Integrating Programming Languages & Databases

Introduction (continued)
& Background material

"Impedance Mismatch"

- Connecting PL and DB is hard because
  - Models don't match
    - Flat tables *versus* Complex objects
    - Declarative queries *versus* Procedural programs
    - Transactions *versus* Semaphores
  - Cultural mismatch
    - DP people don't understand PL research
      - "everything is a database"
    - PL people don't understand DB research
      - "lambda calculus is computationally complete"
  - And anyway, it's not our problem…
    - Industry will figure it out

A Definition of the Problem

- Approach to building persistent systems that satisfies:
  - High performance, scalable, reliable
  - Logical, clean model
  - Support for multiple, concurrent
    - Users (concurrency)
    - Machines (clustering, redundancy)
    - Developers (modularity)
  - Effective maintenance, evolution

Factors for Evaluating Solutions

- Technical (hard) metrics
  - Performance
  - Reliability
  - Scalability
  - Consistency
  - Correctness
- Human (soft) metrics
  - Modularity
  - Encapsulation
  - Development effort
  - Maintenance costs
  - Scalability of group
  - Clarity
  - Beauty
- Most solutions only address some of these factors

Many Dimensions of Scalability

- Amount of data
  - More than fits in memory
- Number of users
  - Hundreds, millions
- Complexity of problem
  - Not of solution
- Rate of change
  - Change in requirement
  - Ebay updates software weekly
- Number of developers
  - Software engineering issues

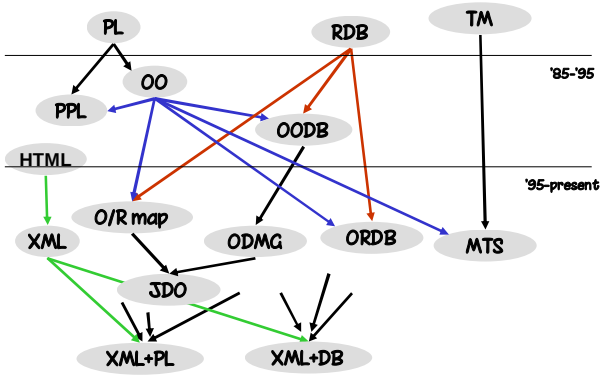
What Are Databases For?

- Search algorithm compiler
  - Queries specify what to find, not how
  - Optimizations
    - Content heuristics
    - Physical characteristics (e.g. page size)
    - Indexes, Etc…
  - Runtime compiler
- Concurrency control
  - Manage concurrent read/write
  - Transactions
  - ACID: Atomic, Consistent, Isolated, Durable

Approaches to PL/DB Integration

- Add DB features to PL
  - Persistent programming languages (PPL)
    - Focus on persistent state
  - Database programming languages (DBPL)
    - Focus on queries
- Add PL features to DB
  - Active databases (no papers at this point)
  - Object-Oriented Databases (OODB)
  - Object-relational database (ORDB)
- Put something between the two
  - Object-relational mapping (O/R mapping)
  - Maybe XML will help?
    - A common ground between the two?

History (reconstructed)



Roadmap to Reading

- Persistent Programming Languages
  - Make objects persistent
  - Trouble validating claim to be "better programming model" & deep logical questions
- Object-Oriented Databases
  - Originally: Make objects persistent in DB
  - Later: Standard OO Query/Data languages
  - Now: standard migrated to "JDO"
    - Confusion about what it is, standard not adopted
- Object-Relational Databases
  - Slight update to dominant relational standard
  - Incumbents fight back

Roadmap to Reading II

- Database Programming Languages
  - Integrated models of queries
  - What about transactions?
- Type Systems
  - For DBPL and OODB standard
  - Necessary, but not there yet
- Object-Relational Mapping
  - Fairly clean semantics, poor performance
  - Five new proposals a year for 10 years
- Middleware
  - A high-level model for *client* transactions
- XML
  - Trees are back in fashion; semi-structured data
  - Something both DB and PL people consider important

Research Topics

- What kind of research can we do?
  - Well-defined metrics
    - Performance, dataset size
  - Proofs
    - Semantics, type theory, correctness
- Result
  - Clear path to publication
- Comments
  - Not all *important* questions are *easily* expressed in this form
  - Not all proofs/metrics are *important*

Data/Query Models

Review

Data/Query Models

- Relational Model
  - Data model: Tables
    - Viewed as *physical* data model
  - Query model: Relational Algebra/Calculus
    - SQL is dominant form
- Entity-Relationship (ER) Model
  - A *logical* data model (see web), no explicit query model
  - With mapping to relational model
- Object-Oriented Modeling
  - Unified Modeling Language (UML)
  - Nearly identical to ER Modeling
  - OQL is closest thing to query model
- Other variations
  - Semantic data model, etc
  - Useful variations on ER/OO model

Call Level Interfaces

Background

Call Level Interface

- Set of APIs to run SQL commands
  - These are the workhorse of database interfaces technologies
- Basic operations
  - Connect to database
  - Execute SQL commands (with parameters)
  - Iterate over result set (if there is one)
- Variations
  - Access meta-data, convert data,
- Note
  - An interface to the database engine, not to a particular logical database

History of DB Interface APIs

Embedded SQL	???	Required preprocessor
ODBC	1992	For "C". Basis of "SQL/CLI New Binding Style" in 1995
DAO	~1992	VB and Jet DB engine
JDBC	1996	Java version of ODBC
RDO	~1996	VB and any DB
OLE DB	~1996	high-performance, C level
ADO	~1996	VB and web scripting
ADO.NET	~2001	All languages, uses

ADO Example

```
Dim db as new ADODB.Connection
Call db.Open("ODBC;DSN=" & DatabaseName
        & ";UID=" & UserName & ";PWD=" & UserPassword)

Dim rs as new ADODB.recordset

Call rs.Open("SELECT Name, Phone FROM Employee")
Write "<Table>"
Do while not rs.EOF
   Write "<TR><TD>" & rs.Field("Name").value & "</TD>"
   Write "<TD>" & rs.Field("Phone").value & "</TD></TR>"
   rs.MoveNext
Loop
Write "</Table>"
```


Calling Stored Procedures

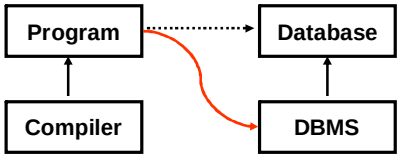
```
Set objCon = New ADODB.Connection
Set objCom = New ADODB.Command

'Creating the DB connection string
'Please change the below connection string as per your
server and database being used.
objCon.ConnectionString =
"PROVIDER=SQLOLEDB.1;PASSWORD=;PERSIST
SECURITY INFO=TRUE;USER ID=sa;INITIAL
CATALOG=TestSQL;DATA SOURCE=Rockets"

'Opening the connection
objCon.Open objCon.ConnectionString

'assigning the command object parameters
With objCom
 .CommandText = "GetRecords"
    'Name of the stored procedure
 .CommandType = adCmdStoredProc
    'Type : stored procedure
 .ActiveConnection = objCon.ConnectionString
End With

'Create 2 output parameters
Set objPara = objCom.CreateParameter("rows", adInteger,
adParamOutput)
Set objpara2 = objCom.CreateParameter("Status",
adVarChar, adParamIn, 50)

objpara2.Value = InputStatus
```

(List, NumRows) = GetRecords(Status)

```
'Append the output parameters to command object
objCom.Parameters.Append objPara
objCom.Parameters.Append objpara2

'Store the result in a recordset
Set objRS = objCom.Execute

'Open the recordset
Do While Not objRS.EOF
 For k = 0 To objRS.Fields.Count - 1
  write objRS(k).Name & ": " & objRS(k).Value
 Next
 objRS.MoveNext
Loop

'retrieve the output parameters values
MsgBox "Total records returned: " & objPara.Value
MsgBox

'close connection
objRS.Close
objCon.Close
```


CLI Issues

- Program text in strings
  - Call rs.Open("SELECT Name, Phone FROM Employee")
- Access to bindings via strings
  - rs.Field("Phone").value
- No semantics connection



Summary

- Everyone knows it is terrible
- Lots of effort to do better
- Yet it is still ubiquitous

Transactions

Review of basic concepts

Transactions

- Transaction is a *unit of work*
  - *Begin* Transaction
    - Do work…
  - *Commit* or *Abort*
- Key issues
  - Concurrency
    - Multiple transactions running together
  - Failure
    - Handling catastrophic system failures

ACID Transaction Principles

- Properties that must be preserved by DBMS

A	Atomic	Either *all* the operations in a transaction are performed or *none* are
C	Consistent	The database must be in a consistent state at the start and end of every transaction
I	Isolated	There is no interference between concurrent transactions
D	Durable	Once a transaction completes, its affect is permanent even in the event of complete system failure

Analysis of Concurrent Transactions

- Analyze *schedule* of operations
  - Operations (R, W) specify
    - Transaction they are part of
    - Data object that they affect
- Define equivalence on schedules
  - Contain same operations and conflicting operations are in same order
  - W conflicts with any other operation
- Serial schedule
  - Performs transactions in some linear order
- Schedule is *serializable*
  - if it is equivalent to a serial schedule

Ensuring Serializability

- Pessimistic Locking
  - Two-phase locking
    - Lock all data that is *read* or *updated*
    - Hold locks until end of transaction
    - Block other transactions that need locks
    - Abort if deadlock detected
- Optimistic Locking
  - Timestamp all udpates
  - Abort if accessing invalid data
- Combination of above
  - For example
    - Optimistic locking for web pages
    - Pessimistic locking when processing submit

Issues with Consistency

- Consistency can be defined *locally*
  - Local
    - Salary of manager > Salary of employees
    - Nobody can check out more than 10 books
  - Nonlocal
    - VLSI layout must represent a planar graph
- Reality
  - Very few consistency constraints are actually expressed in most databases
  - It is up to the application program to ensure consistency

Issues with Atomicity

- Transactions are assumed short
  - A lot of work on long transactions
  - Typical database systems limit transactions to 30 seconds
    - What happens if you have an atomic operation that takes longer than this?
      - Better start thinking hard!

Issues with Durability

- Durable
  - Store everything immediately
- Atomic & Isolated
  - Don't store right away, to prevent interference with other queries
- Database systems deal with this problem pretty well
  - Strictness: only read or overwrite *committed* data
  - Prefix-reducability

From Dave Eckhardt, CMU

Issues with Isolation

- Phantoms
  - 1. Find all employees with salary > 50000
  - 2. Computer average
  - 3. Update all employees with salary > 50000
  - No *conflict* if another transaction adds an employee between #1 and #3
  - Locking a predicate
- Support for different levels of isolation

Transactions from Client Viewpoint

- Client code must indicate transaction boundaries
  - BeginTransaction
    - Do work…
  - EndTransaction
- This is a problem for modularity
  - How do we assemble a composite transaction from multiple parts, if each is beginning/ending its own transaction
- Review solutions in Middleware area