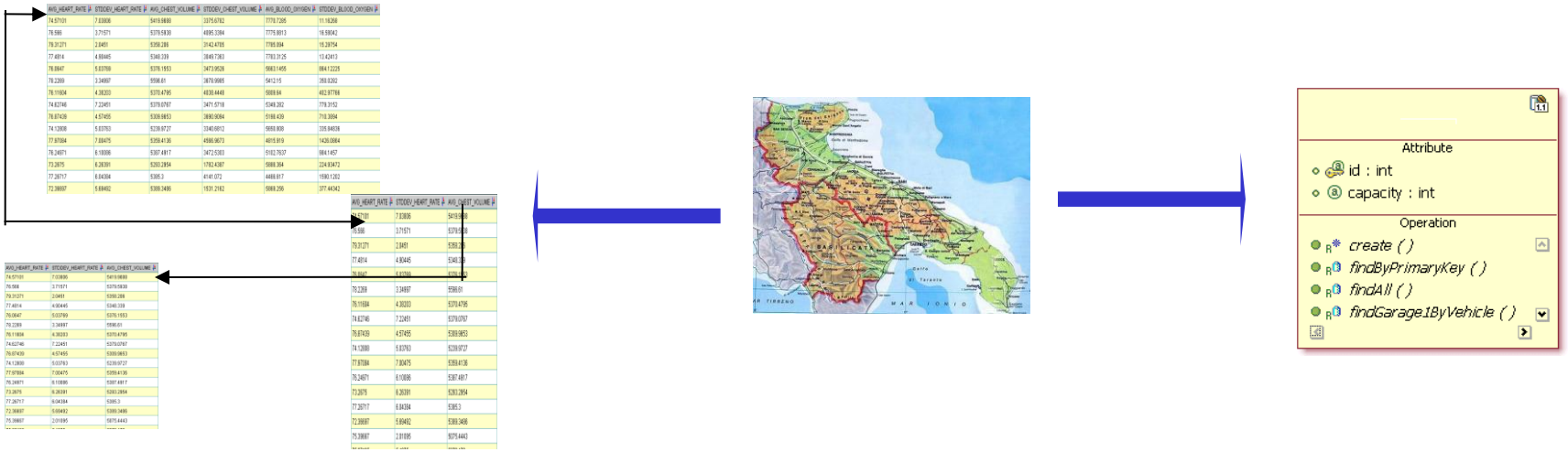
Exercise 2: Object-relational data model in Oracle

Why object-relational model?

There are contexts (maps, images) where the use of relational model seems too restrictive due to the necessary complex representation.

With the object-relational approach, objects can be represented easily with structured data types, also associating methods to them.



The abstract data types in Oracle ...

- user-defined data types;
- usually allow grouping columns or related fields as a object;
- the syntax is:

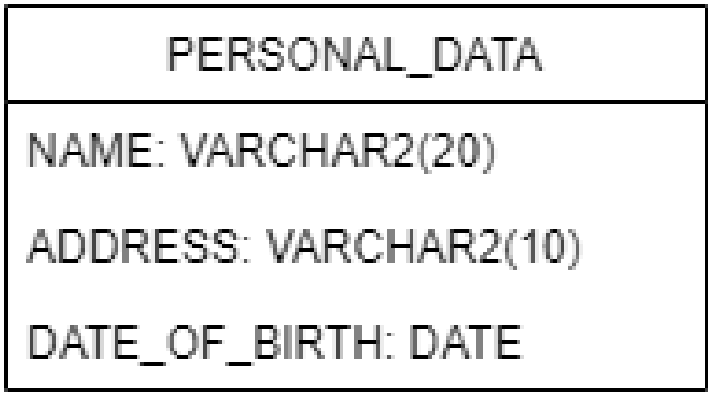
```
CREATE TYPE typeName AS OBJECT
(
column1 TypeCol1,
column2 TypeCol2,
.
.
columnK TypeColK
);
```

...The abstract data types in Oracle ...

For example, the columns NAME, ADDRESS, DATE_OF_BIRTH of the table CUSTOMER

NAME	ADDRESS	DATE_OF_BIRTH

can be grouped together to define the abstract data type PERSONAL_DATA with the specification of its attributes



...The abstract data types in Oracle ...

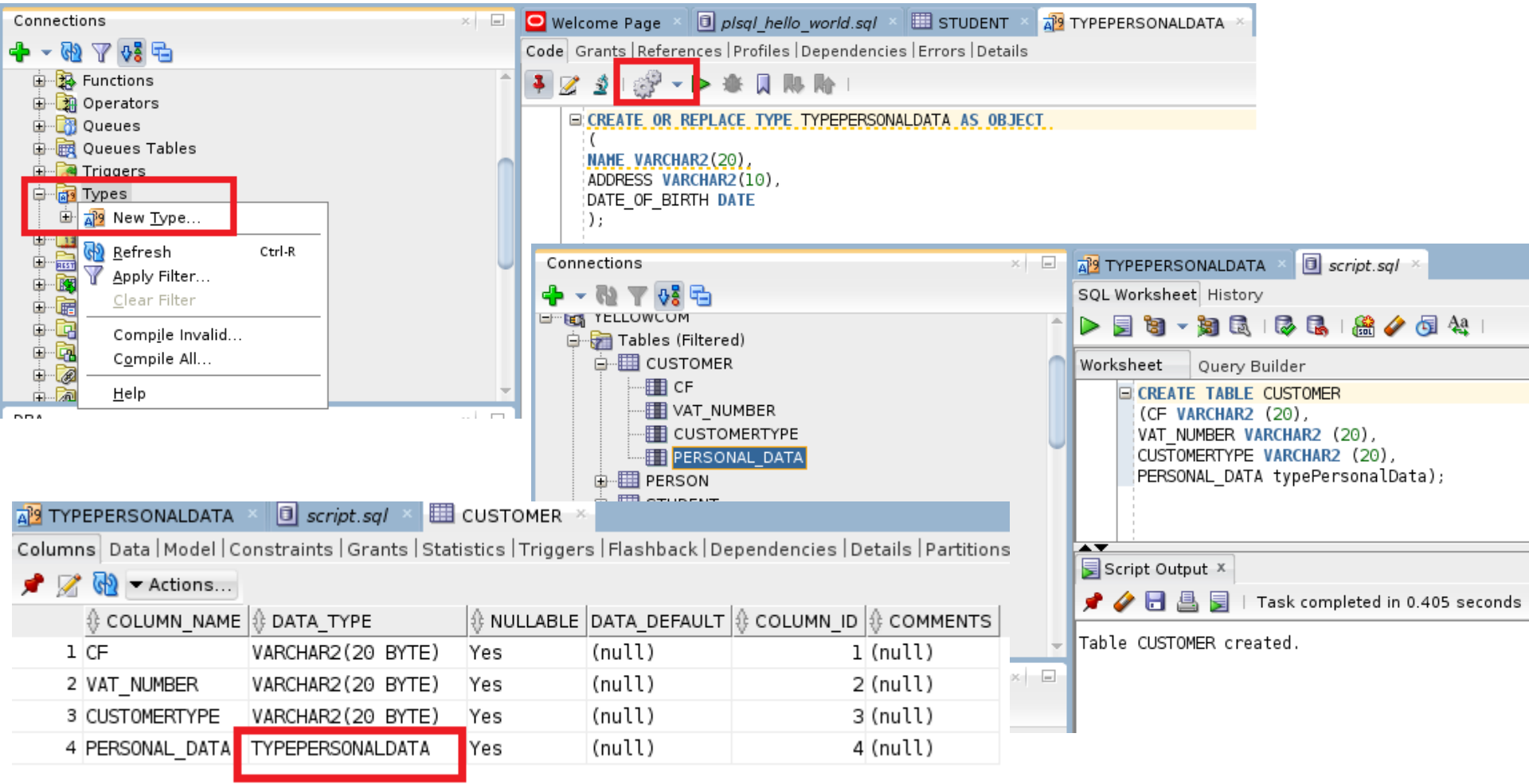
```
CREATE TYPE typePersonalData AS OBJECT
( NAME VARCHAR2(20),
ADDRESS VARCHAR2(10),
DATE_OF_BIRTH DATE);
```

and the CUSTOMER table may be amended as follows:

```
CREATE TABLE CUSTOMER
(CF VARCHAR2 (20),
VAT_NUMBER VARCHAR2 (20),
CUSTOMERTYPE VARCHAR2 (20),
PERSONAL_DATA typePersonalData);
```

...The abstract data types in Oracle ...

The abstract data types can be created or modified via Oracle graphics tool (SQL Developer)



...The abstract data types in Oracle ...

Once you have created an abstract data type and a table defined also on the type created, any operation to be performed on that table will consider the new structuring of the table

INSERT INTO CUSTOMER values (*'SSN', 'VAT identification number', 'Typecustomer', typePersonalData ('Name', 'Address', '10-DEC-95')*)

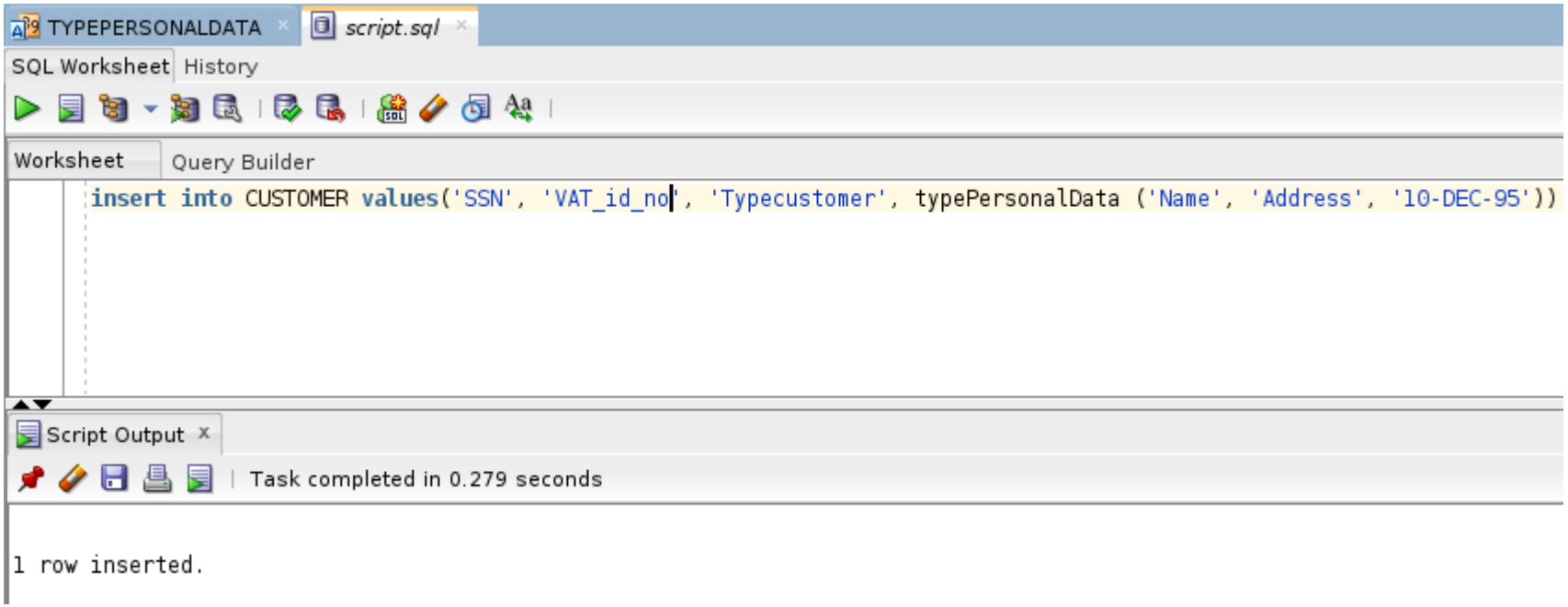
type constructor

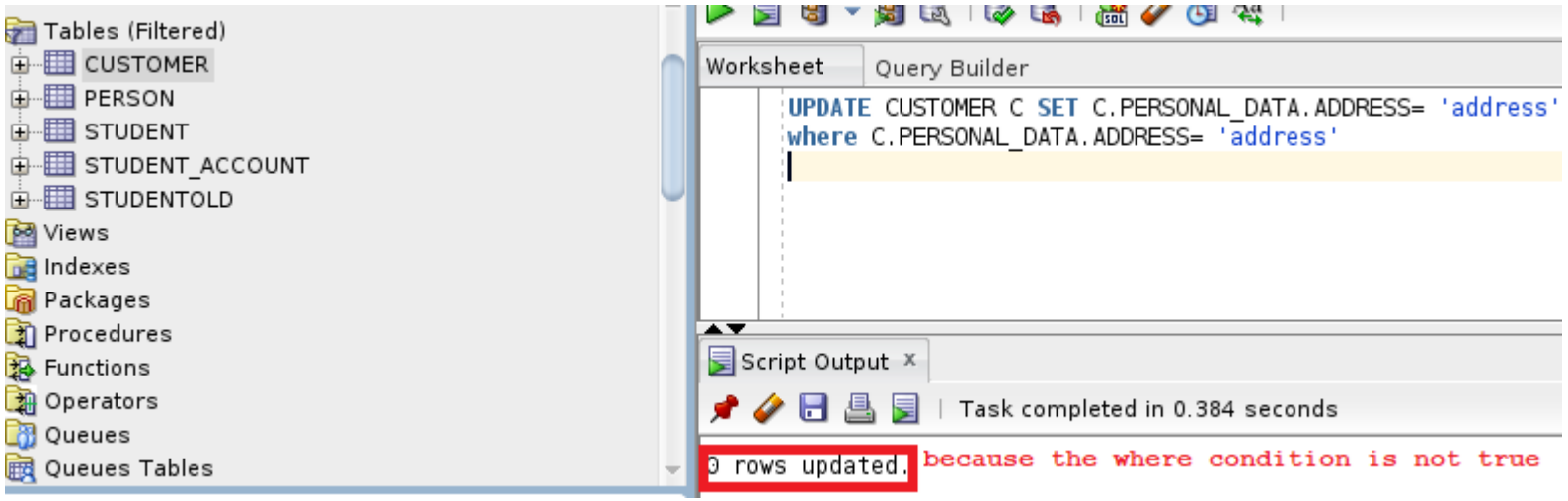
dot notation to refer to the column in question

name of the abstract type column

table alias

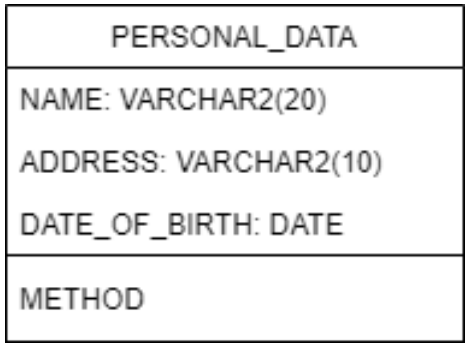
UPDATE CUSTOMER C SET C.*PERSONAL_DATA*.ADDRESS= '*address*'
where C.*PERSONAL_DATA*.ADDRESS= '*Address*'


TYPEPERSONALDATA
script.sql
SQL Worksheet
History
Worksheet
Query Builder
insert into CUSTOMER values('SSN', 'VAT_id_no', 'Typecustomer', typePersonalData ('Name', 'Address', '10-DEC-95'))
Script Output
Task completed in 0.279 seconds
1 row inserted.


Tables (Filtered)
CUSTOMER
PERSON
STUDENT
STUDENT_ACCOUNT
STUDENTOLD
Views
Indexes
Packages
Procedures
Functions
Operators
Queues
Queues Tables
Worksheet
Query Builder
UPDATE CUSTOMER C SET C.PERSONAL_DATA.ADDRESS= 'address'
where C.PERSONAL_DATA.ADDRESS= 'address'
Script Output
Task completed in 0.384 seconds
0 rows updated.
because the where condition is not true

...The abstract data types in Oracle ...

In the statement of an abstract data type, we can also declare methods (only their signature) operating on the type attributes



The syntax is:

CREATE TYPE *typeName* AS OBJECT
(...
columnK typeColumnK,
...
MEMBER FUNCTION
methodName
(*typeParameter* IN *parameterName*)
RETURN returnType
);

CREATE TYPE *typeName* AS OBJECT
(...
columnK typeColumnK,
...
MEMBER PROCEDURES
methodName
(*typeParameter* IN / OUT *parameterName*)
);

used parameter (IN) returned (OUT) by the method

...The abstract data types in Oracle ...

To declare the method implementation

```
CREATE TYPE BODY typeName IS
MEMBER FUNCTION methodName (ParameterType parameterName)
RETURN returnType IS
 BEGIN
....
    END;

END;
```

...The abstract data types in Oracle ...

Continuing the above example

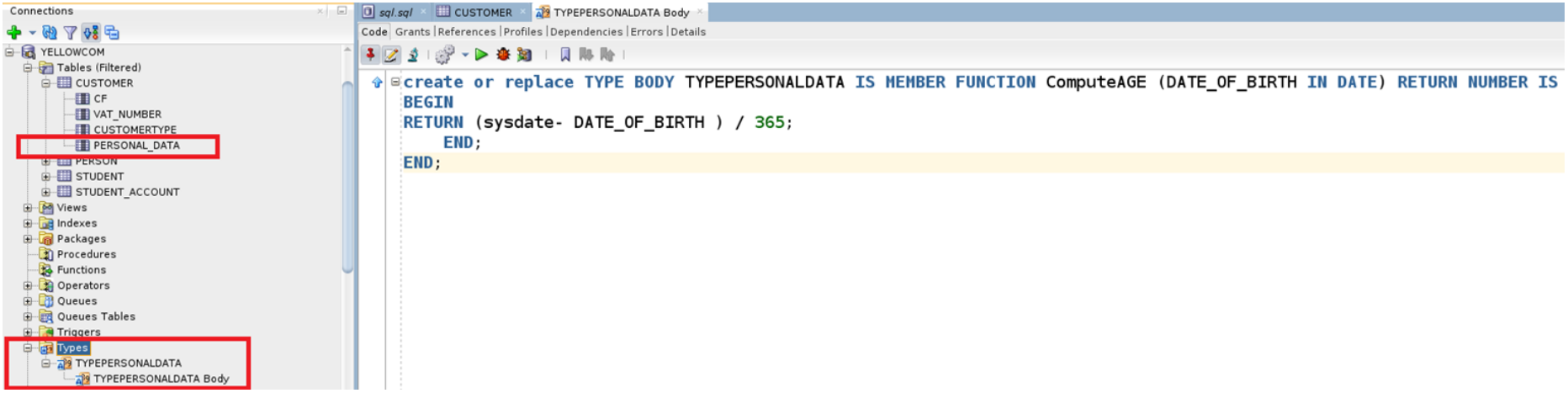
CREATE TYPE *typePersonalData* AS OBJECT

(NAME VARCHAR2(20),
ADDRESS VARCHAR2(10),
DATE_OF_BIRTH DATE, MEMBER FUNCTION ComputeAGE
(*DATE_OF_BIRTH* IN DATE) RETURN NUMBER);

CREATE TYPE BODY typePersonalData IS
MEMBER FUNCTION ComputeAGE (DATE_OF_BIRTH IN DATE) RETURN NUMBER IS

BEGIN
RETURN (sysdate - DATE_OF_BIRTH) / 365;
END;
END;

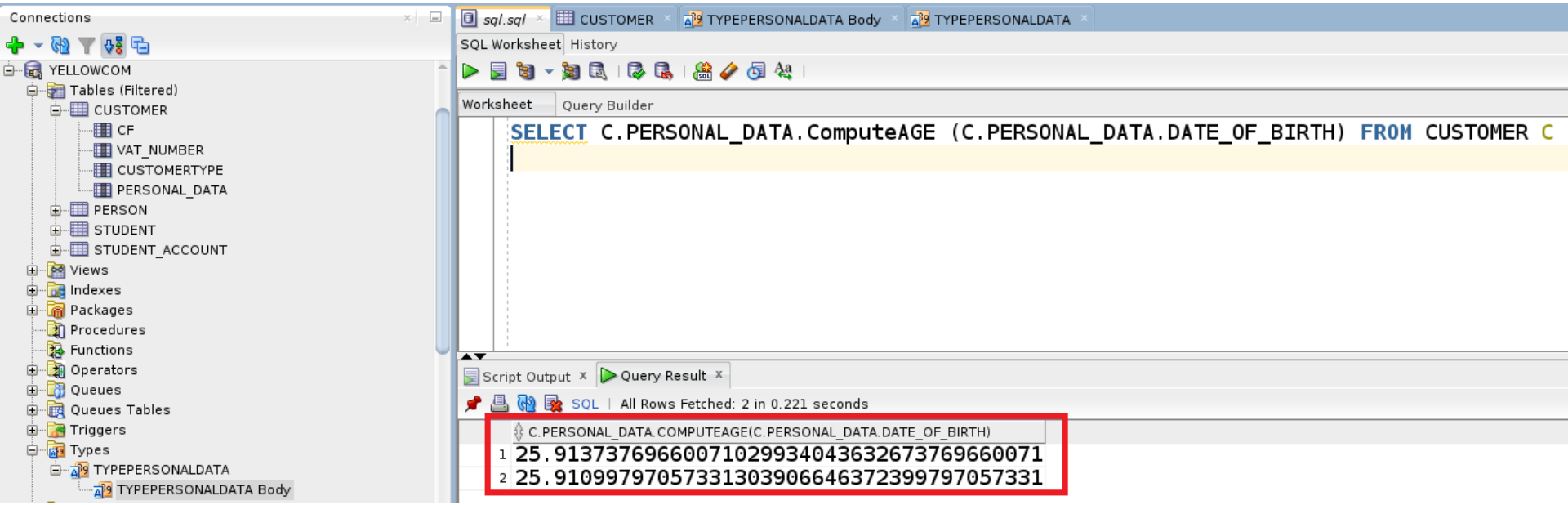
...The abstract data types in Oracle ...



...The abstract data types in Oracle

We can call the declared method:

SELECT C.personalData.ComputeAGE
(C.personalData.*DATE_OF_BIRTH*) FROM CUSTOMER C



SELECT **ROUND**(C.personalData.ComputeAGE
(C.personalData.*DATE_OF_BIRTH*)**, 2)** FROM CUSTOMER C

→25.91
→25.91

VARRAY...

• Allow us to create aggregations (multivalued) of the same type in the same column objects;

• The syntax that lets us create the aggregation is:

CREATE TYPE *nameVARRAY*
AS VARRAY (*size max*) OF *ObjectType*

• The syntax for defining a column of aggregation type of objects is:

CREATE TABLE *tableName*
(...
columnname *nameVARRAY*,
....)

... VARRAY...

For example, given the state of CUSTOMER table

Code	DoB	Address	Name	Tel. No	VAT No	Type

it is possible to add a VARRAY *arrayPHONE*

CREATE TYPE *arrayPHONE* AS VARRAY(*10*) OF NUMBER

and recreate the table

```
CREATE TABLE CUSTOMER
(...
telephoneNumber arrayPHONE;
...)
```

... VARRAY

Thus, any operation can take advantage of the new structure

insert into *Customer* ('StringFiscale', '31-Dec-05', 'stringInd', 'stringNome', *arrayPHONE*(080544,3280000), 1234, 'stringTipo')

constructor

component objects of the VARRAY

Select C.telephoneNumber from customer C

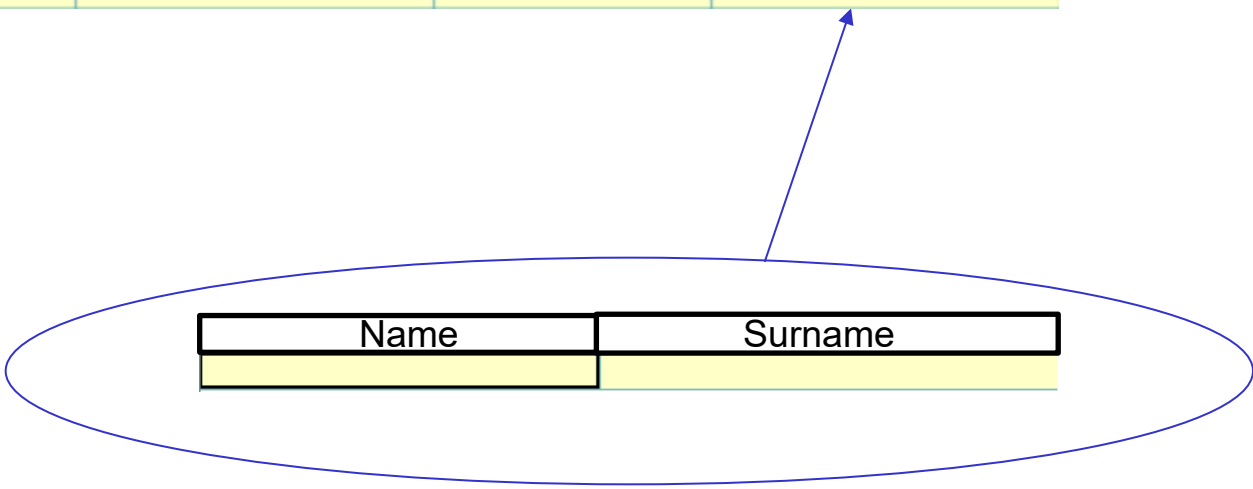
name of the column of type variable array

Nested Tables...

- They make possible multiple relationships (1-N) within a table;

- A nested table is represented by a column in another table

Code	Total Cost	Date	No Participants	Event Name	Guest List



...Nested Tables...

• The syntax for creating a data type on which a nested table can be based is:

CREATE TYPE *NESTED_TABLE_BASE*
AS TABLE OF *type_nested_table_objects*

• The syntax for creating a table that contains a column that is a nested table is

CREATE TABLE *table*
(...
nomeNestedColumn NESTED_TABLE_BASE,
...)
NESTED TABLE *nomeNestedColumn* STORE AS *nestedTableNT*

• The type *type_nested_table_objects* can also be an abstract type

...Nested Tables...

For example, given the type abstract *typeSpouse*

CREATE TYPE *typeSpouse* AS OBJECT
(Name VARCHAR (50), weddingDate DATE)

We create the type of data on which to base the nested table

CREATE TYPE *baseSpouse* AS OF TABLE *typeSpouse*

below we create the table that will contain a column of type nested table

CREATE TABLE Customers1
(*Name* VARCHAR (50), *date_of_birth* DATE, *place_of_birth* VARCHAR (50), *date_of_death* DATE, *place_of_death* VARCHAR (50), *listSpouses* baseSpouse)
NESTED TABLE *listSpouses* STORE AS *baseSpouse_TAB*

...Nested Tables...

To insert customers, we will use the consructors of the new data types, in our example we have:

```
INSERT INTO VALUES Customers1
( 'Paolo Rossi', '1-Jan-1962', 'Milano', null, null,
baseSpouse (
typeSpouse ( 'Paola Bianchi', '12-May-1980'),
typeSpouse ( 'Franca Verdi', '27-Jun-1997')
           )
)
```

constructor of the abstract data type used by the base type

...Nested Tables.

To refer to the columns contained in the nested table, we must use the flattening operator (TABLE)

```
SELECT ES.Name FROM TABLE (SELECT listSpouses FROM Customers1
                  WHERE Birthdate < DATE '2006-06-01') ES;
```

provided that identifies which of nested table consider from the main table

```
INSERT INTO TABLE (select listSpouses from Customers1 where
name = 'Paolo Rossi') VALUES (typeSpouse ( 'Liliana Blue', '12 Oct 2002'))
```

If the query retrieves more than one instance, the DBMS raises an exception

ORA-01427: single-row subquery returns more than one row

```
SELECT ES.Name FROM Customers1 C,
TABLE(C.listSpouses) ES
WHERE C.Birthdate < DATE '2006-06-01';
```

In this case, the **TABLE(C.listSpouses)** takes the listSpouses collection of each customer C and expands it into rows.
This query performs a classical implicit join between the upper table and the nested one

Using CAST and MULTISET

Remember that: if you try to directly enter the values of the columns of the nested table with an INSERT command in the main table, you get an error because the main table contains one column for each nested table.

To get the right result we have to use the operators CAST MULTISET in the form:

CAST (MULTISET (<subquery>) AS <TipoNT>)

Exercises

Let us consider the case study information system of a mobile telephone operator.

- Create Abstract Types
- Create the corresponding tables
- Use nested tables.