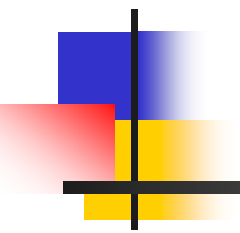
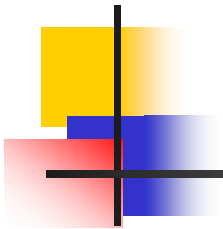


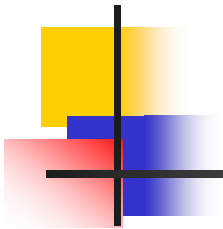
Object Relational and Extended-Relational Systems

Chapter 22



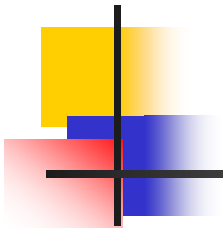
Outline

- Introduction
- OR – Data types
  - Simple
  - Complex
    - Structures
    - Collections
  - Referencing (pointers)



Why OR

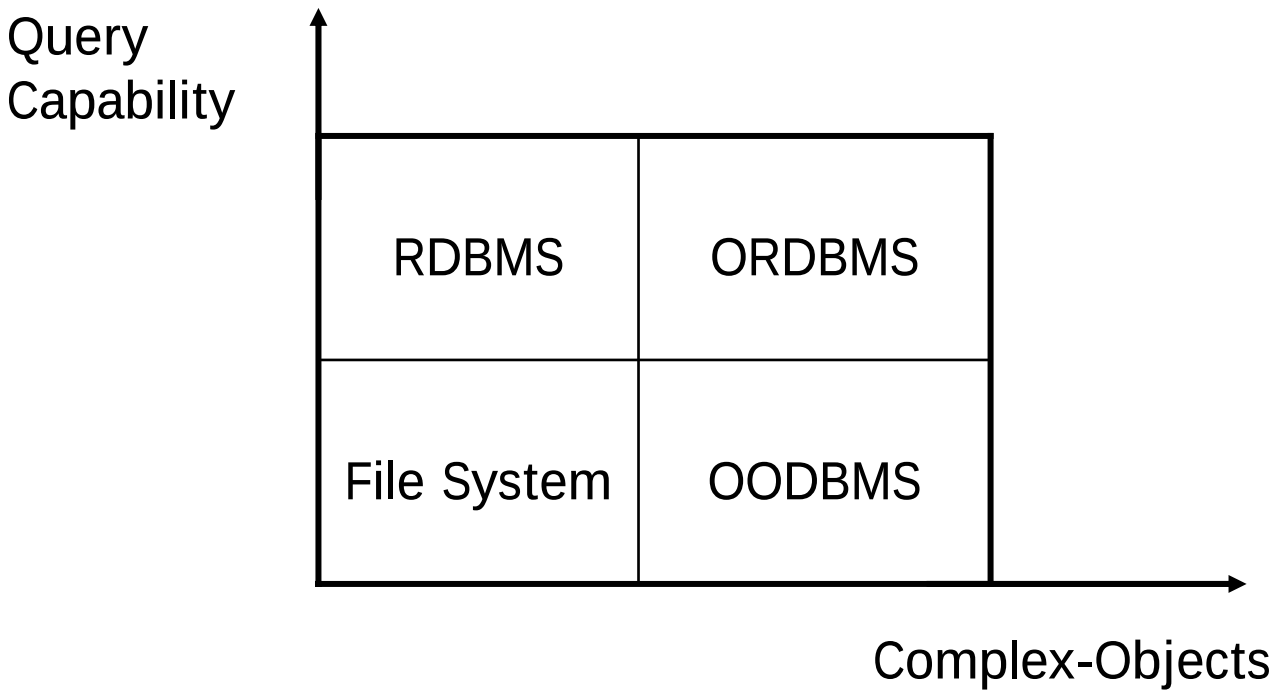
- OO paradigm exposes weakness of simple Relational Model;
  - Abstraction, encapsulation inheritance
- RDBMS products and SQL language enjoy a large market share and wide acceptation within users
- Pressure on Relational DBMS software vendors to respond to OO qualities:
- Success of Berkeley Univ. *Postgres DBMS*
- Advances on query processing with large-objects and user functions;
- Evolution not a Revolution;
  - Support to new applications;



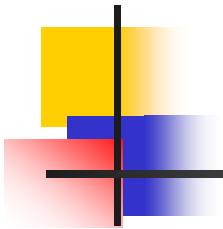
OR-DBMS – SQL Extension

- SQL extension -> the way to go (Stonebreaker96):
- But SQL has to support:
  - Abstract Data Types (ADTs)
  - Complex objects(construtors)
  - Functions and procedures
  - Large objects support (BLOBS and CLOBS)
  - Inheritance

Perspective

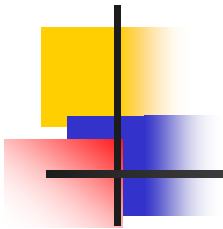


M. Stonebreaker 1996



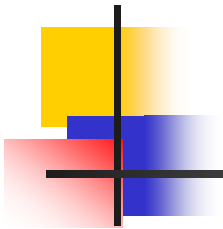
Relational – Simple Type

- Relational Model
  - Attributes of simple types;
  - SQL standard types: Integer, Character, Date, Time etc..
  - Relation
    - name
    - Attribute definitions – 1stNF
    - Constraints (primary, foreign keys, simple domains,...)
    - instances: set of tuples of Relation type



OR - Complex - Structures

- OR abandons restrictions concerning First normal form;
- Attributes are defined according to *Domains* that can be complex;
  - Structures
  - Collections
  - References (pointers)
- Domains are modeled through User Data Types (UDT)



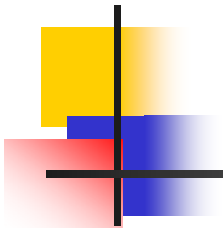
OR- Attribute Types

- **Primitive data types**
  - Name string
  - Telephone Number(10)
  - Country char(30)
- **Complex Type**
  - **Structured** information:
    - Address – {street, number, PObox, city}
  - **Collections sets:**
    - Set of courses
  - **Collections list: arrays**
    - List of telephone numbers

OR - User Defined Types (UDT)

- Structure:
  - Attributes
    - Simple types (SQL datatypes)
    - Complex types (other UDT, row, array)
    - References (point to types)
  - Methods signature
- Example

```
create type newperson_ty as object
      (firstname          varchar2(25),
      lastname            varchar2(25),
      birthdate           date,
member function AGE(BirthDate in DATE) return NUMBER
```

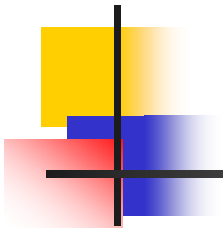


- UDT in Oracle 8i

- Syntax:

  Create Type *nameOfType* AS [object, table, varray]
  <attribute and method declaration>

- **Object**: for structures
- **Table**: for Sets
- **Varray**: for bounded and ordered list



- Object Type Implementation ...

- Creating Types Similar to creating a “class” with attributes:

```
CREATE TYPE addr_ty AS OBJECT
(street varchar2(60),
city    varchar2(30),
state   char(2),
zip     varchar(9));
```

... - Object Type Implementation ...

- Imbedding Objects and Nesting
  - Create a person type with address type nested inside:

```
CREATE TYPE person_ty AS OBJECT
(name     varchar2(25),
address  addr_ty);
```

  - Create a student type with person type nested inside:

```
CREATE TYPE student_ty AS OBJECT
(student_id       varchar2(9),
person    person_ty);
```

... - Object Type Implementation ...

- Creating an Object Table
  - Now that the student_ty object type has been defined it can be used in creating an object table like the following:

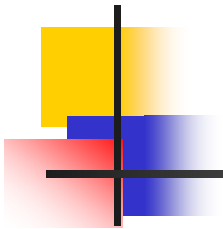
```
CREATE TABLE STUDENT
  (full_student          student_ty);
```

... - Object Type Implementation ...

- To extract data, the following query can be entered:

```
SELECT s.full_student.student_id ID,
s.full_student.person.name NAME,
s.full_student.person.address.street STREET
FROM student s
WHERE s.full_student.student_id = 100
```

```
ID         NAME                         STREET
---------- ---------------------------- -------------
100        John Q. Student              1000 Chastain Rd.
```



... - Object Type Implementation

- Updating and deleting is similar to what one would do in the relational model:

```
UPDATE STUDENT s
SET  s.full_student.person.name = 'JOHN NEWNAME'
WHERE s.full_student.student_id = 100;

DELETE FROM STUDENT s
WHERE s.full_student.student_id = 100;
```

- Implementing Methods ...

- To define a method in a type object:

```
create or replace type newperson_ty as object
(firstname    varchar2(25),
lastname      varchar2(25),
birthdate     date,
member function AGE(BirthDate in DATE) return NUMBER;
```

- Then define the method itself (PL/SQL):

```
create or replace type body newperson_ty as
member function AGE(BirthDate in DATE) return NUMBER is
begin
RETURN ROUND(SysDate - BirthDate);
end;
end;
```

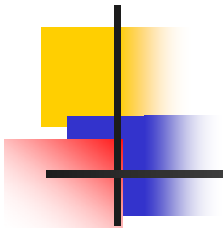
... - Implementing Methods

- To test the method first set up a table holding the person_ty object type:

```
create table NEWPERSON of newperson_ty;

insert into NEWPERSON values
(newperson_ty('JOHN', 'DOE',
 TO_DATE('03-FEB-1970', 'DD-MON-YYYY')));
```

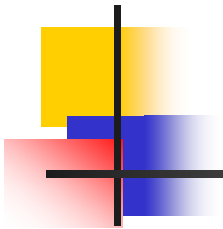
- To call the AGE function we can do the following:

```
select P.PERSON.AGE(P.PERSON.Birthdate)
from NEWPERSON P;

P.PERSON.AGE(P.PERSON.Birthdate)
----------------------------------------
                                     12005
```



- Table Type Implementation

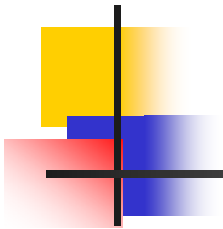
```
create type item_type as object (
itemnb              numeric(3),
productid                  integer,
qty                 numeric(3))

create type itemcollection_type as table of item_type;

create type invoice_type as object (
invoicenb integer,
total numeric(6,2),
items itemcollection_type);

Create table invoice of invoice_type;
```



- Varray Type Implementation

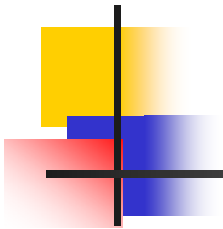
- Specifies a multiple occurrence of a certain type
- Fixed max size;

```
create type tels_type as varray(3) of varchar(30)
```



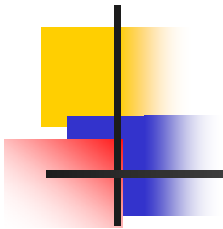
Example: INSERT ...

```
Create table item of item_type;

Insert into item values (item_type(1,10,100));

Create type invoice_type (nr integer,item item_type)

Create table invoice of invoice_type;

Insert into invoice values(1,item_type(1,10,100));
```



... Example: INSERT ...

```
create type itemcollection_type as table of item_type;

create type invoice_type as object (
invoicenb integer,
total numeric(6,2),
items itemcollection_type);

Create table invoice of invoice_type;

Insert into invoice values
(1,10,itemcollection_type(item_type(10,100,1000),item_type
(20,200,2000),item_type(30,300,3000)))
```



... Example: INSERT

Create type tel_type **as object** (numtel varchar(20));

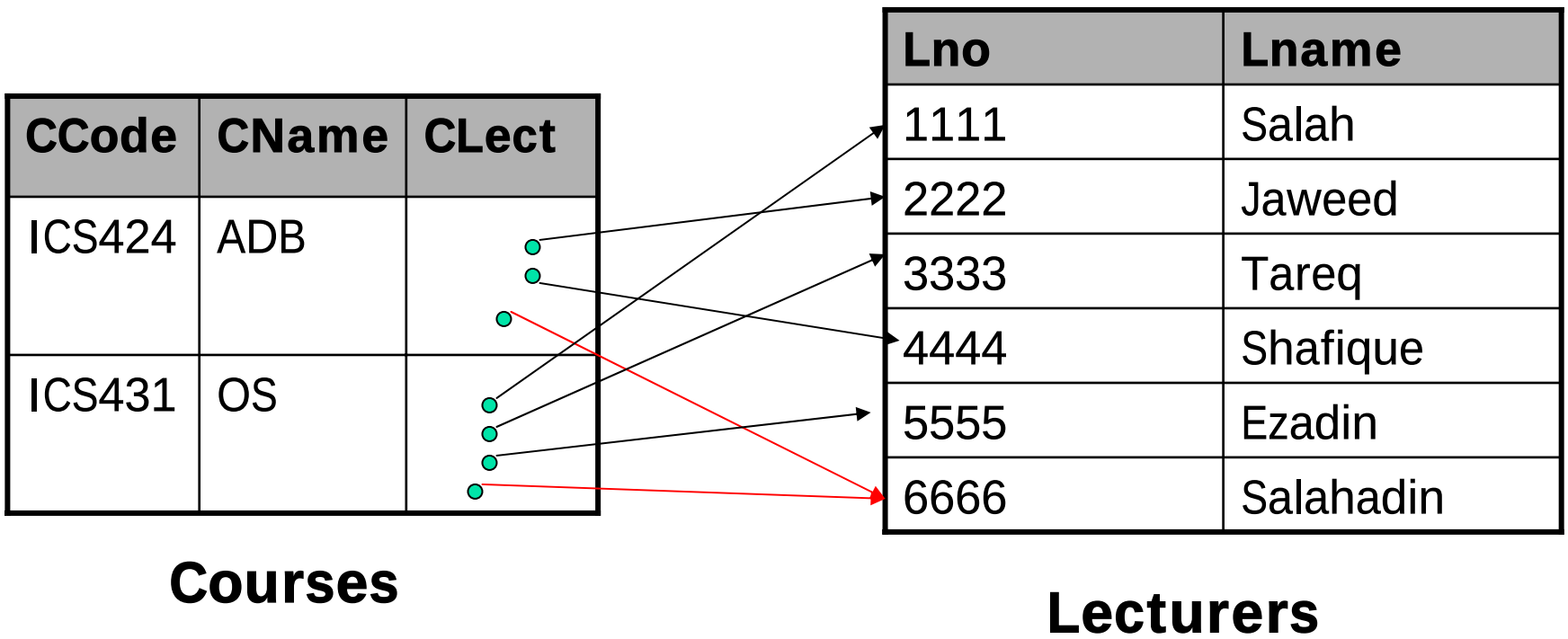
Create type tels_type **as varray**(3) of tel_type;

Create type student_type **as object** (
id integer, address address_type,tels tels_type)

Create table student of student_type;

Insert into student values (
10,
address_type(1,'rue y','lausane',1015),
tels_type(**tel_type**('2222'), **tel_type**('3333')))

- Referencing ...

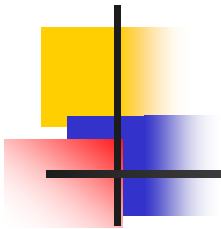


CCode	CName	CLect
ICS424	ADB	
ICS431	OS	

Courses

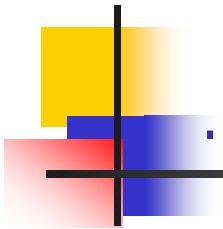
Lno	Lname
1111	Salah
2222	Jaweed
3333	Tareq
4444	Shafique
5555	Ezadin
6666	Salahadin

Lecturers



... - Referencing ...

```
create type Lecturer_ty as object (
  Lno                   Char(7),
  Lname                 Varchar(30));

create table Lecturers of Lecturer_ty;

 create table Courses (
   CCode  CHAR(6),
   CName     VARCHAR(30),
   CLect     REF Lecturer_ty);
```



... - Referencing ...

- Every row object has a unique identifier called the object identifier (OID).
- OID allows other objects to reference an existing row object.
- REF function can be used to reference an OID:

```
create table NEWDEPARTMENT
(DeptName VARCHAR(30),
PersonIn   REF NEWPERSON_TY);
```

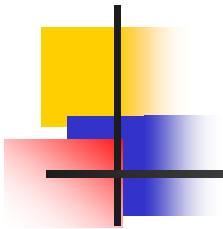
- Table `NEWDEPARTMENT` holds a reference to a `NEWPERSON_TY` object, but does not hold any real values.

... - Referencing ...

- To get a full description of the table just created:

```
Set describe depth 2
Desc NEWDEPARTMENT
```

```
Name                 Null?    Type
 ---------------- -------- --------------------
 DEPTNAME                   VARCHAR2(30)
 PERSONIN                   REF OF NEWPERSON_TY
   FIRSTNAME                VARCHAR2(25)
   LASTNAME                 VARCHAR2(25)
   BIRTHDATE                DATE
```



... - Referencing ...

- To insert a record into NEWDEPARTMENT, the REF is needed to store the NEWPERSON reference in the PersonIn column:

```
insert into NEWDEPARTMENT
select 'Research',
REF(P)
from NEWPERSON P
where LastName = 'DOE';
```

- The literal value “Research” is inserted into the NEWPERSON table.
- The REF function returns the OID from the query on the selected NEWPERSON object.
- The OID is now stored as a pointer to the row object in the NEWPERSON object table.

... - Referencing ...

- The referenced value cannot be seen unless the DREF function is used. The DREF function takes the OID and evaluates the reference to return a value.

```
select DEREF(D.PersonIn)
from NEWDEPARTMENT D
where DEPTNAME = 'Research'

DEREF(D.PERSONIN)(FIRSTNAME, LASTNAME, BIRTHDATE)
-------------------------------------------------
NEWPERSON_TY('JOHN', 'DOE', '03-FEB-70')
```

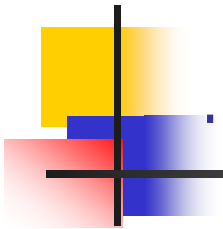
- This shows that the NEWPERSON record JOHN DOE is referenced by the Research record in NEWDEPARTMENT.

... - Referencing ...

- To gather the same structure of the object type of an object table the VALUE function is required.

```
select value(p)
from newperson p
where lastname = 'DOE'
```

```
VALUE(P)(FIRSTNAME, LASTNAME, BIRTHDATE)
----------------------------------------
NEWPERSON_TY('JOHN', 'DOE', '03-FEB-70')
```



... - Referencing

PL/SQL Sample:

```
set serveroutput on
declare
   v_person      NEWPERSON_TY;
begin
    select value(p) into v_person
    from NEWPERSON p
    where lastname = 'DOE';
    DBMS_OUTPUT.PUT_LINE(v_person.firstname);
    DBMS_OUTPUT.PUT_LINE(v_person.lastname);
    DBMS_OUTPUT.PUT_LINE(v_person.birthdate);
end;

JOHN
DOE
03-FEB-70
```

- Inheritance

- Create a root type of an object hierarchy:

```
create type PERSON_TY as object
(name          varchar2(25),
 birthdate     date
 member function AGE() return number,
 member function PRINTME() return varchar2) not final;
```

- To create a subtype the following syntax can be used:

```
create type EMPLOYEE_TY under PERSON_TY (
salary number,
member function WAGES() return number,
overriding member function PRINTME() return varchar2);
```

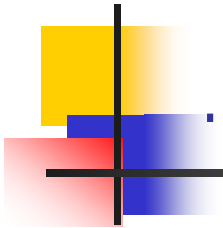
References ...

- Burleson, Donald (2001). *The object/relational features of Oracle.* TechRepublic, Inc. Retrieved November 2, 2002, from http://www.dba-oracle.com/art_oracle_obj.htm

- Cáceres P., Cavero J., Marcos E., & Vela B. (2002) Aggregation and Composition in Object – Relational Database Design. Kybele Research Group - Rey Juan Carlos University. Retrieved November 4, 2002, from http://www.science.mii.lt/ADBIS/local1/marcos.pdf

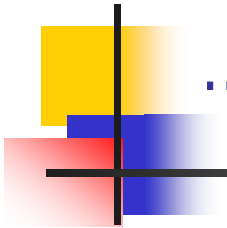
- Donaldson, John (2001). *Nested tables and Object Tables.* Retrieved November 4, 2002, from http://cs.oberlin.edu/faculty/jdonalds/311/lecture27.html

- Hanson, Robert (2002). *Object-Relational Databases in Oracle 8i*. Retrieved November 12, 2002, from http://www.technology.niagarac.on.ca/courses/comp708/roboo.htm



... References ...

- Koch, G. & Loney, K. (2002). Oracle 9i: The Complete Reference. : McGraw-Hill.
- Lassen, A. & Olsson J. (1999, February) *Experiences from Object-relational Programming in Oracle8*. Center for Object Technology. Retrieved November 4, 2002, from http://www.cit.dk/COT/reports/reports/Case4/06-v1.4/cot-4-06-1.4.pdf
- Lindstrom, Gary (2002, September). *Lecture 14: Oracle Extended Relational Features.* Retrieved November 2, 2002, from http://www.cs.utah.edu/classes/cs5530/lectures/lecture14x2.pdf
- Oracle Corporation (*1997, June*). *Oracle8™ Object Relational Database: An Overview*. Retrieved November 4, 2002, from http://technet.oracle.com/products/oracle8/info/objwp3/xoo3twp.htm



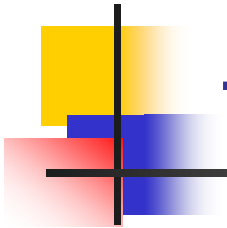
... References ...

- Oracle Corporation (1997). *Oracle8 Concepts, Release 8.0*. Retrieved November 2, 2002, from

http://storacle.princeton.edu:9001/oracle8-doc/server.805/a58227/ch_ordb.htm

- Oracle Corporation (2002, May). *Simple Strategies for Complex Data: Oracle9i Object-Relational Technology*. Retrieved November 4, 2002, from http://otn.oracle.com/products/oracle9i/pdf/simple_strat_for_complex_rel2.pdf

- Oracle Corporation (2002). *Oracle9i Object-Relational Technology Feature Overview* . Retrieved November 10, 2002, from http://otn.oracle.com/products/oracle9i/htdocs/ort_twp.html

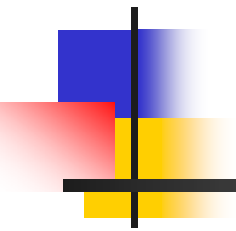


... References

- Oracle Corporation (2002). *Basic Components of Oracle Objects* . Oracle9*i* Application Developer's Guide - Object-Relational Features Release 2 (9.2). Retrieved November 4, 2002, from http://csis.gvsu.edu/GeneralInfo/Oracle/appdev.920/a96594/adobjbas.htm

- Oracle Corporation (2002). *Introduction to Oracle Objects*. Oracle9i Application Developer's Guide - Object-Relational Features Release 2 (9.2). Retrieved November 6, 2002, from http://otn.oracle.com/products/oracle9i/htdocs/ort_twp.html

- Robbert, Mary Ann (2002, April). *Oracle Objects*. Retrieved November 2, 2002, from http://cis.bentley.edu/mrobbert/CS652/Oracleobj.ppt



END