

# Choosing Database Technology for Object-Oriented Applications

TU-Wien, Sommersemester 2007  
Rudolf Lewandowski

# Überblick

- Motivation
- Sicherheit, Verfügbarkeit
- Objekt Datenbanken
- Object Relational Mapping
- Relationale Datenbanken
- Zusammenfassung

## Motivation

### Was ist das typische Problem?

- Sie haben so 50 – 80 Geschäftsobjekte
- Zusätzlich viele Hilfsobjekte
- Die Daten sollen sicher gespeichert werden
- Sie bringen es dann schnell auf 200 oder mehr Datenbanktabellen
- Sie wollen Objektorientiert entwickeln
- Daraus folgt:  
OODBMS oder RDBMS verwenden

## Motivation

### Was erwarte ich von einer Datenbank?

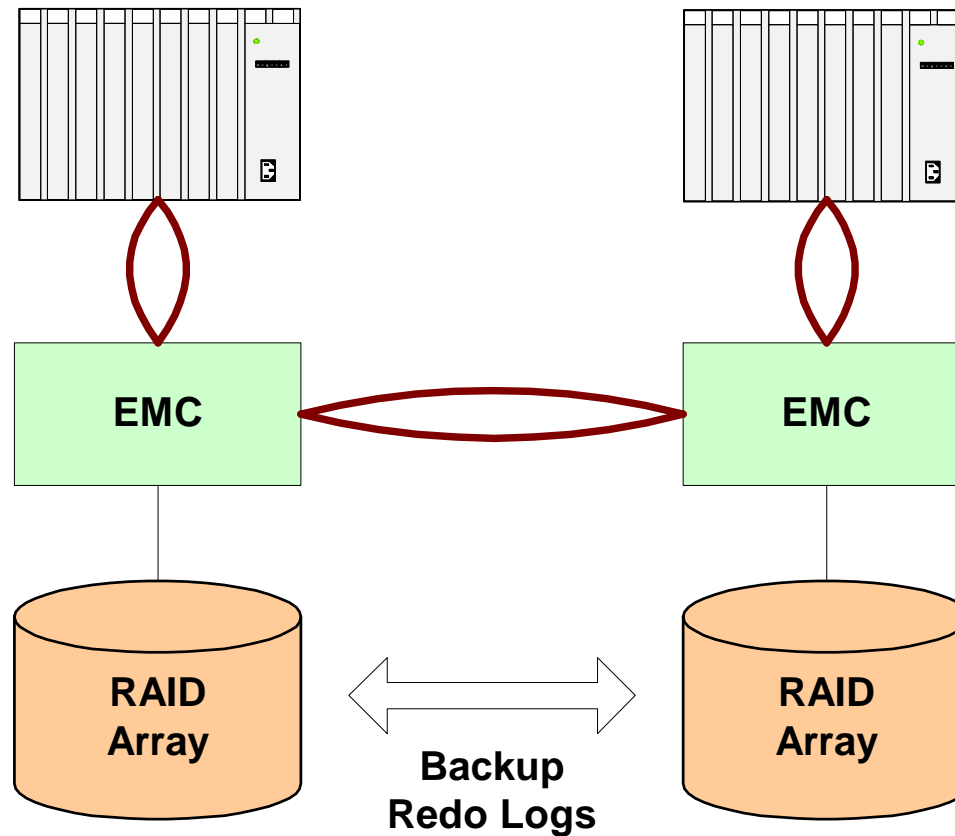
- Sicherheit, Verfügbarkeit
- Die Geschäftsobjekte sollen (einfach) gespeichert werden können.
- Performance (viele Sachbearbeiter zugleich)
- Transaktionen (kurze, lange)
- Historie (Lebenszyklus von Objekten)
- Wartung (DBA)

# Datenbank Sicherheit, Verfügbarkeit

- SAN (Storage Area Network)
- Regelmäßige Backups
- Redo Logs
- 2 Standorte
- 2 Trassen für Lichtleiter
- RAID Arrays
- ...

# Datenbank Sicherheit, Verfügbarkeit

## Storage Area Networks



## Object Database

<http://www.service-architecture.com/object-relational-mapping/articles/index.html>

Storing objects you want OO features to be preserved...

- Complex Objects
- Object Identity
- Encapsulation
- Classes
- Class Hierarchy
- Polymorphism
- Extensibility

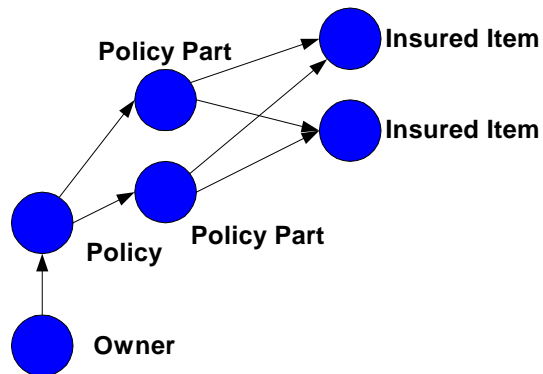
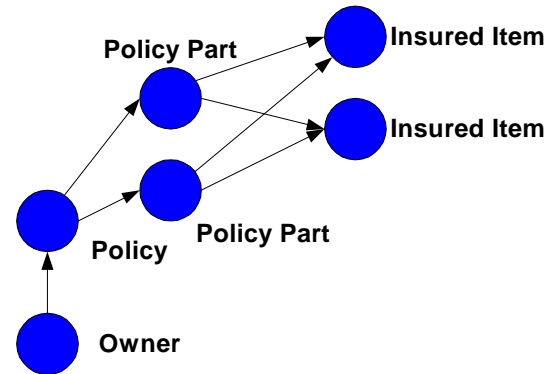
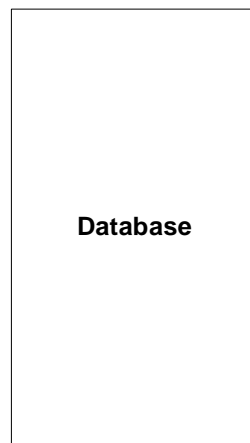
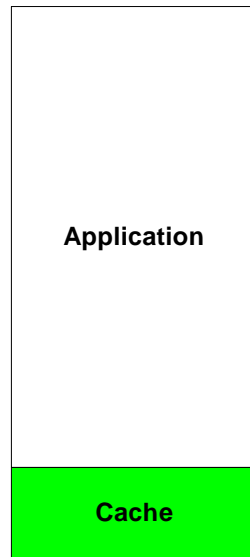
# Object Database

...while new requirements emerge

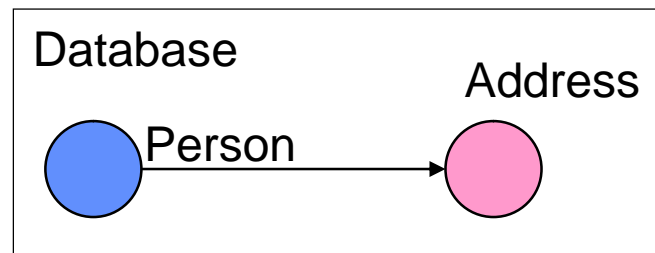
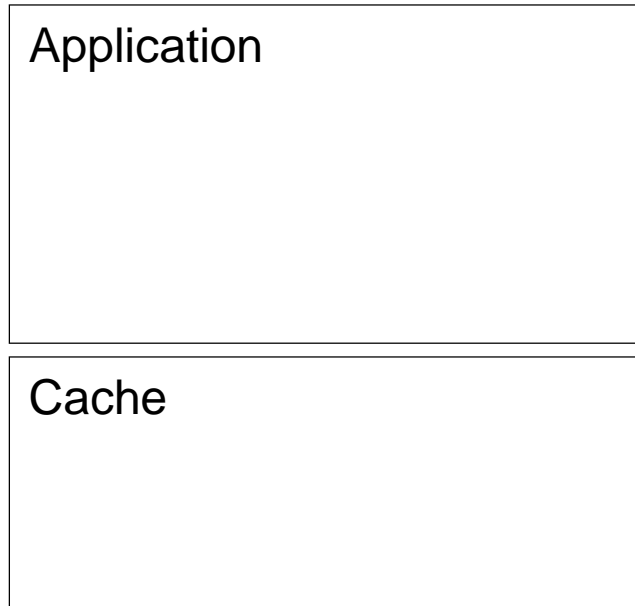
- Persistence
- Query Facility
- Concurrency
- Recovery
- Security
- Secondary Storage Management

# Object Database

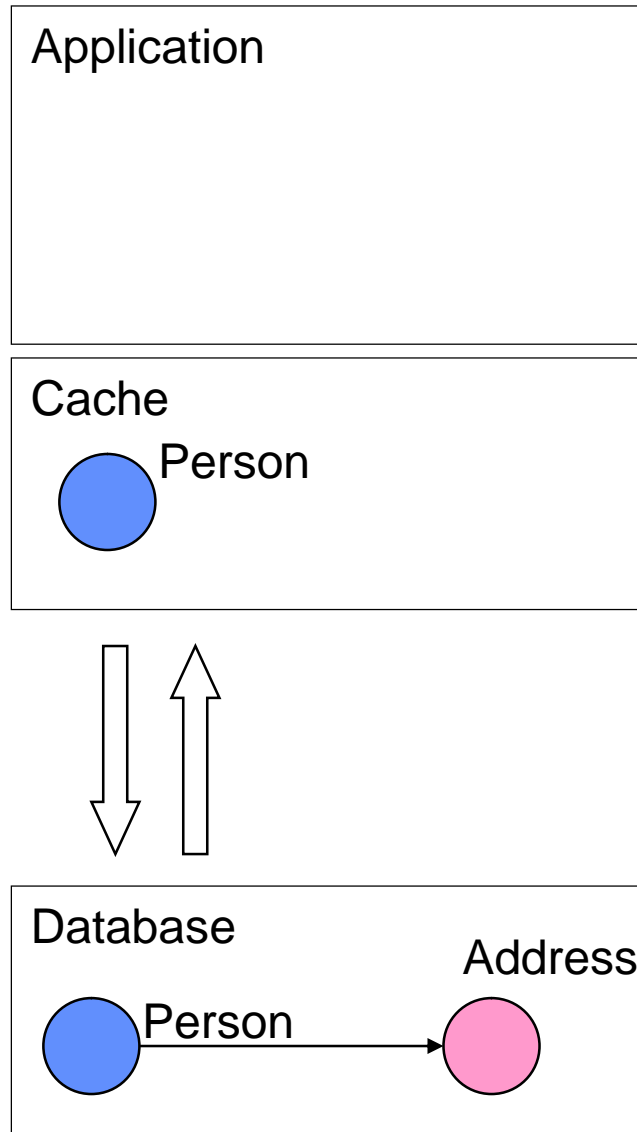
Products:  
ObjectStore,  
Versant,  
Objectivity/DB,  
Poet,  
GemStone,  
...



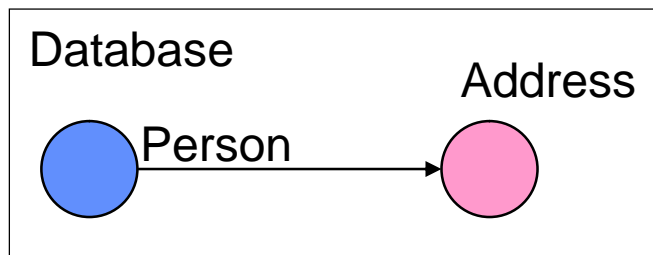
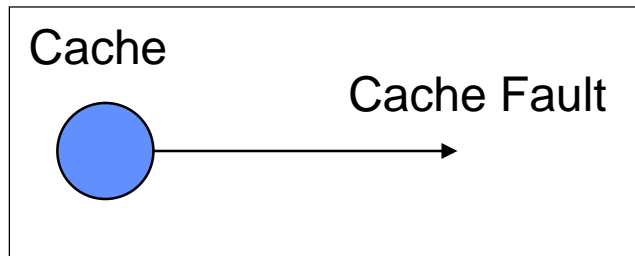
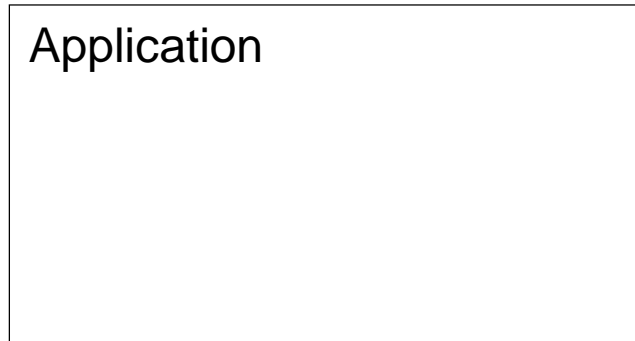
# Object Database



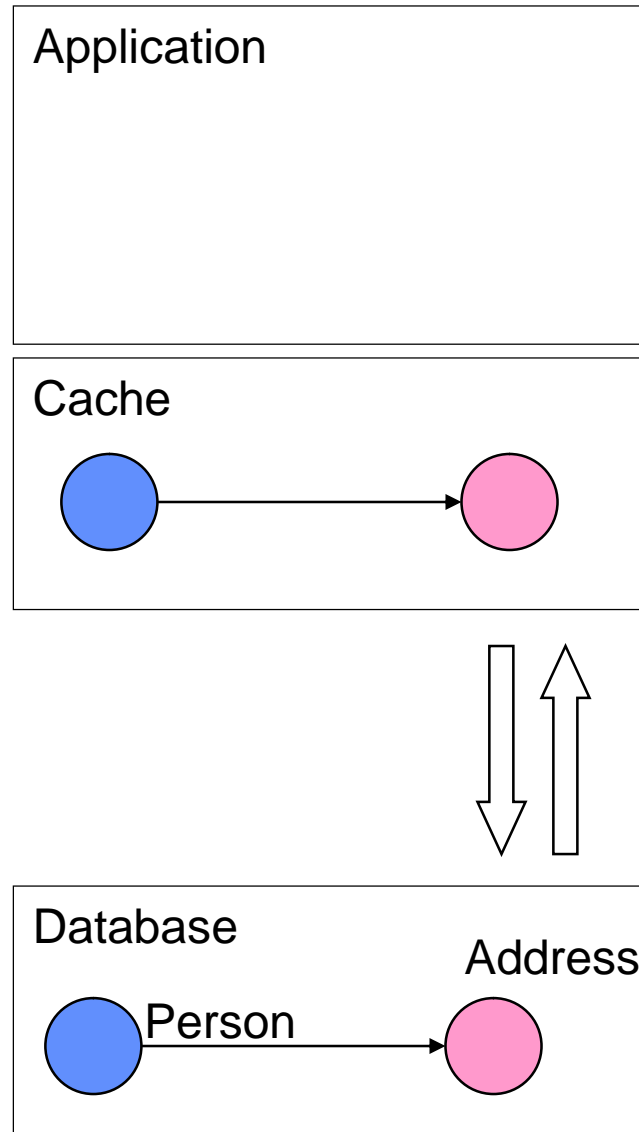
# Object Database



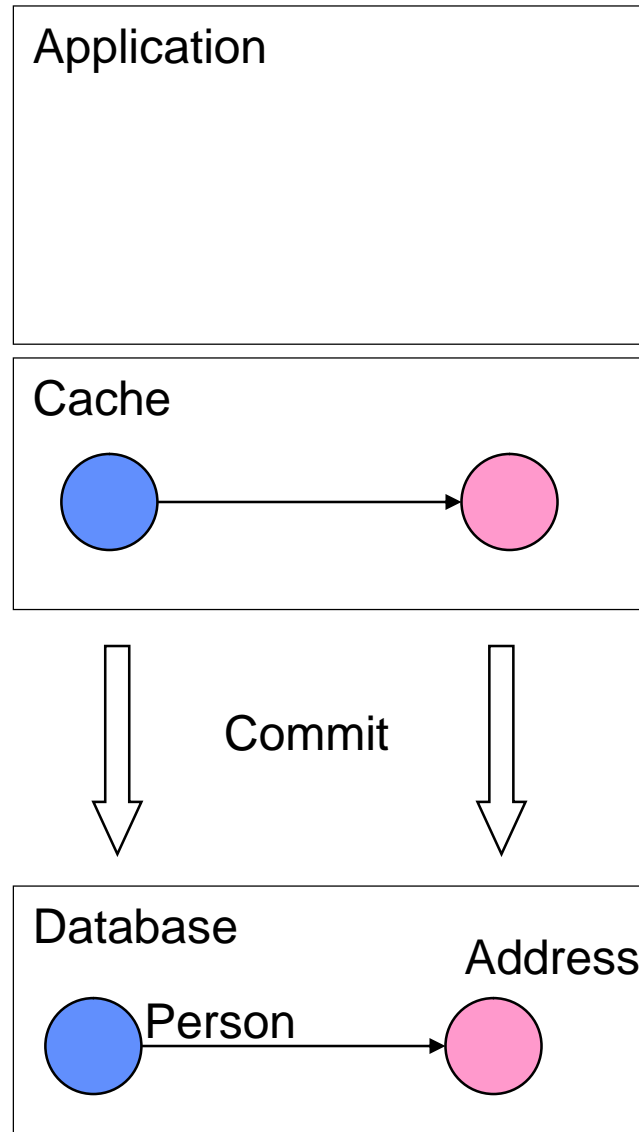
# Object Database



# Object Database



# Object Database



## Object Database

- Eine Objektorientierte Datenbank lässt persistente Objekte wie normale Objekte der Sprache erscheinen -> transparent Persistence
- Wenig Lines of Code für Persistence
- Gut für komplexe Objektmodelle
- Bieten oft lange Transaktionen
- Bieten Historie
- Query by Example
- Schnelle Navigation (Traversierung)

# Object Database Transparent Persistence



```
txn.begin();
Collection people = pm.getExtent(Person.class, false);

// perform query
Query query = pm.newQuery(Person.class, people, "name == \"Tester\"");
Collection result = (Collection) query.execute();
Iterator iter = result.iterator();
// iterate over the results
while ( iter.hasNext() ) {
    Person person = (Person) iter.next();
    // now traverse to the address object and update its value
    person.address.street = "13504 4th Avenue South";
}
txn.commit();
```

# Object Database Erfahrungen

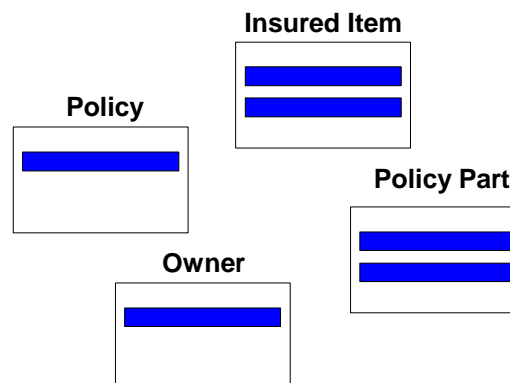
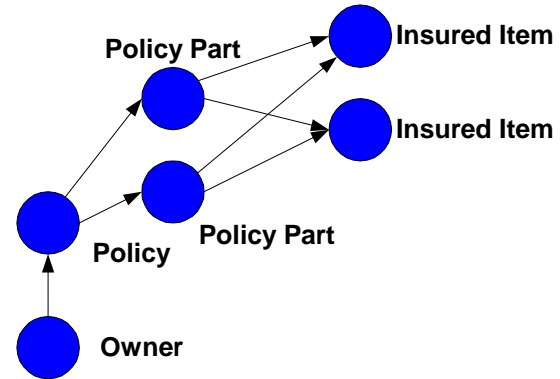
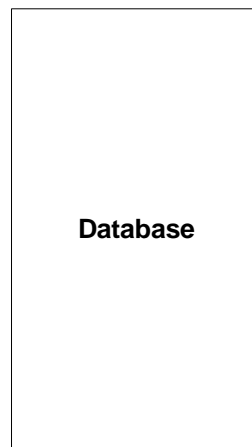
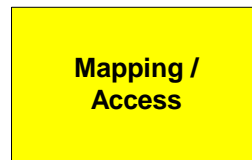
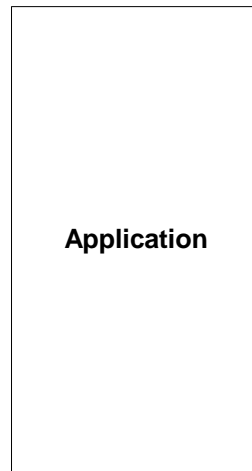


- Versant für Smalltalk
- Sehr komplexe Produktstrukturen
- Fehler: Event Dependents mit auf DB
- Lösung: Methode vor Speichern und nach Lesen aufrufen  
(Nicht mehr ganz *transparent persistence*)
- Nicht behebbarer Fehler auf OS, Version von VA...
- Nicht auf Großrechnern verfügbar
- Keine DBA Erfahrung
- Derzeit *kein* OODBMS bei der Generali im Einsatz

# Object Relational Mapping



Lack of  
Impedance  
Mismatch

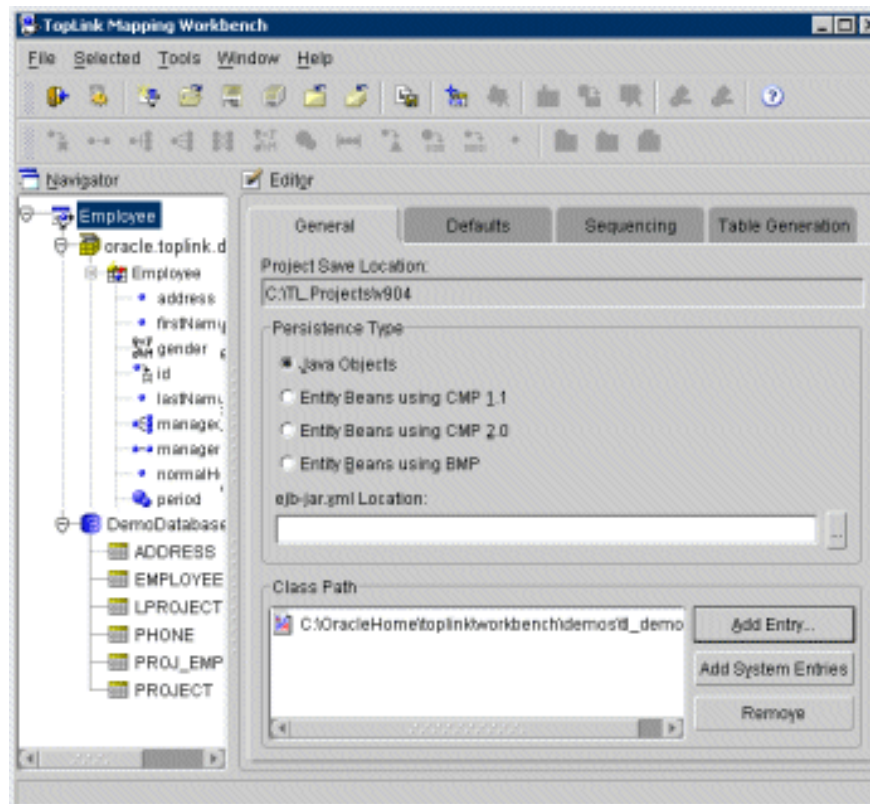


## Object Relational Mapping Framework

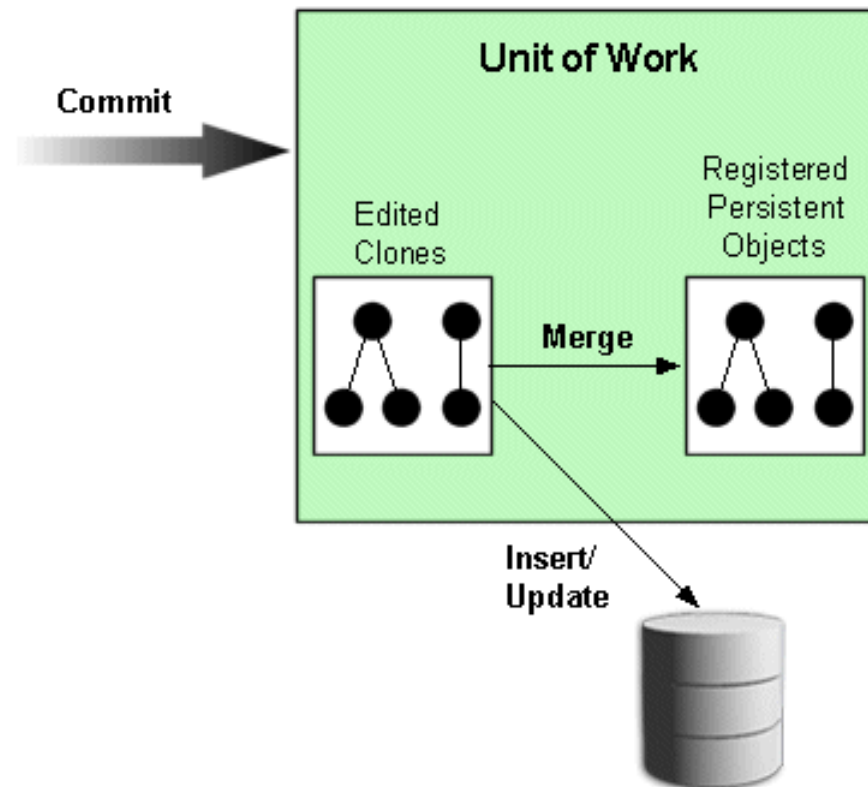
- Werkzeugunterstützt (Mapping Workbench)
- Ermöglicht transparent Persistence für den Programmierer
- Kann bestehende (Legacy) Tabellen mappen
- Inheritance führt zu vielen Tabellen oder einer Tabelle mit leeren Feldern
- N zu M Relationen, Hashtables... benötigen Hilfstabellen
- BLOBs machen andere Programmiersprachen oder Tools „blind“
- Mehrere Performancefallen

# Object Relational Mapping Framework TOPLINK Workbench

<http://otn.oracle.com/products/ias/toplink/content.html>



# Object Relational Mapping Framework TOPLINK Unit of Work



# Object Relational Mapping

## Performancefallen



- Benutzt intern Call Interface z.B. JDBC
- Viele kleine Einzelzugriffe
- Unterstützt nicht alle Möglichkeiten der Datenbank (z.B: DB Cursor)
- `SELECT COUNT(*) FROM ADDRESS WHERE...` versus `Person.getAdresses().size()`
- Im Batch müssen mehrere tausend Objekte verändert werden
- Der Entwickler weiß oft gar nicht was passiert...

# Performancefallen

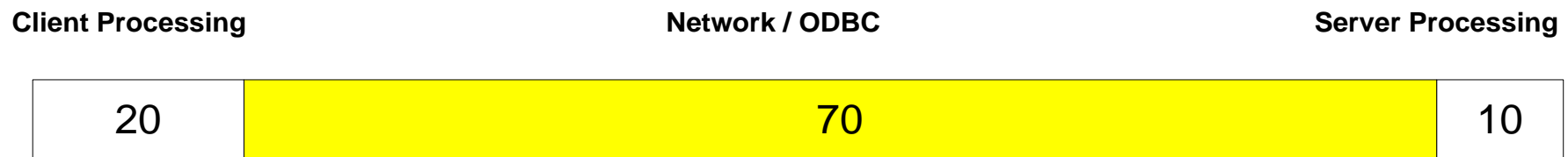
## Daher...



- Objektmodell *nicht unabhängig* von Datenmodell designen.
- Datenbanken haben Limitierungen, die sich im Objektmodell widerspiegeln.
- Handgeschriebene Services, die die Features der Datenbank ausnutzen (ORACLE oder DB2) können viel effizienter sein
- Beachtung der Datenbank führt oftmals zu einem einfacheren Objektmodell

# Performance Lösungen in Phoenix

- Performance
  - Static SQL from Client side
  - Clustering SQL Requests (Several SQLs in one large request)
  - Compression (Transmission)
  - Compiled Procedures in C



# Relationale Datenbank Alternative



- Service (Data Access) Layer selbst implementieren
- Services sind handgeschrieben und optimiert (SQL Preprozessor)
- Services lesen „Cluster“ anstelle einzelner Objekte
- Nicht benötigte Attribute werden nicht gelesen
- Alle Möglichkeiten der Datenbank werden ausgereizt.
- Im Batch ist die Hälfte des Codes in SQL
- ORACLE kann sortieren, Maximum suchen, Datumsarithmetik...

# Zusammenfassung

## OODBMS vs. Mapping vs. Call Level



- Sprache
- Performance
- Lange Transaktionen
- Entwicklungszeit
- Batch
- Tools
- Sicherheit, Verfügbarkeit
- Investitionsschutz
- Versionierung
- Plattformübergreifend