Enterprise Applications Introduction

Microsoft Transaction Server

- Problem
  - Programs that use begin/end transaction cannot be composed easily
  - Transactions may involve multiple machines and distributed computation
  - How do transactions and objects interrelate?
- Need for
  - Compositional distributed transactions

MTS – Approach

- Declare certain classes as *transactional*
  - new/require/support transaction
- Unify object and transaction lifetime
  - first new/required object starts transaction
  - supporting objects enlisted in transaction
  - transaction commits when main object is freed
- Resource dispensers track operations
  - database, email, message queue, (file system)
- Distributed 2-phase commit
- No explicit entity-relational mapping

Microsoft Transaction Server

- Implemented as class wrappers
  - replace class factory for transactional objects
  - modify resources (e.g. databases) to be implicitly aware of transaction
- Evaluation
  - Good model of *orthogonal transactions*
- Basis for design of EJB
  - session beans = MTS transactional objects
  - entity beans were added
    - (have to be different in some way)
    - Used for entity-relational mapping

Example Application

- Allegis E-Business Suite
  - Partner Relationship Management
- Issues
  - large application (lots of kinds of data)
  - rapid development
  - integration with existing systems
  - flexible customization
  - flexible configuration
  - 2 or 3 new versions a year for 3 years

Allegis E-Business Suite

- Approach
  - Used MTS for transactions
  - Tried but rejected object-relational mapping
  - Extended declarative approach
    - data model (similar to OQL)
    - UI model
    - security model
    - workflow model
    - rule-based triggers
      - used procedural languages for writing triggered actions

Data Model

- 307 entity types
  - 311 parent relationships
- 982 data fields
- 665 relationships
  - 177 M:M relationships
  - 488 M:1 relationships
- Physical
  - 488 tables
  - 2131 columns

User Interface

- Approximately 600 page types
  - each with many variations
- Typical pages
  - for each entity type

• overview	summarizes info on records
• create	often a wizard
• search	often multiple related entities
• list	search results, selectable columns
• edit	tabs show partial views
• reporting	"slice and dice" summary pages

- Some special cases
  - Lead distribution
  - De-duplication
  - Bulk upload/download

Search Pages

- Arbitrary conjunctions of conditions
  - string starts with/contains
  - number/date min and/or max
- Related entities
  - Search based on existence of related object that meets conditions
  - Example: Accounts, Contacts, Leads
    - Account name contains “acme”
    - Lead value > 1,000
    - Contact named “Anshu”
    - Return accounts
      - find accounts where there exists lead > 1,000 and the account has a contact who named contains Anshu

List Pages

- Selectable columns
  - Small part of object usually displayed
    - (dynamically chosen)
  - Show detail on related objects
- Chasing multiple levels of sub-objects
  - People + their projects + milestones
  - Naïve approach requires many queries
- Paging
  - Always a problem

Edit Pages

- Concurrency
  - Optimistic between page load and store
    - If timestamps change, user is shown both updated data and their unsaved changes
    - Works uniformly on all pages/fields
  - Pessimistic actually performing update
- Any page can support editing
  - Particularly useful on list pages

Lead Distribution

- Distribute *leads* to *resellers*
  - user-specified criteria
    - p1(lead) -> q1(reseller) and m1(lead, reseller)
    - else p2(lead) -> q2(reseller) and m2(lead, reseller)
    - …
  - lead conditions p1, …, pN
    - specified by lead search criteria
  - reseller conditions q1,…, qN
    - specified by reseller search criteria
  - matching conditions m1,…,mN
    - based on fields that reseller and lead have in common
      - address, products, etc
- Batch process
  - may be 10,000 or more leads in a batch
  - 50,000 resellers
  - 50 distribution criteria
  - how fast can you distribute them?

Projects

- Topics
  - Comparison
  - Language Design
  - Survey
  - Benchmarks
  - Theory
  - Implementation
  - Others?…
- Output
  - 10-page report
    - or
  - 5 page report and implementation
- Proposal by 10/23

Comparison

- 2-way Comparison
  - Implement small application 2 of using
    - Object-relational mapper
    - PJama
    - JDO
  - Compare performance and subjective factors
  - Small application can be
    - Address book
    - Simplified Petshop
    - etc

Language Design

- Integrating OQL into..
  - Design a language extension to integrate variant of OQL into Java / Haskell
    - Could be same style as Meijer paper, but using high-level object definitions instead of relations
- XML
  - Use transitivity of XML encodings to bridge between PL and DB
    - PL ß à XML ß à DB è PL ß à DB

Projects

- Survey
  - Write a paper that focuses on clarifying the issues and problems in PL/DB integration, rather than presenting a solution
- Benchmark
  - Do an initial design of a better benchmark for PL/DB integration. Not just type systems, and not just performance

Implementation

- OQL -> SQL
  - write an OQL to SQL translator
  - Use mini-OQL, not full language
- Clustered Read
  - Extend a persistence model (O/R mapping, OPJ) to have a notion of clustered read
    - Use some form of "hint" or "tag" to keep track of which transactions involve clusters

Theory

- Formalize Meijer's type system proposal
- Show confluence of comprehension transformations
- Apply criteria from Type Survey paper to two other languages' type systems