INTRODUCTION TO OBJECT-ORIENTED DATABASES

Sources include:

- Chaudhri, A.B. (1993) **Object Database Management Systems: An Overview** in "*BCS OOPS Newsletter*", No.18 Summer '93
- Chaudhri, A.B. & Osmon, P. (1996) **Databases for a New Generation** in *"Object Expert"* March/April '97 pp 33-38
- Graham, I. (1994) **Object Oriented Methods**. Published by: Addison-Wesley, ISBN 0-201-59371-8
- Graham, I. (1995) **Migrating to Object Technology**. Published by: Addison-Wesley, ISBN 0-201-59389-0
- Vadarparty, K. (1996) **Developing an ODBMS Application: Basic Steps** in *"Journal of Object Oriented Programming"* January '96 pp 19-21

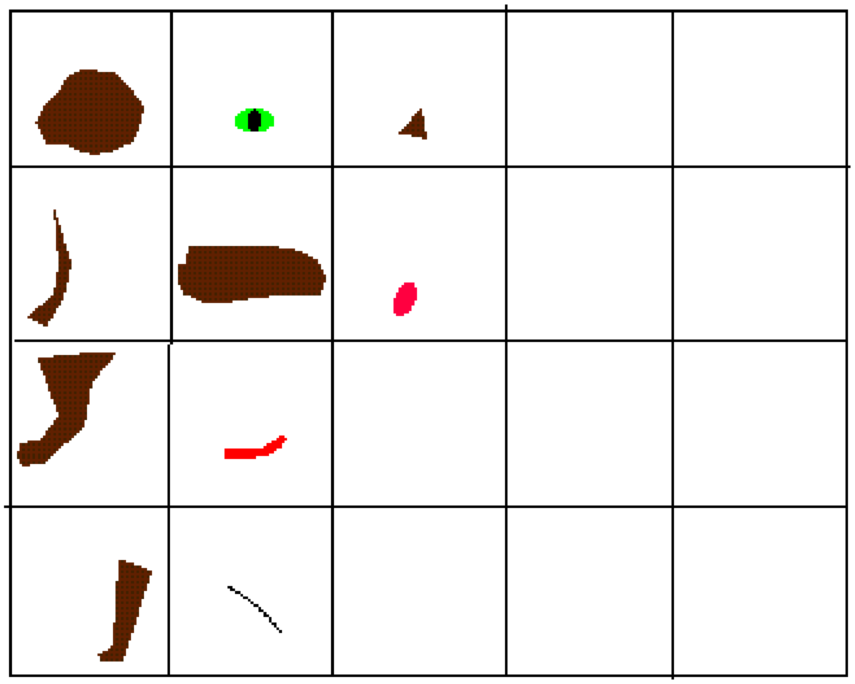
What is an Object Oriented Database?

- Back to the Beginning

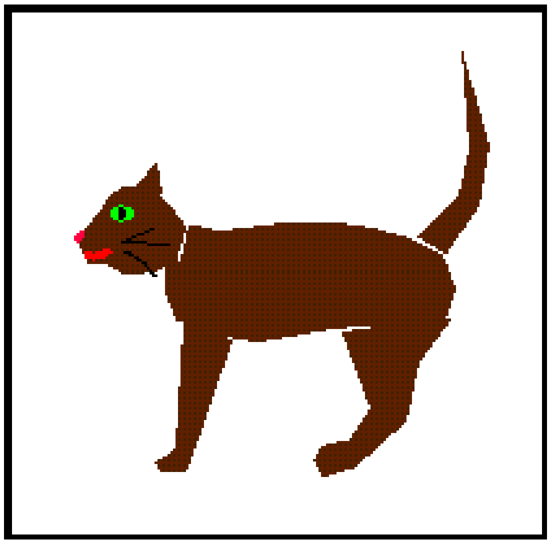
Object Oriented Databases (ODBMS) store ***data*** together with the appropriate ***methods*** for accessing it i.e. ***encapsulation.***

Relational databases " *hammer the world flat*" by normalisation.

Relational database of a cat:

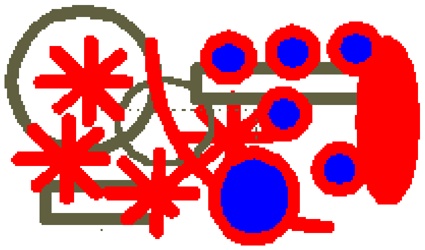


Object-oriented database of a cat:

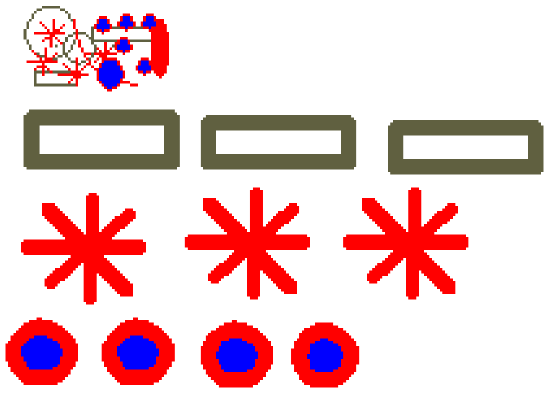


This enables:

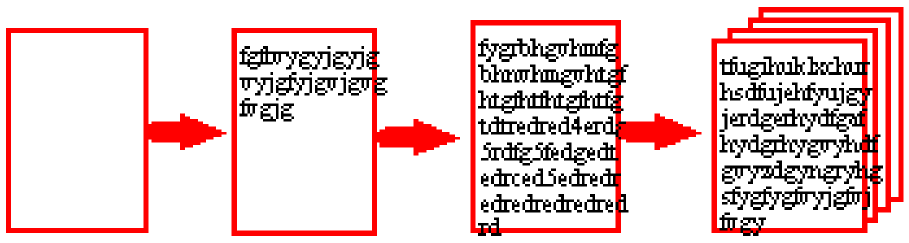
- complex data types to be stored *(e.g. CAD applications)*



- a wide range of data types in the same database *(e.g. multimedia applications)*



- easier to follow objects through time *(e.g. "evolutionary applications")*



Applications

- Back to the Beginning

The first areas where ODBMS were widely used were:

- CASE
- CAD
- CAM

Increasingly now used in:

- telecommunications
- healthcare
- finance
- multimedia
- text/document/quality management

These are the '*next-generation applications*" where traditional IT methods have often not impacted. *For example, it has been estimated that up to 90% of the data held in businesses is still paper-based, because a large amount of data is not 'record oriented'.*

History of Databases

- Back to the Beginning

file systems (1950s)	• store data after process created it has ceased to exist
hierarchical/ network (1960s)	• concurrency • recovery • fast access • complex structures
relational (1970-80s)	• more reliability • less redundancy • more flexibility • multiple views
ODBMS (1990s)	• better simulation • more (and complex) data types • more relationships (e.g. aggregation, specialisation) • single language for database AND programming • better versioning • no 'reconstruction' of objects • other OO advantages (reuse, inheritance etc.)

How do ODBMS Work?

- Relational Model
- Object Model
- Back to the Beginning

Example:

Relational Model of Student-Course Relationship

STUDENT

Student#	StudentName	Address
1	jane jones	6 The High Street
2	brian brown	104 Park Avenue
3	clara clarke	97 Gilmore Street
4	sally smith	68 Lemon Grove
5	tom taylor	53 London Road

STUDIES

Student#	Course#
1	C1
2	T2
3	T2
4	Q9
5	F3

COURSE

Course#	CourseName
C1	computing
F3	flower arranging
Q9	quantum mechanics
T2	theology

To query:

"What course does Student 2 work in?"

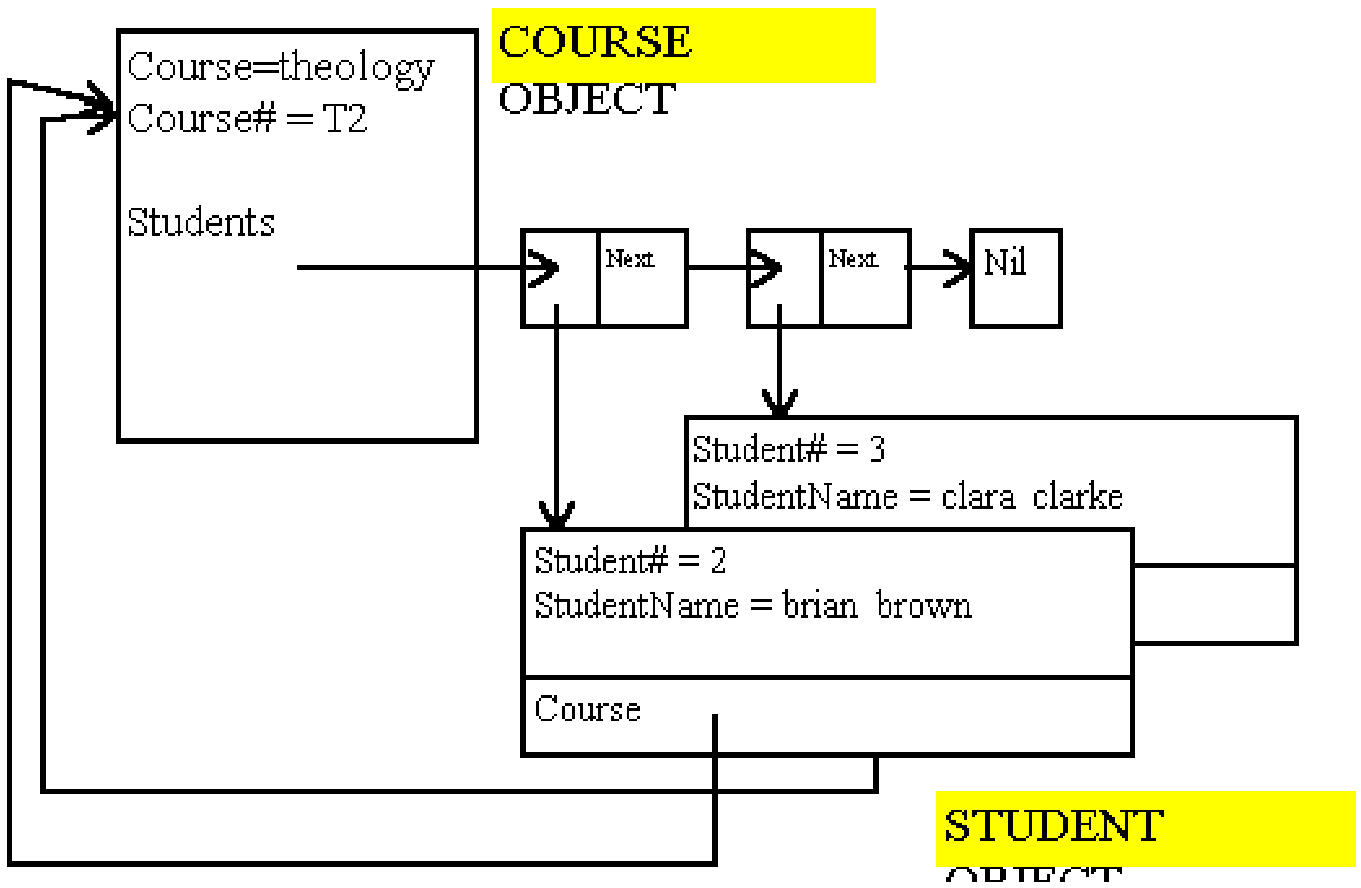
- go to STUDIES and look up Student 2 and return Course# (i.e. T2)

- go to COURSE and look up T2 and return Theology

"Name all students doing Theology"

- go to COURSE and find Course# (i.e. T2)
- go to STUDIES look up T2 and return all Student#s
- go to STUDENT and find each Student# and return each StudentName

Object Model of Student-Course Relationship



To query:

"What course does Student 2 work in?"

- search Students index for pointer to Student 2
- follow CoursePointer to T2 and return CourseName (Theology)

"Name all students doing Theology"

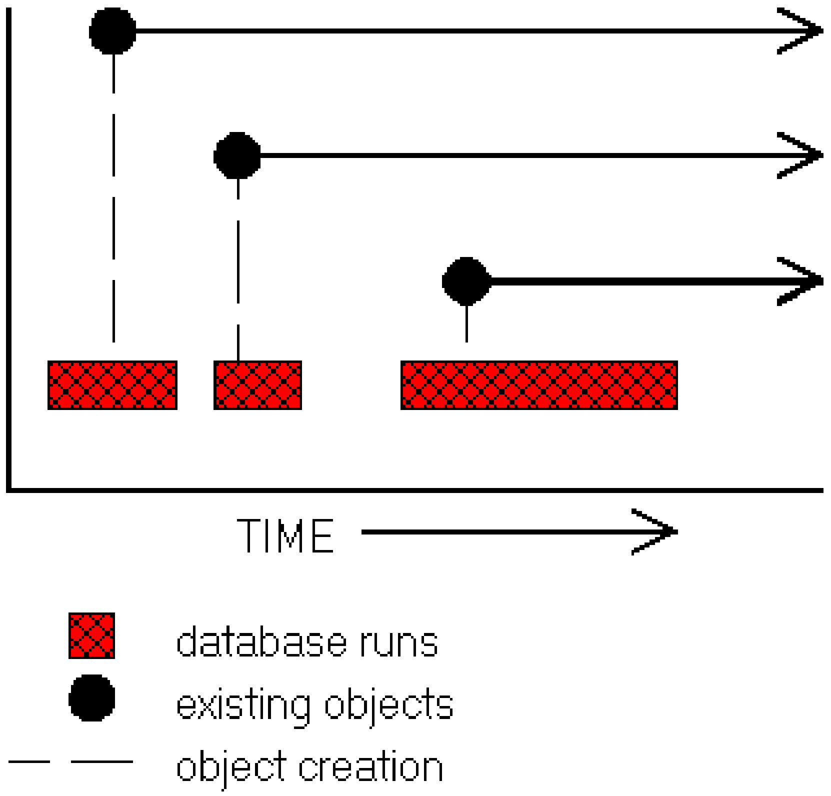
- search Course Index and find Course# (i.e. T2)
- follow Student Pointers, looking up each Student#

This process is called ***navigation,*** note that it relies heavily on ***pointers*** and that a large proportion of pointers ***must be*** persistent.

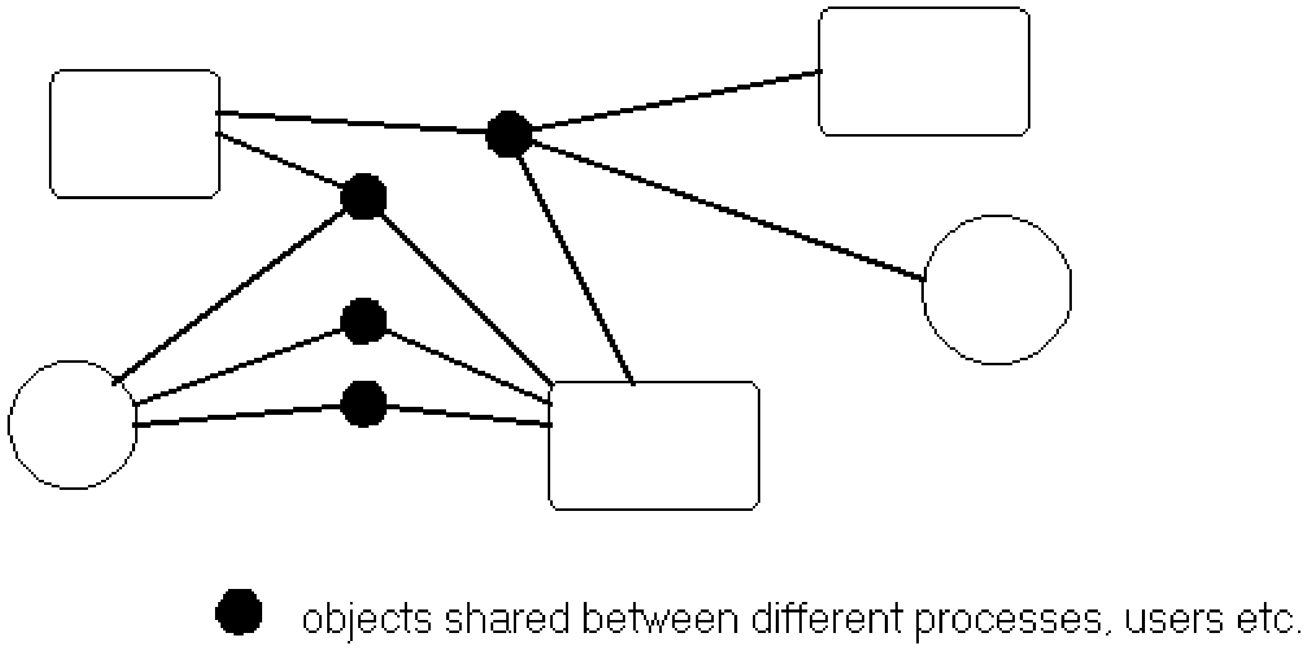
Implementation Issues

- Persistence
- Sharing
- Paging
- Back to the Beginning

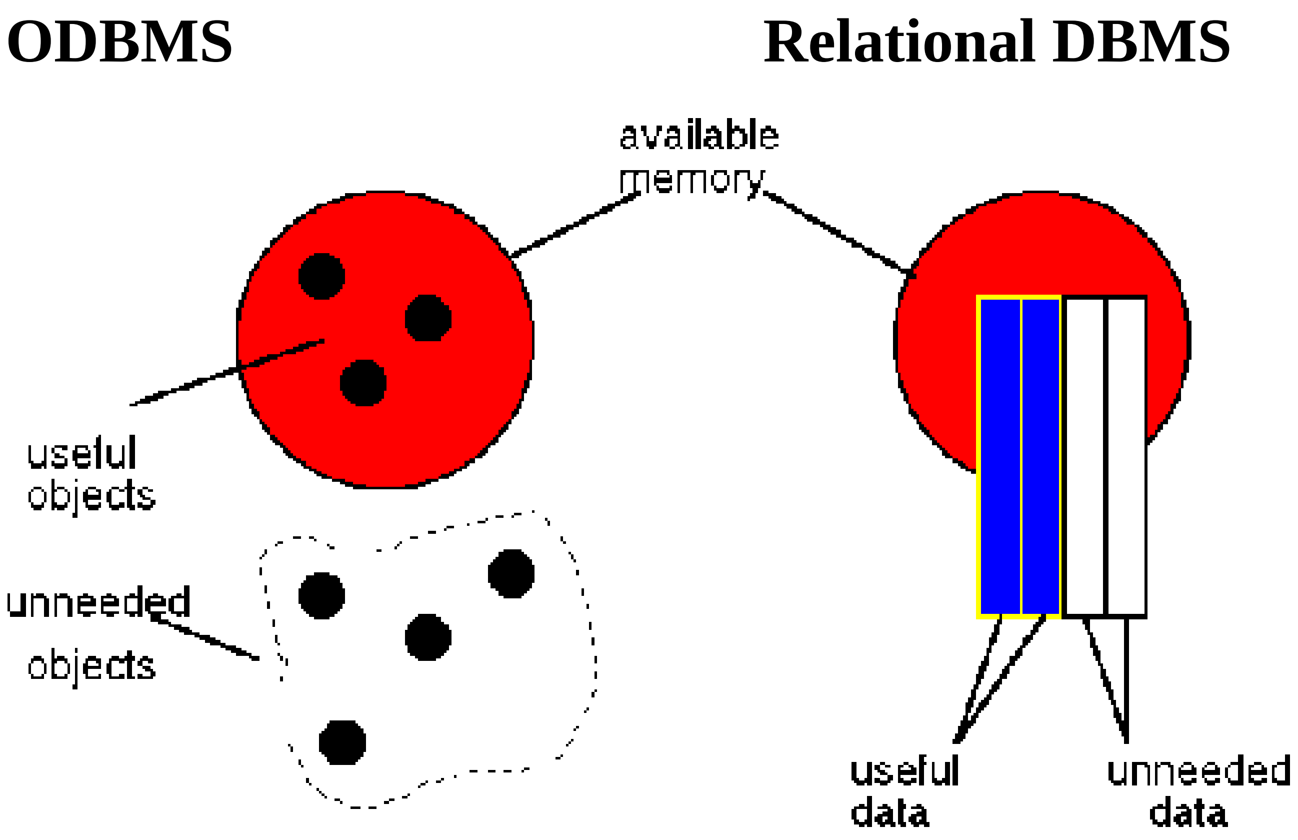
1. Object-oriented databases give objects ***persistence***, which enables objects to be stored **between** database runs. *(NOTE: in the context of ODBMS,* **PERSISTENCE** = ***POST RUN TIME PERSISTENCE****)*, this facilitates ***versioning*** (i.e. a new, additional object is stored each time changes are made).



1. Object-oriented databases allow objects to be ***shared between processes in a distributed environment.***



1. Object-oriented databases can ***reduce the need for paging by enabling only the currently required objects to be loaded into memory (relational databases load in tables containing both the equired data AND other unnecessary data)***



Relationships

Object-oriented databases model **four** standard relationships between the objects they contain:

1.*Inheritance*

i.e. an object is **a kind of** something else

2. *Association*

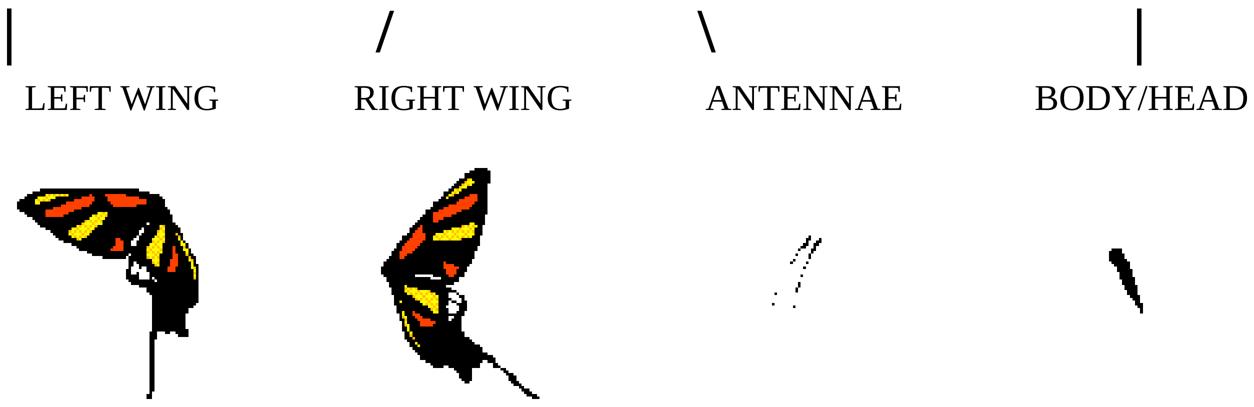
i.e. an object **has a connection with** another object

eats

3.*Aggregation*

i.e. an object **is made up of** other objects

______________________ **BUTTERFLY** _______________



This is the 'usual' view - a butterfly object knows what parts it is made up of.

4.*Inverse (or Parent) relationship*

i.e. an object **is part of** another object

is_part_of

Here we are saying explicitly that the wing 'knows' it is a part of the butterfly, but the butterfly does not necessarily know it is made up of wings.

Benefits and Drawbacks of ODBMS

- Benefits
- Drawbacks
- Back to the Beginning

Benefits

1. the objects **do not require re-assembling** from their component tables each time they are used thereby reducing processing overheads by increasing access speeds *e.g. up to 100 times faster for some applications (Sun Cattel benchmark)*

2. **Paging** is reduced

3. **Versioning** is easier

4. **Navigation** through the database is easier and more natural, with objects able to contain pointers to other objects within the database

5. **Reuse** reduces development costs

6. **Concurrency control** is simplified by the ability to place a single lock on an entire hierarchy (while still retaining the choice to lock individual objects)

7. **Better data model** as based on the 'real world' instead of the 'flattened' relational model

8. Good for applications where the ***relationships*** between items in the database carry **key information** *e.g. in the student database, we were particularly interested in what students studied (i.e. the STUDIES relationship). This is handled very efficiently by navigation.*

9. Relationships and constraints on objects **can be stored in the server application** rather than the client application therefore any changes need only be made in *one place* thus reducing the need for and risks involved in making multiple changes

10. Fit in well with client/server and distributed architectures

Drawbacks

1. Poor for applications where the ***values*** of items in the database carry **key information** *e.g. if we had been more interested in student age (e.g. to calculate the mean age) than the courses they study then relational database would clearly be more efficient*

2. **Speed of access** may be reduced by late binding which may cause extensive searches through the inheritance hierarchies

3. Present **lack of standards** including the lack of a common query language such as SQL (though OQL on its way?)

4. There are as yet **no formal semantics** for ODBMS. Relational databases can be 'proved' correct by means of set theory and relational calculus

5. The simplicity of **relational tables** is lost

6. The object oriented **paradigm shift** can make the move to ODBMS difficult

---

Examples of ODBMS

- Back to the Beginning

1. Many **pure** ODBMS are available
2. Also there is a whole group of **hybrid** Relational/Object DBMS.

	DATABASE ENGINE	
	Relational	*Object*
Relational Data Model	● Oracle ● Ingres ***1**	● Illustra ● UniSQL/X
Object Data Model	● OpenODB ● Odaptor ● Persistence ***2**	● Ontos ● ITASCA ● Versant ● ObjectStore ● Objectivity ● O_2 ● Gemstone ● Objectory/DB ***3**

***1** *These are traditional relational databases but new/projected versions have object-oriented 'add-ons' e.g. Ingres Version 6 (OpenIngres) Oracle Version 8*

***2** *Different ways to achieve this: Object Mediator (Persistence) - maps objects to relations, Object Wrapper (OpenODB) - object server sits on relational database, Odaptor like OpenODB but sits on Oracle*

***3** *Different 'OOP language dependencies': ObjectStore/Objectivity - based on C^{++} with Smalltalk support; Gemstone - based on Smalltalk with C^{++} support; O_2/ITASCA - language neutral, with several language support*

- Back to the Beginning

Steve Hand and Jane Chandler
26th February 1998

[University Home Page] [Faculty Home Page] [Departmental Home Page] [Jane's Home Page] [Disclaimer & Copyright]

This Page is maintained by *Jane Chandler*
Email: Jane.Chandler@port.ac.uk
Page last updated: *3rd September 1998*