REXX, Distributed Systems and Objects

John Tibbetts
Kinexis

Pages 174-193
Rexx, Distributed Systems and Objects

John Tibbetts
Kinexis
(415) 558-9277
e-mail: john@kinexis.com

Rexx Symposium
May 2, 1995
Rexx, Distributed Systems and Objects

- Rexx + Client/Server Database
  - Simple architecture for simple C/S apps
- ORexx + SOM
  - Beginning of strong client/server platform
- Current technology (ORexx)
  - Functions as SOM requester
  - Adequate for client-side activity
- Coming technology
  - Exporting OREXX classes as SOM classes
  - Scripting language for OpenDoc
  - Suitable as server platform
Our approach...

- Discuss paradigm issues
  - Evolution of distributed architectures in Four Phases
- Discuss transaction issues
  - Agenda of TP
- Examine Rexx C/S implementation strategies
Computing Architecture Phases

1. Centralized
2. Clients to Database Server
3. Clients to Function Server
4. Objects

*Code* and *Data* in varying combinations
Phase 1. Centralized Computing

- Strong control & manageability
- Good security
- Weak user empowerment
- Weak on distributed computing
- Limits business "reach"
Phase 2. Clients to Database Server

- Power to the user
- Power to the user interface
- Uneven performance and integrity
- Weak 3-tier architecture
- Trust problems
Phase 3. Clients to Function Server

- Improved performance and integrity
- Stronger 3-tier architecture
- Trust tuning
- *But* significant software complexity
Phase 4. And Then There Are Objects...

- Inately partitioned
- Semantic continuity
- Limited transactional awareness

An Object is *Data* surrounded by a protective layer of *Code*
Transaction = “The Deal”

- **In clay**
  - Baked invoices at Ebla (3rd millennium BC)

- **On paper**
  - Sales orders and invoices
  - Double-entry ledgers
  - Contracts and deeds

- **Online**
  - Reservations for travel, hotels, cars, etc
  - Banking & stock trading documents
  - Order entry, inventory planning, accounting
  - Telephone call setup and billing, email
ACID Test for Transactions (And All Deals)

- **Atomicity**
  - Transactions are “all or nothing” (integrity principle)
  - Wedding vows (two-phase commit)

- **Consistency**
  - Transactions are a correct transformation of state
  - Debits = credits

- **Isolation**
  - Concurrent transactions behave as if executed serially
  - Transactions don’t see other transactions partial results

- **Durability**
  - Once committed, transactions are not forgotten
  - Bound to honor COMMITments

Transactions are the computer equivalent of contract law
The Transactional Discipline

Non-transactional: state changes continuously

Problem State

Solution State

Transactional: orderly, coordinated, audited state change

Problem State

Solution State
How TP Monitors are organized

RM = resource manager

Copyright Kinexis 1988-1993. All rights reserved.
Full-Fledged TP: X/Open DTP Model

Application

Native

Resource Manager (RM)

TX

Trans Mgr (TM)

Native

Comm Manager (CM)

XA

XA+
TP-Lite: Transactions Inside Database

- Today's client/server databases bundle TM and DM together

- TM should be *unbundled* for open systems
  - Coordinate multi-vendor DBMS
  - Coordinate user-written function
  - Coordinate other resources
Imagine Transactional Objects

- Objects distributed about network
- Send messages to Debit savings acct object Credit load account
- Commit changes all object states
- Simultaneous to multiple consumers

Objects are microscopic Resource Managers: *Subsystem driven by a formal API that has state.*
OMG Transactional Object

Transactional Client

Demarcation

Propagation
Transaction Operation

Registration

Transaction Service

Transactional Object

Propagation
Transaction Operation

Registration

Transaction Service

Recoverable Object

Registration
### Mapping Paradigm & Transactionality

<table>
<thead>
<tr>
<th>Non-TP</th>
<th>Phase 1</th>
<th>Phase 2</th>
<th>Phase 3</th>
<th>Phase 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monolithic program</td>
<td>Any client/server DBMS</td>
<td>RPC Msg Queue Sockets</td>
<td>CORBA, DSOM, COM, DOE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Monolithic program under TP: CICS, IMS, Guardian, ACMS</td>
<td>Any client/server DBMS</td>
<td>Dist TP: TRPC, TMQ, LU6.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CORBA (w/OTS), DSOM, (COM)</td>
</tr>
</tbody>
</table>

17
Steps to Distributed, then Transactional, Objects

1. **Compatibility** among differing object models in same machine
   - CORBA (coarse-grain)
   - SOM (fine-grain)

2. Distributed homogeneous objects
   - CORBA
   - DSOM

3. Distributed heterogeneous objects
   - CORBA 2.0
   - DSOM

4. Distributed transactional objects
   - CORBA w/OTS
Rexx implementation strategies

- Rexx or ORexx Client to Client/Server database
  - Phase 2 or Phase 4/2 hybrid
- Rexx or ORexx Client to Function server
  - Phase 3 or Phase 4/3 hybrid (non-transactional)
- Rexx or ORexx Client to TP Monitor (eg. CICS ECI)
  - Phase 3 or Phase 4/3 hybrid (transactional)
- ORexx Client to DSOM
  - Phase 4 (non-transactional)
- ORexx modifying Server behavior
  - Phase 4 (non-transactional)
- ORexx Client or Server with ORB transaction services
  - Phase 4 (transactional)