ODBMS: PROTOTYPES & PRODUCTS

- Systems providing object databases programming languages
  - **Prototypes**: Encore-Ob/Server (Brown Univ.), IRIS (Hewlett-Packard), EXODUS (Wisconsin Univ.), Zeitgeist (Texas Instrument), Orion (MCC), ODE (Bell Lab), Gbase (Graphael), Vodak (GMD), TI Open OODB (Texas Instrument)
  - **Products**: GemStone (Serviologic), ObjectStore (Object Design), ONTOS (Ontologic), Versant (Versant Object Tech.), O₂ (Ardent Software), Objectivity/DB (Objectivity), POET (Poet Software), Jasmine ii (Computer Associates Inter.)

- Object Extensions of Relational DBMS
  - Illustra (Illustra Technologies), UniSQL (UniSQL Inc.), OpenODB (Hewlett-Packard), Matisse (ABC), Omniscience (Omniscience), Polyhedra (Polyhedra PLC), PostgreSQL (PostgreSQL Organization)

The ODBMS Manifesto [M. Atkinson & al 89]

- Open Choices:
  - Uniformity of the Model (Classes/Methods/Objects)
  - Naming and Persistence Model
  - Typing System (Classes and/or Types)
Uniformity of the Data Model

- Three independent choices:
  - At the implementation level: are classes and methods represented as objects?
  - At the language level: are classes and methods treated as objects (syntactical and semantic uniformity)?
  - At the interface level: are classes and methods presented as objects?
- Unlike O2 and ONTOS, in Orion, Gemstone and IRIS the classes are objects

Towards Persistence Modeling

- How objects become persistent (i.e., database objects): two approaches
- Persistence is defined independently from the language: it is a property of the data model (using names)
  - By class names: classes instances are automatically persistent (e.g., ORION)
  - By persistent root names: each object referenced by a root becomes persistent (e.g., O2)
- Persistence is a property of the underlying language
  - By library classes provided by the system: objects becomes persistent if there classes inherit from a specific class `Object` (Ontos) or `PObj`ect in (Versant)
  - By an expression of the language: e.g., `persistent` (Object Store), `persist` (Zeitgeist) `pnew` and `pdelete` (ODE)
Aspects of Persistence

- **Orthogonality of persistence and types**: data can be persistent independently of its type
  - Instances of types can be persistent or transient (e.g., O2, ObjectStore)

- **Orthogonality of persistence and objects creation**: definition and methods for memory allocation are the same for persistent or transient objects
  - There is not a need to copy objects from the program work space to the database in order to make objects persistent (O2)

- **Behavioral transparency**: uniform manipulation of persistent or transient objects in programs
  - No distinction in access and update operations nor in variable declarations

- **Persistence is a property which can be attached either to types or instances**: two paradigms of persistence propagation

Persistence Propagation by Inheritance

- **Root Class of Persistence**: `PObj ect`
  - All subclasses of persistent classes became persistent
  - `New` and `Delete` messages are overloaded
  - A `Lookup` primitive is added for searching

- **Non orthogonal to types**
  - Only the types which inherit from `PObj ect` persist
  - Duplication of persistent and transient classes
Persistence Propagation by Reference

- Definition by the programmer of Persistent Roots
  - Keyword "database" or "persist" added:
    ```
    Artist* art = new database Artist( "Monet");
    ```
  - Persistence objects are catalogued
    - accessible by a Lookup method

- All objects referenced by a persistent object become persistent
  - References become persistent during object storage
    - references are replaced by oids

Object Deletion: DB vs. PL view

- **DB Approach**: Explicit “delete” operation
  - Dangling references can be managed automatically or by the user
- **PL Approach**: Implicit deletion when objects are not any more referenced by others
  - The difficult deletion of individual objects implies the use garbage collector techniques
The Issue of Naming in ODBMS

- Naming and Persistence in ODBMS are closely related.
- Names specify how to enter the database graph.
  - Impact of the naming model on the simplicity and expressiveness of the QL.
- Two naming approaches:
  - Implicit => query only the class extensions i.e., set of instances.
  - Explicit => query any object or value in the database.
- NOTE: In the relational model only the relations are named and the queries are on collections of tuples.

Typing System: Object/Class vs. Value/Type

- An object:
  - is encapsulated in a class (structure & behavior).
  - is manipulated by methods (encapsulation).
  - has an identifier (data sharing).
- A value:
  - has a type (just a structure no methods).
  - is manipulated by functions (no encapsulation).
  - hasn’t an identifier.

- Types are used for:
  - documentation.
  - correctness (type checking).
  - efficiency (optimization).
- Classes add:
  - object factory (new).
  - object behavior (method implementation).
  - collection (extent).

➲ A Compromise: Imperative vs. Object-Oriented
  Two paradigms: All Object and Object/Value ODBMS.
A Compromise

➊ All Object DBMS
- Structuring primitives are objects (e.g., collection classes)
- Meta-Classes & Inheritance at class level
- Class extensions are managed by the system
- Possible explicit object deletion
- Example: GemStone
  ➡ Provides for complex behaviors

➋ Object/Value DBMS
- Structuring primitives using type constructors
- Inheritance defined by sub-typing
- Class extensions can be managed by the programmer
- Object deletion using a Garbage Collector
- Example: O2
  ➡ Provides for complex structures

A Classification of the Systems

➊ LANGUAGE-ORIENTED SYSTEMS
- Decrease the distance between the type system (data model) of the PL and that of the database (e.g., ONTOS)

➋ SYSTEMS WITH PERSISTANT PL
- In addition to the reduction of the impedance mismatch between the PL and the DB the frontier between the language and the database disappears (e.g., ObjectStore, ODE, Zeitgeist, Versant)

❼ APPLICATION-ORIENTED SYSTEMS
- Providing functionality suited to specific applications for example (CAD, CAM, CAE) versions, schema evolution, long transactions, etc. (e.g., Objectivity)

➍ COMPLETE ODBMS
- Integrate the functionality of a DBMS with the object-oriented technology (e.g., GemStone, O₂)
VIII) REFERENCES

- M. J. Carey and D. J. DeWitt: “Of Objects and Databases: A Decade of Turmoil”. In Proc. of VLDB’96, pp. 3-14, Bombay, India, Sept. 1996
- F. Bancilhon, C. Delobel, S. Gamerman: “SGBD Orientés-Objet”, GIP Altair
- B. Amann: “Des SGBD Relationnels aux SGBD Orientés-Objet”. Course Slides, CNAM Puteaux
- Y. Viénon: “Object-Orientation and Databases”. Course Slides, PARIS-VI