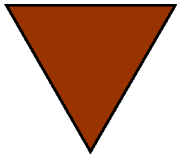


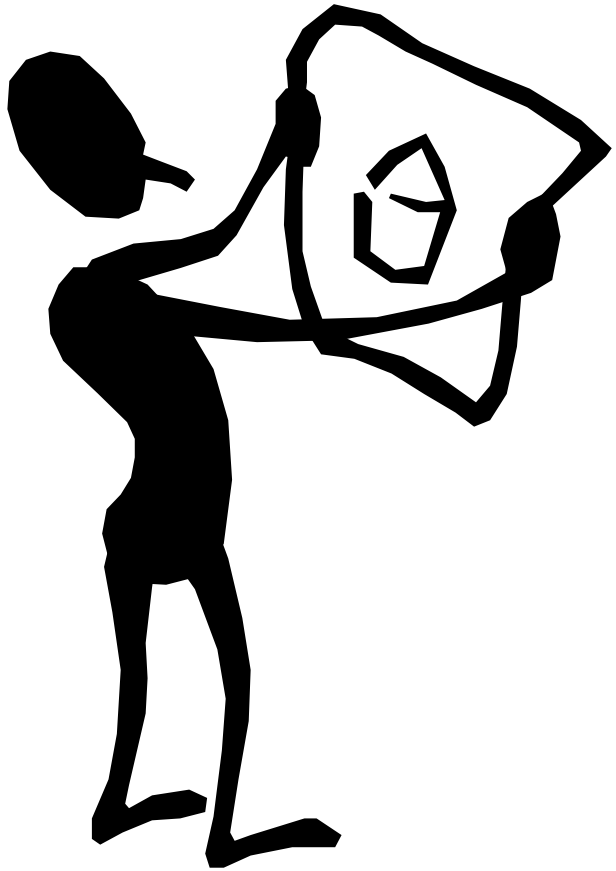
Choosing a Database Technology

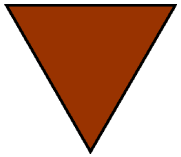
Tutorial at Object World 98
Frankfurt, October, 1st 1998

Jens Coldewey
Coldewey Consulting
Uhdestraße 12
D-81477 München
Germany
jens_coldewey@acm.org
http://www.coldewey.com

Roadmap through the next hours

- Introduction
- The Problem: Storing Objects
- Requirements of the Stakeholders
- The important forces in depth
- Turning forces into a decision



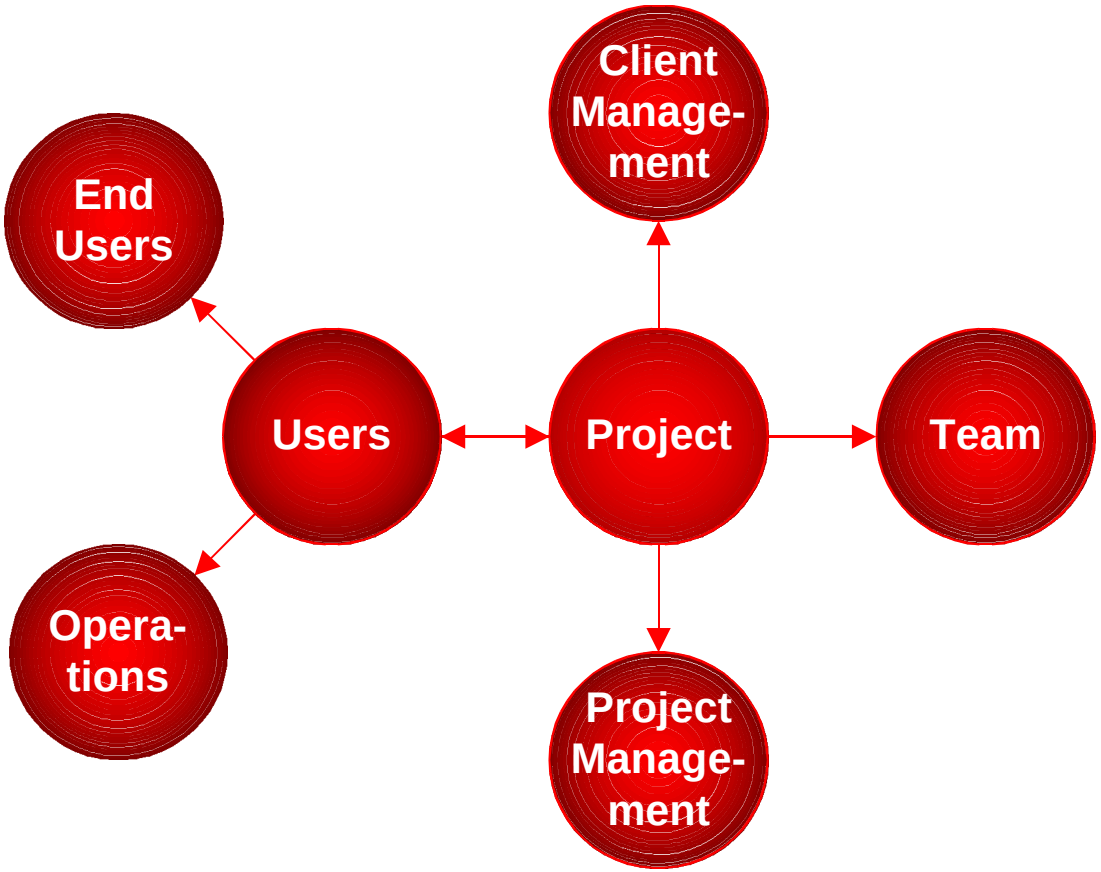


What are we talking about?

- You...
  - develop medium to large size systems
  - use object technology
  - need to store your objects
  - have to decide for a database technology
- It doesn't matter...
  - what domain you are in
  - whether you work for a product or a project
  - what language you use

! I'm not talking about specific products !

Deciding for a database you have to take care of the different stakeholders



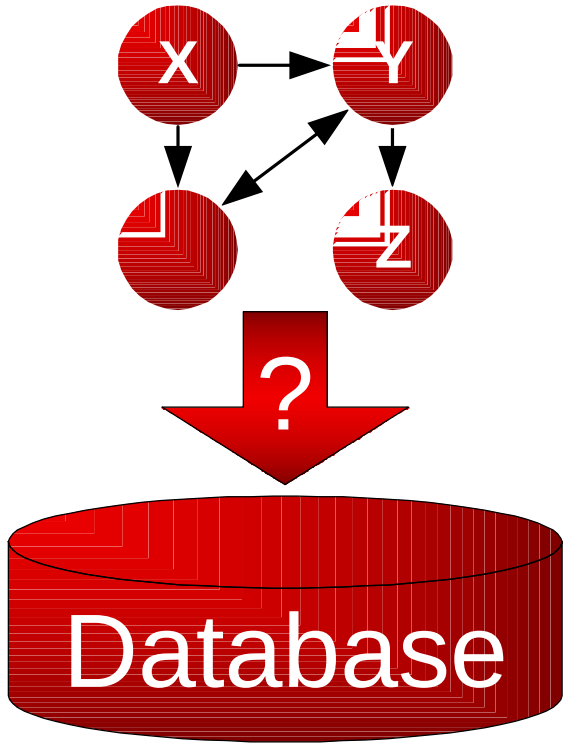
You have to find the best balance

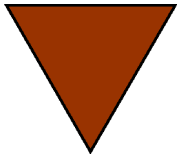
- Every group has different objectives
- These "forces" often contradict each other
- A good decision is a pragmatic balance of these forces

This tutorial analyzes the forces. It's your term to decide.

Before we talk about solutions we have to identify the problem: Storing objects

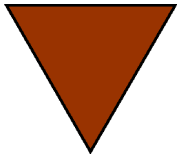
- Introduction
- **The Problem: Storing Objects**
- Requirements of the Stakeholders
- The important forces in depth
- Turning forces into decisions





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

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

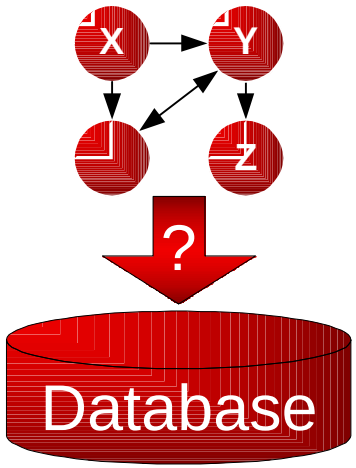


...while new requirements emerge

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

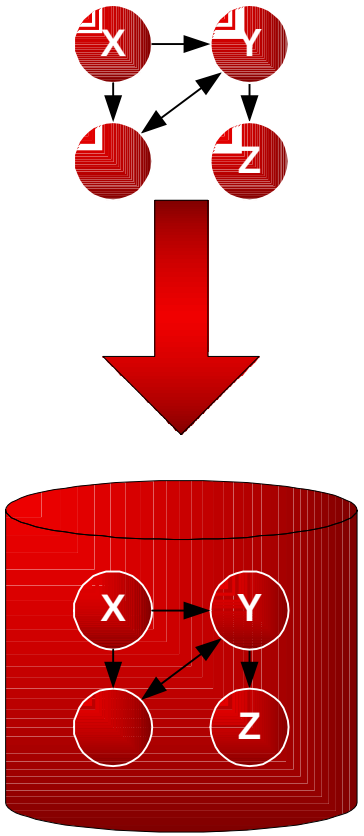
(Modified from [Atk+89])

What are the options?



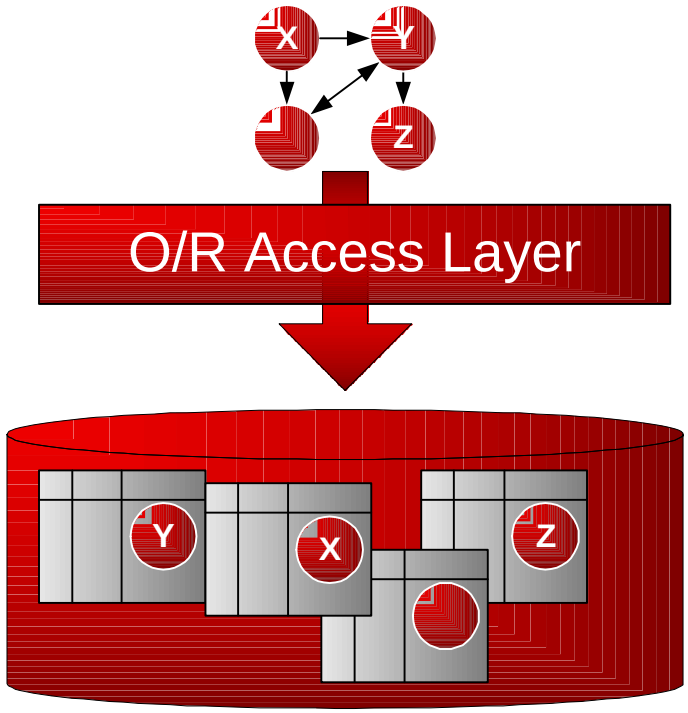
Stream Persistence	Object Database		Object/ Relational Coupling	
	Object Server	Page Server	ORDBMS	Access Layer

Object databases directly store objects in the database



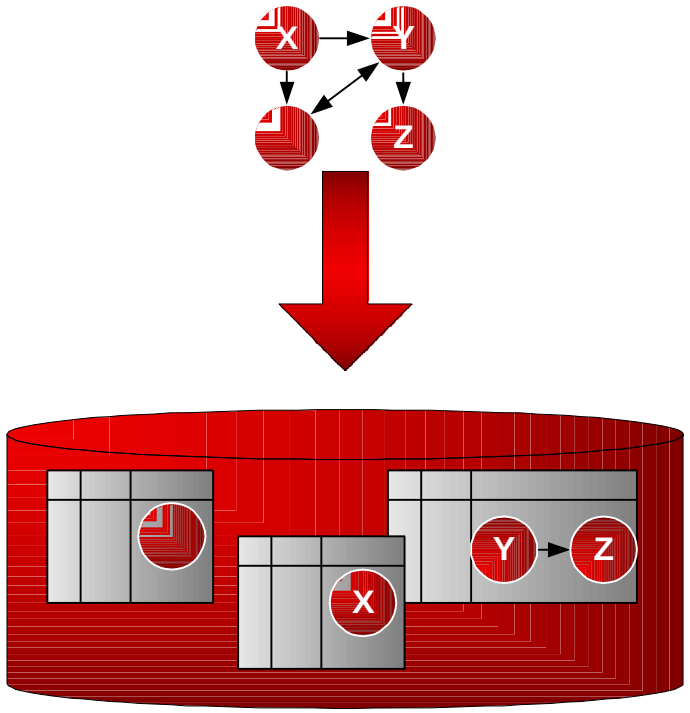
- Tight language integration
  - Access
  - Typing
  - References
- ODMG defines "standard" interface
- Products: ObjectStore, Versant, Objectivity/DB, Poet, GemStone,...
- Literature: [Cat94]

Relational databases need a complex mapping mechanism



- Relational Paradigm
  - Tables and table-operations
  - References via foreign keys
  - No extended typing
  - No inheritance
- SQL - Standardized
- Products: TOPLink, Persistence
- Literature: [Kel98]

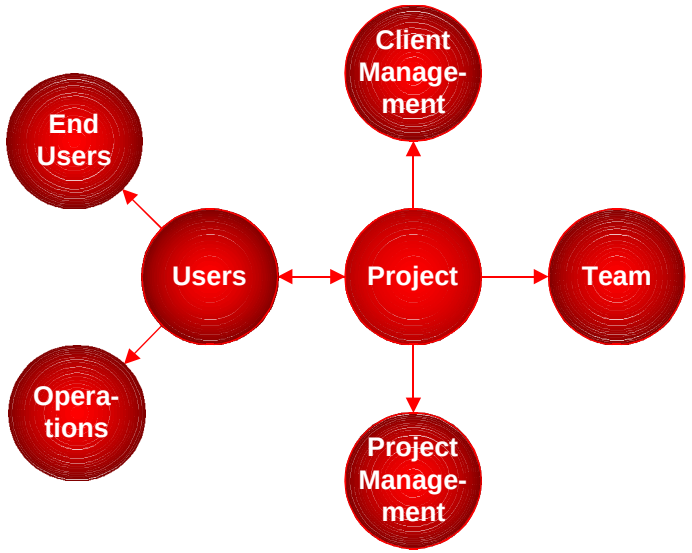
Object / Relational databases try to combine both worlds



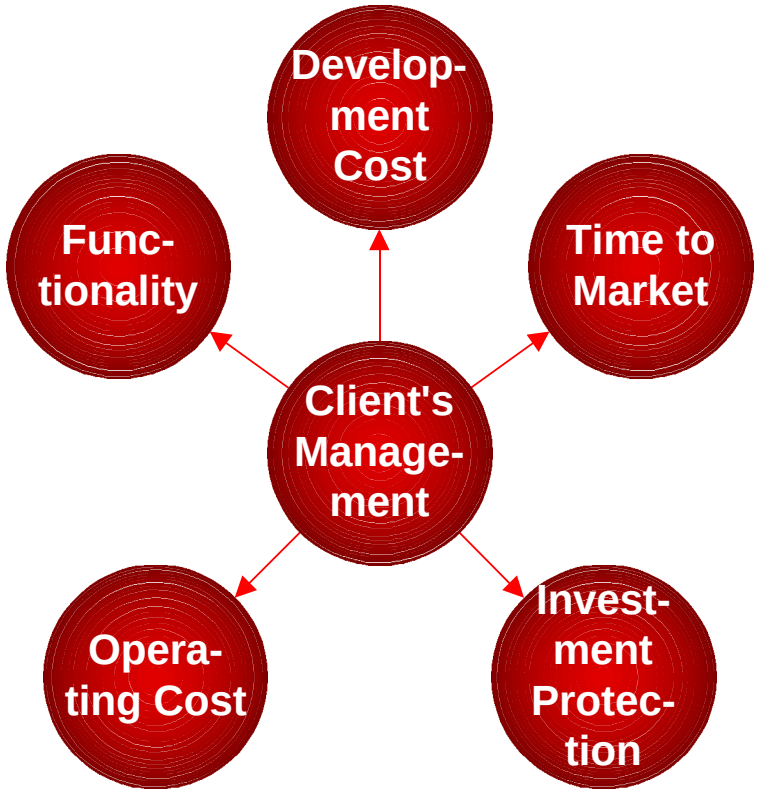
- Extends Relational Paradigm
  - Complex Objects
  - Typing
  - Nested Tables
- E-SQL (vendor specific)
- No standards by now
- Products: DB/2, Informix, Oracle 8, Unidata
- Literature: [StM96]

To start with the analysis of the forces we turn back to our stakeholders

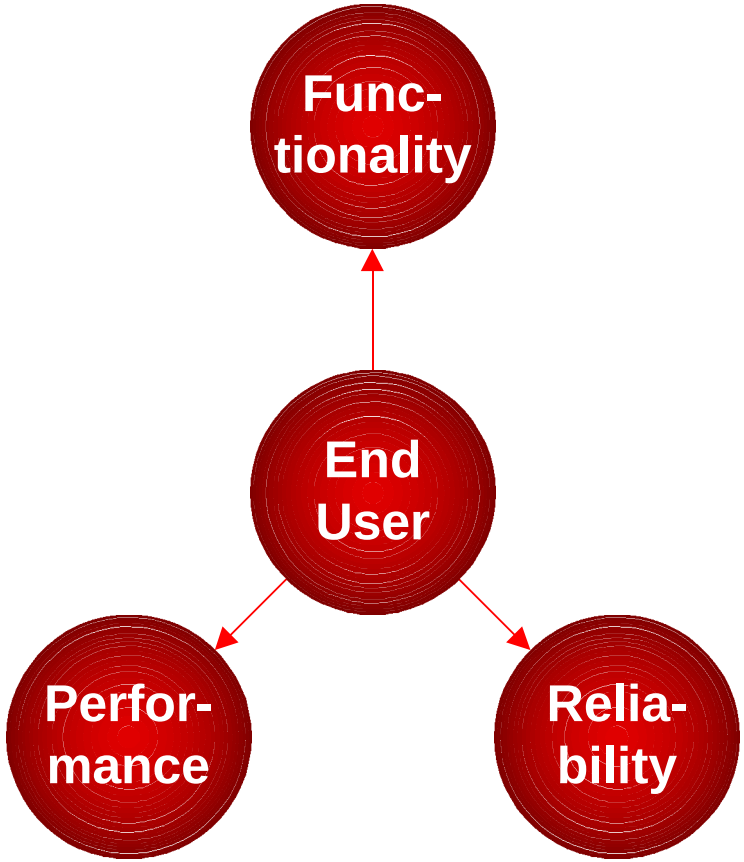
- Introduction
- The Problem: Storing Objects
- **Requirements of the Stakeholders**
- The important forces in depth
- Turning forces into decisions



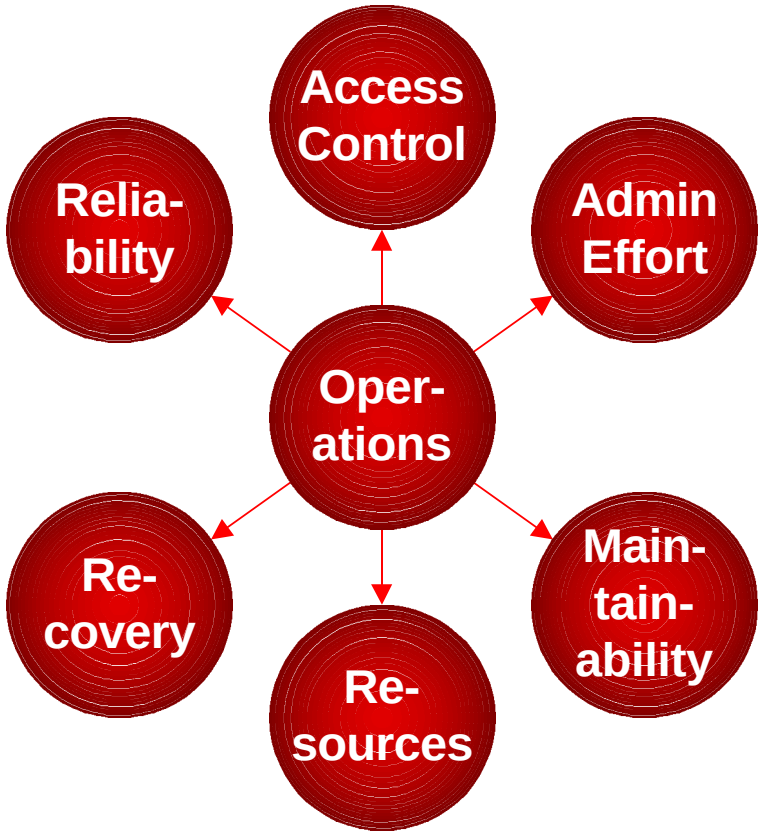
Client's management usually has five things in mind



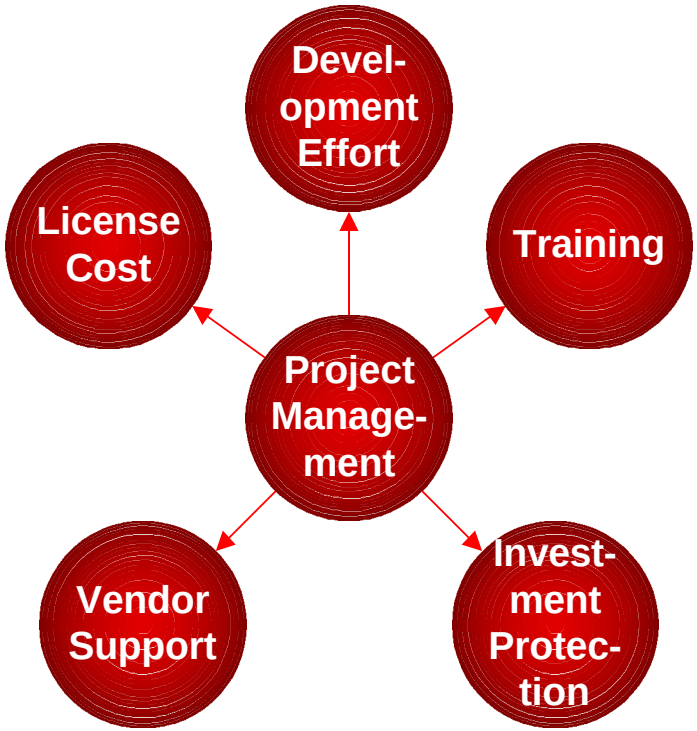
The end user's requirements are usually "obvious"



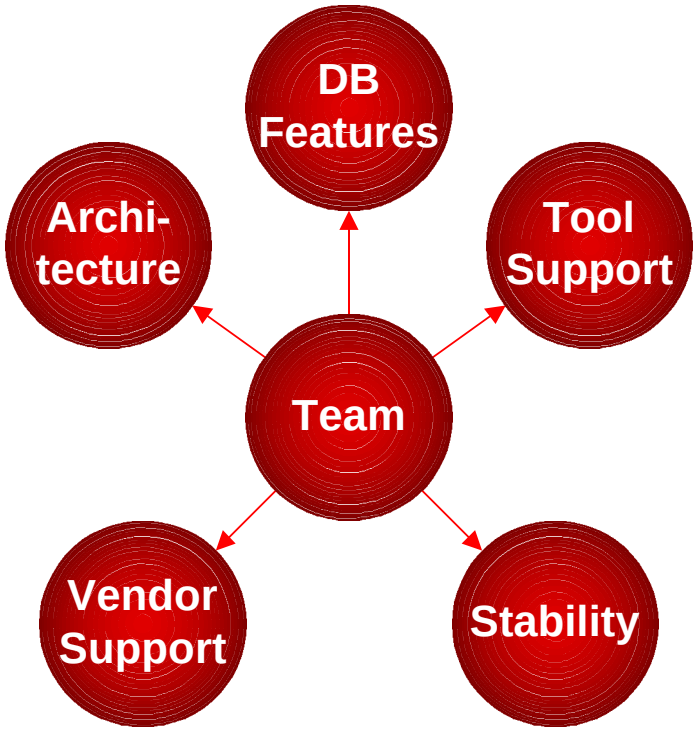
Operations' requirements are often less obvious - and they *will* haunt you!

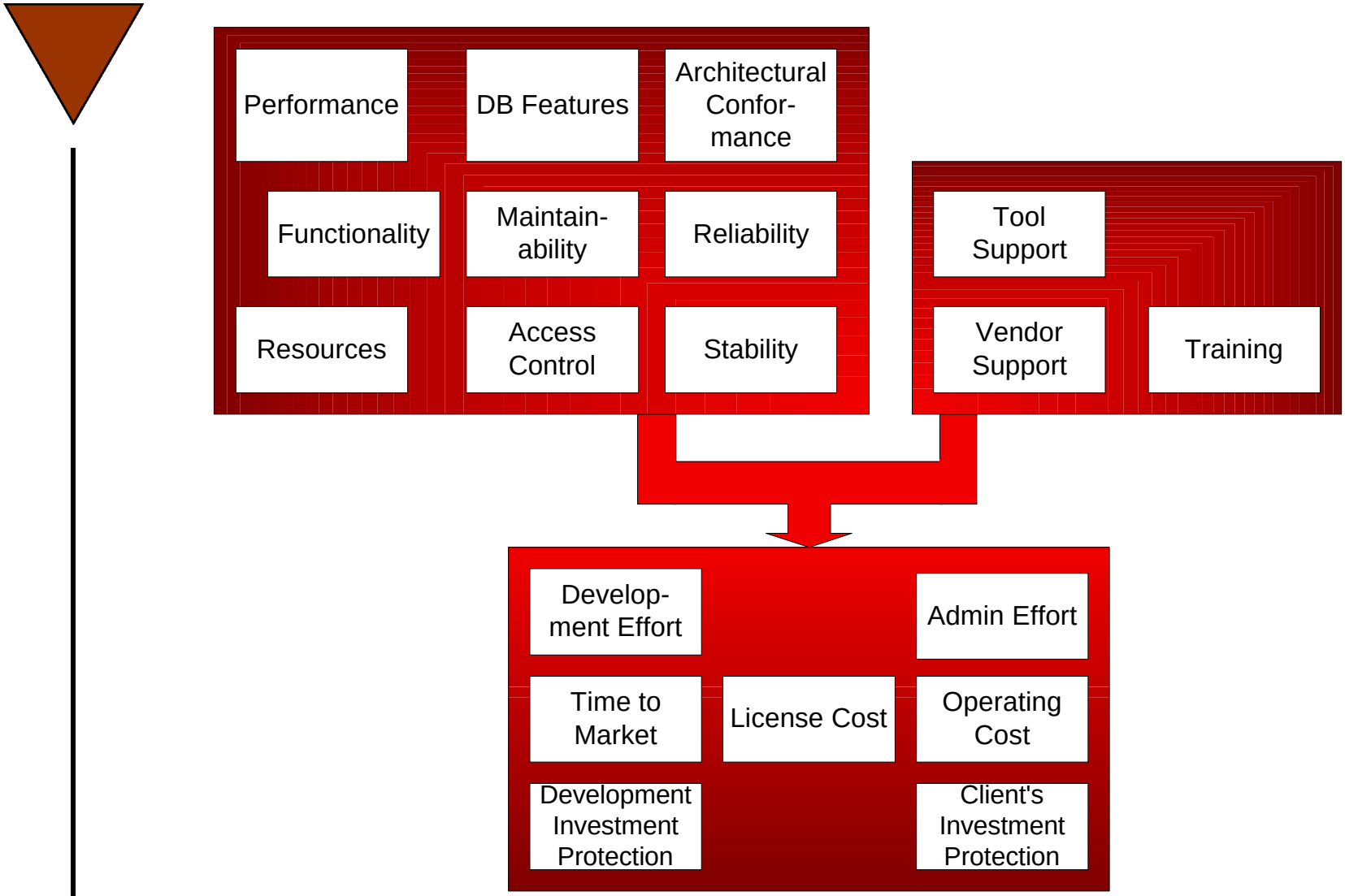


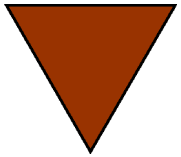
Some forces derive from your own project management



And finally the development team needs to get optimal support for their work



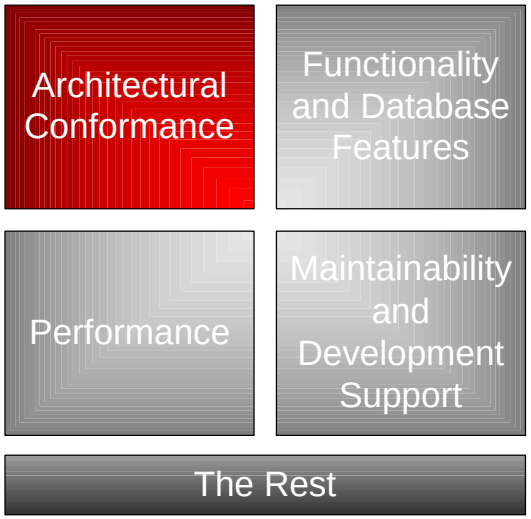




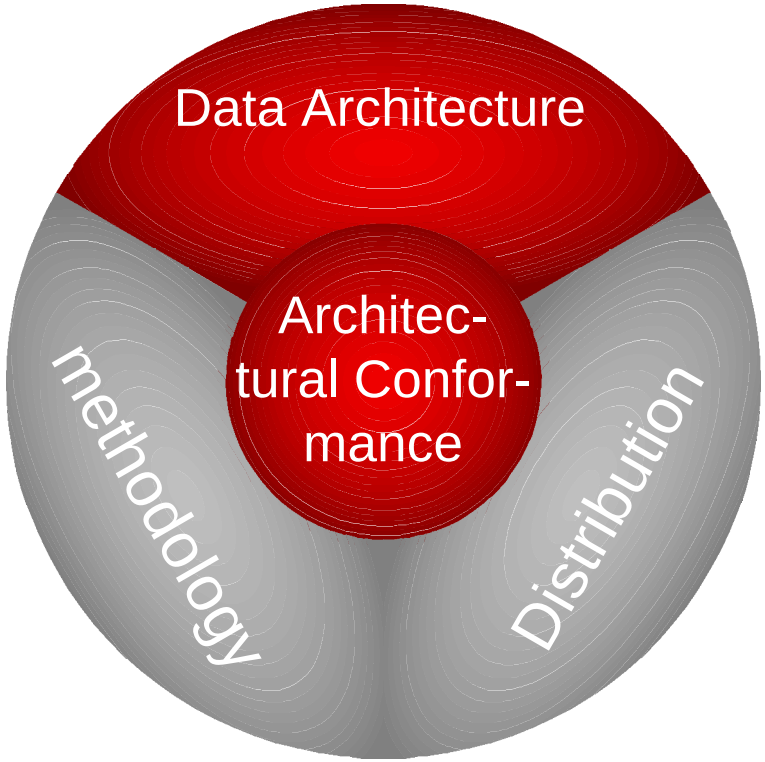
In the following discussion I concentrate on discriminating issues

- Some of these issues depend more on the product than on the technology
- The following concentrates on what technology heavily influences:
  - Architectural Conformance
  - Functionality and DB Features
  - Performance
  - Maintainability
- The rest is just sketched

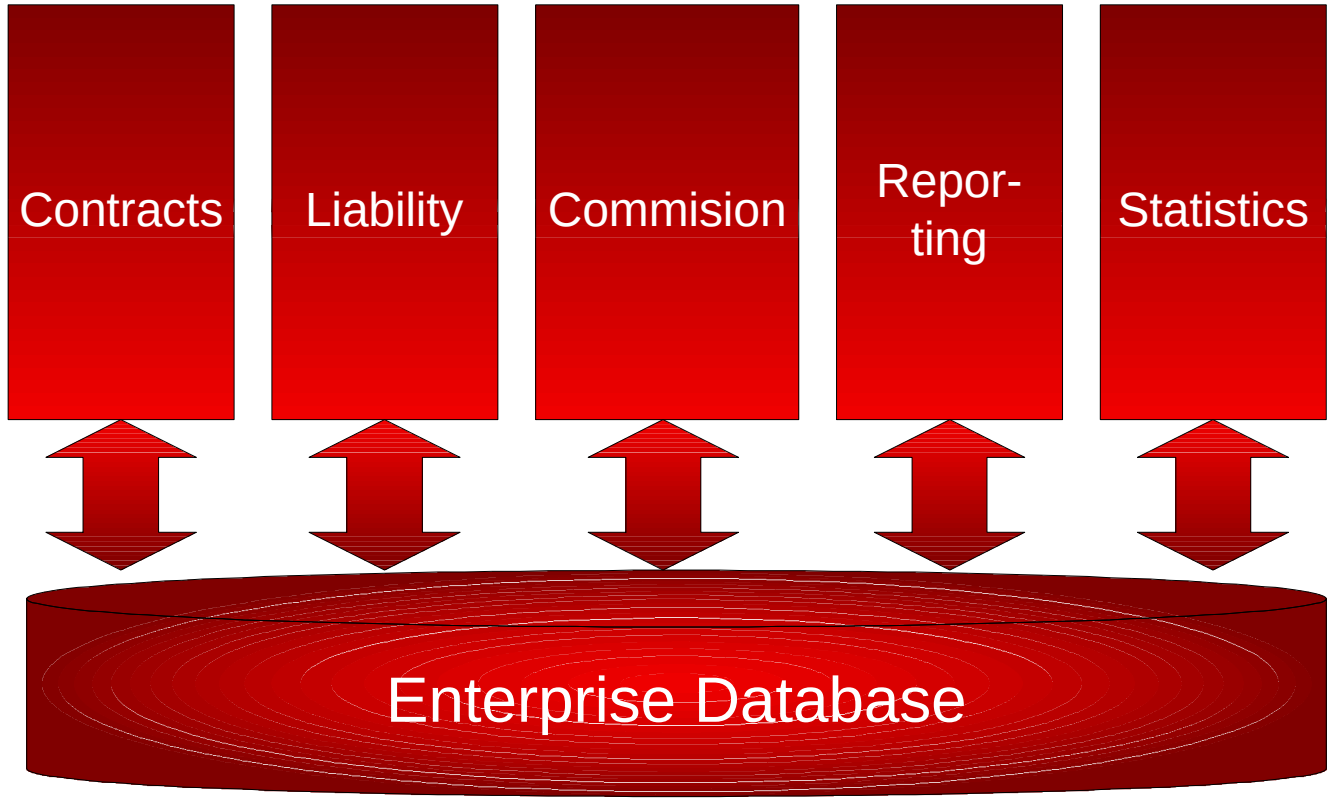
- Introduction
- The Problem: Storing Objects
- Requirements of the Stakeholders
- **The important forces in depth (I)**
- Turning forces into decisions

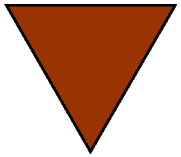


The Architectural Conformance splits into three areas of interest



Many large systems consist of several applications on the same database



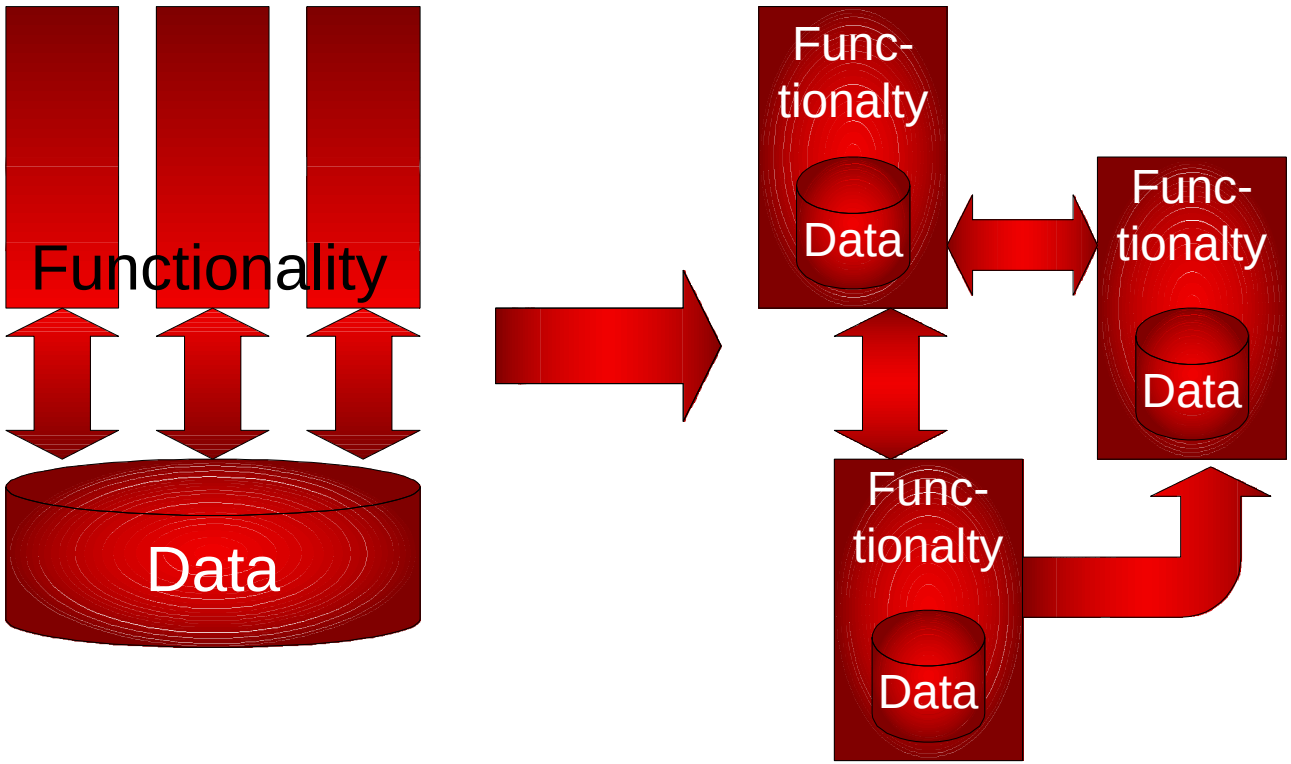


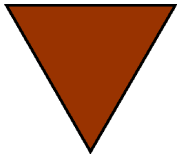
This is the architecture most (relational) database experts have in mind

+ Well-known architecture
+ Easy integration with legacy applications
+ Easy Client/Server design
+ Many useful tools

– Database design changes propagate merciless ⇒ Application specific views needed
– Elephantiasis of the database

The object view is different



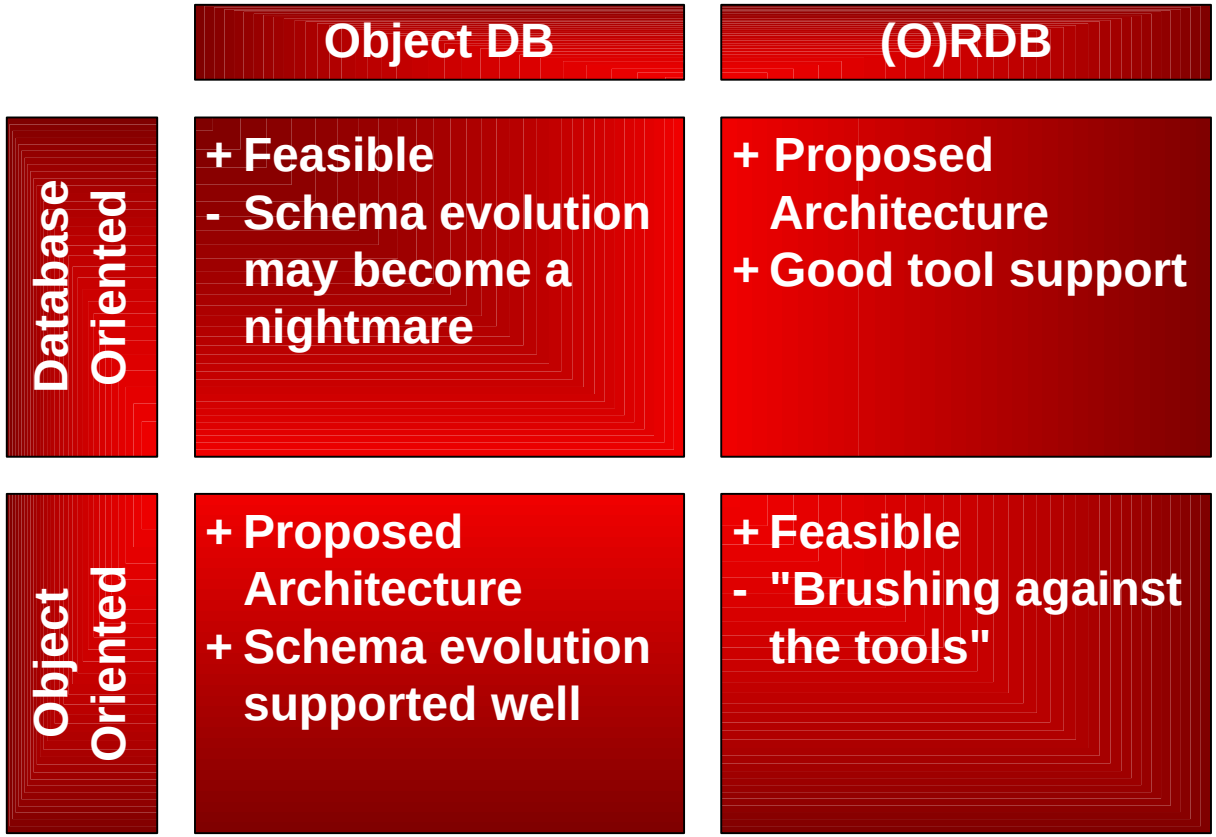


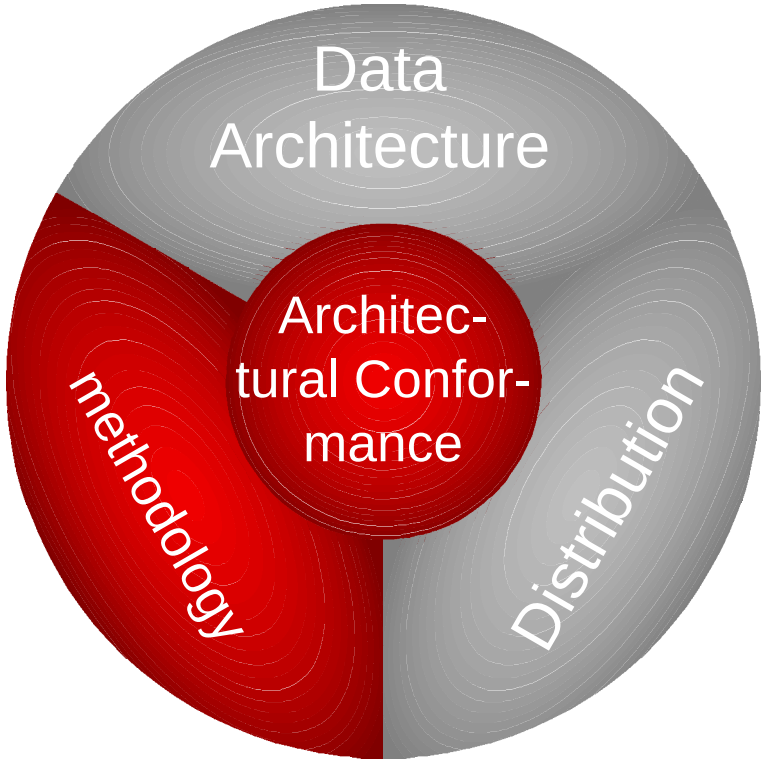
The object approach avoids some problems but generates new ones

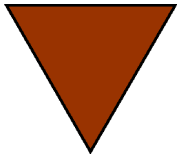
+ Good encapsulation of data
+ Defined responsibilities keep databases manageable
+ Missing central database model speeds up development

– Additional communication is new and complex
– Analyzing information of several applications is hard
– Tool market is still in its infancy
– Paradigm shift is hard

Both DB technologies support both architectures but have different focus





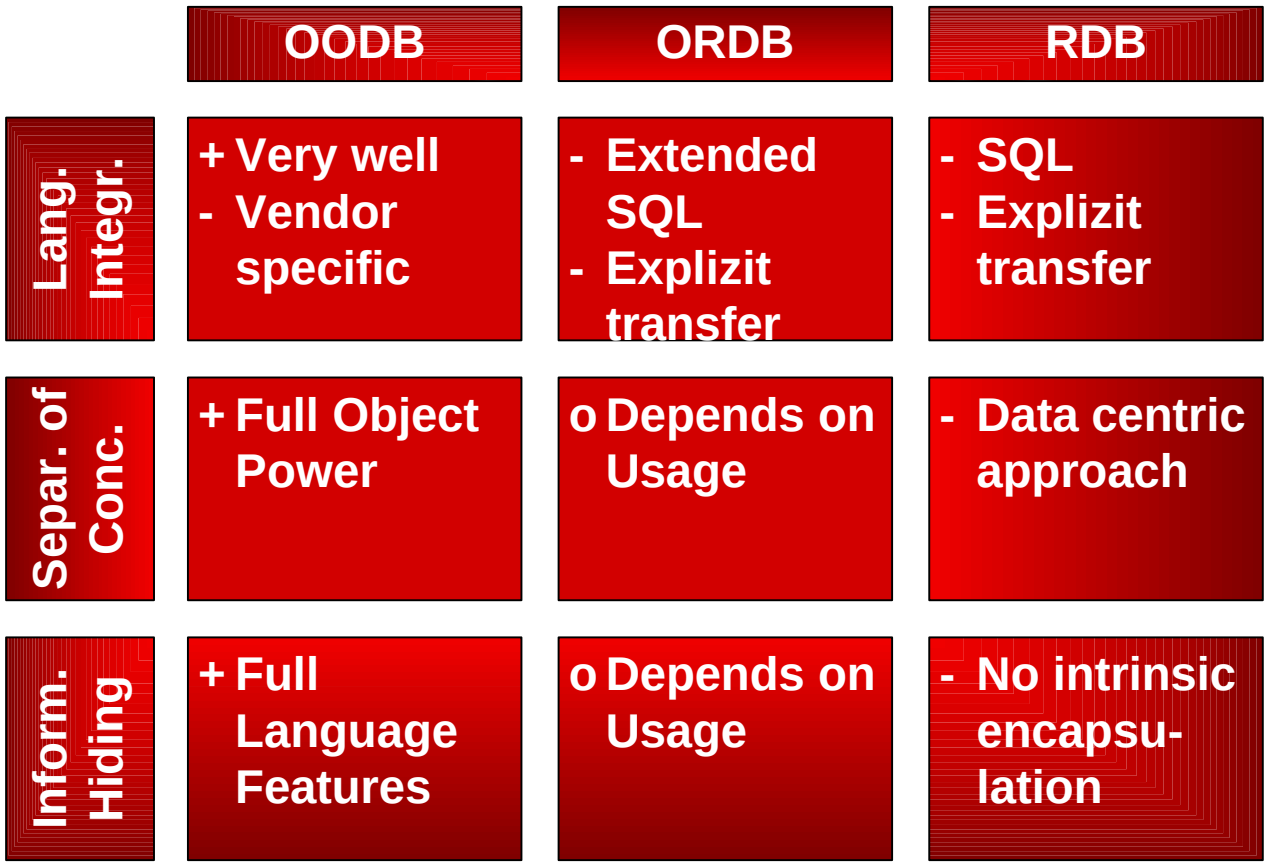


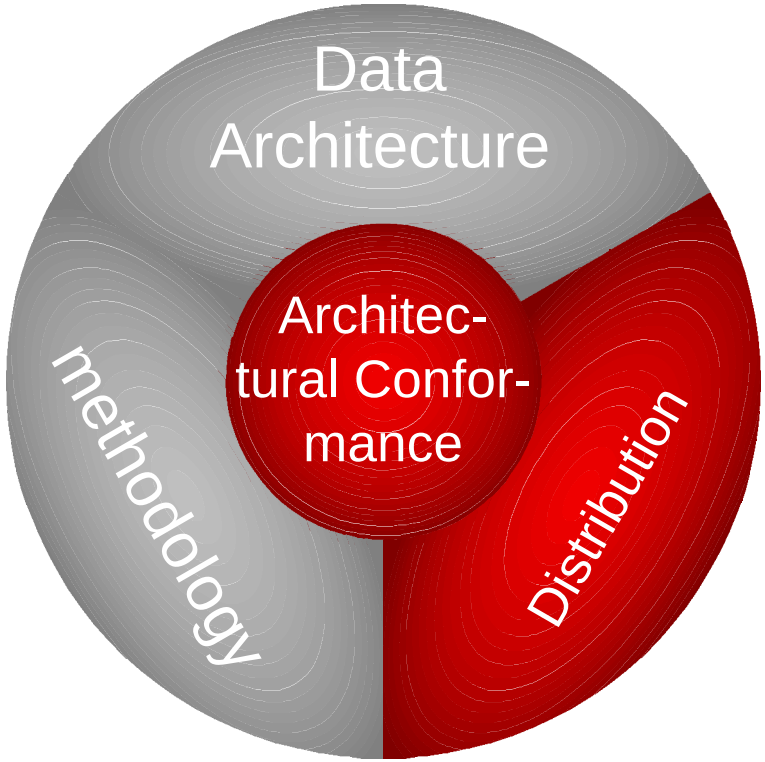
Storing objects you want to preserve your methodology

- Language integration
  - A single language (and type system) should apply, no matter whether information is transient or persistent
- Information Hiding
  - Only the object itself knows what data it is storing
- Separation of concerns
  - Every part of the system should have its well-defined, exclusive responsibility

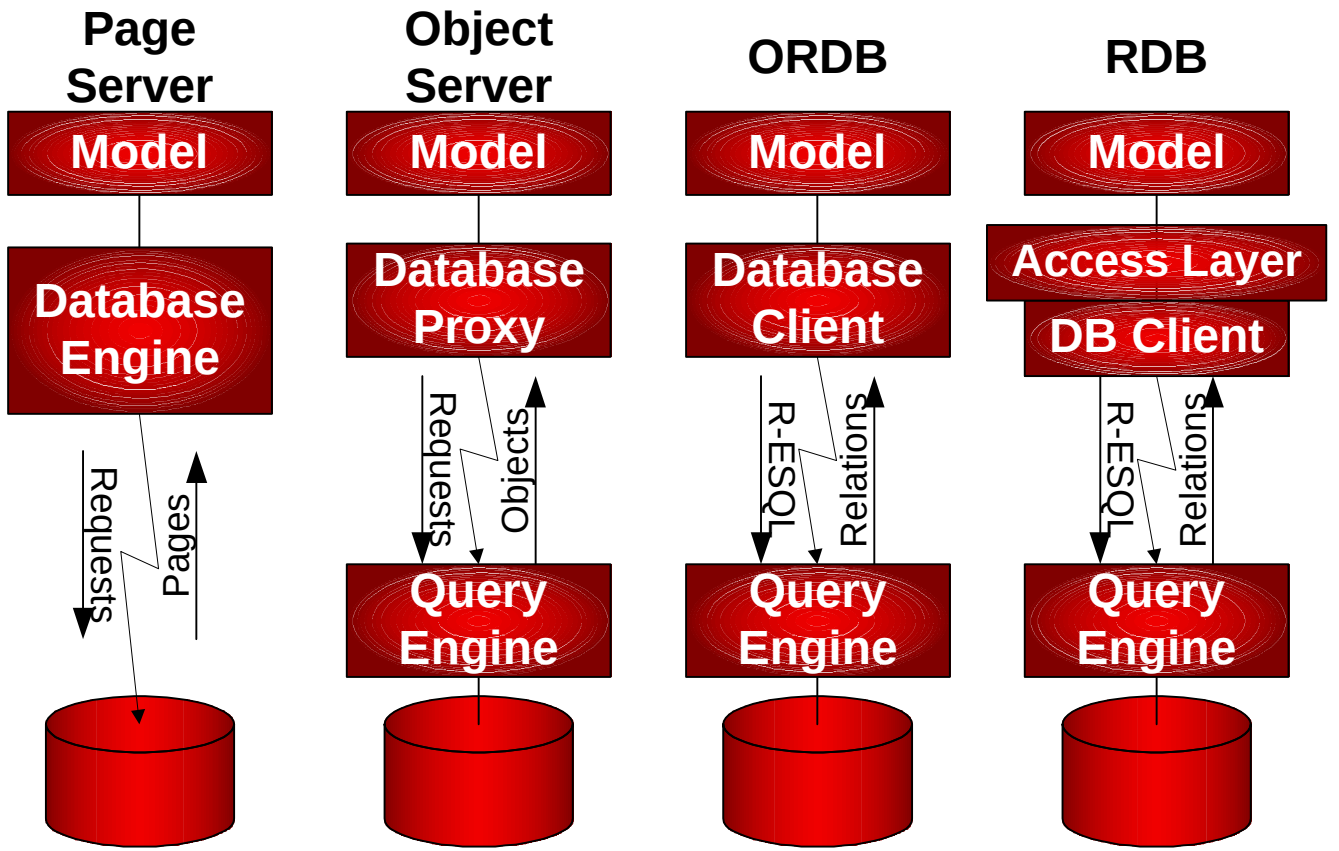
Many successful system have been built without sticking to these rules dogmatically

The paradigm gap is quite obvious here

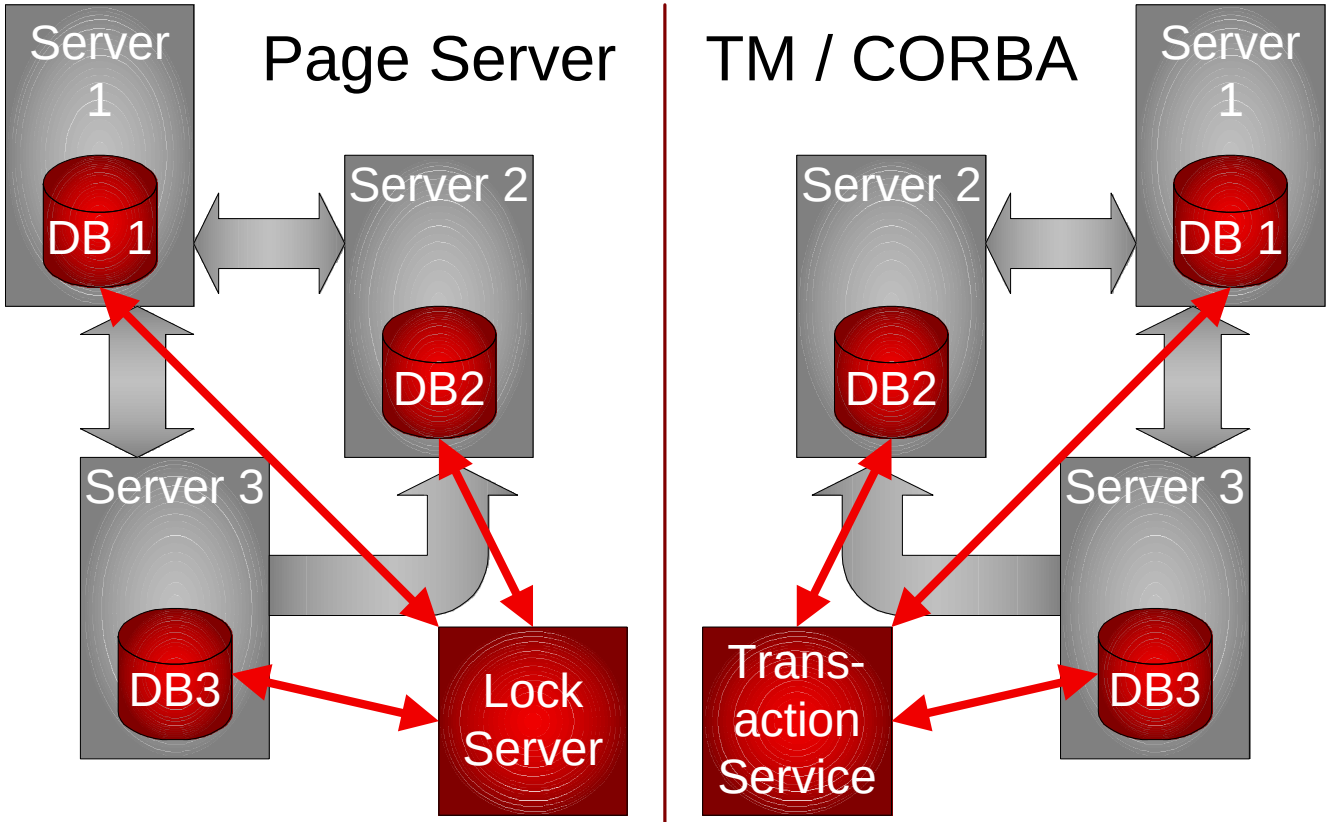




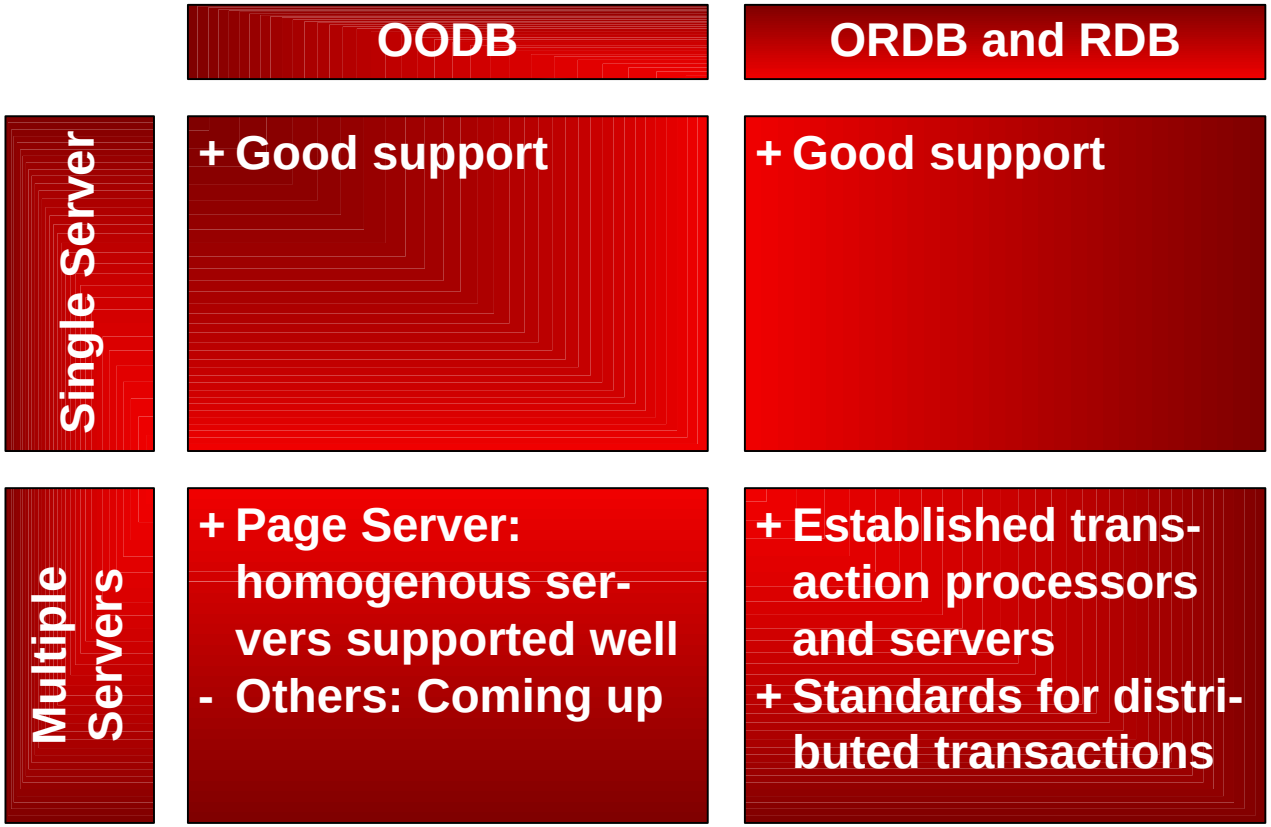
All solutions support single server architectures with specific strengths



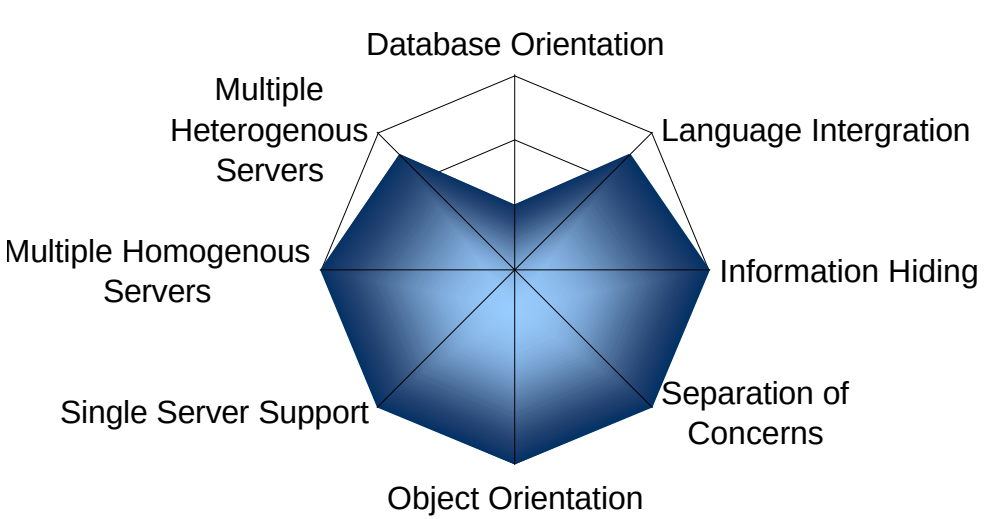
For multi-server architectures you need extra services



The product is more important than the technology



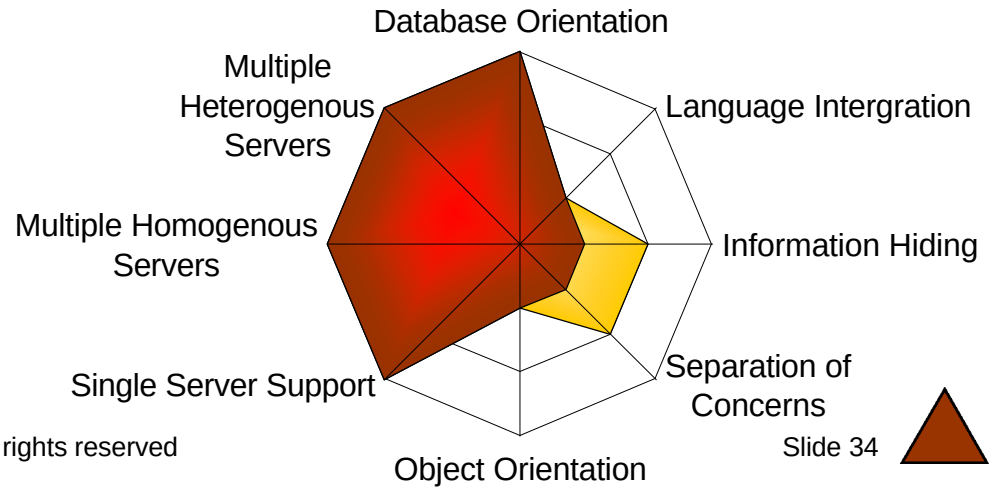
Architectural considerations don't suppose a single solution



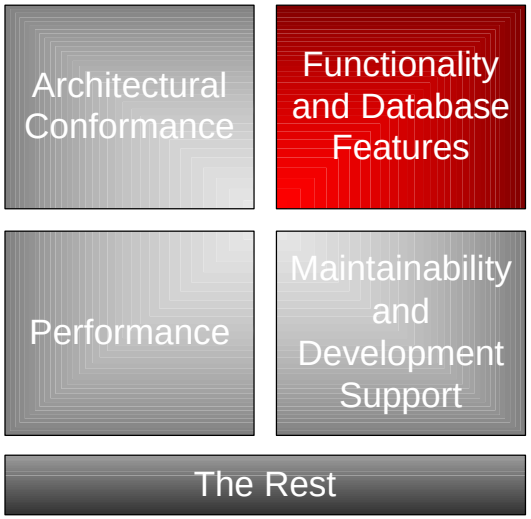
ODBMS

ORDBMS

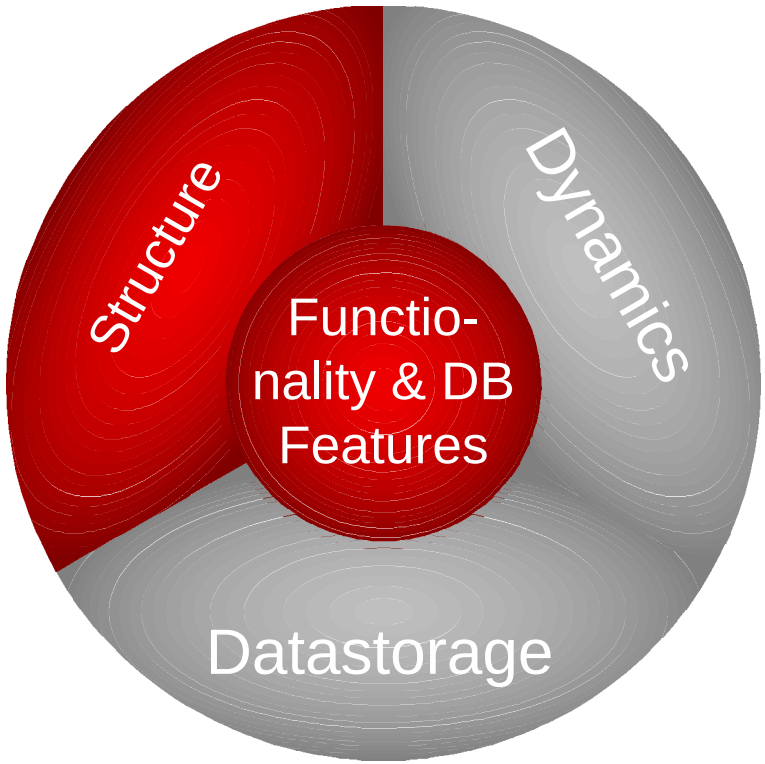
RDBMS



- Introduction
- The Problem: Storing Objects
- Requirements of the Stakeholders
- **The important forces in depth (II)**
- Turning forces into decisions
- Summary



It is important not to miss any of the aspects that comprise functionality



There are a lot of details to be considered

Structure

- Object granularity
- Inheritance
- Number of Classes
- Collections
- Versioning and History

Dynamics

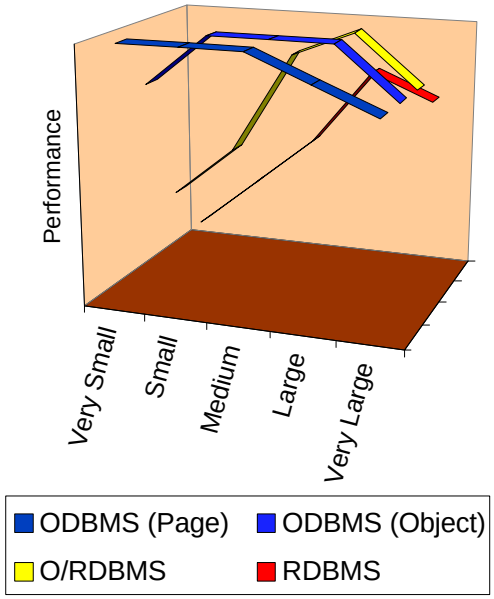
- Access Characteristics
- Query Complexity and Flexibility
- Mass Updates

Data Storage

- Transaction Model
- Concurrency and Locking
- Scalability

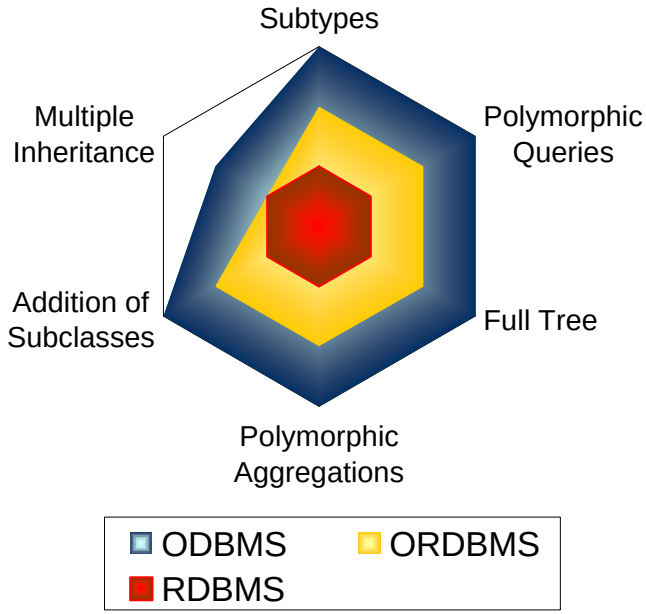
Object granularity is an important factor

- Analysis and design techniques influence object size
  - Large objects result from a data-focussed analysis
  - Small fine-grained objects result from a use-case driven approach
- High reusability often results in fine-grained objects



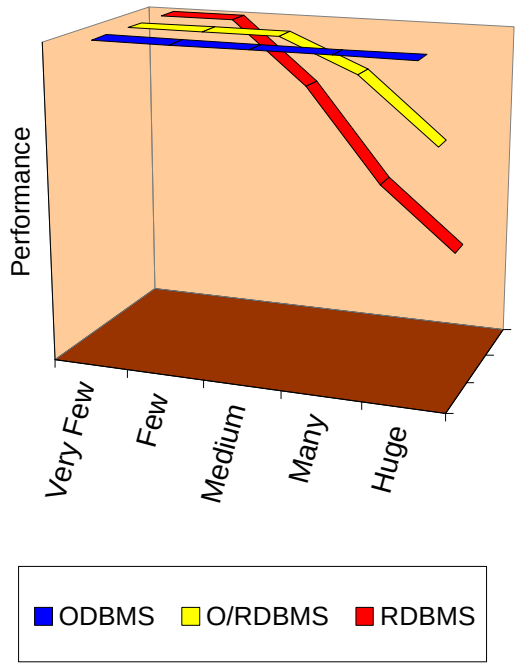
Inheritance support is one of the crucial differences

- Inheritance support implies:
  - Subtypes
  - Polymorphic queries
  - Complete tree support
  - Polymorphic aggregations
  - Fast addition of new subclasses
  - (Multiple Inheritance)



The number of classes prevent a purely relational solution

- A fine-grained design usually results in many classes
- The more classes you have the more you rely on polymorphism
- A direct O/R approach maps classes to tables...



Support for templates and collections is less natural than you may guess

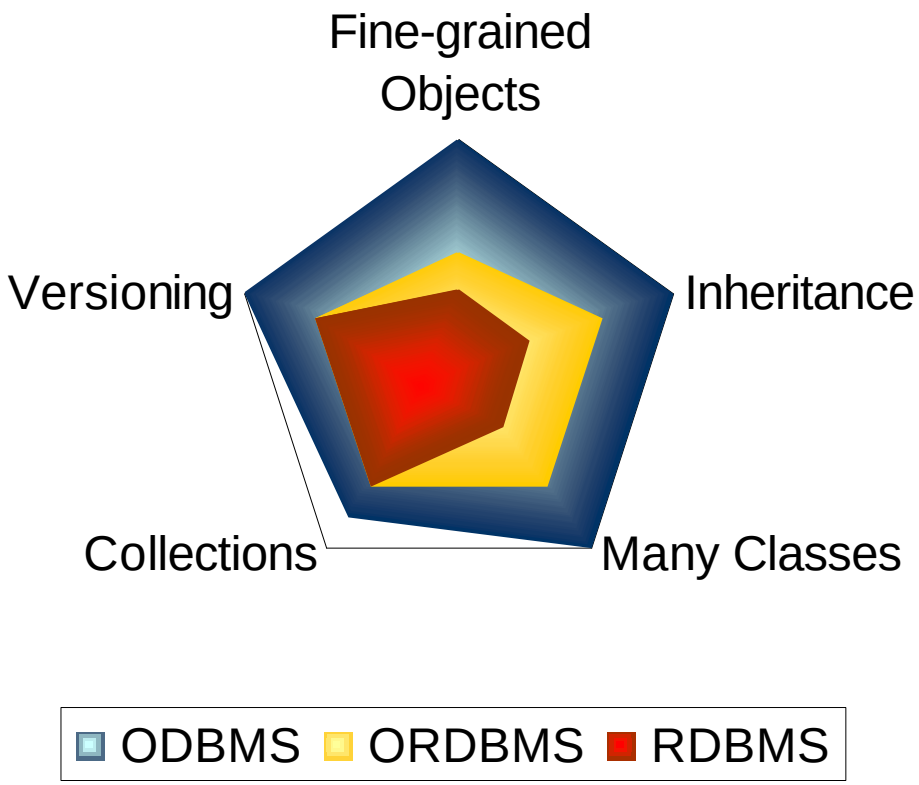
- Compliance to class library
- Different database representations
  - Indexed Collections → Direct Storage (Bi-directional cursors!)
  - Dictionaries →Indices
  - Sets →Special (ODB) or UNIQUE (RDB)
- ODBs offer additional bi-directional Associations

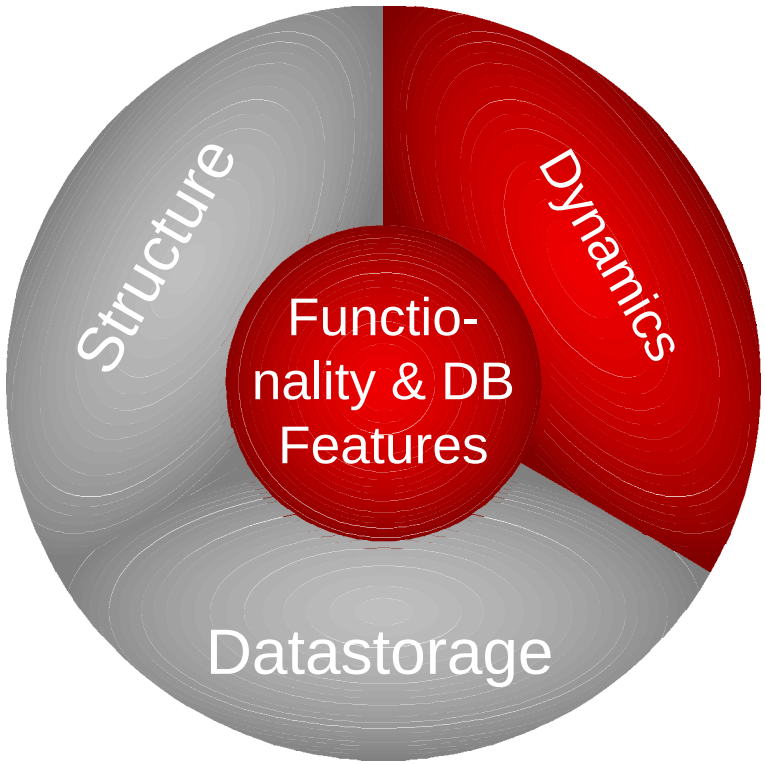
Important differences between products!

Support for versions and history are sometimes helpful

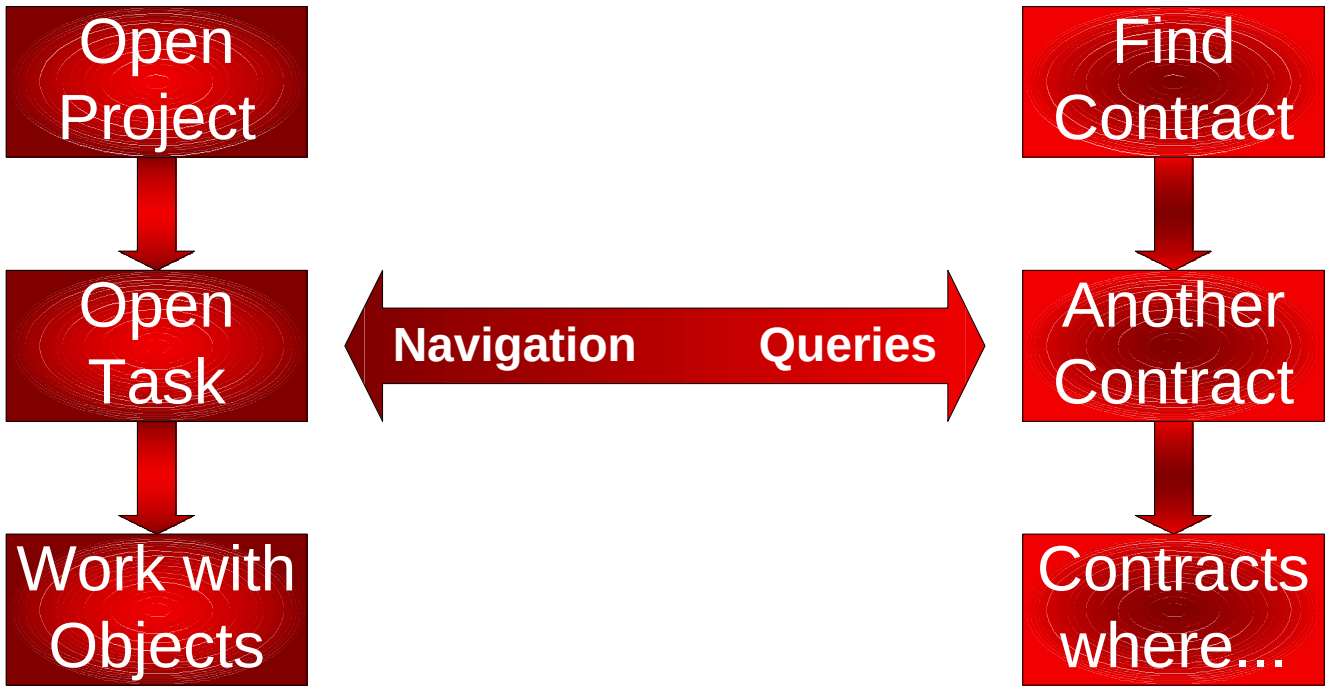
- Implementing versioning with a relational database is well-understood but needs resources
- Many object databases support one-dimensional multi-thread versions
- More of a goodie than a crucial feature

Concerning structure ODBMS fit best





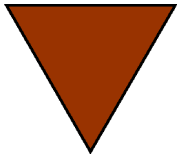
Database applications show typical access characteristics



ODBMS support navigation, (O)RDBMS querying

- ODBMS
  - store objects regardless of class
  - are optimized to follow references
  - provide clustering for additional navigation support
  - have weak query engines
- (O)RDBMS
  - store objects according to class
  - use (slow) joins to navigate
  - have excellent query engines
- ORDBMS
  - can navigate aggregations fast

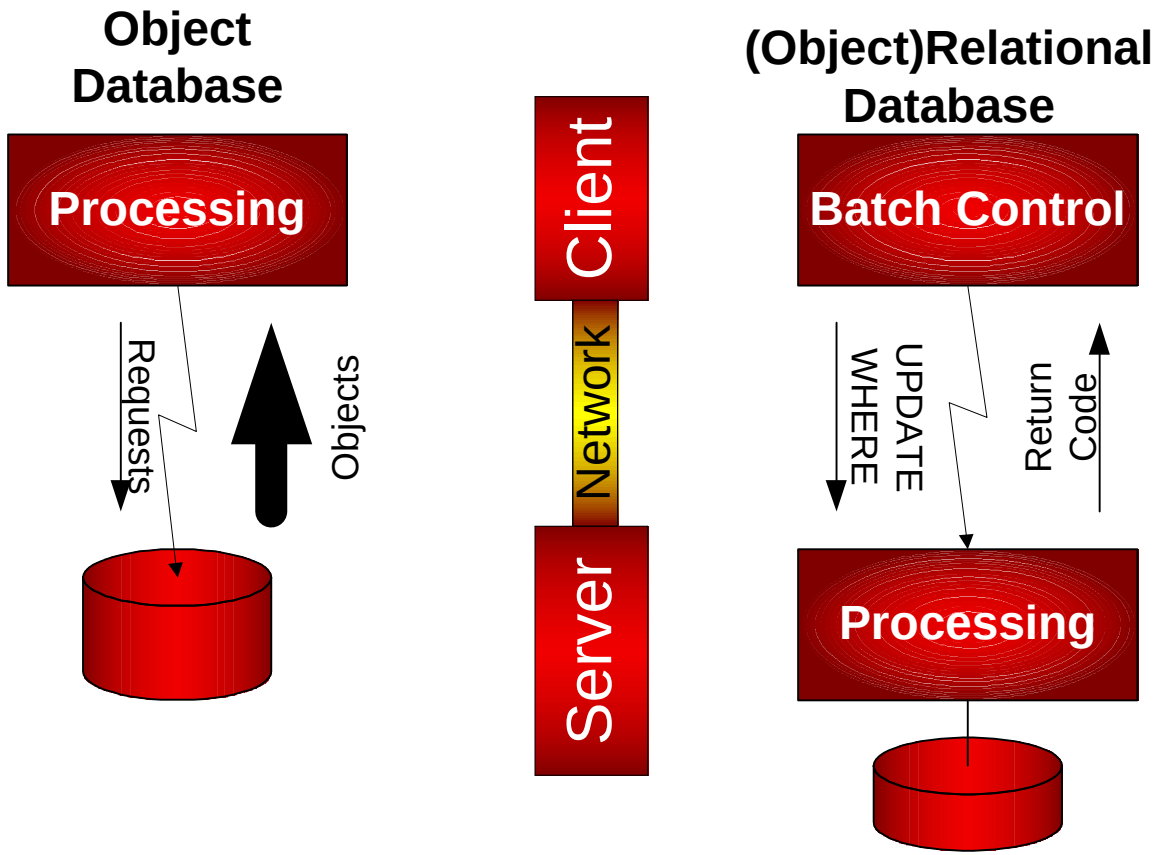
This is one of the most important forces



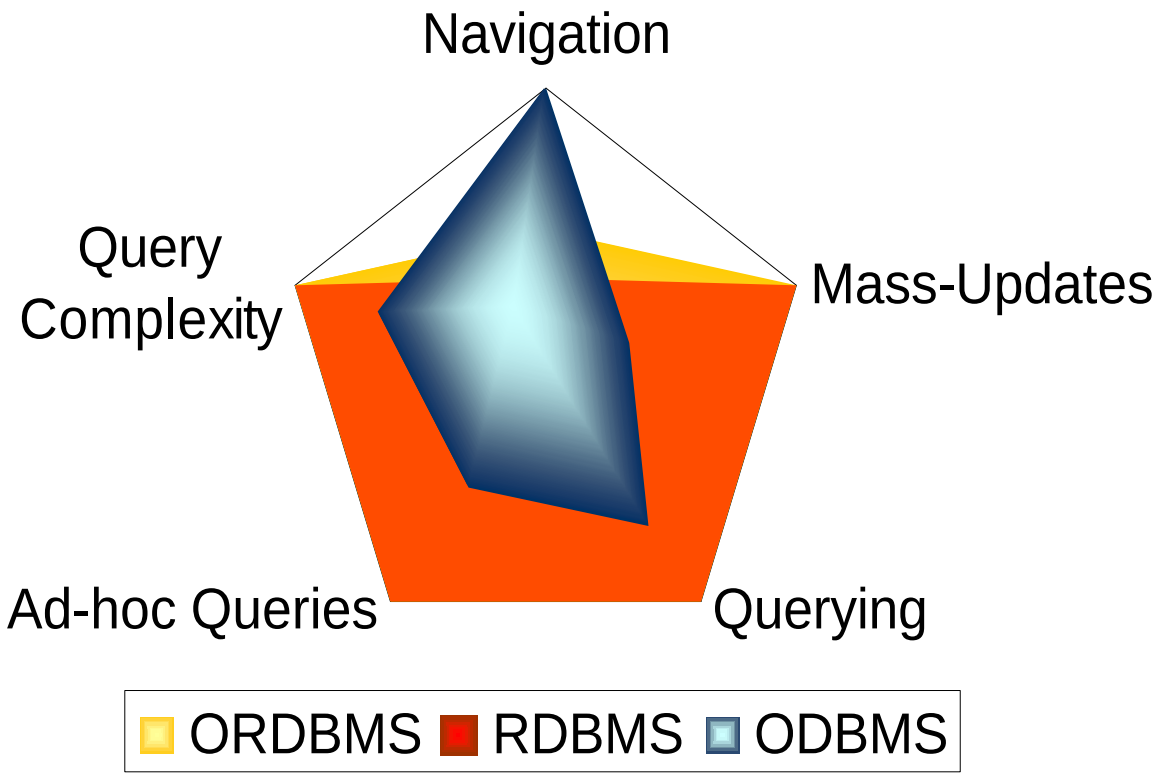
Consequently (O)RDBMS offer much better support for queries

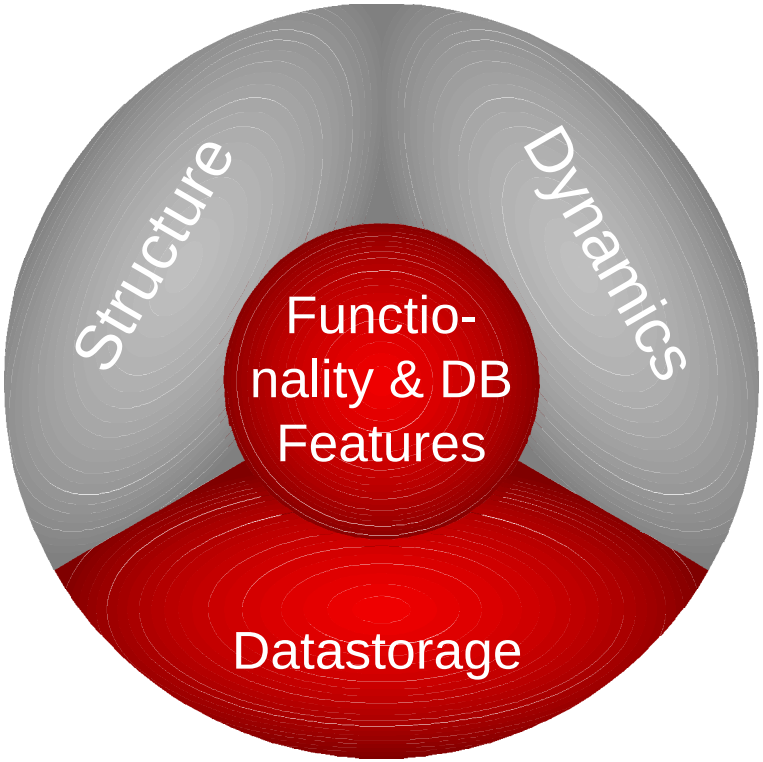
- ODBMS
  - Medium support for complex object selections
  - Attribute queries violate encapsulation
  - No ad-hoc queries (Caution: Data Warehouses...)
- (O)RDBMS
  - Excellent support for complex queries
  - Paradigm supports free queries on all data (everything is global!)
  - Good support for ad-hoc queries

ORDBMS usually don't support mass updates or stored procedures



Summary: Navigational applications call for ODBMS, querying apps for RDBMS

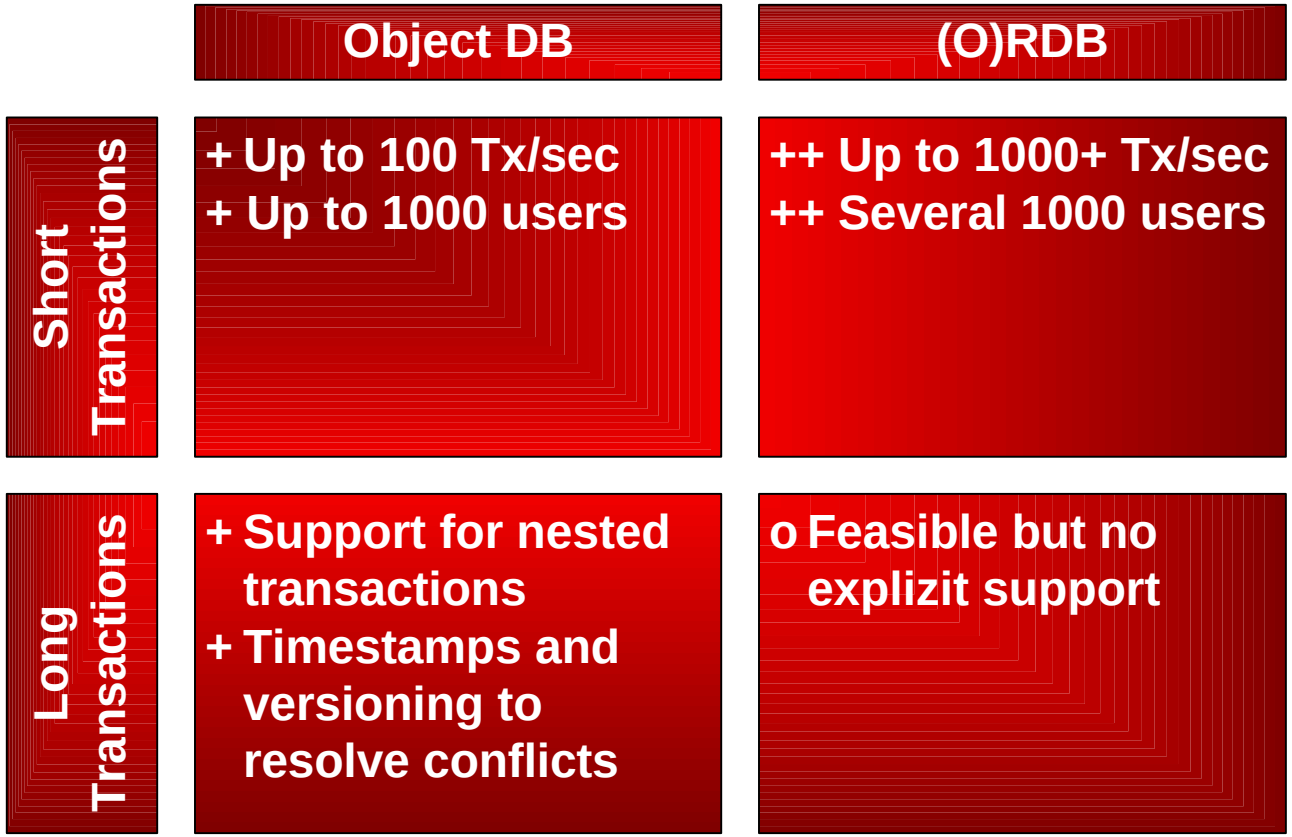


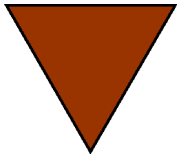


Today's applications feature two different transaction styles

- Check-in/Check-out semantics
- Long Transactions
  - May last for hours or even weeks
  - High probability of conflicts
  - Only few transactions per second
  - Call for nested transactions
- Classic semantics
- Short transactions
  - Last for a fraction of seconds
  - Low probability of conflicts
  - More than 1000 transactions per second

The differences become relevant in extreme areas only



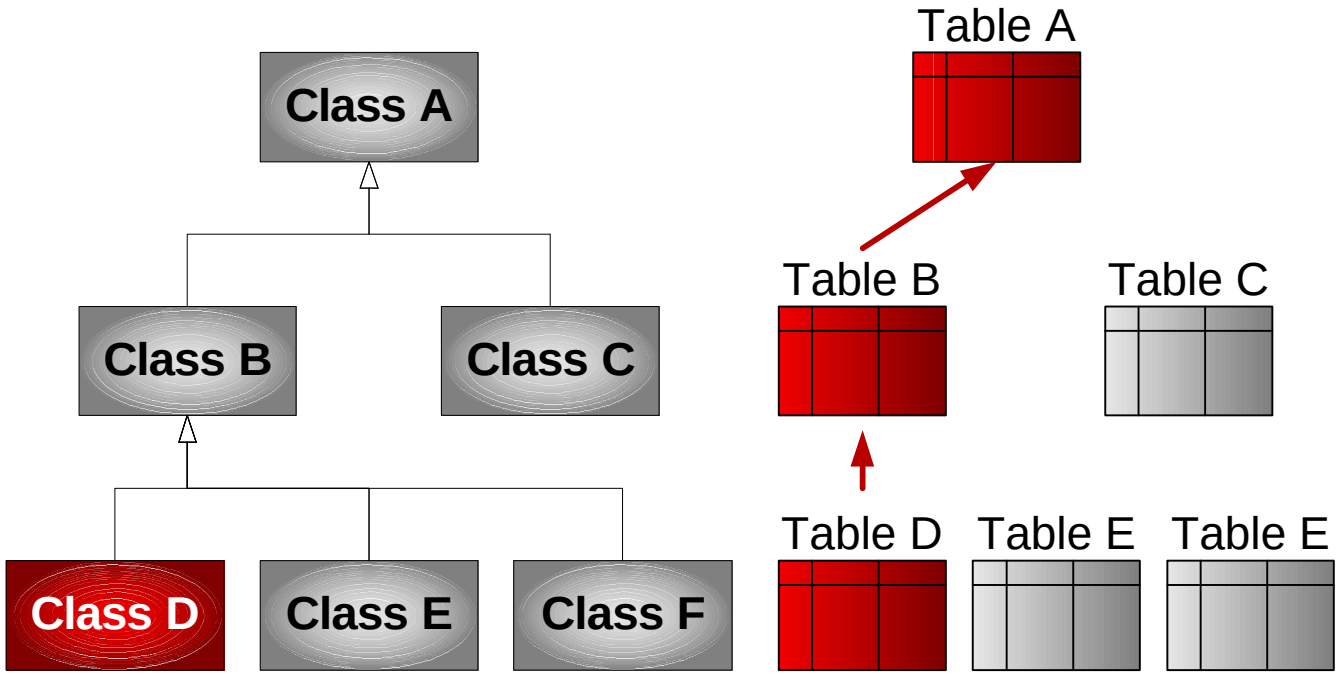


There are two different locking techniques in the ODBMS world

- Object Locking
  - \+ Low risk of conflicts and deadlocks
  - – High overhead for locking
- Page Locking
  - – Higher risk of conflicts and deadlocks
  - \+ Better performance

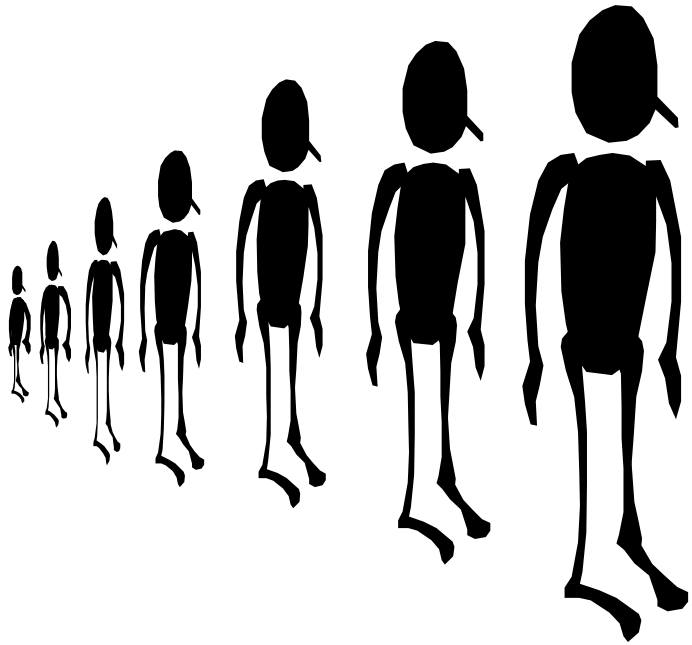
- Unless you have extreme performance requirements, the best tradeoff is hard to find
- Some products offer both strategies
- You have to experiment with your project's profile

Locking can cause disasters with naive O/R Mapping

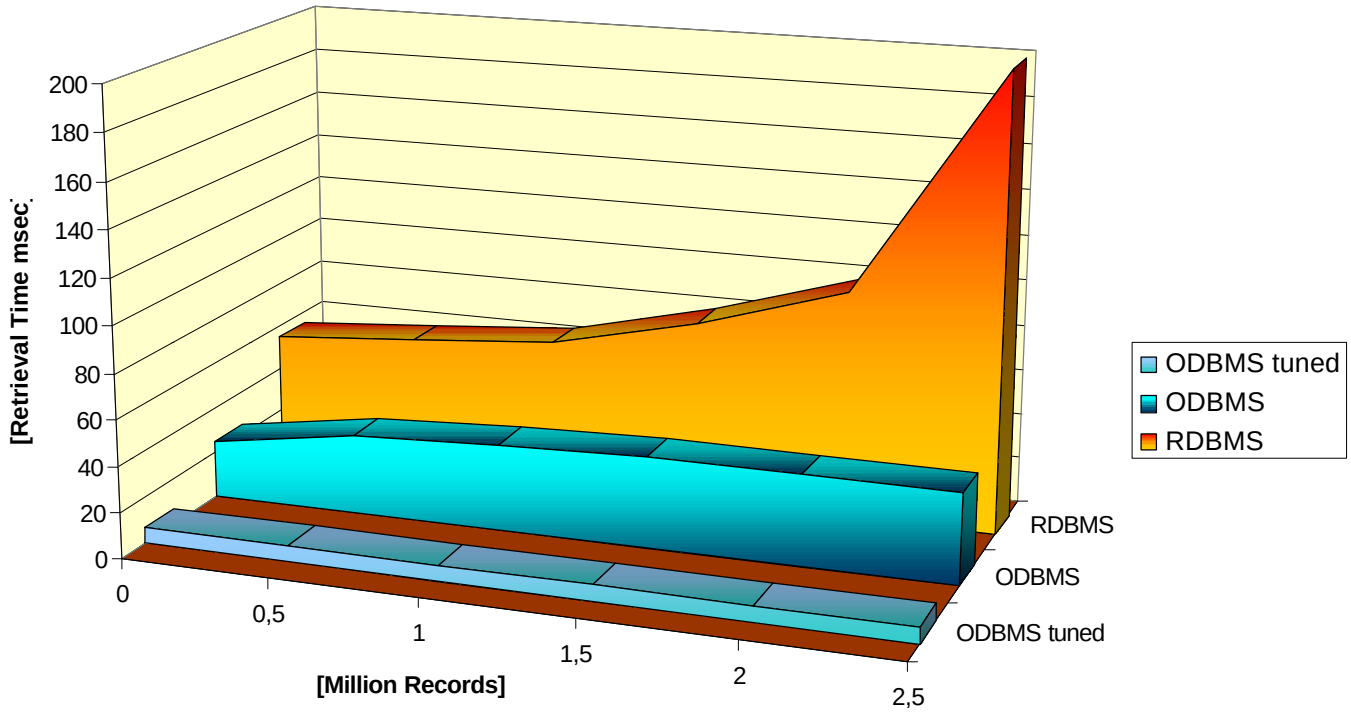


All technologies provide good scalability

- Both technologies work with extremly large databases
- Access time for simple retrieval:
  - ODBMS: const
  - RDBMS: $\log_m n$

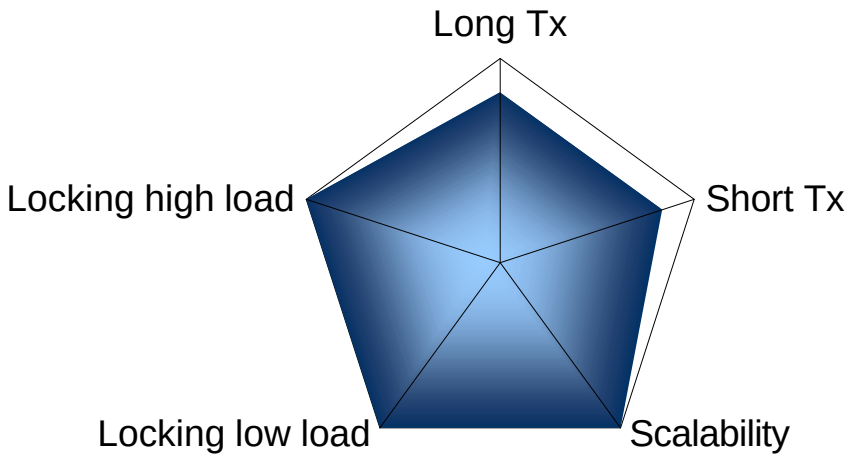


Some hard data from the MMIS project



Source: A. Chaudhri: *Object Databases in Practice - Tutorial;* OOPSLA'97 Workshop on *Experiences using Object Data Management in the Real World*
http://www.soi.city.ac.uk/~akmal/oopsla97.dir/workshop.html

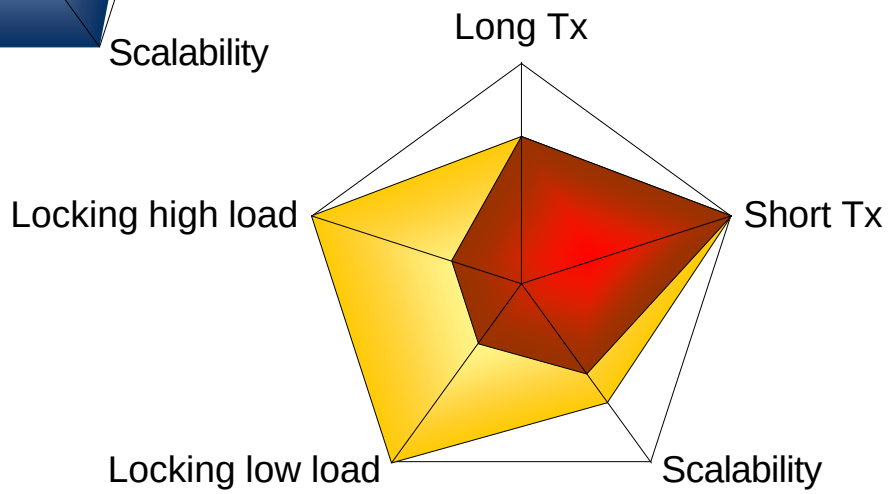
Beware of the locking characteristics of an O/R access layer!



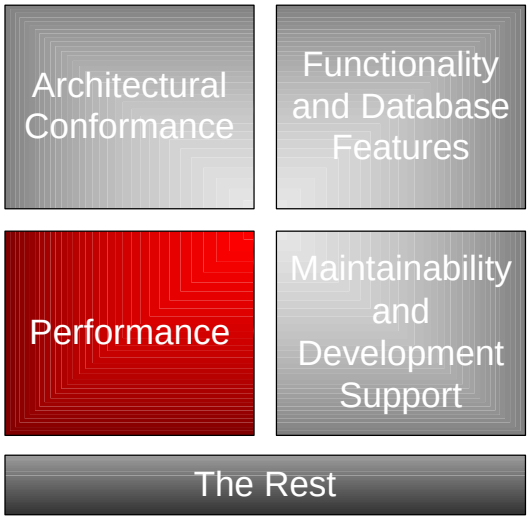
ODBMS

ORDBMS

RDBMS



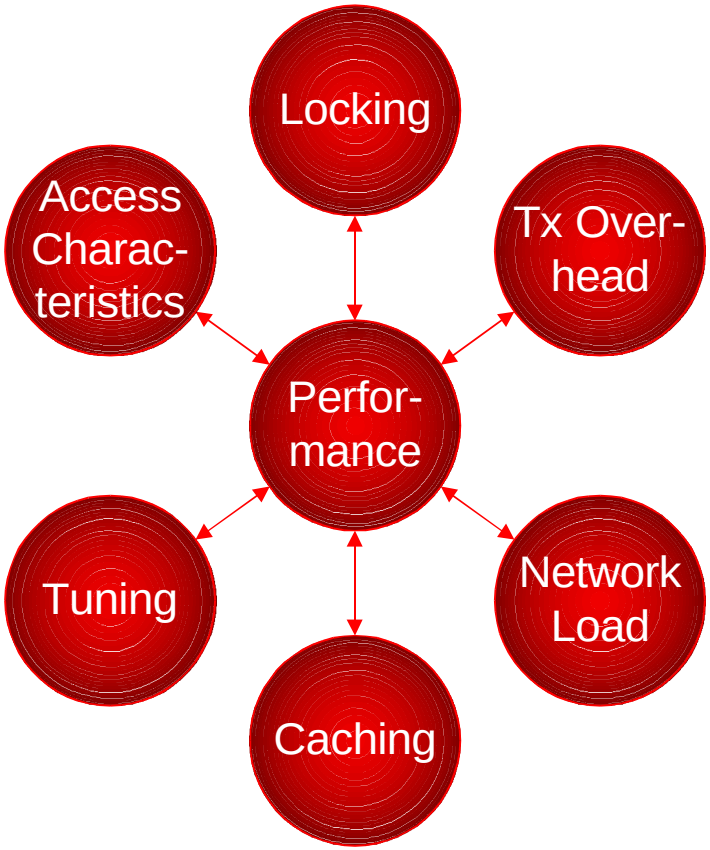
- Introduction
- The Problem: Storing Objects
- Requirements of the Stakeholders
- **The important forces in depth (III)**
- Turning forces into decisions



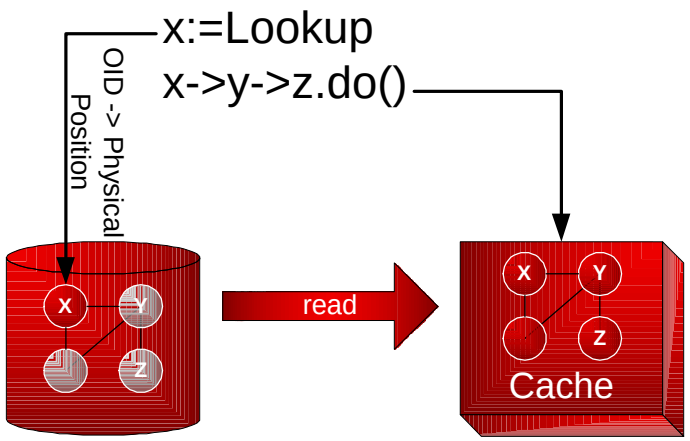
In general ODBMS have significantly better performance

- Ratio between ODBMS and RDBMS
  - Navigational Lookup: 10-100 times
  - Queries: 0,1-10 times
  - Insert: 1-10 times
- Access layers make it even worse
- Advantage decreases with transaction load
- Collection of bench-marks: www.soi.city.ac.uk/~akmal/
- Don't take these numbers for granted: Make your own tests!

Performance depends on several factors



Key to performance: Direct access and clustering instead of foreign keys



- The more the application uses navigation the higher your gain
- Significant differences between products
  - Page servers show faster access but worse locking behavior
  - Moving an object may change OID
- ORDBMS also use OIDs (called REF)

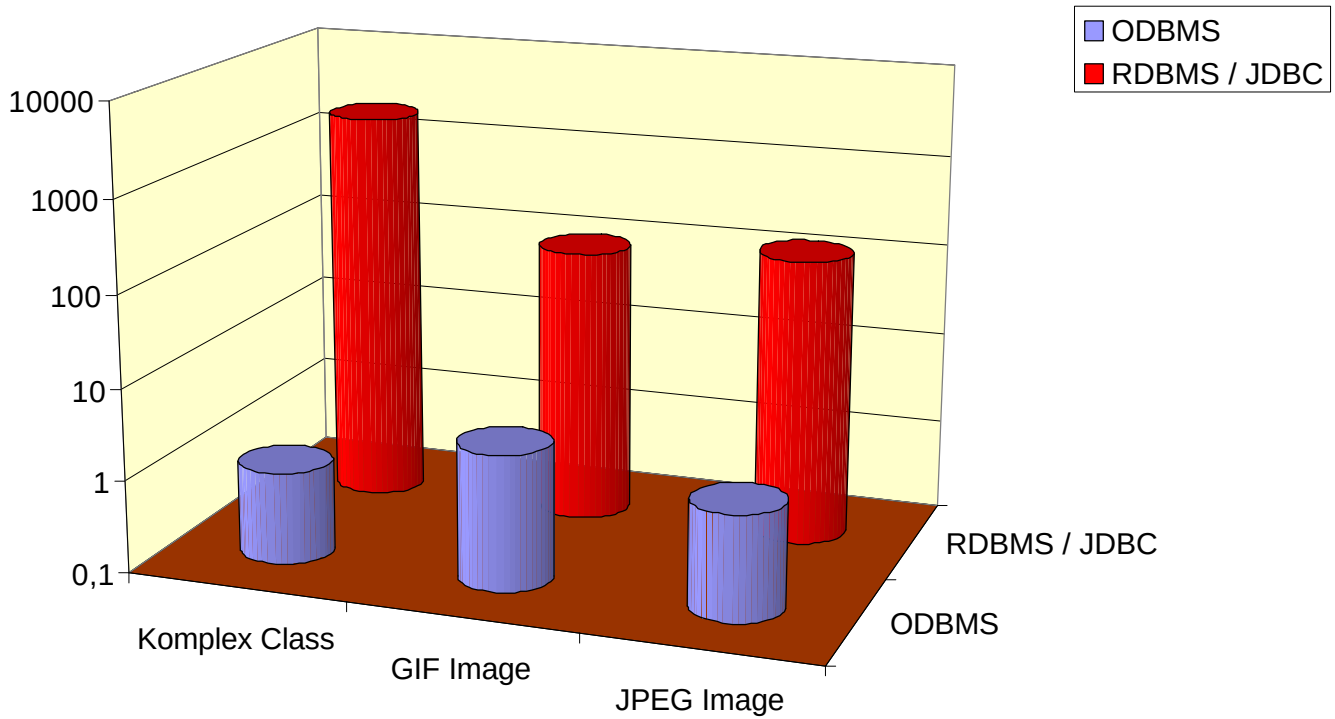
Tuning techniques and support differ

- ODBMS
  - Clustering
  - Cache adjustment
  - Indices and query optimization
- Mediocre tools for profiling

- (O)RDBMS
  - Denormalization (!)
  - Query optimization
  - Indices
  - Technical parameters
- Excellent profiling support

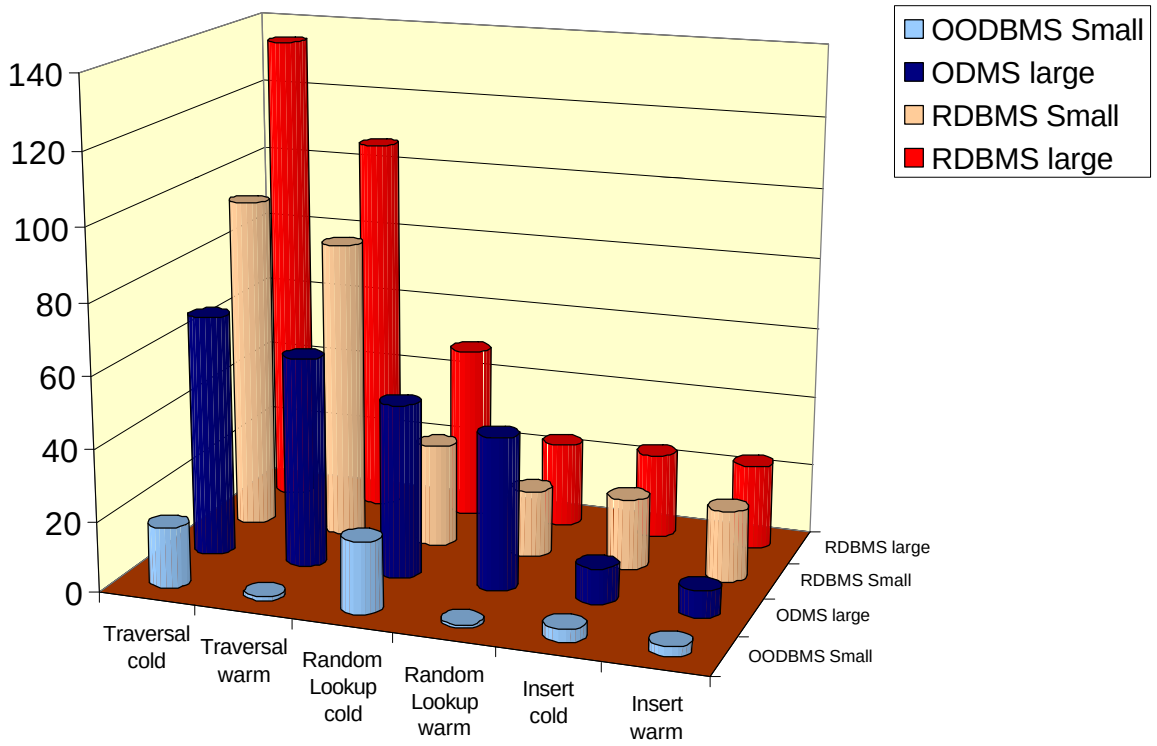
A real world comparison...

(Object Store for NT 5.0 against JDBC 1.1.3/Oracle 7.3)



Source: Strategic Technology Resources: *Java Data Management - Comparing ODBMS and RDBMS Implementations of Quantum Objects*, White Paper, Chicago, Illinois; 1997; http://www.odi.com/content/white_papers/str-odb-rdb.pdf

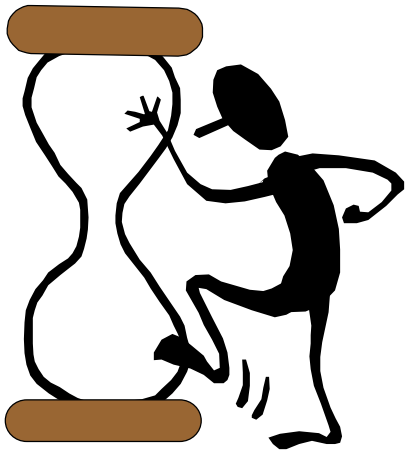
...and a more elaborate benchmark



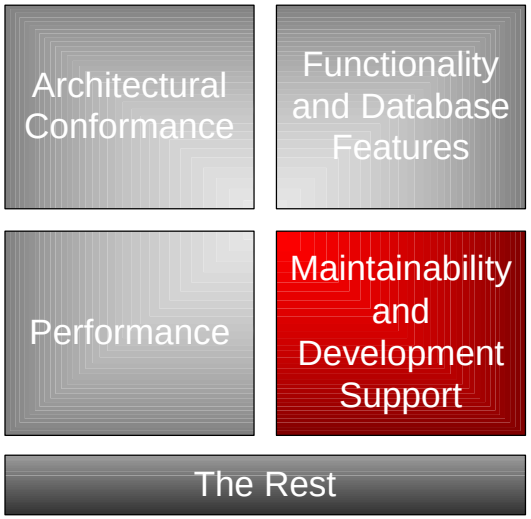
Source: R. Catell: *An Engineering Database Benchmark* in: J. Gray (ed.): *The Benchmark Handbook for Database and Transaction Systems,* Morgan-Kaufman, San Mateo, California. 1991

A good benchmark takes all this into account

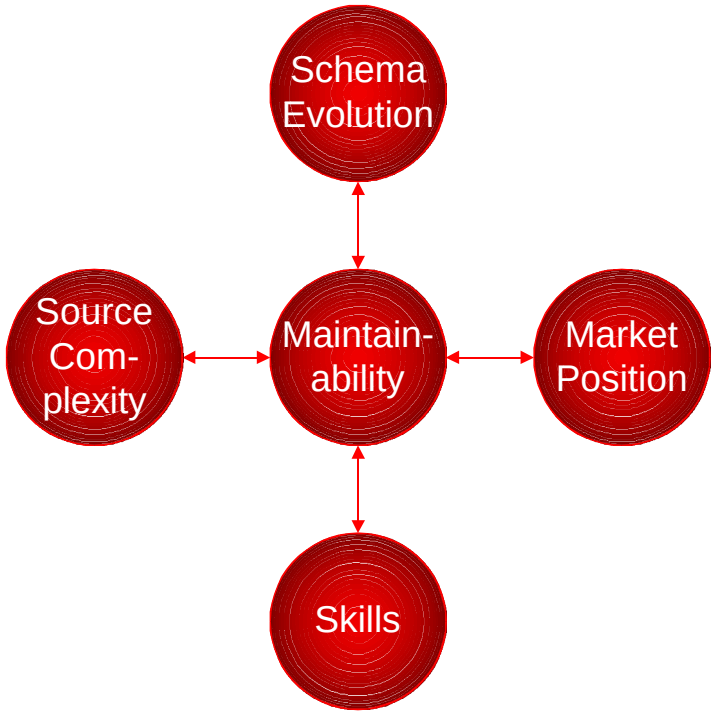
- A benchmark should contain
  - Emulation of access characteristics
  - Database size
  - Simulation of heavy multi-user operation
  - Checks on fragmentation
- Please keep in mind:
  - Performance only needs to be "good enough"
  - Benchmarks should be time-boxed



- Introduction
- The Problem: Storing Objects
- Requirements of the Stakeholders
- **The important forces in depth (IV)**
- Turning forces into decisions

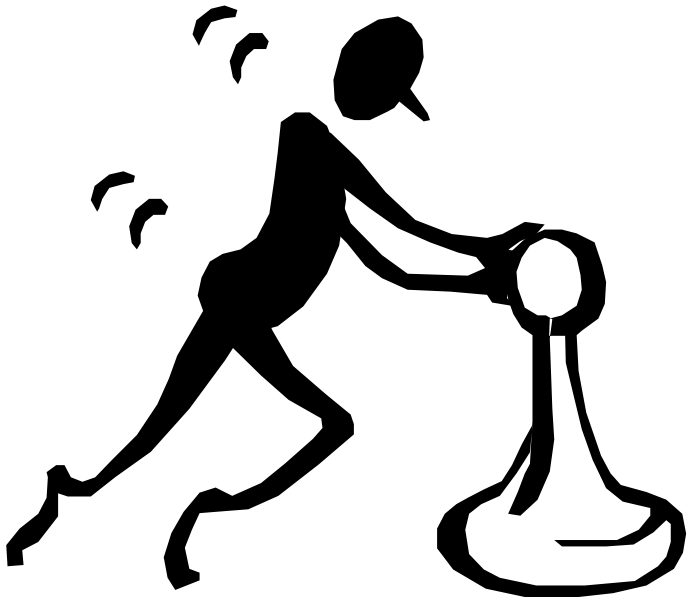


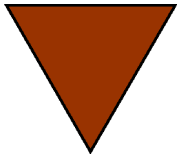
Maintainability is more than just using OO



Over the time the object model changes ("schema evolution")

- ODBMS
  - Provide support for adding attributes
  - Provide APIs to *program* more complex changes (lazy update)
- RDBMS
  - Views can handle most changes via administration





The vendor's market position is important to estimate long-term support

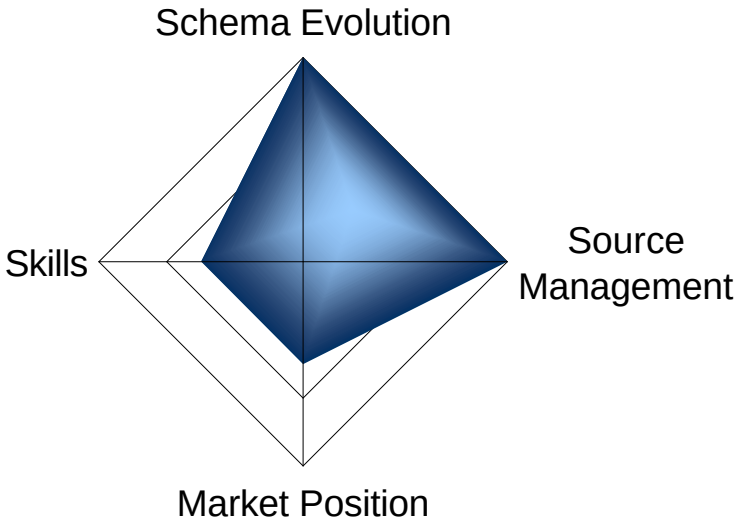
- ODBMS
  - Large vendors exist for a decade now
  - Total revenue less than Oracle's expenses on advertising
  - Long-term survival of some vendors unknown
- (O)RDBMS
  - Vendors have established market positions
  - Long-term survival near to guaranteed
  - O/R Databases have unknown future

Complex sources only annoy developers but may cause real problems later

- More sources mean
  - More complex configuration management
  - More points to change
  - More opportunities for errors

- ODBMS
  - Language integration provides single-source (with exceptions)
- (O)RDBMS
  - Database definition and mapping definition in extra files
  - Behavior in OO language and (E)SQL (and stored procedures)

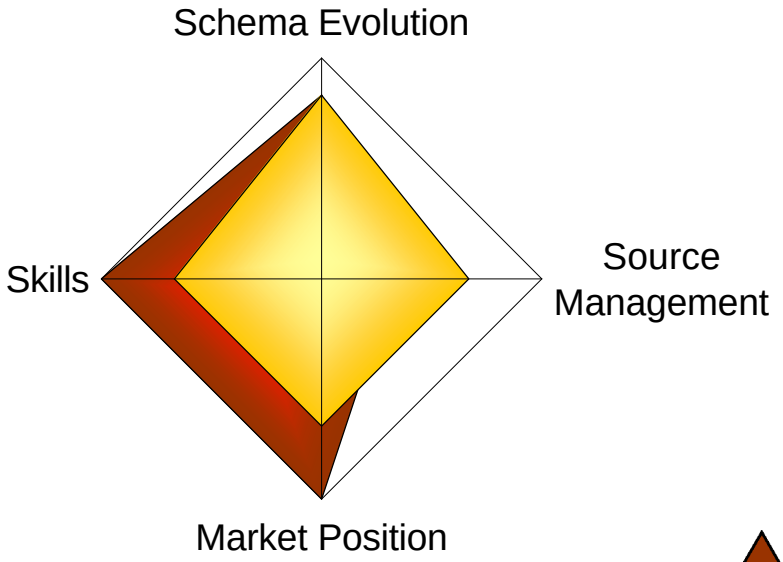
Which solution fits your maintenance needs depends on your requirements



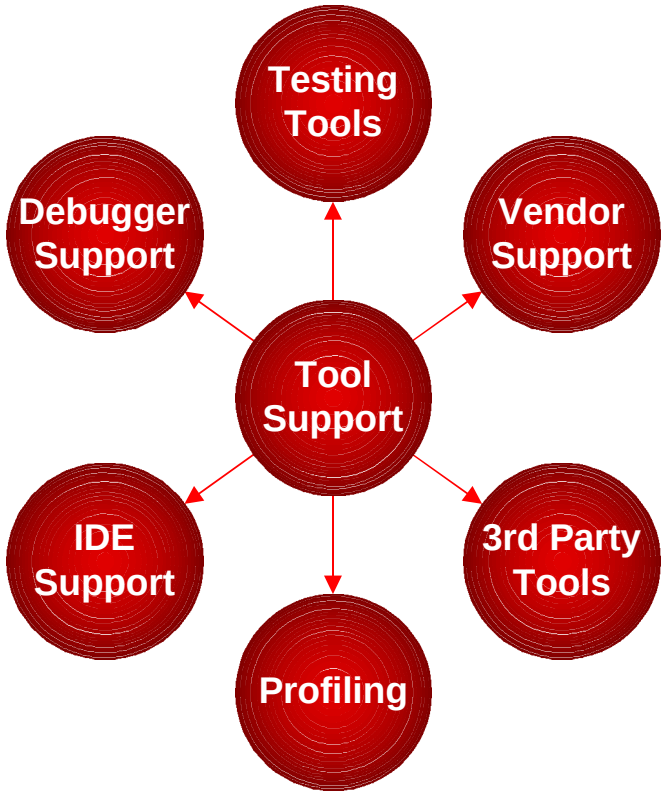
ODBMS

ORDBMS

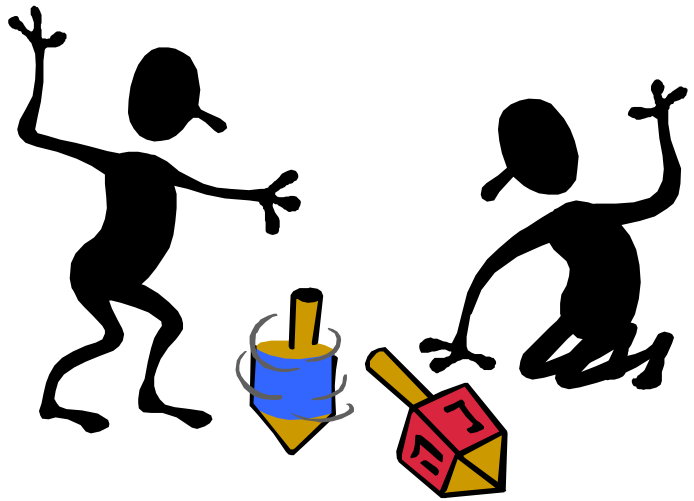
RDBMS



Some final remarks on development support

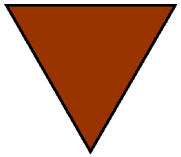


- Introduction
- The Problem: Storing Objects
- Requirements of the Stakeholders
- The important forces in depth
- **Turning forces into decisions**



Now that we have identified the forces, it's up to you to find a balance...

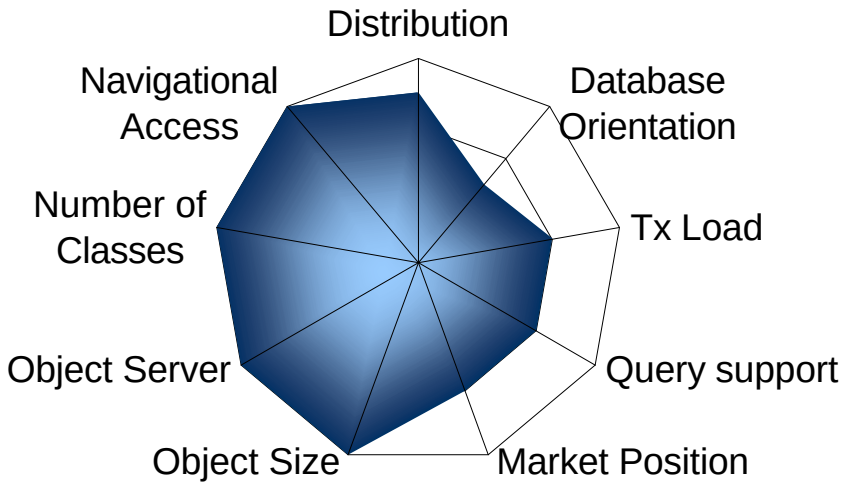
- Identify relevant forces
- Find your project's corresponding needs
- Identify the options that still remain
- Evaluate products
  - attend classes
  - write prototypes
- Come up with suggestions

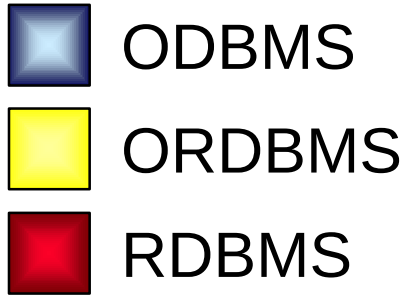


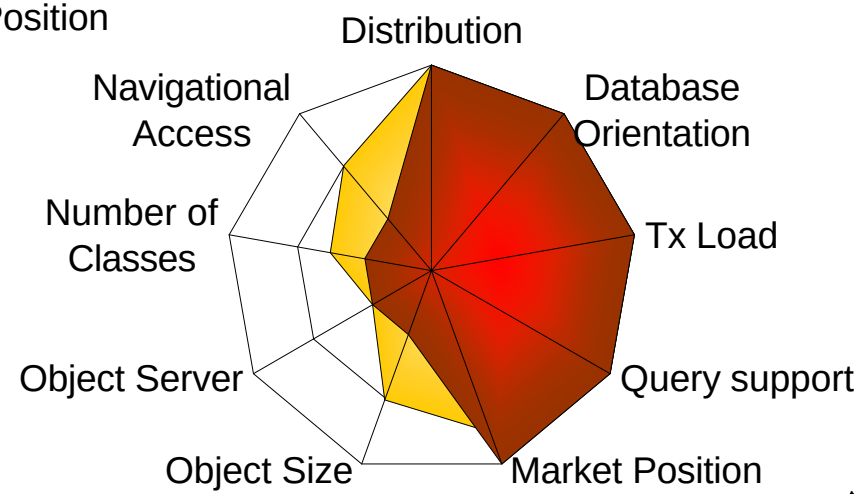
A minimal set of forces to decide technology may look like this

- Basic architecture (Database oriented versus object oriented)
- Distribution architecture
- Object granularity
- Access characteristics
- Number of classes
- Transaction load
- Market Position

This list of forces clearly shows strengths and weaknesses of both







... and it's up to you to fight the political stands

- Try to see the evaluation with the eyes of the stakeholders
- Try to foresee their objections
- Try to understand personal fears
- Don't be dogmatic

"People hate changes"
T. DeMarco

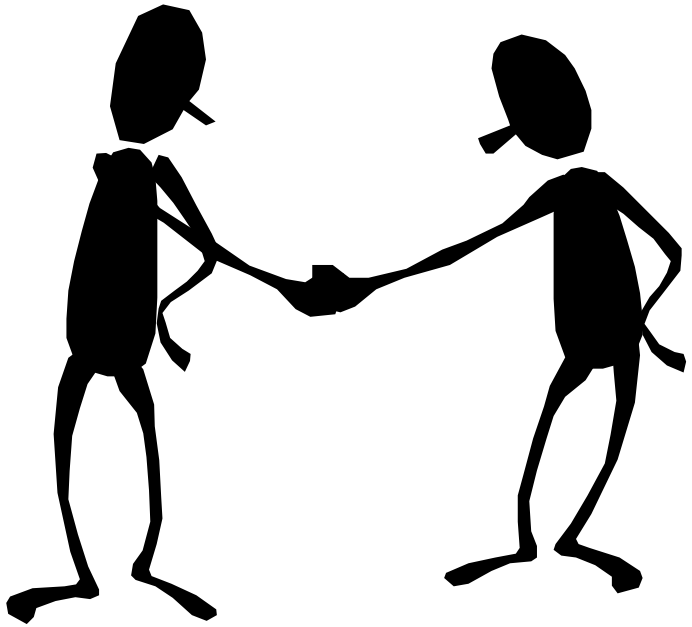
Thanks for their support

For additional input

- Akmal Chaudhri,
  Logica UK Ltd.
- Uwe Beßle,
  Versant
- Rüdiger Eberlein,
  Oracle

For fruitful discussions

- Wolfgang Keller,
  EA Generali



References

[Atk+89] **M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, S Zdonik:** *The Object-Oriented Database System Manifesto* in: *Deductive and Object-Oriented Databases, Proceedings of the First International Conference on Deductive and Object-Oriented Databases (DOOD'89)*, pp. 223-240

[Cat91] **R. G.G. Catell:** *An Engineering Database Benchmark* in: **J. Gray (ed.):** *The Benchmark Handbook for Database and Transaction Systems*, Morgan-Kaufman, San Mateo, California. 1991

[Cat94] **R. G. G. Catell:** *Object Data Management - Object-Oriented and Extended Relational Database Systems - Revised Edition;* Addison-Wesley, Reading, Massachusetts, 1994

[Cha97] **A. Chaudhri:** *Object Databases in Practice - Tutorial;* OOPSLA'97 Workshop on *Experiences using Object Data Management in the Real World* http://www.soi.city.ac.uk/~akmal/oopsla97.dir/workshop.html

[Kel98] **W. Keller:** *Object/Relational Access Layers - A Roadmap, Missing Links, and More Patterns* in: **J. Coldewey, P. Dyson (eds.):** *Proceedings of the 3rd European Conference on Pattern Languages of Programming and Computing*, Universitäts Verlag Konstanz, to be published http://www.coldewey.com/europlop98/

[StM96] **M. Stonebraker, D. Moore:** *Object-Relational DBMSs - The Next Great Wave;* Morgan-Kaufman, San Mateo, California, 1996

[STS97] **Strategic Technology Resources:** *Java Data Management - Comparing ODBMS and RDBMS Implementations of Quantum Objects*, White Paper, Chicago, Illinois; 1997; http://www.odi.com/content/white_papers/str-odb-rdb.pdf