Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (01223) 515010**
INTERNATIONAL: **+44 1223 515010**
FAX: **+44 1223 359779**
E-MAIL: **apm@ansa.co.uk**

Training

ANSAwise - Transactions in Distributed Systems

Chris Mayers

Abstract

Organizations wish to preserve their investments in IT, and databases are a case in point. They also wish to exploit new database technology if it is cost-effective to do so.

Real-world distributed systems will inevitably involve interfacing with databases of various types and ages (hierarchical, network, relational, post-relational/object-oriented). However, the database community has a rather different view of how distribution should be handled.

This module of the ANSAwise training programme compares and contrasts these different views, and outlines the key standards in the database world, and suggests how interfaces between distributed systems and databases can be architected.

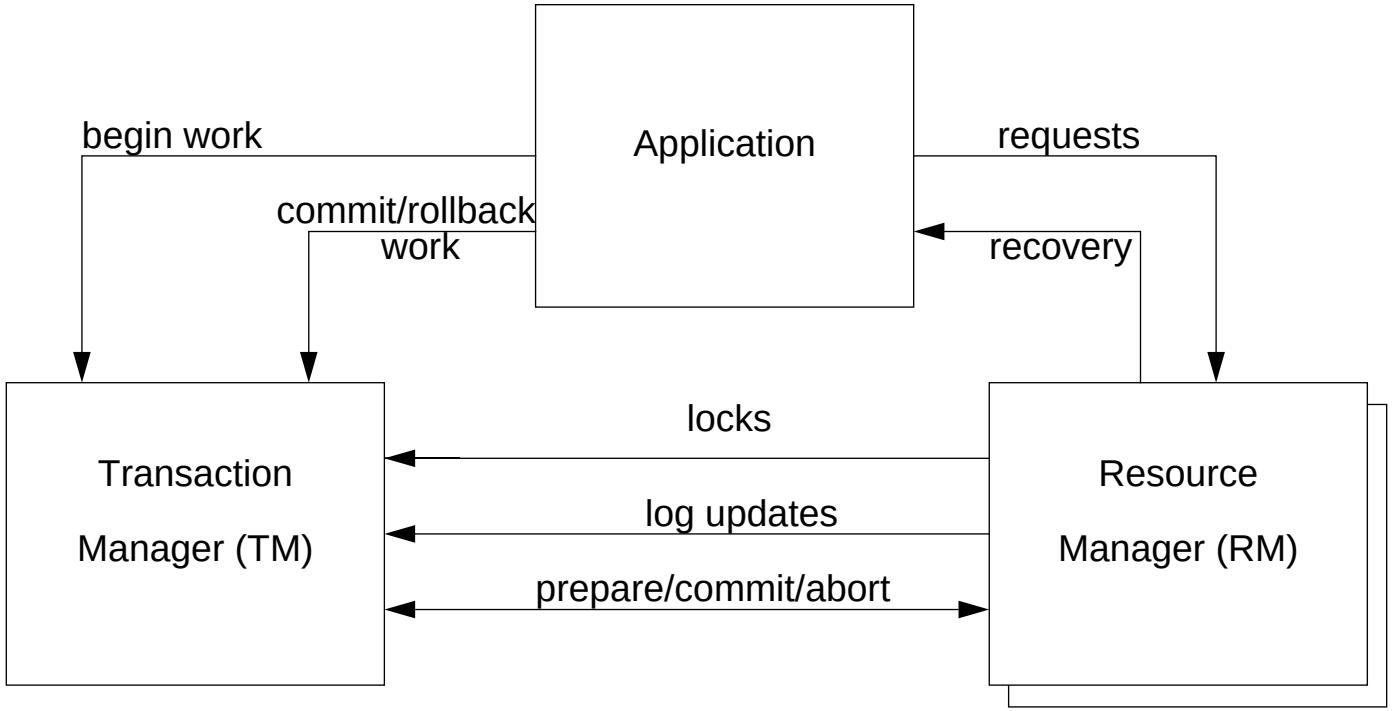
This session ends with a discussion of open distributed transaction processing, including the CORBA transaction service.

[This is a variant of APM.1461, produced for CNET.]

APM.1629.02 **Approved** 1st April 1996
Briefing Note

Distribution:
Supersedes:
Superseded by:

Transactions in Distributed Systems



In this session

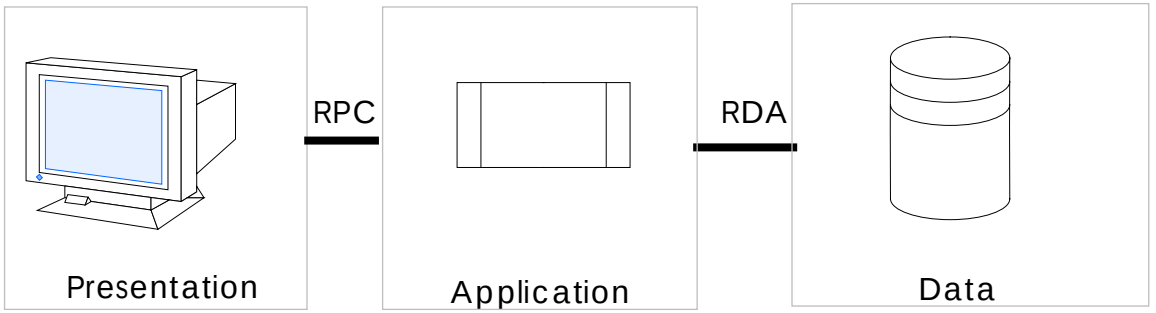
- Explain the Remote Data Access (RDA) technique
- Explain the Remote Procedure Call (RPC) technique
- Show how they can be used together to support templates for application partitioning
- Examine open standards in distributed database...
- ... and in transaction processing

Two worlds

- Two approaches to distributed systems are predominant
  - Remote Data Access (RDA), generally used for databases
  - Remote Procedure Call (RPC), generally used for distributed computing
- RDA is based on the SQL structured query language
  - usually to access a relational database
- RPC is based on IDL interface definition languages
  - usually to access an application server
- Both are ‘client-server approaches’...
- ...Each has unique advantages
  - How can applications take advantage of both?

RPC and RDA for application servers

- **RPC from first to second tier**
- **RDA from second to third tier**



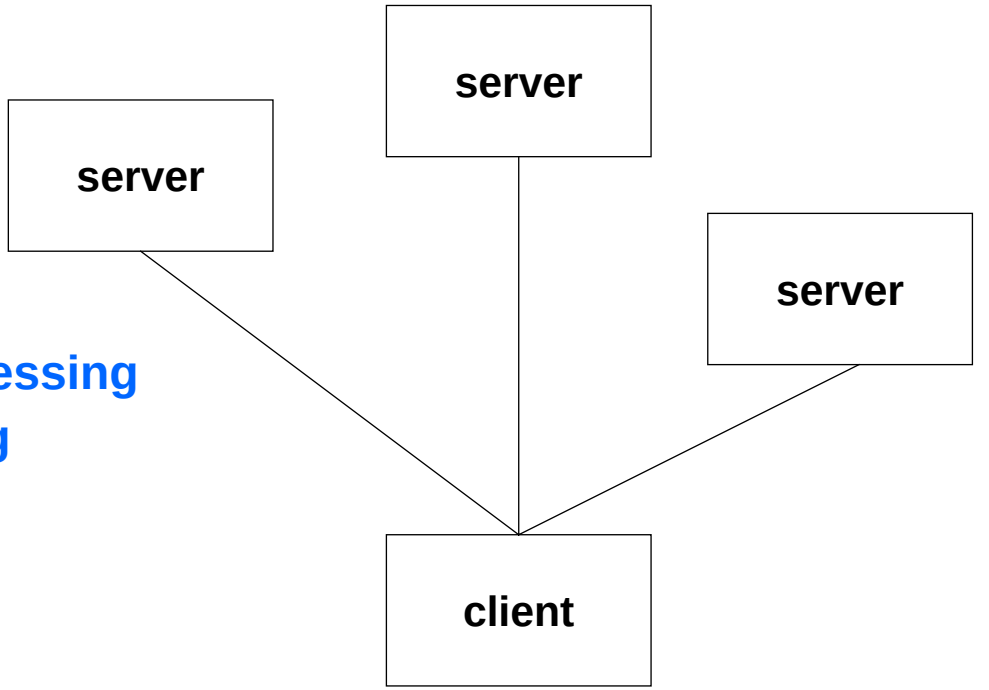
- **... there are other possible combinations too**

Remote Data Access (RDA) Overview

- Application interacts with database using SQL statements
- Statements can retrieve, update, delete, or insert records:
  - a single record
  - a set of records at once
  - a set of records, one at a time
- Statements can be grouped into transactions
  - a transaction succeeds or fails as a whole
- The location of the remote database is transparent

RDA technique

- **Basic Features**
  - Transparent remote SQL
  - Dialog interaction
  - Dynamic SQL
- **Enhancements**
  - Distributed transaction processing
  - Distributed query processing
  - Static SQL
  - Stored procedures



RDA Advantages

- **Flexibility for clients to define application-specific views and high level operations**

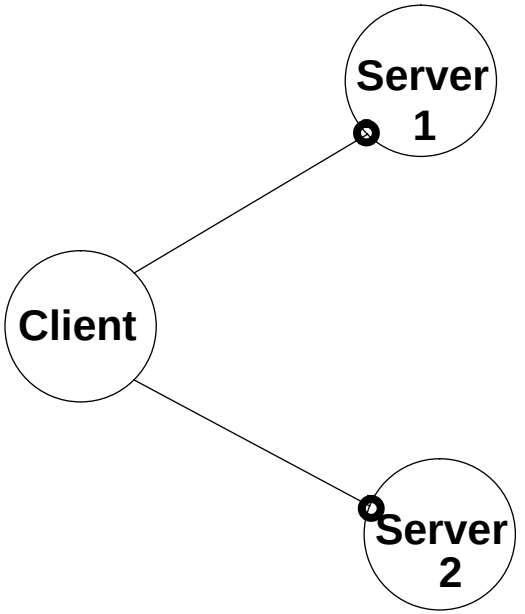
- **Vendor-supported interfaces to application development tools**
  - **data browsers**
  - **report writers**
  - **4GLs**
  - **spread sheets**
  - **graphing packages**
  - **...and so on**

Remote Procedure Call (RPC) Overview

- Application interacts with object using a remote procedure call defined in IDL
- Statements invoke individual operations defined in IDL
- Remote procedure call can do anything an ordinary procedure call could do
- Transaction services can be used if available
- The location of the remote object is transparent

RPC technique

- **Basic features**
  - transparent remote procedures
  - “one-shot” interaction
  - static procedures
- **Enhancements**
  - “transactional RPC” (dialog interaction)
  - distributed transaction processing
  - dynamic procedure definition



RPC Advantages

- **High level of abstraction**
  - **client application programmer convenience**
  - **semantic integrity protection**
  - **low communications traffic**
- **Static compilation**
  - **efficiency**
  - **automated type checking**
- **Applicability to non-database services**
  - **for example, device control**

IDL and SQL

- IDL defines only the interface; the logic can be written in any programming language
- ... SQL defines the logic; it can be executed
- There are various different IDL syntaxes (for example, CORBA IDL and DCE IDL)...
- ... there is one SQL syntax

CORBA IDL (Interface Definition Language)

- **A simple service (Echo)**

```
interface Echo {
// Comment lines start with two slashes

   string Echo (in string Src);
   string Reverse (in string Src);
};
```


SQL (Structured Query Language)

- **A simple SQL statement**

```
SELECT  employee_name, title, commission
FROM    employees
WHERE   employee_name IN
        (SELECT    employee_name
        FROM       employees
        WHERE      commission > 8000
        AND        title = 'account_manager');
```

Evolution of SQL

- 1986: SQL-86
  - now obsolete
- 1989: SQL-89
- 1992: SQL-92
  - Entry SQL: roughly equivalent to SQL-89
  - Intermediate SQL: what most database vendors already supported
  - Full SQL: other well-understood features
- 1998?: SQL3
  - parts of SQL3 will appear earlier, as addenda to SQL-92

Some features of SQL3

- **Stored procedures**
  - **‘computational completeness’**
  - **triggers and assertions**
- **Object data management**
  - **encapsulation, polymorphism, inheritance**
  - **multiple inheritance, generalization/specialization hierarchies**

SQL Diversity

- **Each vendor's SQL has restrictions and extensions**
  - **you will probably have to suffer some restrictions and use some extensions...**
  - **...avoiding vendor lock-in requires care**

Date's 12 Rules for Distributed Database - Autonomy

- 1. Local autonomy
  - administrators retain control over their local databases
- 2. No reliance on a central site
  - all processing is decentralized
- 3. Continuous operation
  - cannot shut down the entire system

Date's 12 Rules for Distributed Database - Distribution Transparency

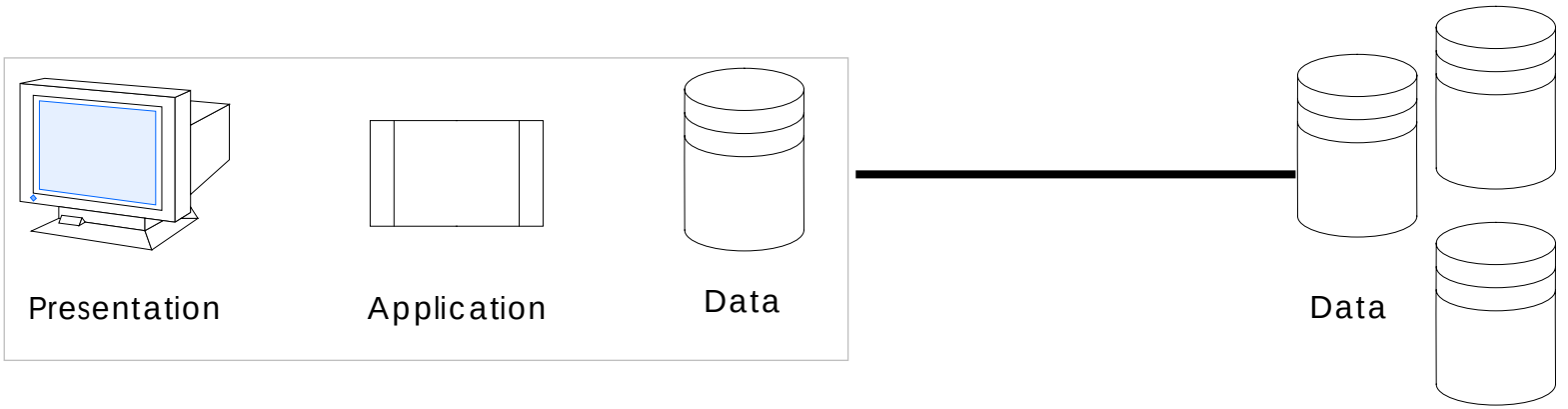
- 4. Location independence (transparency)
- 5. Fragmentation independence (transparency)
- 6. Replication independence (transparency)
- 7. Distributed query processing (optimization)
- 8. Distributed transaction management

Date's 12 Rules for Distributed Database - Heterogeneity

- **9. Hardware independence**
  - one distributed database can include different hardware platforms
- **10. Operating system independence**
  - one distributed database can include different operating systems
- **11. Network independence**
  - one distributed database can span LANs/WANs with different network topologies and protocols
- **12. DBMS independence**
  - one distributed database can include DBMSs from different vendors

Distributed Database

- **This template can support multiple databases transparently**



Distributed Database in the Real World

- Much R&D effort has been sunk into distributed databases
- No vendor has yet delivered a full distributed database solution
- ...However, practical partial solutions do exist

Distributed Database viewed by Distributed Systems

- The 12 rules imply complete (non-selective) distribution transparency
  - many applications don't require this; a partial solution may be enough
  - performance may be unacceptable
  - it is unlikely to be fully achievable, anyway
- There's an implicit assumption that all the databases are relational
  - this assumption would be unrealistic; 80% or more of a typical organization's data is in non-relational form
- There's a need to consider integration with emerging 'data persistence' technologies
  - post-relational and object databases
  - non-database distributed object stores

Database Middleware

- **“Database middleware” products currently aim at a rather different set of issues from distributed systems**
  - **heterogeneous remote data access**
  - **gateways to mainframe databases**
  - **database-to-database extraction**
- **Products supporting query access only (not updates) already provide reasonable performance and stability**

Microsoft ODBC (Open Database Connectivity)

- De-facto specification; ODBC 1.0 specification issued in 1991
- Tackles the problem of heterogeneous remote data access
- Widespread support from database vendors
- Widespread support from application vendors (at least 170 products)
- Supports non-relational as well as relational data
- Also incorporated into Apple Data Access Manager (DAM)

Distributed Systems viewed by Distributed Database

- Any application partitioning introduces complexity
- RPC isn't necessary for small-scale client/server systems
- RPC is useful across a broad range of applications
- RPC/application server development products are unproven
  - products are immature...
  - ...vendors must provide evidence that their products enable rapid development of large-scale heterogeneous client/server systems

Transactions - the ACID properties

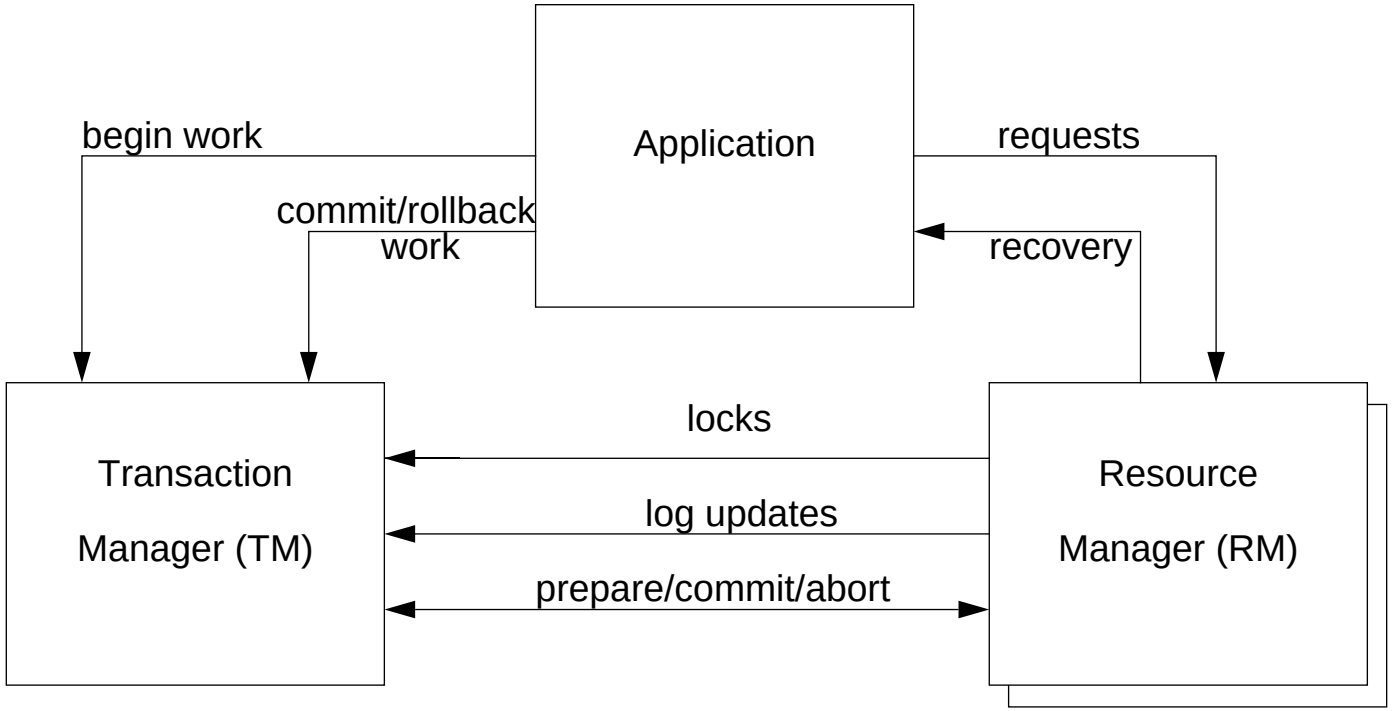
- ***A*tomicity**
  - **all-or-nothing**
- ***C*onsistency**
  - **transactions must be self-contained logical units of work**
- ***I*solation**
  - **concurrent transactions must not affect each other**
- ***D*urability**
  - **what's done must not be undone**

Distributed Transaction Processing

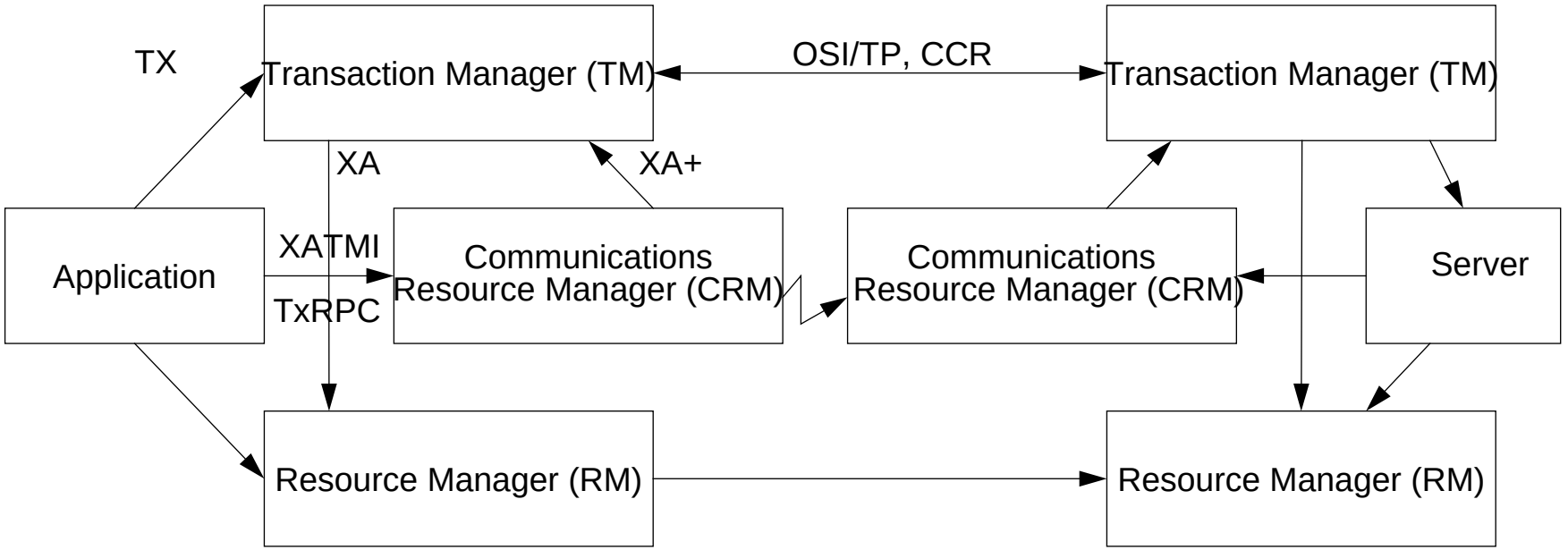
- **In a distributed system, how are transactions involving multiple databases coordinated?**

- **What about transactions involving operations other than database update?**

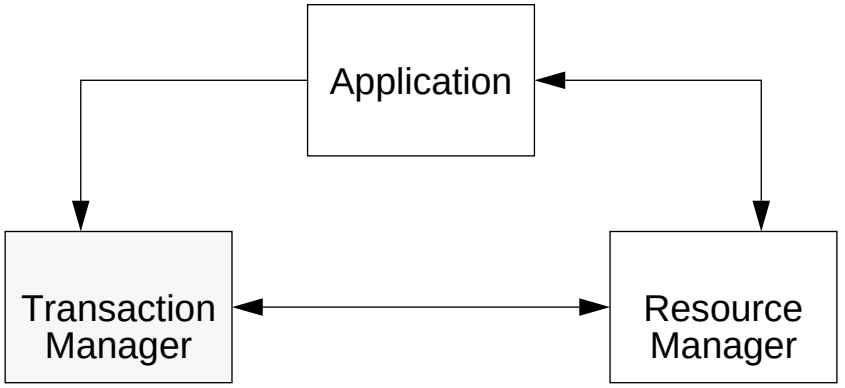
Typical Open TP Core Components (much simplified)



X/Open and ISO TP Model



Example Transaction Managers



- **Encina (Transarc, now owned by IBM)**

- **Tuxedo (Unix Systems Laboratories, now owned by Novell)**

Encina (IBM/Transarc)

- **Application Interface - Transactional C (proprietary language extension)**
- **Transaction Model - Nested**
- **Communications - Transactional RPC (TRPC)**
- **Infrastructure - OSF DCE**

Tuxedo (Novell/Unix Systems Laboratories)

- Application Interface - ATMI (C, C++, COBOL, 4GLs)
- Transaction Model - Flat
- Communications - Tuxedo
- Infrastructure - Most Unix

CORBA Transaction Service

- Is an Object Service
  - see *CORBAservices* by the Object Management Group (Wiley)
- Specifies transactional and recoverable objects
  - similar to Resource Managers in the X/Open DTP model
- Tracks object usage automatically across threads
- Supports flat and nested transactions
- Supports legacy TP standards and protocols
  - see Appendix A of the Transaction Service Specification

Summary

- RPC and RDA can be used together to build large-scale client/server systems
- Distributed databases, even incomplete, may be part of the solution
- Distributed databases alone are unlikely to be sufficient for large-scale client/server systems
- Database standards are continuing to evolve rapidly
- Distributed transaction processing support is needed whether RPC, RDA, or distributed databases are used