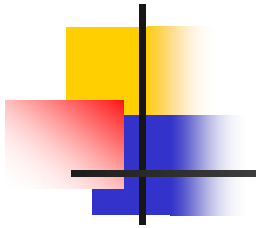
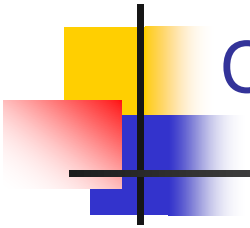


Object and Object Relational Databases

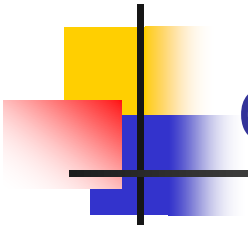
Instructor : Assoc. Prof. Dr. Đặng Trần Khánh

Cao Hữu Vũ Lam -13070241
Trần Thị Mi - 13070247



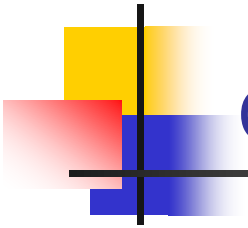
Outline

- Object database
  - Object Database Fundamentals
  - ODMG
    - Object v/s Relational Models
    - ODL
    - OQL
  - Advantages /disadvantages
- Object Relation database
  - ORDBMS Fundamentals
  - SQL3
  - Advantages /disadvantages
- Comparison of ODBMS and ORDBMS



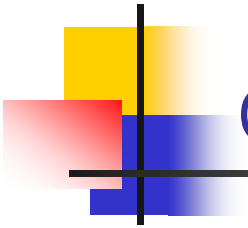
Object database - Fundamentals

- Combines the features of an object-oriented language and a DBMS
- Has its origin in OO programming languages
- Need for complex data types
- Object
  - State – current value
  - Behavior - what operations are permitted



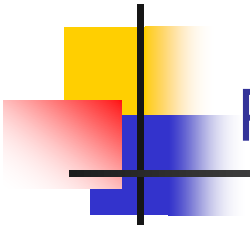
Object database features

- Abstract data types
- Inheritance
- Object identity
- Persistence
- Support of transactions
- Simple querying of bulk data
- Concurrent access
- Resilience



Constructors

- Atom
- Structured (or tuple)
  - Date, Interval, Time, Timestamp
- Collection
  - Set: unordered, no duplicates
  - Bag: unordered, duplicates
  - List: ordered, elements can be inserted
  - Array: ordered, elements can be replaced
  - Dictionary: maps keys to values



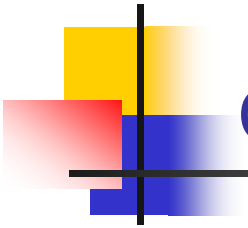
Persistence of Objects

- Transient objects
  - Exist in executing program
  - Disappear once program terminates
- Persistent objects
  - Stored in database and persist after program termination

ODMG

- Object Database Management Group (ODMG) founded in 1991 by Rick Cattell
- Made up of several parts
  - Object Model
  - Object Definition Language (ODL)
  - Object Query Language (OQL)
  - Language bindings

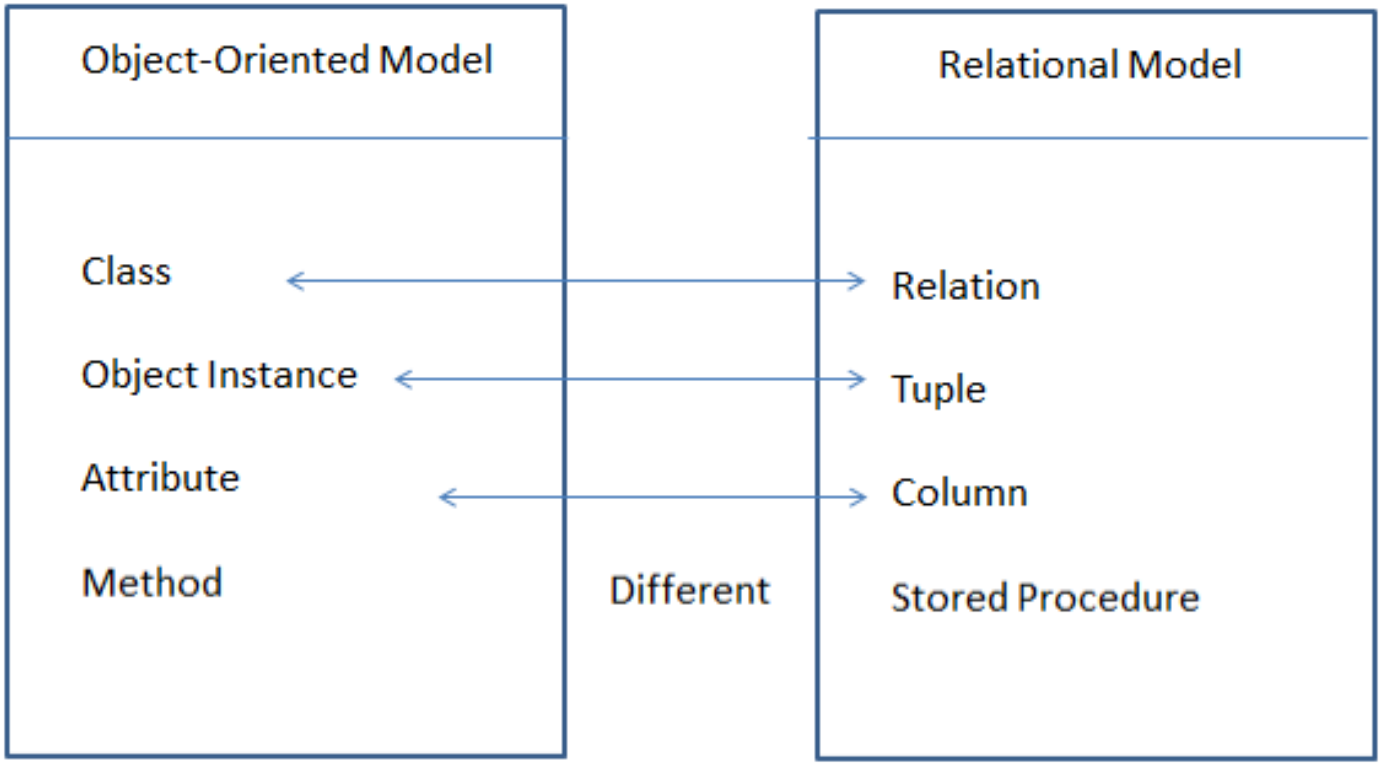
Rick Cattell

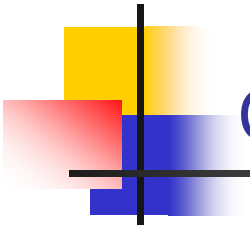


ODMG Object Model

- Data model for object definition language (ODL) and object query language (OQL)
- Basic modelling primitives
  - Object: unique identifier
  - Literal: no identifier
- Object has five aspects:
  - Identifier
  - Name
  - Lifetime
  - Structure
  - Creation

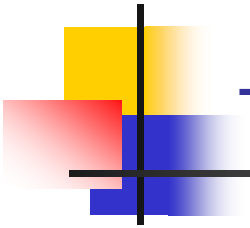
Object v/s Relational Models





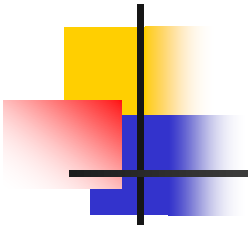
Object v/s Relational Models

- A method in an object model is defined in the class to which the object belongs.
- A stored procedure is a sub-routine available to applications and this is external to the database, defined in the data dictionary.
- OODB is OO language specific whereas Relational DB are language independent via SQL.
- No impedance mismatch in applications using OODB where as object relational mapping must be performed in relational database for use in OO applications.



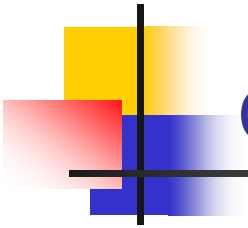
Type Specifications

- Interface
  - Defines only abstract behaviour
  - *Interface Employee {…};*
- Class
  - Defines abstract behaviour and abstract state
  - *Class Person {…};*
- Literal
  - Defines abstract state
  - *Struct Complex {…};*



Inheritance

- IS-A relationship
  - Inheritance of Behavior
  - Multiple inheritance
  - *Interface Professor : Employee {…};*
- EXTENDS relationship
  - *Inheritance of state and behaviour*
  - *Single inheritance*
  - *Class Student extends Person {…}*



Object Definition Language (ODL)

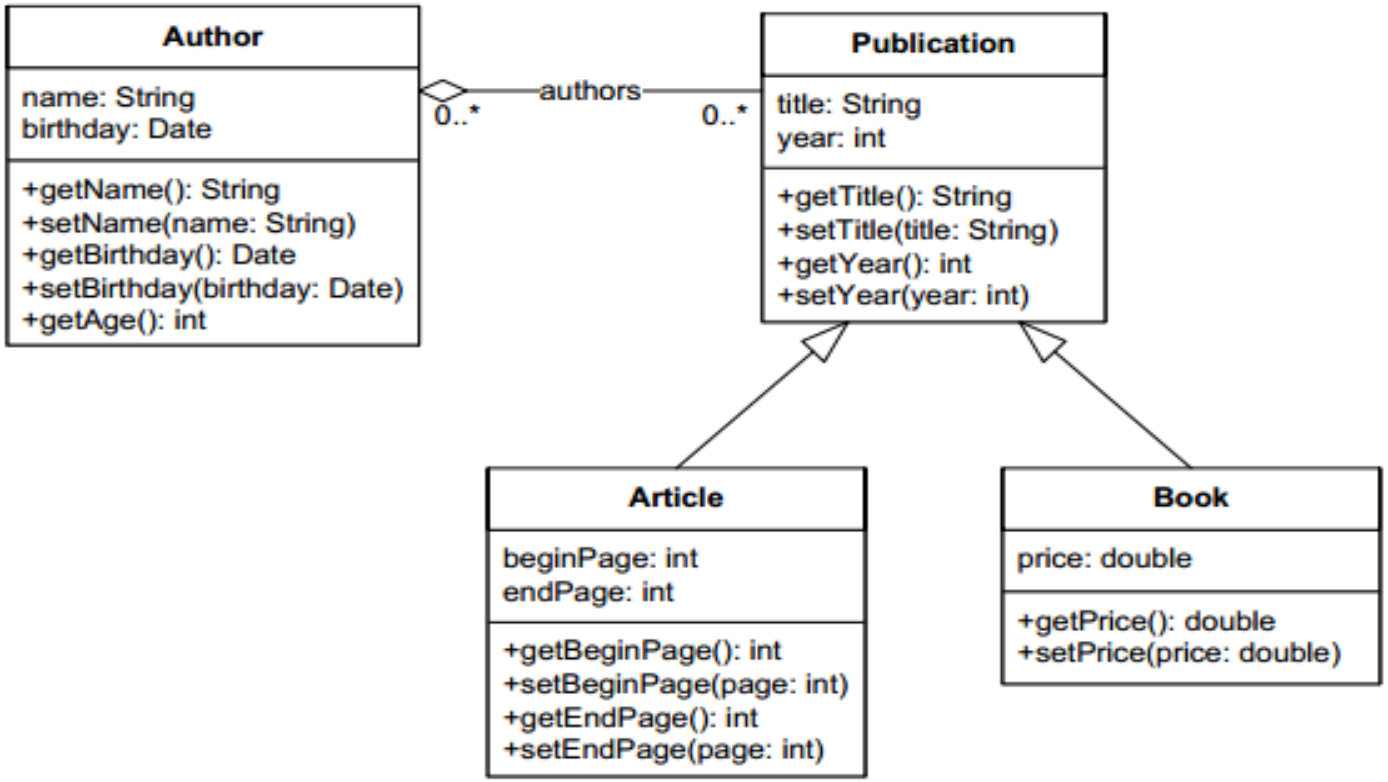
- Used to define object types for a particular database application
- Support semantic constructs of ODMG object model
- Independent of any particular programming language
- ODL object class definition

```
class name [ ( extent name , key name ) ] {
    { exception name { { type name } } }
    { attribute type name }
    { relationship type name inverse relationship }
    { type name ( { ( in | out | inout ) type name } ) [ raises ( { exception } ) ] }
}
```

ODL example

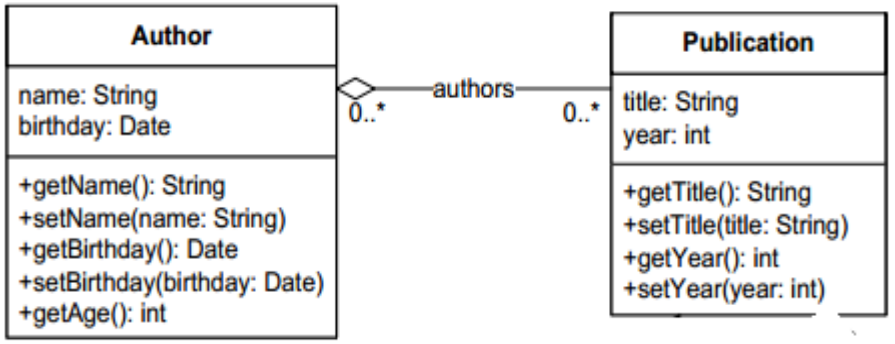
```
class PERSON
(     extent      PERSONS
      key         Ssn  )
{     attribute   struct Pname {    string   Fname,
                                    string   Mname,
                                    string   Lname  }     Name;
      attribute   string                                  Ssn;
      attribute   date                                    Birth_date;
      attribute   enum Gender(M, F)                       Sex;
      attribute   struct Address {  short    No,
                                    string   Street,
                                    short    Apt_no,
                                    string   City,
                                    string   State,
                                    short    Zip  }       Address;
      short       Age();    };
```

ODL example


Author
name: String
birthday: Date
+getName(): String
+setName(name: String)
+getBirthday(): Date
+setBirthday(birthday: Date)
+getAge(): int
0..*
authors
0..*
Publication
title: String
year: int
+getTitle(): String
+setTitle(title: String)
+getYear(): int
+setYear(year: int)
Article
beginPage: int
endPage: int
+getBeginPage(): int
+setBeginPage(page: int)
+getEndPage(): int
+setEndPage(page: int)
Book
price: double
+getPrice(): double
+setPrice(price: double)

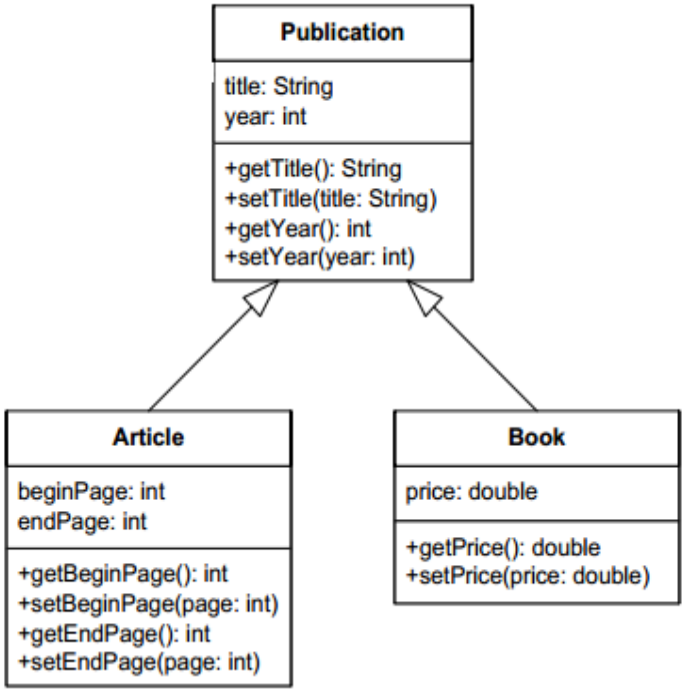
ODL example

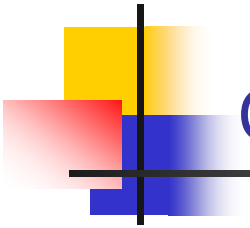
```
class Author (extent Authors) {
    attribute string name;
    attribute date birthday;
    relationship set<Publication> authors
    inverse Publication::authored_by;
    integer get_age();
};
class Publication (extent Publications) {
    attribute string title;
    attribute integer year;
    relationship list<Author> authored_by
    inverse Author::authors;
    string GetTitle();
};
```



ODL example

```
class Article extends Publication (extent
    Articles) {
    attribute unsigned short begin_page;
    attribute unsigned short end_page;
};
class Book extends Publication (extent
    Books) {
    attribute double price;
};
```



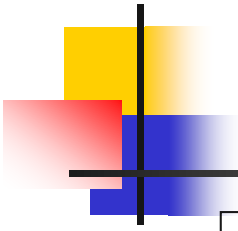


Object Query Language (OQL)

- Based on ODMG Object Model and SQL-92

```
select    list of values
from      list of collections and typical member
where     condition
```

- Query structures look very similar in SQL but the results returned are different.



OQL example

```
Select   distinct v.name
From     voters v
Where    v.state = “Colorado”
```

Voter Id	Name	State
V1	George Love	Colorado
V2	Winnie the Pooh	Florida
V3	John Lewis Hall	Colorado

Result from SQL table with rows

Name
George Love
John Lewis Hall

Result from OQL collection of objects

String	String
George Love	John Lewis Hall

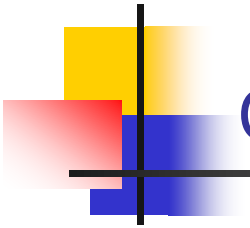
OQL example - Basic

```
select   a.authors.title
from     Authors a
where    a.name = “Tilmann
    Zaeschke”
```

Illegal as the “dot”
operator cannot be applied
to a collection of objects

```
select   p.title
from     Authors a, a.authors p
where    a.name = “Tilmann
    Zaeschke”
```

Correct solution based
on correlated variables



OQL example - Return Types

- Queries with DISTINCT return a set

```
select    distinct  a.name
from      Authors a
Set<Struct { string name }>
```

- Queries with ORDER BY return a list

```
select       s.title
from         Publications s
order by     p.year desc
List<Struct { string name }>
```

OQL example - Subqueries

- Find the names of all co-authors of Michael Nebeling

```
select   distinct  a.name
from     (select     mp
           from      Authors m, m.authors mp
           where     m.name = “Michael Nebeling”) p, p.authored_by a
```

- Find the titles of the articles that were published in the same year at the book on the ODMG 3.0 standard

```
select    p.title
from      Articles p, (select     o.year
                       from       Book o
                       where      o.title = “ODMG 3.0”)  y
Where p.year in y
```

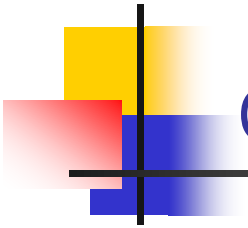
OQL example - Universal and Existential Quantification

- Find the names of authors who have written a book that costs less than 20 €

```
select    a.name
from      Authors a
Where     exists b in Books:
          b.price < 20 and b in a.authors
```

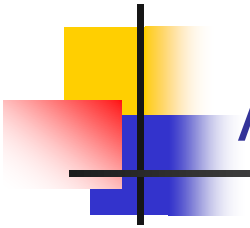
- Find the names of authors who have not published anything since 2000

```
select    a.name
from      Authors a
Where     for all p in a.authors:
          p.year <= 2000
```



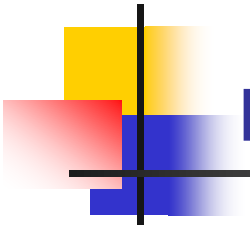
C++ Language Binding

- Specifies how ODL constructs are mapped to C++ constructs
- Uses prefix d_for class declarations that deal with database concepts
- Template classes
  - Specified in library binding
  - Overloads operation new so that it can be used to create either persistent or transient objects



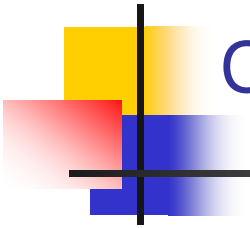
Advantages

- More semantic information in the database
- Support for complex objects
- Better performance in complex applications
- Reusability of classes
- Extensibility of the data types supported by the database



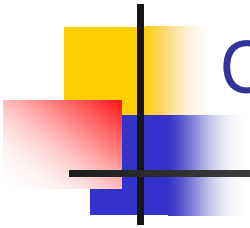
Disadvantages

- Lack of universal data model
- Lack of experience
- Complexity
- Lack of support for security



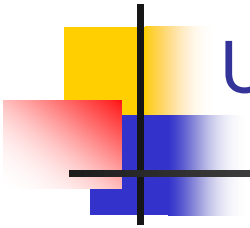
ORDBMS Fundamentals

- Combination of both relational and object databases
- Relation is still central
- Query language – SQL3



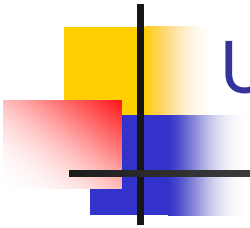
ORDBMS Fundamentals

- OO Features in SQL3
  - Objects: User Defined Types(UTDs), Row Objects
  - Type constructors
  - Collection types
  - User-defined functions
  - Support for large objects
  - Inheritance



User Defined Types

- A key feature of ORDBMS
- Combination of atomic data types and associated methods
- DBMS doesn’t need to know about how UDT are stored or their methods work? It just has to know about UDT signature
- Hiding UDT internals is called encapsulation



User Defined Types

- General form of UDT specification
  - CREATE TYPE <type-name> (
    <list of component attributes and their types>
    <declaration of functions (methods)>
    );
- Example

CREATE TYPE CustomerUDT(
Id CHAR(12),
Name NVARCHAR(100)
)

Row Objects

- Directly create a structured attribute using the keyword ROW
- General form

```
CREATE ROW TYPE <type-name> (
 <list of component attributes and their types>
 );
```

- Example

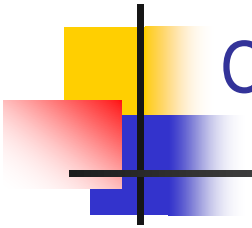
```
CREATE ROW TYPE AddressRT(
 City NVARCHAR(100),
 Country NVARCHAR(100),
);
```

Creating Tables Based on the UDTs, Row Objects

- General form
  - *CREATE TABLE <table-name>* OF TYPE <type-name /row-type-name>
- Example 1:

  CREATE TABLE Address OF TYPE AddressRT;
- Example 2:

```
CREATE TABLE Customers (
  Id CHAR(12) NOT NULL PRIMARY KEY,
  Name NVARCHAR(100) NOT NULL,
  Address AddressRT
)
```



Collection Types

- ARRAY works approximately as in programming languages
  - Reference elements using []
- Example

```
CREATE TABLE Customers (
        Id CHAR(12) NOT NULL PRIMARY KEY,
        Name NVARCHAR(100) NOT NULL,
        Address AddressRT ARRAY[10]
)
SELECT Address[1].City FROM Customers
```

Object Identifiers Using Reference Types

- Create unique system-generated object identifiers
- A component attribute of tuple may be a reference to a tuple of another (or posibly same) relation.

Object Identifiers Using Reference Types

- Example:

```
CREATE TABLE Customers (
        Id CHAR(12) NOT NULL PRIMARY KEY,
        Name NVARCHAR(100) NOT NULL,
        Address  REF AddressRT
    )
INSERT INTO Customers(Id , Name , Address )
    SELECT  '1', 'Marry' , REF(T) FROM AddressTable  T
    WHERE ROWNUM = 1
```

Object Identifiers Using Reference Types

SELECT C.Address FROM Customers C

WHERE ID = 1;

Result :

ID :

00002202086F2A5BC83EC7440E97984B2C9EA9D9FEB23
93991A58C4229897D0F775122F006

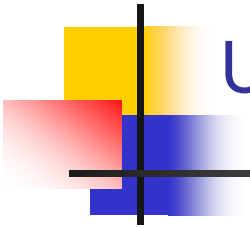
SELECT C.Address.City FROM Customers C

WHERE ID = 1;

Result :

City :

NewYork



User-defined functions

- Two types :
  - Internal
  - External
- Defining UDT's methods
  - FUNCTION <name> (<arguments>) RETURNS <type>;

Internal Functions

- Internal functions are written in extended SQL
- Example
  - FUNCTION *AddressRT*(:dCity *NVARCHAR(100)*, :dCountry *NVARCHAR(100)*)

```
RETURNS AddressRT;
:d AddressRT;
BEGIN
        :d := AddressRT();
        :d.city := :dCity;
        :d.Country := : dCountry;
        RETURN :d;
END;
```

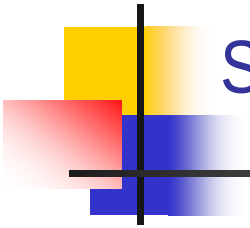
External functions

- UDT's can have methods that are written in host language (e.g. C, C++, ...)
- Only signature appears in UDT definition
- General form

  DECLARE EXTERNAL <functionName> <signature>

  LANGUAGE <language name>
- Example

```
DECLARE EXTERNAL Square
    POLYGON
    RETURNS BOOLEAN
LANGUAGE C;
```



Support for large objects

- Support for large objects
  - Blob : binary large object
  - Clob : character large object
- Consider class recording
  - Class_video_recordings(cid: integer, cmdate: date, loc char(10), video: BLOB)

Inheritance

- Provided using keyword UNDER
- Example:

  CREATE TYPE Student UNDER Person (addr address)

  - Creates an explicit relationship between subtype Student and supertype Person.
  - An object of subtype is considered to be an object of supertype.

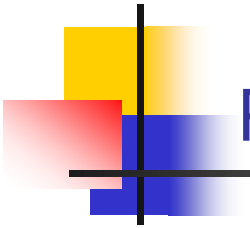
Comparison of ODBMS and ORDBMS

Object Databases

- Complex data types
- Integration with programming language
- High performance

Object Relation Databases

- Complex data types
- Powerful query languages
- High protection



References

- *[1] R. Elmasri, S.B. Navathe: "Fundamentals of Database Systems", 6th Edition, Pearson Addison-Wesley, 2011, pp.353-413.*
- *[2] Object & Object-Relational Databases , Dr.Ranga Raju Vatsavai*
- *[3]Wikipedia:"Object database", 7 Sep 2014*
- *[4]Wikipedia:"Object-relational database", 7 Sep 2014*

Thanks for your attentions!

Q & A