    new DocMessage( conta2, valor );
 Context ctx = new InitialContext();
 Queue docQueue = (Queue)
  ctx.lookup("java:comp/env/jms/DOCQueue");
// mensagem é colocada na fila durante TX1
 queue.send( mensagem );
}
```

Exemplo 3 – Código (DocListenerBean)

```
void onMessage( Message jmsMsg ) {
// TX2 begin (implicitamente)
  DocMessage mensagem = (DocMessage) jmsMsg;
  double valor = mensagem.getValor();
  int conta = mensagem.getConta();
  Context ctx = new InitialContext();
  BankFacadeHome home = (BankFacadeHome)
    ctx.lookup("java:comp/env/ejb/MyBank");
  BankFacadeBean mybank = home.create();
// TX2 propagada automaticamente para mybank
  mybank.deposita( conta, valor );
// TX2 commit (implicitamente)
}
```

Referências Java

- JDBC
  - http://java.sun.com/products/jdbc/
  - http://java.sun.com/docs/books/tutorial/jdbc/index.html
  - http://java.sun.com/docs/books/tutorial/jdbc/basics/transactions.html
- JTA
  - http://java.sun.com/products/jta/
- JTS
  - http://java.sun.com/products/jts/

Referências DTP

- Distributed Transaction Processing: Reference Model, Version 3
  - http://www.opengroup.org/bookstore/catalog/g504.htm
  - http://www.opengroup.org/onlinepubs/9294999599/toc.pdf
- Distributed TP: The XA Specification
  - http://www.opengroup.org/publications/catalog/c193.htm
  - http://www.opengroup.org/onlinepubs/9698909699/toc.pdf
- Distributed TP: The XA+ Specification, Version 2
  - http://www.opengroup.org/bookstore/catalog/s423.htm
  - http://www.opengroup.org/onlinepubs/8095979699/toc.pdf

Referências DTP

- Distributed TP: The TX (Transaction Demarcation) Specification
  - http://www.opengroup.org/bookstore/catalog/c504.htm
  - http://www.opengroup.org/onlinepubs/9694999599/toc.pdf
- Distributed TP: The XCPI-C Specification, Version 2
  - http://www.opengroup.org/publications/catalog/c419.htm
  - http://www.opengroup.org/onlinepubs/9695989099/toc.pdf
- Distributed TP: The TxRPC Specification
  - http://www.opengroup.org/bookstore/catalog/c505.htm
  - http://www.opengroup.org/onlinepubs/9694999499/toc.pdf
- Distributed TP: The XATMI Specification
  - http://www.opengroup.org/bookstore/catalog/c506.htm
  - http://www.opengroup.org/onlinepubs/9694999399/toc.pdf

Dúvidas ???

Felipe Leme

felipeal@gmail.com
ra930886@ic.unicamp.br
http://felipeal.net