O_2: an Object-Oriented Database System

including: The story of O_2 by *O. DEUX et al.* and

O_2, an Object-Oriented Data Model

by *Christophe Lècluse, Philippe Richard and Fernando Velez*

Presented by: Sheng-Yih Wang

October 24, 1996

Introduction

O_2 consists of eight functional modules:

- OOPE, the object-oriented programming environment.
  - provides graphical tools to database operations such as edit, update, query and browse.
  - support DDL and DBPL.
- The alphanumeric interface, a non-graphical user interface.
- LOOKS, the user interface generator.
- The language processor
- The query interpreter
- The schema manager
- The object manager
  - mapping between abstract object data model and disk representation.
- The disk manager
  - I/O
  - data placement
  - indexing
  - buffering

A Programmer's View of the System

- Data Model and Data Definition language
  - Objects and Values

    i_0: **tuple**(**name:** "Eiffel tower",
    address: i_5,
    description: "A famous Paris monument",
    closing_days: **list**("Christmas", "Easter", "August 15th"),
    admission_fee: 25)
    i_1: **tuple**(**name:** "Paris",
    map: i_2,
    hotels: **set**(i_3, i_4))
    i_5: **tuple**(**city:** i_1,
    street: "Champ de Mars",
    number: 1)

  - Types and Classes

    tuple(**name:** string,
    map: Bitmap,
    hotels: **set**(Hotel))

  - Naming and Persistence
    * Persistence is attached to Names
    * extension — set of all objects in the class
    * example:

      add class City **with** extension
      type tuple(**name:** string,
      map: Bitmap,
      hotels: **set**(Hotel))
    * transient object — classes without extension

- Methods
  - declare method by giving its *signature*
  - **add method** check_rates(): Boolean **in class** hotel
- Subtyping and Inheritance
  - inheritance mechanism based on subtyping
  - subtype — more defined, more attributes

    tuple(**name:** string,
    address: Address,
    description: Text,
    closing_days: **list**(string),
    admission_fee: integer,
    facilities: **list**(string),
    stars: integer,
    rate: float)

    is a subtype of

    tuple(**name:** string,
    address: Address,
    description: Text,
    closing_days: **list**(string),
    admission_fee: integer)
- The "Object" class
  - pre-defined
  - root of the class hierarchy
  - pre-defined methods for equality testing, copying, etc.

- Exceptional Attributes and Methods
    - a tuple value can have extra attributes

      i_0: **tuple**(**name:** “Eiffel tower”,
      address: i_5,
      description: “A famous Paris monument”,
      closing_days: list(“Christmas”, ”Easter”, “August 15th”),
      admission_fee: 25,
      height: 315)

    - a named object can have specific methods
    - **add method** increase_fee(amount: integer) **in object** Eiffel_tower
    - **add method** increase_height(amount: float) **in object** Eiffel_tower

- The O_2 language
  - DBPL
    - write codes of methods
    - language independent design
    - define an O_2 layer for object/value declaration/manipulation
    - add O_2 layer to L language → LO_2
    - **body** increase_fee(amount: integer)
      in class Monument
      co2 { *self.**admission_fee** += amount; }
    - Iterator: **for** (x **in** S [**when** condition]) < statement >
    - **o2** Monument x;
      for (x **in** Paris_monuments
      when (*x.**admission_fee** <= 20))
      [x increase_fee(3)];

- Query language
  - two-mode language
  - return object or value
  - functional and first-order
  - subset of programming language

- Development and Execution Modes
  - Development Mode — interactive
  - Execution Mode — schema is frozen, performance is better, safer
  - Example: **compile** `Tour_operator`

- Distribution — visible to the application programmer.
  - `[` **on server** `Eiffel_tower_increase_fee(3) ]`
  - `[` **on workstation** `Eiffel_tower_increase_fee(3) ]`
  - `[` **on other** `Eiffel_tower_increase_fee(3) ]`

LOOKS, the user interface generator

- A two-dimensional space of generic display algorithm
- Specific presentations
- Structured edition
- Interactive method activation
- Browsing and querying the database
- example

OOPE, the object-oriented programming environment

- implemented in CO_2, interface is built using LOOKS
- Including *browser*, *O_2 shell*, *workspace*, *debugger*

The Implementation

- development mode — all modules are present
- execution mode — only object manager, schema manager and LOOKS
- System Decomposition and Process Layout
  - upper layer: schema manager
  - middle layer: object manager
  - innermost layer: WiSS (Wisconsin Storage System)
  - mirror process
- The schema manager
  - create class in any order
  - deletion of non-leaf/used class is forbidden
- The object manager
  - Complex Object Manager
    - Object Identifier — persistent
    - Object Identifier = RID in WiSS
    - use forward-marking
    - RID assigned when commit
    - distinguish objects from values
    - structured values are stored as objects
    - primitives for values / message passing for objects

    * Tuples: record stored on a page
    * Long data item (LDI) format for large tuples
    * Lists: ordered trees
    * Set: B-tree
    * Multimedia Objects: unstructured byte string
- Message Passing Manager
    * implemented as function call whenever possible in execution mode
- Transaction Manager
    * transactions are dealt with at the OM level
    * distinguish concurrency on schema/objects
    * recovery can be switched on/off
    * transactions can run in *object fault/resident* modes
    * O_2 transaction maps to WiSS transaction
    * optimistic approach — transaction update objects in workstation without CC
- Communication Manager
    * simplicity — Ethernet, TCP/IP
    * Transparency
    * Performance — execution migration, optimized object moving
    * Reliability — failure detection
- Persistence
    * defined as *reachability from persistent root objects*
    * named objects or values
- O_2 Buffer Manager — dual buffer
- Clustering manager — placement tree
- Index Manager — class-hierarchy indexing

Performance of the O_2 Prototype

Table 1: Object Creation Time

Type	Creation Time
Tuple	1.8
Set	1.6
List	1.6

Table 2: Time to read or write an attribute of a tuple object

Object Type	New Class	Persistent Class
Temporary Object	0.29 ms	0.44 ms
Named Object	0.58 ms	0.97 ms

Table 3: Time to apply a null method to an object

Method#	New Class	Persistent Class
1	0.60 ms	0.90 ms
10	0.65 ms	0.93 ms
20	0.70 ms	0.98 ms

Table 4: Time to scan a set of 10,000 tuple objects

Selectivity Factor	Elapsed Response Time
0%	28.7 s
10%	35.3 s
100%	99.1 s

O_2 object-oriented data model