**THE O₂ OBJECT DBMS**

**O₂ Technology, Inc**
3600 West Bayshore Road - Suite 106
Palo Alto, CA 94303
USA

**O₂ Technology Ltd**
North Heath Lane - Horsham
West Sussex RH12 5UX
Great Britain

**O₂ Technology**
7 rue du Parc de Clagny
78035 Versailles Cedex
France

---

**ODBMS History**

- **1984**
  *Birth of the ODBMS concept*
  (D. Maier, G. Copeland)

- **1989**
  *ODBMS definition: "The Manifesto"*
  (M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, S. Zdonik)

- **1993**
  *ODBMS standard: ODMG 93*
  (T. Atwood, J. Duhl, G. Ferran, M. Loomis, D. Wade)
ODBMS Definition

The ODBMS Manifesto, December 1989

- Persistence
- Disk management
- Data sharing
- Data reliability
- Data security
- Query language

- Complex object
- Inheritance
- Overloading
- Object identity
- Encapsulation
- Extensibility

ODBMS Definition:

- DBMS
- OO System

ODMG 93: The ODBMS Standard

- Founded in 1991 by Rick Cattel and five ODBMS vendors
- Guarantee the portability of applications written for object databases across most of object database systems in use today.
- Standardized interfaces:
  - Object data definition language ODL
  - Object Query language OQL
  - C++ and Smalltalk bindings
- Adopted by OMG (IDL, CORBA), currently studied by ANSI SQL3
The O₂ Product Line: A Complete Object DBMS

- Scalable database engine
- Complete set of development tools
- Openness

The O₂ Product Line

- O₂ development tools
- O₂ database engine
- Connectivity & Third Party Tools
The O₂ Product Line

- O₂ set of tools
- O₂C
- C++
- Smalltalk
- O₂Store
- O₂Engine
- O₂DBAccess
- O₂Web
- O₂Report
- OQL
- Develop. tools
- Develop. tools

© O₂ Technology - May 1996
The O₂ System: Benefits

- Supports all types of data (pictures, sound, graphics, video, programs, documents, etc.)
- Improve the software production cycle (analysis, design, coding, maintenance and evolution)
- Produces better quality applications
  - performance
  - maintainability
  - user-friendly

A Powerful Database Engine

O₂ development tools

O₂ database engine

Connectivity & Third Party Tools
Database Engine: Benefits

• Improve system performance
  – increased performance for complex data manipulation

• Accommodates all sorts of data
  – text, pictures, images, sounds, etc.

• Accommodates all sorts of behavior
  – design environments, programming environments

O₂ Database Engine: Strong Points

• High performance
• Large databases
• Large number of users
• Language independence
• Adapted to multiple platforms
O₂Engine Presentation

1) Schema management
   Object data model, ODMG compliance

2) Client/server architecture

3) Database features
   Transactional mechanisms, DB administration, tuning

1) O₂Engine: Schema Management

- Object data model fully ODMG compliant
  - Complex object management
  - Simple and multiple inheritance, late binding
- Methods associated to object
- Multiple schemas, multiple bases
- Version and configuration management
- Schema and data evolution
O₂Engine: Object Data Model

- Object model
  - class, method, simple and multiple inheritance, late binding
  - atomic types
- Relationships
  - 1-1 relationships
  - 1-n, n-m relationships
- Collection
  - Set, bag, varray, list
  - Operations (scan, assignment, union, intersection, difference, etc)

Complex Object Example

```
struct
  "Mercedes"
  country
  sales

set
  cars
  Car

list
  struct
  car
  sales
  Car 94,500
```
**O₂Engine:**
Multiple Databases, Multiple Schemas

- import schema S1
- import base

- Base B1
- Base B2
- Base B3

**O₂Version:** O₂ Version and Configuration Management System

- Any class can have versionable and not versionable objects
- Any object can be versioned
- Any versionable object can be referenced to or from non versioned object
- Integrated in the kernel of O₂Engine
  - virtual copy, concurrency, index
- Customizable for specific needs
**O₂Version**

- Version unit: collection (configuration) of O₂ objects
  - complex object, dynamic construction
- Operations: methods of the class `o2_version`
  - creation, deletion, modification, browsing, merge, naming
- Management of the version derivation graph

```
configuration
  v0 --v1 --v1.1 --v1.1.1
  |         |         |
  v1 --v1.1 --v1.2
  |         |         |
  v2 --v2.1
```

**Schema Evolution**

- Schema modification (creation, modification, deletion) on attributes, types, inheritance
- Can be done at compile time or at run time
- Automatic propagation on existing data at access time (lazy update)
- Simple transformations are automatic
- Complex transformations can be programmed
O₂ Engine: Persistence

Supports 2 models of persistence:

- Direct persistence (ODMG C++ model):
  - status declared at creation time
- Persistence by reachability (ODMG Smalltalk model):
  - names are entry points to the database; everything reachable from the names is persistent.
  - dynamic: persistence status can change at run time
  - solves the referential integrity problem
  - automatic garbage collection is supported

2) O₂ Engine: Client/Server Support

[Diagram showing client/server architecture with PC, O₂ Server, Unix Workstation, O₂ database connections]
**O₂Engine: Architecture**

- **Client Process**
  - Cache
  - schema, object, queries, locks and local logs
- **Server Process**
  - Cache
  - I/O Manager
  - global locks, disk, transaction

**O₂Engine: Detailed Architecture**

- O₂Engine API
- OQL
- Schema Manager
- Object Manager
  - index, clusters
- Page buffer

- Transaction Manager
- Lock Manager
- Log Manager
- Page buffer
- DB
- log
Inside O₂Engine

- Schema manager (SM):
  - creation, selection, update and deletion of classes, types, methods, functions, applications and names
  - multiple databases and schema management,
- Object manager (OM):
  - object manipulation with identity, message passing
  - management and manipulation of complex values
  - persistence model, garbage collection
  - index and cluster

Inside O₂Engine: Physical Representation

tuple 32 'a' PID1
set, bag, list
  - small col_info 2, 3, 5, 14, 23
  - large col_info
byte string
  - small abcdefghijklmno...
  - large

B-tree

Large Data Item
3) O₂Engine: Database Administration

- Cold and warm recovery based on a log mechanism
- Garbage collector
- On line back up
- Incremental database replication
- Transaction monitoring
- Utilities to save/restore schema and bases, copy databases, restructure database

O₂Engine: Tuning

Logical/physical independence
- Cache management on both client and server sides
- Statistics (disk access, network access, swapping, number of objects, size of collections, etc.)
- Dynamic control on locking and logging
- Query optimizer
- Index
- Cluster
O₂Engine: Index

- Index on named collections of tuples
- Collections can be indexed on:
  - atomic attribute
  - attribute referencing an object
  - collection attribute
  - via a composition path
- Extensible (do it yourself) index

Index: Example

Index on an atomic attribute:
create index TheCarManufacturers on name

Index on an attribute referencing an object:
create index TheCarManufacturers on Country

Index on a composition path:
create index TheCarManufacturers on Country.name
**O₂Engine: Cluster**

- Goal: Manage the location of data on disk
- Clustering strategy based on the notion of placement tree (subtree of the composition graph, defined by the DBA)
- Many clustering strategy can be defined on objects; level of priority among them
- Sorted clusters: use index to organize the physical location of a set of objects
- Default clustering strategy based on the way objects are made persistent

Cluster: Example

```sql
create cluster CarManufacturer on (country(name))
```
O₂Engine: Transactions

- Classical transactions (ACID), roll back
- Read only transactions
- Long transactions
- Multiprocess distributed transactions
- 2 phase commit
- XA support
- Page and object locking, implicit or explicit
- Call back locking

O₂Engine API

- C or C++ functions library
- Direct access to the system: schema manipulation, complex object manipulation, index and clusters, transactions, ...
- High performance
- Useful for software integrators and VARs
O₂ Engine API Example

```c
#include <o2.h>
main (argc, argv) {
    Handle hd_cl_Person, hd_Smith, hd_name;
    /* Connection with the server */
    o2_init(argc, argv, &sinit);
    o2_set_base("my_base");
    hd_cl_Person = o2_name_to_id("Person", O2_CLASS);
    /* new of the object Smith */
    o2_create(&hd_Smith, hd_cl_Person);
    /* Access and update of the attribute name */
    o2 = name_to_pid(hd_cl_Person, "name", &pid, &kind);
    int off_set = o2_solve_attribute(hd_Smith, pid);
    o2_tuple_put(hd_Smith, off_set, O2_STRING, "Smith", V_FLAG);
    /* Assignment of the persistent root Smith with the initialized object */
    hd_name = o2_get_root("Smith");
    o2_tuple_put(hd_name, O2_ROOT_OFFSET, O2_COBJ, &hd_Smith, V_FLAG);
    o2_end(); }
```

O₂ Engine: Performance

- O₂ is a scalable system
  - time to add new data to an existing O₂ DB is constant
  - time to scan data is linear with respect of the volume of data to be read
  - time to retrieve a specific object from a DB is constant, irrespective of the size of the O₂ DB

![Graphs showing the performance of O₂ Engine](image)
O₂Engine: Performance

- Two official benchmarks:
  - OO1 benchmark (R. Cattell, Sun)
  - OO7 benchmark (D. DeWitt, M. Carey and J. Naughton, University of Wisconsin)
- Representative of complex data applications and based on a CAD application
- Accepted by both vendors and users

O₂Engine: OO1 Benchmark

<table>
<thead>
<tr>
<th>Operation</th>
<th>RDBMS</th>
<th>O₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC</td>
<td>94</td>
<td>8.14</td>
</tr>
<tr>
<td>TW</td>
<td>84</td>
<td>0.07</td>
</tr>
<tr>
<td>LUC</td>
<td>29</td>
<td>13.01</td>
</tr>
<tr>
<td>LUW</td>
<td>19</td>
<td>0.02</td>
</tr>
<tr>
<td>RTC</td>
<td>95</td>
<td>6.23</td>
</tr>
<tr>
<td>RTW</td>
<td>-</td>
<td>0.10</td>
</tr>
<tr>
<td>IC</td>
<td>20</td>
<td>3.88</td>
</tr>
<tr>
<td>IW</td>
<td>20</td>
<td>1.06</td>
</tr>
</tbody>
</table>
O₂Engine: OO7 Benchmark

<table>
<thead>
<tr>
<th></th>
<th>O₂</th>
<th>ODBMS 1</th>
<th>ODBMS 2</th>
<th>ODBMS 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB size</td>
<td>80,9</td>
<td>122,3</td>
<td>74,9</td>
<td>55,4</td>
</tr>
<tr>
<td>TA</td>
<td>689,7</td>
<td>2516,4</td>
<td>1269,4</td>
<td>808,9</td>
</tr>
<tr>
<td>TB</td>
<td>950,0</td>
<td>2101,2</td>
<td>1468,8</td>
<td>961,7</td>
</tr>
<tr>
<td>TC</td>
<td>1133,0</td>
<td>6467,3</td>
<td>1389,7</td>
<td>1040,0</td>
</tr>
<tr>
<td>TD</td>
<td>0,2</td>
<td>4,8</td>
<td>11,0</td>
<td>1,1</td>
</tr>
<tr>
<td>QA</td>
<td>1,5</td>
<td>9,7</td>
<td>9,6</td>
<td>6,6</td>
</tr>
<tr>
<td>QB</td>
<td>19,7</td>
<td>39,5</td>
<td>37,1</td>
<td>60,5</td>
</tr>
<tr>
<td>QC</td>
<td>12,7</td>
<td>22,7</td>
<td>36,2</td>
<td>64,4</td>
</tr>
<tr>
<td>QD</td>
<td>134,6</td>
<td>52,6</td>
<td>136,3</td>
<td>90,4</td>
</tr>
<tr>
<td>Insert</td>
<td>58,8</td>
<td>126,1</td>
<td>64,9</td>
<td>72,5</td>
</tr>
<tr>
<td>Delete</td>
<td>75,6</td>
<td>130,0</td>
<td>44,7</td>
<td>91,2</td>
</tr>
</tbody>
</table>

O₂Engine Performance:
Case Study

- Storage and retrieval of notes in documents
- 10 000 000 notes, 400 000 000 objects, 50 users, 30 Gigabytes
- Operations:
  - load, query (on key words, id, author), navigation, insertion, deletion
- Conclusion: O₂ is a scalable system
  - time to load extra notices is constant and independent of the volume already loaded
  - in the multi-user context, O₂ reacts gracefully to the increase in the number of simultaneous connections.
Performance: Case study (cont)

• Lookup one document objects based on its identifier:

<table>
<thead>
<tr>
<th></th>
<th>O₂ (5 Giga)</th>
<th>Competitor (5 Giga)</th>
<th>O₂ (25 Giga)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 clients</td>
<td>0.057 s</td>
<td>0.080 s</td>
<td>0.080 s</td>
</tr>
<tr>
<td>10 clients</td>
<td>0.337 s</td>
<td>0.456 s</td>
<td>0.484 s</td>
</tr>
<tr>
<td>25 clients</td>
<td>0.830 s</td>
<td>2.200 s</td>
<td>1.180 s</td>
</tr>
<tr>
<td>50 clients</td>
<td>1.740 s</td>
<td>* (failed)</td>
<td>2.440 s</td>
</tr>
</tbody>
</table>

• Lookup all document objects belonging to a time interval:

<table>
<thead>
<tr>
<th></th>
<th>O₂ (5 Giga)</th>
<th>Competitor (5 Giga)</th>
<th>O₂ (25 Giga)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 clients</td>
<td>23 s</td>
<td>300 s</td>
<td>24 s</td>
</tr>
</tbody>
</table>

• Traversal:

<table>
<thead>
<tr>
<th></th>
<th>O₂ (5 Giga)</th>
<th>Competitor (5 Giga)</th>
<th>O₂ (25 Giga)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 clients</td>
<td>1 s</td>
<td>1 s</td>
<td>0.9 s</td>
</tr>
<tr>
<td>25 clients</td>
<td>4 s</td>
<td>20s</td>
<td>4 s</td>
</tr>
</tbody>
</table>

O₂Store: O₂ Transactional Structured Record File System

- O₂ development tools
- O₂Engine
- O₂Store
- Connectivity & Third Party Tools

© O₂ Technology - May 1996
O₂Store: What’s in there

- Persistent data management
- Data model neutral
- Structured, variable length records
- Indices: B-Trees, positional B-Trees, your own
- Concurrency (page and record)
- Multi-volume
- Recovery
- Client/server: page mode
- Distribution: XA support
- Administration tools
- Small footprint: .5 mb for server, 1 mb per client.

O₂Store API

- Transaction and session commands
- File manipulation primitives
- Record manipulation primitives
- Index manipulation primitives
- Administration commands
O₂Store: What’s not in there

- No objects, just records
- No transient data, just persistent data
- No schema or schema manager, left to the user
- No automatic index maintenance, left to the user
- No development tools, just an API (set of C functions)

O₂Store: Detailed Architecture
O₂Store: Who needs it?

Any tool builder who needs a persistence engine:

• with high performance, small footprint
• with transactional capabilities (more than a file system) and recovery
• without a predefined data model (less than an object database system)

  e.g. developers of DBMS, knowledge based systems, dedicated applications, etc..

A Complete Set of Tools

O₂ development tools
O₂ database engine
Connectivity & Third Party Tools
Set of O₂ tools: Benefits

- **Improved productivity:**
  - fast GUI development thanks to O₂Look
  - comfortable development thanks to O₂Tools
  - fast application development thanks to O₂C and OQL
  - reusable components thanks to O₂Kit

- **Improved applications:**
  - user friendly through O₂Look

- **Standard applications thanks to ODMG compliant C++, Smalltalk and OQL**

---

**O₂ Development Tools**

- Lower Case
- GUI
- Languages
O₂ Development Tools

- C++, SmallTalk, C, Eiffel, and Objective C interfaces
- O₂ C: object 4GL
- OQL: object query language
- O₂ Look: GUI generator for objects
- O₂ Graph: graph editor for objects
- O₂ Report: report generator
- O₂ Kit: library of reusable components
- O₂ Tools: programming environment
The Same Object From Disk to Screen

Traditional          O₂

Disque

begin
  x = y
  y = z
  z = z + 1
end

GS QL: Entitly Relationship Viewer

Title
mdv::1

Locations
Supply_Lines
Distributors
Retailers

Town
Latitude
Longitude
Population
Size

GS QL: Graphics Model Viewer

Programming Languages:
C++, SmallTalk, C, O₂C, Eiffel, ObjectiveC

- Developing a C++, SmallTalk, C, Eiffel or ObjectiveC application on top of O₂
- Adding persistence to a C++, SmallTalk, C, Eiffel or ObjectiveC program
- Using the object 4GL O₂C: database management, programming language, GUI management
- Mixing languages to develop O₂ applications
**C++ Interface**

- ODMG standard
- C++ 3.0 data model
- Add persistence to existing C++ applications
- Program a C++ application which will benefit from all O₂ Engine functionalities (persistence, strong typing, collections, relationship, ...)

**C++ Programming**

- C++ class description
- C++ appli. files
- O₂ lib
- o2import

- C++ files
- C++ - O₂ application
C++ Programming: Example

main () {
    d_List < d_Ref < Car > > garage("TheGarage");
    Car *mercedes550;
    d_Transaction *trans;   ....
    database->open("o2db");
    for (i=0; i < garage.cardinality(); i++) {
        garage[i]->display(); }
    trans->begin();
    garage->insert_element_last (mercedes550);
    trans->commit();
}

C++ Programming: Connection with O₂ tools

- Query the O₂ database with OQL
- Display C++ objects with O₂Look
- Extend a C++ application with O₂C (classes, methods)
- Display the persistent C++ classes hierarchy and browse the structures with O₂Tools
O₂C Features

• Creation of O₂ objects, messages sending, complex data manipulation
• All the functionality of C
• Integration of O₂Look and OQL
• Incremental compiler and dynamic loader
• Methods stored in the database

O₂C: Example

class CarManufacturer
   type tuple (read name: string,
                country: Country,
                cars: set (Car),
                sales: list (tuple (car: Car,
                                   sales: integer )))

   method public turnover: real,
             private add_sales(car: Car,
                                quantity: integer)

   end
O₂C: Example

```cpp
method body  turnover : real  in class  CarManufacturer
{
  float result = 0.;
  o2 tuple ( car: Car, sales: integer ) sale;

  for ( sale in self->sales)
    result += sale.car->price * sale.sales;

  return result;
}
```

C++, C, Eiffel, ObjectiveC or Lisp Programming: Benefits

- Standard languages
- Standard ODMG binding (for C++)
- Objects are language independant
- Standard OQL for C++ and Eiffel objects
- Automatic display for any C++ and Eiffel object
- Add persistence to C or C++ existing applications at a lower cost
O₂C Programming: Benefits

- Simplicity of the language (direct collection support, no memory management, ...)
- Transparent access to the database
- Dynamic applications (meta-schema)
- Programmer’s productivity

OQL Query Language

- O₂Tools
- O₂Kit
- O₂Look
- O₂Graph
- O₂C
- C++
- Smalltalk
- Develop. tools
- Develop. tools
- OQL
OQL Features

- Object-oriented SQL
- Method activation
- Complex object manipulation
- ODMG compliant
- Query optimizer
- Two modes:
  - interactive
  - embedded in C, C++, C, Eiffel, ObjectiveC, Lisp and O₂ Engine API

Interactive OQL

```sql
select struct ( auto: car.title,
  mileage: car.mileage)
from car in Mercedes.cars
where car.mileage in [50 .. 300] and
  exist c in car.characteristics.transmission:
  c.name == "Automatic"
```
OQL Embedded in C++

```cpp
{  
  d_List < d_Ref < Car > > garage;
  char *val = "Automatic";
  d_OQL_Query q1("select car from car in Mercedes.cars
      where car.mileage in [50 .. 300] and
      exist c in car.characteristics.transmission: c.name == $1");
  q1 << val;
  d_oql_execute (q1, garage);
  for (i=0; i<garage.cardinality(); i++)
    garage[i]->display();
}
```

OQL Embedded in O₂C

```o2
run body {
  o2 list(Car) garage;
  o2 string val = "Automatic";
  oql(garage, "select car from car in Mercedes.cars
      where car.mileage in [50 .. 300] and
      exist c in car.characteristics.transmission: c.name == $1", val);
  for (i in garage)
    garage[i]->display;
}
```
OQL Benefits

- Simple and powerful
- Fully ODMG compliant
- Performance (optimizer)
- Active objects on the screen
- Productivity

Object Presentation Tools

- O2Tools
- O2Kit
- O2Look
- O2Graph
- Develop. tools
- Develop. tools
- O2C
- C++
- Smalltalk
- OQL
The O₂Look GUI builder

- Generator of X11, Motif graphical presentations;
  Automatic mapping of objects of the database into Motif widgets.

- Open to other X11/Motif tools:
  - GUI builders (TeleUSE, UIMX, XFM, etc)
  - 2D/3D tools (PV-Wave, DataViews, IlogViews, SL/GMS, Phigs, etc)

O₂Look Features

- Display any complex structure without programming and without drawing
- Customization
- Edit, cut, copy, paste
- Trigger methods by menu
- Open to any Motif or X tool
O₂Graph Features

- Graph creation, manipulation and display
- Nodes and links are objects of the database
- Various placement mode (coordinate, grid, tree, graph, DAG.)
- Customization (resources)
- Triggers (dialog manager)

O₂Look, O₂Graph: Benefits

- Quick development of nice GUIs
- Tool for non experts
- Transparent access to the database
- Interactive manipulation of data base objects
- Low development cost
- Can be used in conjunction with other GUI tools
Programming Tools

O₂Tools    O₂Kit
O₂Look    O₂Graph

Develop. tools    Develop. tools

O₂C    C++    Smalltalk

OQL

O₂Tools Features

A set of integrated tools:

- Graphical browser/editor in the schema and in the database
- Object symbolic debugger
- Source management tool
- Library of reusable components (O₂Kit)
- OQL language interface
An Open System

- Development tools
  - Programming languages: C, C++, Eiffel, Lisp, ObjectiveC
  - GUI tools: TeleUSE, SL/GMS, UIMX, ILOG Views
  - Lower CASE tools: SparcWorks, Softbench, ObjectCenter
  - Upper CASE tools: Softeam/Objecteering, Verilog/LOV, Rational/Rose, Concis/Tramis

- Existing Environments:
  - Report generator: O₂Report
  - World Wide Web: O₂Web
  - SQL interface to RDBMS(Oracle, Sybase, DB2, ...): O₂DBAccess
Open to Third Party GUI tools

- The O₂ database engine can be used in conjunction with any GUI tool.
- In the X/Motif environment, the O₂Look generator can be used in conjunction with any X/Motif tool:
  - every O₂Look presentation is a Motif widget (and as such can be incorporated in any Motif tool)
  - every Motif widget can be inserted in any O₂Look presentation (e.g. video display)

O₂Report Features

- Report generator based on objects or collections of objects of O₂ databases
- Available for any O₂ object
- Latex, RTF and HTML generation
- Dedicated to application developers
Open to Existing Environments

- **SQL: O₂DBAccess**
  - ability to interact with existing RDBMS
- **ISO/DTP compliant (XA interface)**
  - through the XA standard compliance, the ability to support distributed and interoperable applications with other traditional data servers
- **OMG/CORBA compliant**
  - ability to access an O₂ database from any CORBA application
- **Web interface: O₂Web**
  - ability to access an O₂ database from any platform connected to a LAN or WAN
**O₂DBAccess**

- Access to existing RDBMS (query, update)

![Diagram](image)

- Legacy application on a remote site
- Specify the remote site hardware
- Specify the remote site software
- Specify the network
- O₂DBAccess provides a means to
  - establish a connection
  - start a session (begin/commit/abort transaction)
  - send an SQL query
  - map the result onto O₂ objects
Interoperability and Distribution

- Allows the development of applications distributed over several object servers and data servers.
- O₂ supports 2-phase commit
- O₂ supports the XA standard
- XA is part of DTP
- Allows a transactional monitor (e.g. Encina or Tuxedo) to operate on top of several servers

Interoperable application

Transactional Monitor

XA interface

XA
O₂
Server

XA
Relational
Server

XA
Relational
Server
O₂Web : Access to the Web

O₂Web features

- Access any information stored in an O₂ database (text, images, sound, video)
- Support all OQL facilities: navigation through objects, predicates, methods execution
- Automatic HTML generation for any O₂ objects
- Coexistence of Web clients and traditional database clients, distributed applications

URL : http://host/cgi-bin/base name?OQL query
Platforms

- Unix:
  - SUN: SUN OS and Solaris
  - HP 9000: HP UX
  - IBM RS 6000: AIX
  - Bull DPX 20: BOSX
  - DEC Alpha: OSF1
  - SGI: IRIX
  - SNI: RM (SINIX) and RW (IRIX)

- PC:
  - NEXTSTEP, Solaris, SCO
  - Windows NT, Windows 95

Conclusion

O₂, an object database system:
- ODMG compliant
- Scalable database engine
- Tool completeness

O₂, a system adapted to the enterprise network