JDO

Java Data Objects

« Write once, persist everywhere »

New tools & standard for Java Persistence

LIBeLIS

Eric Samson

President

✉ eric@libelis.com

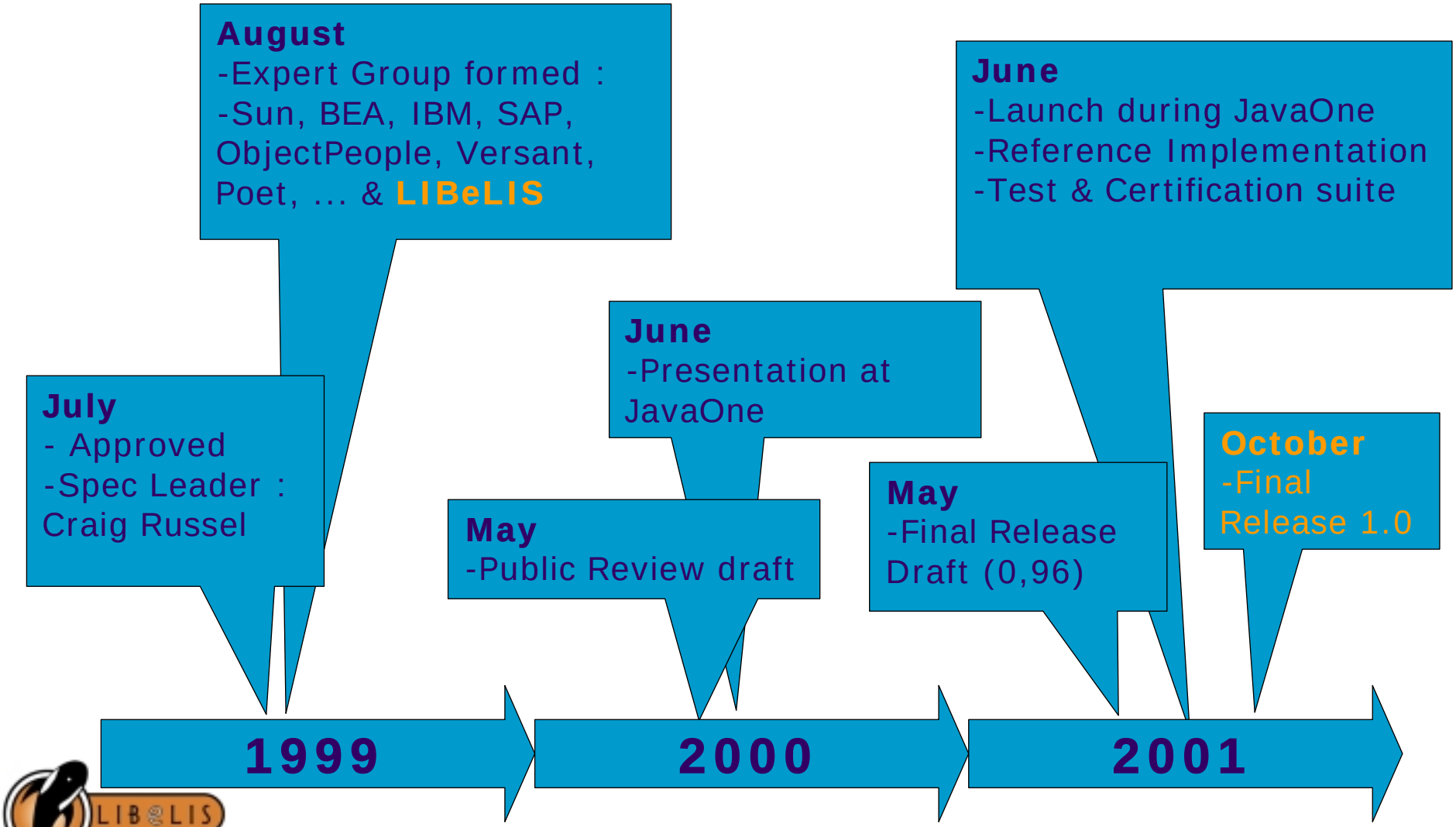
www.libelis.com

ObjectWeb

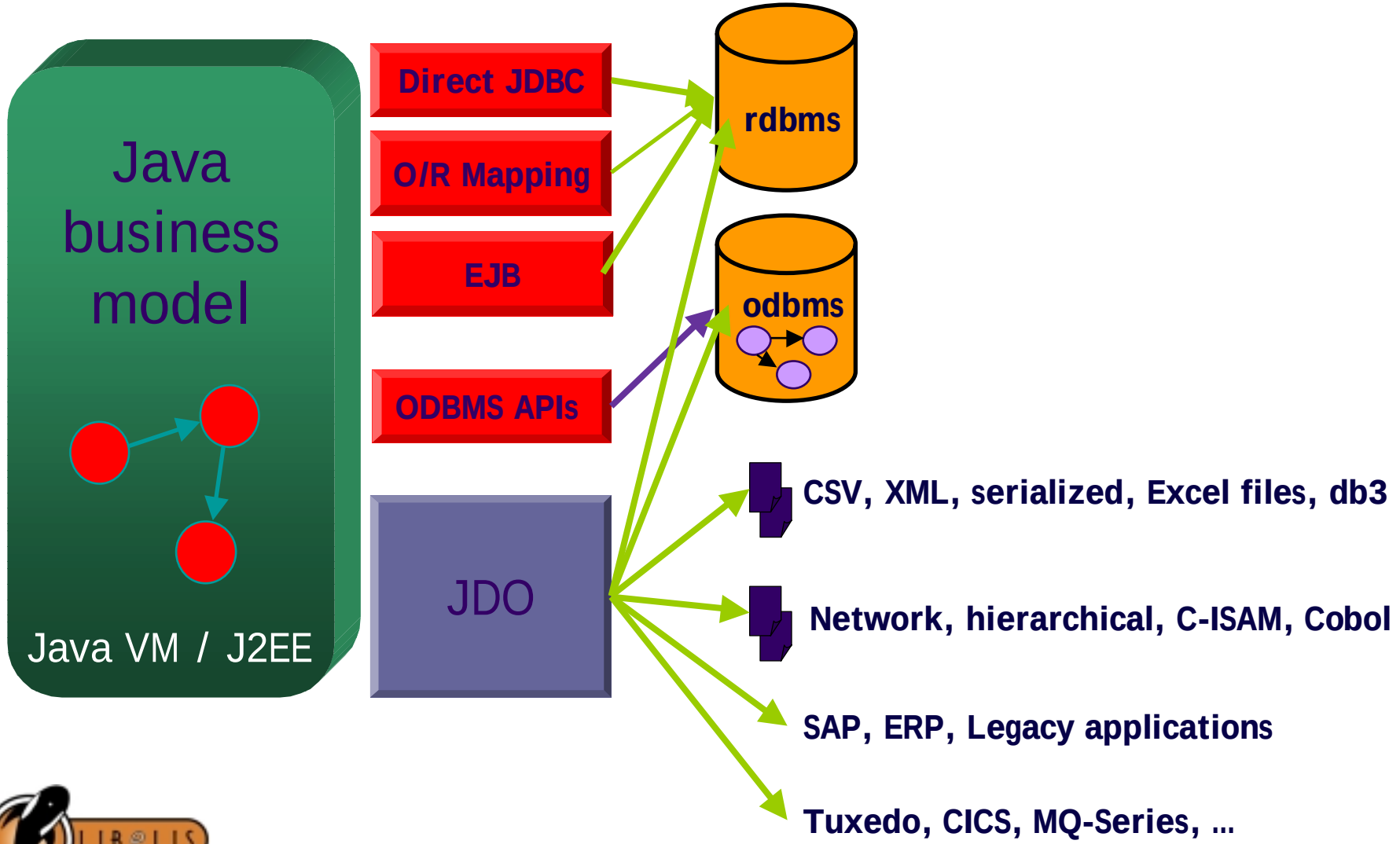
Open Source Middleware

www.objectweb.org

JDO history


August
-Expert Group formed :
-Sun, BEA, IBM, SAP, ObjectPeople, Versant, Poet, ... & LIBeLIS
July
- Approved
-Spec Leader :
Craig Russel
June
-Presentation at JavaOne
May
-Public Review draft
May
-Final Release Draft (0,96)
June
-Launch during JavaOne
-Reference Implementation
-Test & Certification suite
October
-Final Release 1.0
1999
2000
2001

Java Persistence



JDO goals

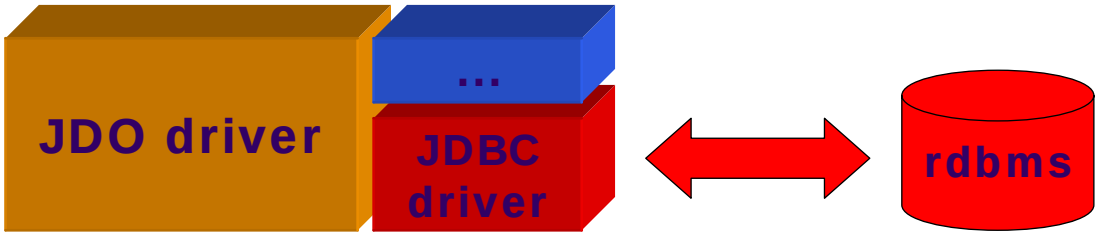
- **Transparent**
  - Any Java class is potentially persistent, with no limit on models
  - Java is the DML, JDO is a non-API
- **Universal**
  - Databases, files, TP monitors, legacy, proprietary
- **Java level**
  - J2ME embedded
  - J2SE client-server model
  - **J2EE** enterprise, EJB

JDO versus JDBC

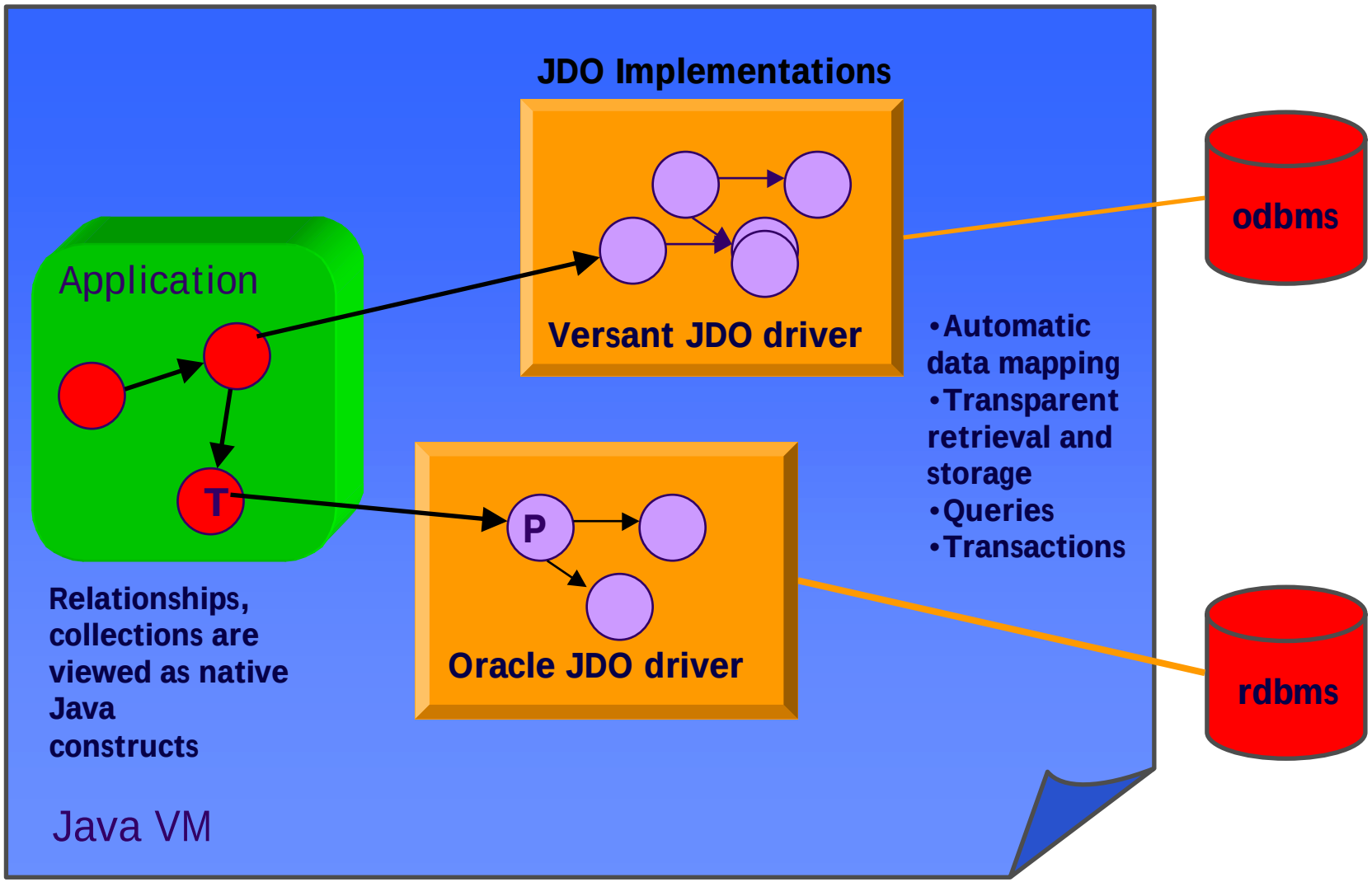
JDBC	JDO
SQL-oriented	Object**-Oriented**
Explicit intrusive code	**Fully** transparent
Manual cache management	Advanced **cache management**
Manual mapping	Automatic **Mapping**
RDBMS centric	Universal

- **Faster than JDBC !**
  - Cache management
  - Efficient mapping for real object models

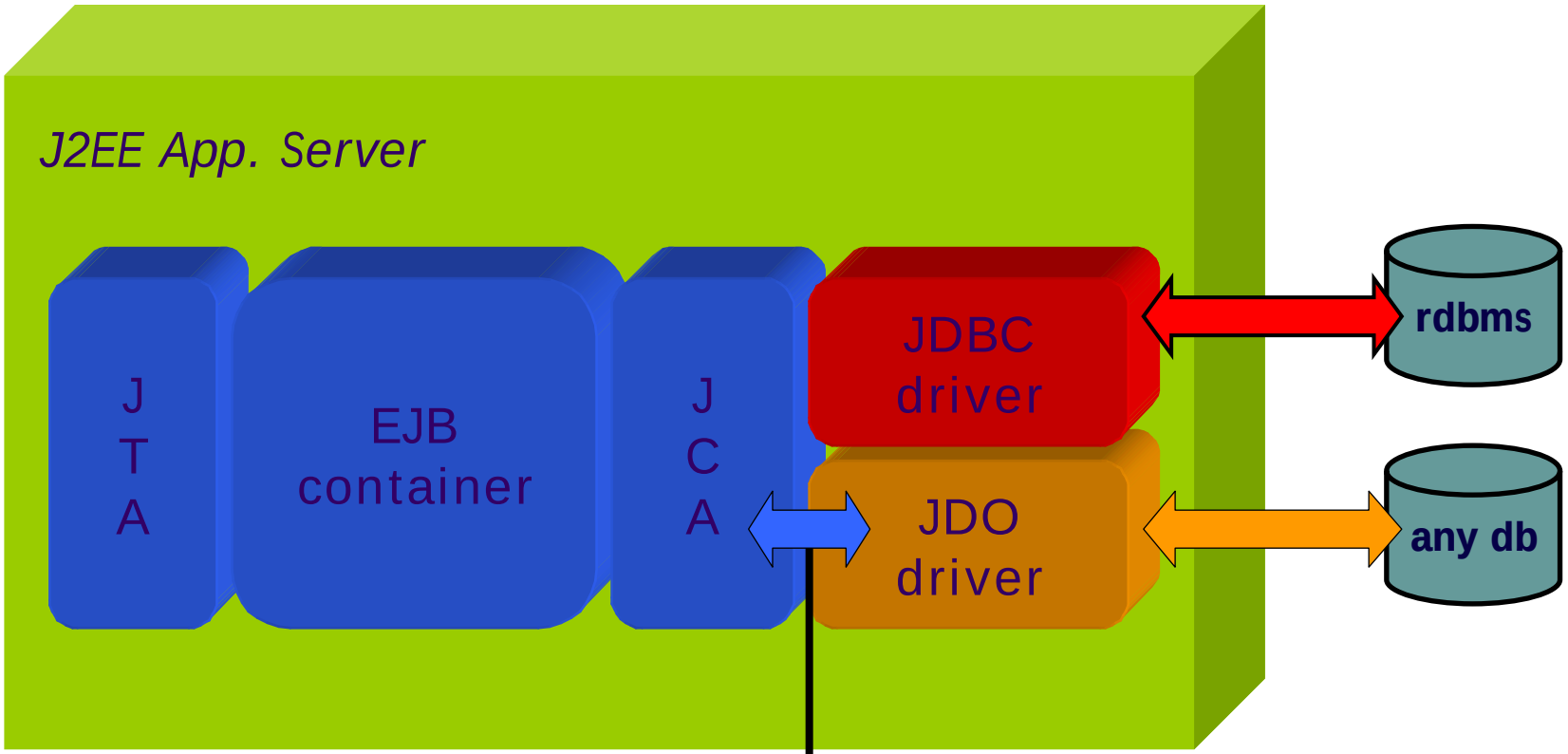
complimentary technologies



2-tier “non-managed” overview



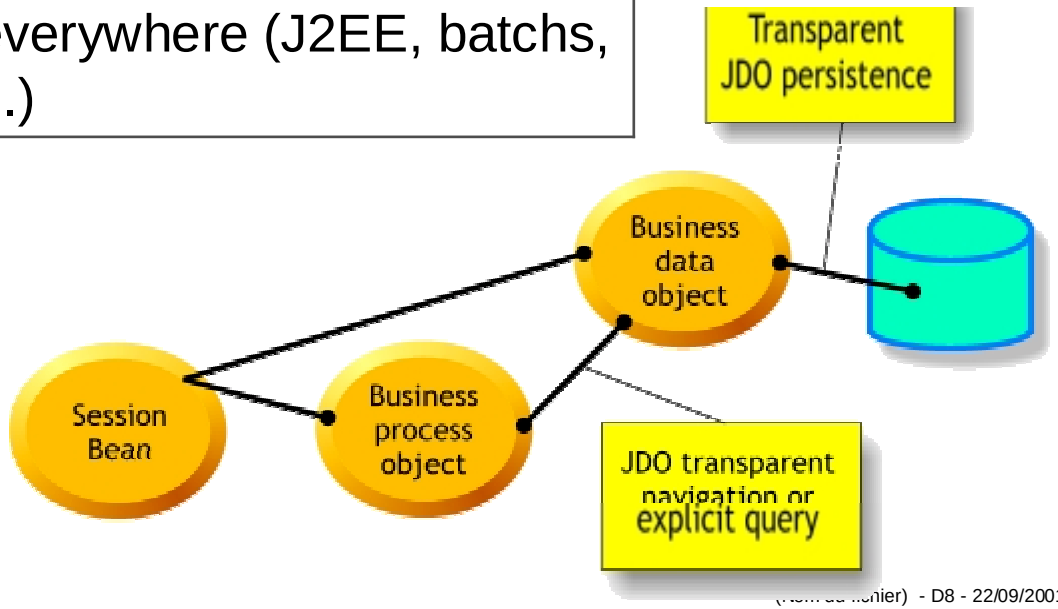
3-tier “managed” JDO overview



JCA contracts :
- Transactions
- Connections
- Security
- Synchronization

JDO versus EJB persistence

EJB persistence	JDO
Intricates distribution and persistence	Separation of concepts
RDBMS centric	Universal
Simplistic models	Complex models
Business objects are J2EE dependent	Business objects can run everywhere (J2EE, batchs, ...)



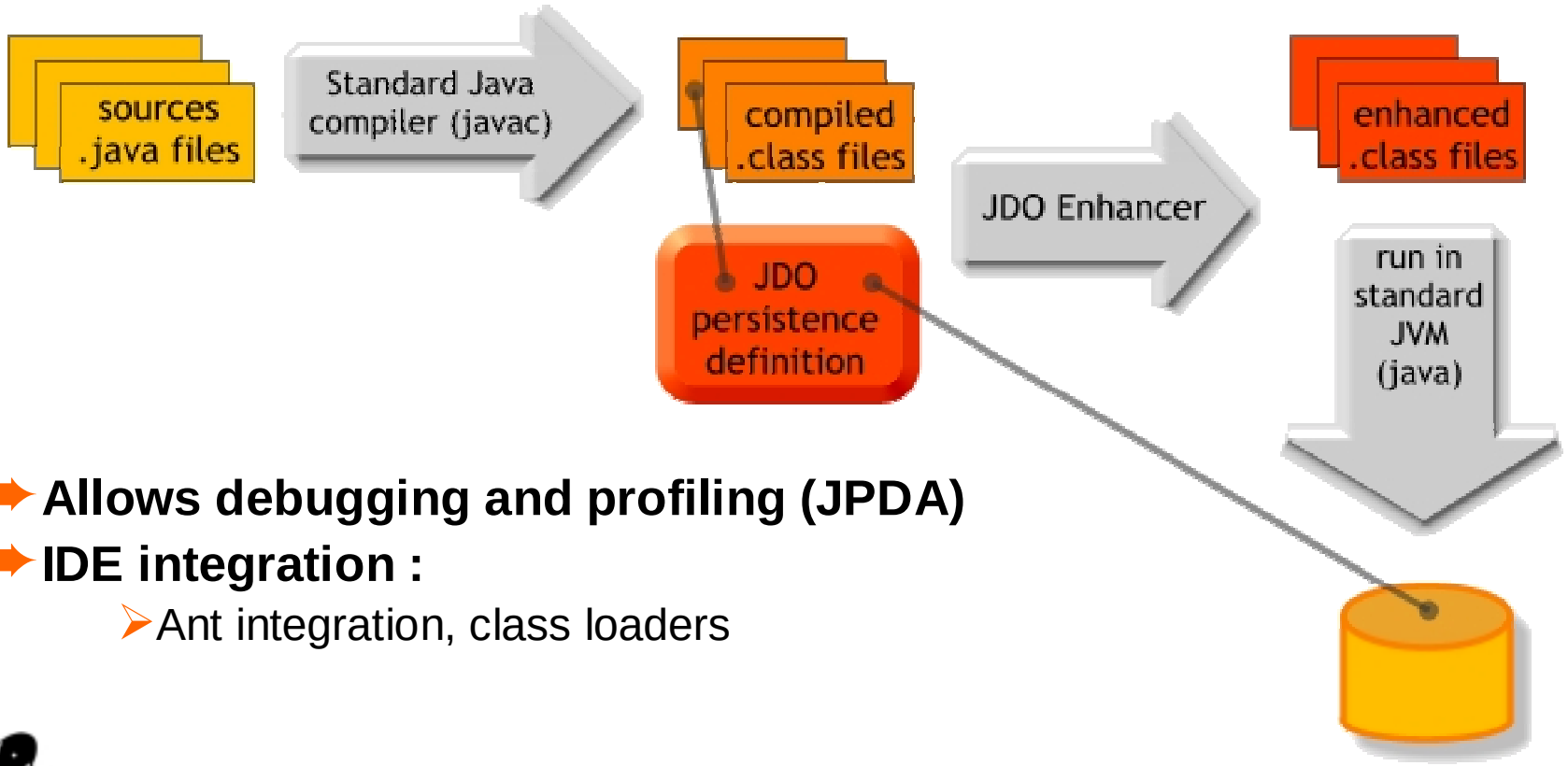
JDO Object model

```
public class Employee extends Person {
  public String name;
  public transient int age;
  private int salary;
  public Company company;
  public IActivity activity;
  Vector awards;
}
```

This is a persistent class !

JDO development cycle

- **Enhancement is fully defined by the specification**
  - Enhanced byte-code must be portable across implementations
- **Allows fully transparent persistence**



- **Allows debugging and profiling (JPDA)**
- **IDE integration :**
  - Ant integration, class loaders

JDO identity

How to uniquely identify an object ?

- Primary key
  - Defined by application, enforced in database
  - Used for access to legacy RDBMS
- Managed by datastore
  - Not tied into any instance values
  - New RDBMS or ODBMS
- Managed by implementation
  - To guarantee uniqueness in the JVM, not datastore
  - For property files, ASCII files, XML files, ...

JDO interfaces

Implicit in J2EE	**PersistenceManager** ➢ Connection to data sources, data caching, object identity management, life-cycle management (StateManager) ➢ PersistenceManagerFactory ● Connection pool, supports JNDI ● Standard properties : – Connection**UserName**, Connection**Password**, Connection**DriverName**, Connection**URL** – Optimistic, … **Transaction** ➢ 1-1 relationship with PersistenceManager : begin, commit, rollback
Explicit	**Query**

JDO queries

- **Neutral, portable, simple, java query language**
  - Optimizations possible for specific query language
    - SQL, OQL, EJBQL, …
  - Allows mixed architectures
    - Client memory / back-end (data-store query engine)
  - Support for large result sets
    - Cursors, “Compiled” queries, …
- JDO QL **filter and return Collections**
  - Input = Collection of candidate instances (class extent or collection attribute)
  - Output = subset of matching instances

Query filters

- **Filter is a Java boolean expression**
  - Identifiers are in the scope of candidate classes
  - Example : find well-compensated employees
    - `Query q =pm.newQuery(Employee.class,“salary>100000”);`
    - `Collection emps = q.execute();`
- **Can navigate through references**
  - Example : find over (fairly ?) compensated employees
    - `filter = “salary > boss.salary”;`

```
class Employee {
  String name;
  float salary;
  Employee boss;
}
```

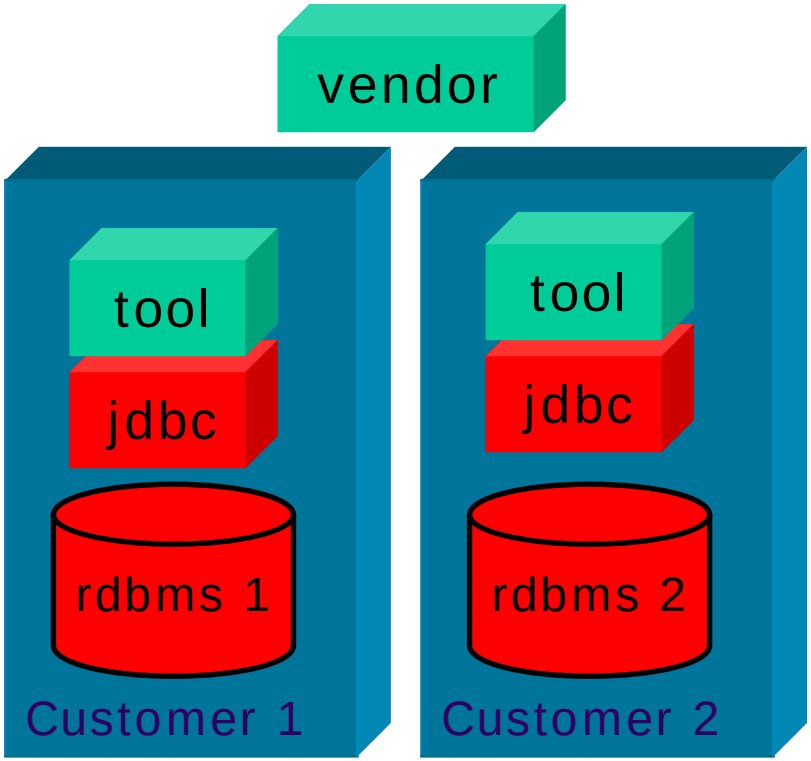
Query : parameters, collections

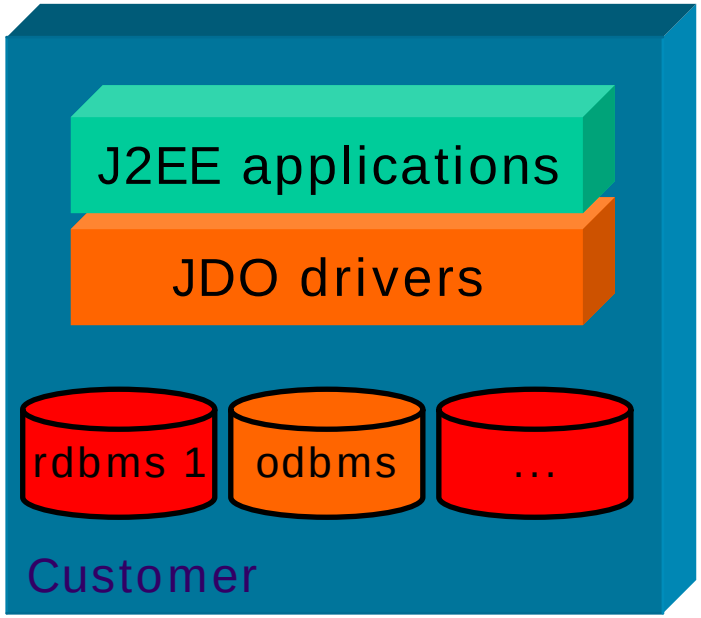
- **Parameters and variables**
  - `query.declareParameters("float sal");`
  - `query.declareVariables("Employee well_comp");`
- **Collection methods supported (contains, …)**
  - Example : find Departments with at least one well-compensated Employee
  - `query.setFilter("emps.contains(well_comp) && well_comp.salary > sal");`
  - `result = query.execute (new Float (150000));`

```
class Department
  Collection emps = new Vector();
}
```

A new vision of persistence

- **Each rdbms vendor must publish a JDBC interface in order to allow standard tools to access data**
- **JDO embraces a wider view of the global IS**



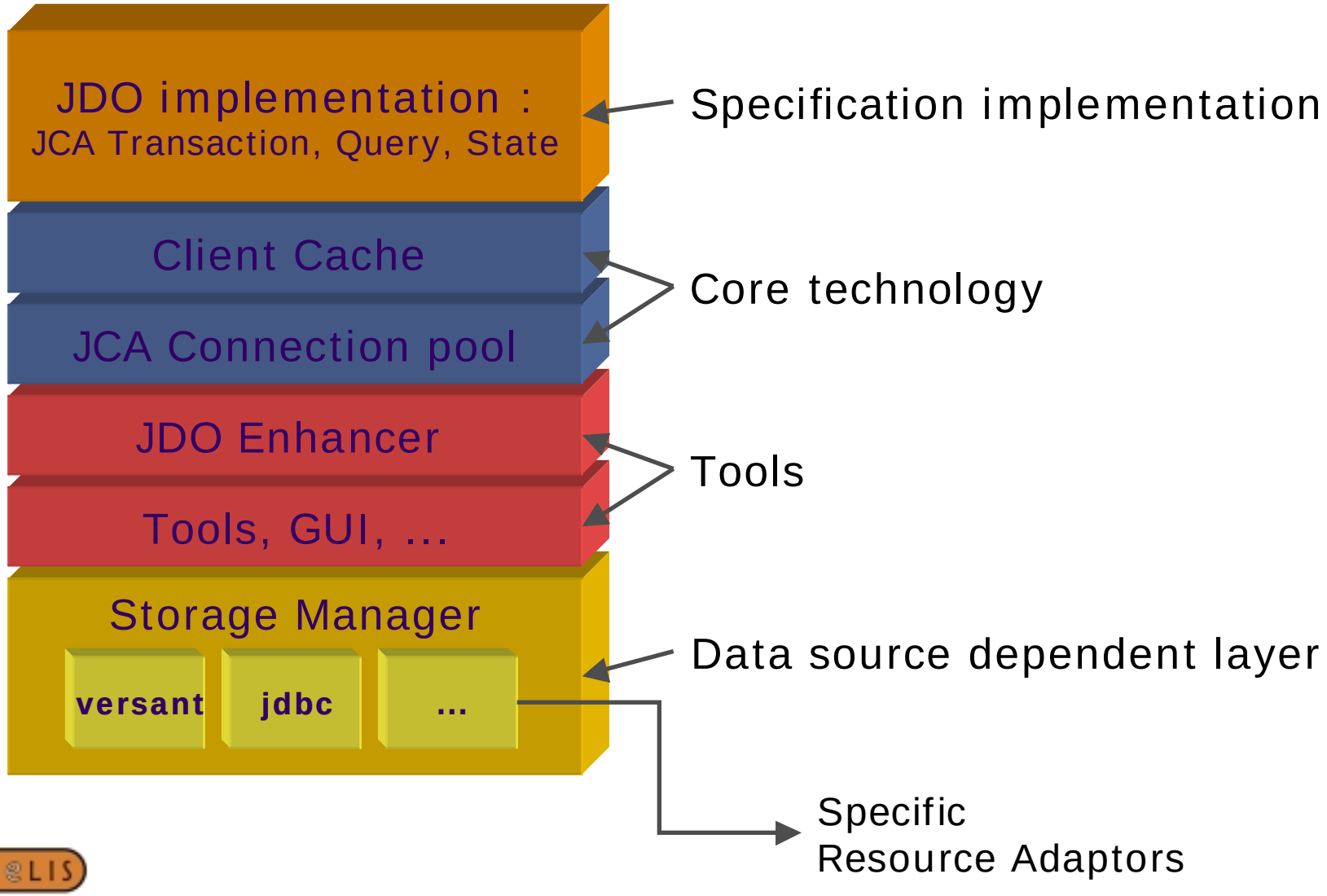


LiDO

LIBeLIS
JDO Drivers

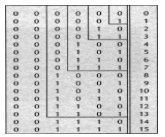
www.objectweb.org

LiDO architecture



Supported data sources

- **Now**
  - Relational databases
  - Object database (Versant)
  - Binary files (Java serialization)



- **Under development**
  - Legacy
    - CICS, Tuxedo, MQ-Series
    - JCA data sources (ERP, applications, mainframes, TP monitors, ...)
  - Files
    - XML files, CSV files
    - Property files, excel files, db3, C-ISAM, ...

LiDO and J2EE

Integration with

- Works with any J2EE compliant App. Server (JCA connection)

LiDO addresses the EJB persistence limitations

- Separation of distribution and persistence
- No more limits on the component model
- Performance & scalability
- Business data independent from J2EE

LiDO relational

- **JDO compliant O/R mapping tool**
  - Automatic Mapping
    - When you have a new Java model you want to automatically persist
    - Supports multiple mapping strategies
  - Explicit mapping
    - When you have existing legacy databases
- **Lido Mapper**
  - RDBMS schema capture/creation
  - GUI papping tool

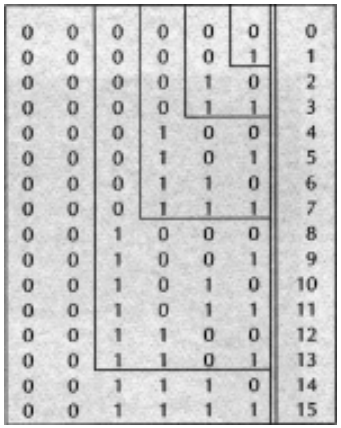
LiDO Versant

- **Based on Versant C layer**
- **Specifics**
  - Supports Versant Query Language (VQL)
  - Schema manager
- **Benefits**
  - Faster than transparent JVI
    - Better collection support
  - Standard API
    - JCA compliance
  - Flexible mapping

LiDO binary file

A light JDO implementation

- Allows to test JDO without any DBMS engine
- Relies on Java serialization
- Might be useful for unstructured data, documents, ...



In memory DB

- All objects are ridden at open and rollback
- All objects are written at close and commit
- Metadata are stored
- Tool to dump the file content

LiDO roadmap

- **New data sources**
- **JDO extensions**
  - Support for database constraints (not null, unique, ...)
  - Referential integrity (on delete, reverse link, ...)
  - Explicit locking
  - Cache management

LiDO universal

- **Access to multiple heterogeneous data sources from the same application**
  - Navigation between objects stored in different data sources supported
  - Polymorphic object mapping