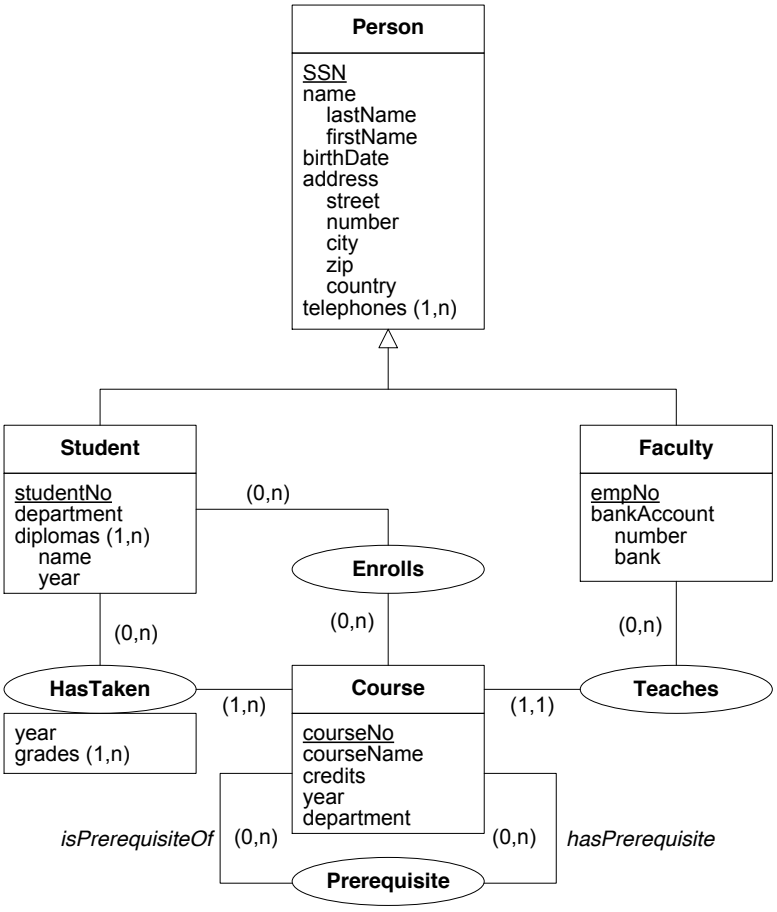
# Object-Relational Databases
# Exercices

## Application Modeling

Translate the entity-relationship schema below into an object-relational schema in Oracle 11g. It is supposed that in the application for each relationship type there are queries that traverse each role in both directions.




## Updating the Database

1. Insert a new person with SSN 123456789, with name Michelle Pfeiffer, born on 1/1/1980 and living in 22 rue de la Gare, 1000 Bruxelles, Belgique, and telephones 12345 and 54321.
2. Insert a new student with SSN 123123123, with name Brad Pitt, born on 1/5/1985, living in 123 avenue Huysmans, 1050 Ixelles, Belgique, no telephones, student number 999, department of Computer Science, with no diplomas, and not courses taken or enrolled.
3. Insert a new student with SSN 111222333, with name Jeremy Irons, born on 12/12/1965, living in 1 rond Point de l'Etoile, 1050 Ixelles, Belgique, no telephones, student number 888, department of Computer Science, with no diplomas, and not courses taken or enrolled.

4. Insert a new student with SSN 333222111, with name Scarlett Johansson, born on 22/11/1984, living in 1 rond Point de l'Etoile, 1050 Ixelles, Belgique, no telephones, student number 777, department of Computer Science, with no diplomas, and not courses taken or enrolled.
5. Insert a new faculty with SSN 666999666, with name Angelina Jolie, born on 4/6/1975, living in 99 avenue de l'Université, 1050 Ixelles, Belgique, no telephones, employee number 777, bank account 310 1234567 89 at ING, and teaching no courses.
6. Add a new faculty with SSN 987987987, with name Clint Eastwood, born on 31/5/1930, living 3 place de la Petite Suisse, 1050 Ixelles, telephone number 2222, employee number 666, bank account 33333333 at ING, and teaching no courses.
7. Add telephones 2222 and 3333 to the student Brad Pitt.
8. Set the telephones of Angelina Jolie as those of Brad Pitt.
9. Add a new course identified by "INFO-H-415", entitled "Advanced Databases" of 5 credits, taught in the fourth year in the Department of Computer Science, having no prerequisites and being the prequisite of no course, having as professor the newly inserted professor Eastwood, and for the moment no students enrolled or that took the course in the pasts.
10. Add a new course identified by "INFO-H-888", entitled "Advanced Data Warehouses" of 5 credits, taught in the fifth year in the Department of Computer Science, having as prerequisite the course "Advanced Databases", as professor Angelina Jolie, and for the moment no students enrolled or that took the course in the pasts.
11. Add a new course identified by "INFO-H-999", entitled "Data Management in the Cloud" of 5 credits, taught in the fifth year in the Department of Computer Science, having as prerequisite the course "Advanced Databases", as professor Angelina Jolie, and for the moment no students enrolled or that took the course in the pasts.
12. Add the course identified by "INFO-H-415" as being taken by the student Brad Pitt in 2010 with grades 18.5 and 19.
13. Add the course identified by "INFO-H-888" to the courses to which the student Brad Pitt is enrolled.
14. Add the course identified by "INFO-H-999" to the courses to which the student Brad Pitt is enrolled.
15. Add the courses identified by "INFO-H-888" and "INFO-H-999" to the courses to which the student Jeremy Irons is enrolled.
16. Add the course identified by "INFO-H-888" to the courses to which the student Scarlett Johansson is enrolled.

## Querying the Database

1. Give the names of the persons.
2. Give the names and student number of the students.
3. Give the names of the persons who are not students nor faculty.
4. Find the name and address of the person whose SSN is 123456789.
5. Find the couples of SSN of persons having the same name.
6. Find the value of the person Michelle Pfeiffer.
7. Give the telephone numbers of the student Brad Pitt
8. Give the name of persons having the telephone number '2222'.
9. Give the name of persons having no telephone number.
10. Find the couples of names of persons sharing at least one telephone number.

11. Find the couples of SSN of persons having exactly the same set of telephones numbers.
12. Give the numbers and names of students who have successfully completed the course "Advanced Databases".
13. Give the prerequisites of the course "Advanced Data Warehouses".
14. Search inconsistencies in prerequisites: Courses that have as prerequisite a course of a subsequent year. Give the name of the course and the prerequisite.
15. Search inconsistencies in prerequisites: Students enrolled in courses that have as prerequisite a course they have not taken. Give the name of the student, the name of the enrolled course and the name of the missing prerequisite.
16. Give the couples of names of a student and one of his teachers.
17. Give the course number of the courses in which Brad Pitt is enrolled.
18. Names of students who are enrolled in at least one course in which Brad Pitt is enrolled.
19. Names of students who are enrolled in (at least) all courses in which Brad Pitt is enrolled.

# Object-Relational Databases
# Answers to Exercices

## Application Modeling

```
/* Creation of Types */
CREATE TYPE Tname AS OBJECT (firstName VARCHAR2(30), lastName VARCHAR2(30));
CREATE TYPE Taddress AS OBJECT (street VARCHAR2(30), streetNumber CHAR(5),
    city VARCHAR2(20), zip CHAR(4), country VARCHAR2(20));
CREATE TYPE TTelephones AS VARRAY(6) OF VARCHAR2(15);
CREATE TYPE TPerson AS OBJECT (SSN CHAR(9), name Tname, birthDate Date,
    address Taddress, telephones TTelephones) NOT FINAL;
CREATE TYPE TCourse;
CREATE TYPE TCourseTaken;
CREATE TYPE TFaculty;
CREATE TYPE Tdiploma AS OBJECT (name VARCHAR2(15), year CHAR(4));
CREATE TYPE Tdiplomas AS VARRAY(10) OF Tdiploma;
CREATE TYPE TSetRefCourses AS TABLE OF REF TCourse;
CREATE TYPE TSetRefCourseTaken AS TABLE OF REF TCourseTaken;
CREATE TYPE TStudent UNDER TPerson (studentNo CHAR(6), department VARCHAR2(20),
    diplomas Tdiplomas, courseEnrolledRefs TSetRefCourses,
    courseTakenRefs TSetRefCourseTaken);
CREATE TYPE TSetRefStudents AS TABLE OF REF TStudent;
CREATE OR REPLACE TYPE TCourse AS OBJECT (courseNo CHAR(10), courseName VARCHAR2(40),
    credits INTEGER, year INTEGER, department VARCHAR2(20),
    isPrerequisiteOfRefs TSetRefCourses, hasPrerequisiteRefs TSetRefCourses,
    courseTakenRefs TSetRefCourseTaken, enrolledStudentRefs TSetRefStudents,
     professorRef REF TFaculty);
CREATE TYPE TGrades AS VARRAY(10) OF DECIMAL(3,1);
CREATE OR REPLACE TYPE TCourseTaken AS OBJECT (grades Tgrades,
    year CHAR(4), studentRef REF TStudent, courseRef REF TCourse);
CREATE TYPE TbankAccount AS OBJECT (accountNumber VARCHAR2(20), bank VARCHAR2(20))
CREATE OR REPLACE TYPE TFaculty UNDER TPerson (empNo CHAR(10),
    bankAccount TbankAccount, coursesTaughtRefs TSetRefCourses);

/* Creation of Tables */

CREATE TABLE Person OF TPerson;
CREATE TABLE Course OF TCourse (
    PRIMARY KEY (courseNo),
    FOREIGN KEY (professorRef) REFERENCES Person )
    NESTED TABLE isPrerequisiteOfRefs STORE AS Table_isPrerequisiteOfRefs,
    NESTED TABLE hasPrerequisiteRefs STORE AS Table_hasPrerequisiteRefs,
    NESTED TABLE courseTakenRefs STORE AS Table_courseTakenRefs,
    NESTED TABLE enrolledStudentRefs STORE AS Table_enrolledStudentRefs;
CREATE TABLE CourseTaken OF TCourseTaken (
    FOREIGN KEY (studentRef) REFERENCES Person,
    FOREIGN KEY (courseRef) REFERENCES Course);
```

## Updating the Database

1. Insert a new person with SSN 123456789, with name Michelle Pfeiffer, born on 1/1/1980 and living in 22 rue de la Gare, 1000 Bruxelles, Belgique, and telephones 12345 and 54321.

   **You need to use to_date() function.**

```
INSERT INTO Person VALUES (
    TPerson('123456789', Tname('Michelle', 'Pfeiffer'), '1/1/1980',
    TAddress('rue de la gare', '22', 'Bruxelles', '1000', 'Belgique'),
    TTelephones ('12345', '54321') ) );
```

2. Insert a new student with SSN 123123123, with name Brad Pitt, born on 1/5/1985, living in 123 avenue Huysmans, 1050 Ixelles, Belgique, no telephones, student number 999, department of Computer Science, with no diplomas, and not courses taken or enrolled.

```
INSERT INTO Person VALUES (
    TStudent('123123123', Tname('Brad', 'Pitt'), '1/5/1985',
    Taddress('avenue Huysmans', '123', 'Ixelles', '1050', 'Belgique'), TTelephones(),
    '999', 'Computer Science', Tdiplomas(), TSetRefCourses(), TSetRefCourseTaken() ) ) ;
```

3. Insert a new student with SSN 111222333, with name Jeremy Irons, born on 12/12/1965, living in 1 rond Point de l'Etoile, 1050 Ixelles, Belgique, no telephones, student number 888, department of Computer Science, with no diplomas, and not courses taken or enrolled.

```
INSERT INTO Person VALUES (
    TStudent('111222333', Tname('Jeremy', 'Irons'),  '1/5/1985',
    Taddress('rond Point de l''Etoile', '1', 'Ixelles', '1050', 'Belgique'), TTelephones(),
    '888', 'Computer Science', Tdiplomas(), TSetRefCourses(), TSetRefCourseTaken() ) ) ;
```

4. Insert a new student with SSN 333222111, with name Scarlett Johansson, born on 22/11/1984, living in 1 rond Point de l'Etoile, 1050 Ixelles, Belgique, no telephones, student number 777, department of Computer Science, with no diplomas, and not courses taken or enrolled.

```
INSERT INTO Person VALUES (
    TStudent('333222111', Tname('Scarlett', 'Johansson'), '22/11/1984',
    Taddress('rond Point de l''Etoile', '1', 'Ixelles', '1050', 'Belgique'), TTelephones(),
    '777', 'Computer Science', Tdiplomas(), TSetRefCourses(), TSetRefCourseTaken() ) ) ;
```

5. Insert a new faculty with SSN 666999666, with name Angelina Jolie, born on 4/6/1975, living in 99 avenue de l'Université, 1050 Ixelles, Belgique, no telephones, employee number 777, bank account 310 1234567 89 at ING, and teaching no courses.

```
INSERT INTO Person VALUES (
    TFaculty('666999666', Tname('Angelina', 'Jolie'), '4/6/1975',
    Taddress('avenue de l''Université', '99', 'Ixelles', '1050', 'Belgique'),
        TTelephones(), '777', TbankAccount('310123456789', 'ING'), TSetRefCourses() ) ) ;
```

6. Add a new faculty with SSN 987987987, with name Clint Eastwood, born on 31/5/1930, living 3 place de la Petite Suisse, 1050 Ixelles, telephone number 2222, employee number 666, bank account 33333333 at ING, and teaching no courses.

```
INSERT INTO Person VALUES (
TFaculty('987987987', Tname('Clint', 'Eastwood'), '31/5/1930',
    TAddress('place de la Petite Suisse', '3',  'Ixelles', '1050', 'Belgique'),
        TTelephones('2222'), '666', TBankAccount('33333333', 'ING'), TSetRefCourses() ) ) ;
```

Here we must use the format for insertion of typed object values as an object is inserted into a table that can contain objects of different types.

7. Add telephones 2222 and 3333 to the student Brad Pitt.

```
UPDATE Person
Set telephones = TTelephones('2222','3333')
WHERE name = Tname('Brad', 'Pitt')
```

8. Set the telephones of Angelina Jolie as those of Brad Pitt.

```
UPDATE Person
Set telephones =( select telephones from Person where name = Tname('Brad', 'Pitt'))
WHERE name = Tname('Angelina', 'Jolie')
```

9. Add a new course identified by "INFO-H-415", entitled "Advanced Databases" of 5 credits, taught in the fourth year in the Department of Computer Science, having no prerequisites and being the prequisite of no course, having as professor the newly inserted professor Eastwood, and for the moment no students enrolled or that took the course in the pasts.

```
INSERT INTO Course
VALUES ('INFO-H-415', 'Advanced Databases', 5, 4, 'Computer Science',
    TSetRefCourses(), TSetRefCourses(), TSetRefCourseTaken(), TSetRefStudents(),
    (SELECT TREAT(REF(f) AS REF TFaculty) FROM Person f WHERE f.name.lastName='Eastwood') );
```

Update the reverse `REF` attribute: Put in the attribute `coursesTaught` of professor Eastwood the identifier of this new course. This attribute `oursesTaught` is a collection of type nested table. We must therefore make an insertion into the nested table as follows:

```
INSERT INTO TABLE (SELECT TREAT(VALUE(f) AS TFaculty).coursesTaughtRefs
     FROM Person f WHERE f.name.lastName = 'Eastwood')
(SELECT REF(c) FROM Course c WHERE c.courseName = 'Advanced Databases') ;
```

In this statement, the expression `TABLE (SELECT TREAT ...)` defines the nested table in which to make the insertion, and the expression `(SELECT REF(c) FROM ...)` defines the identifier to be inserted.

10. Add a new course identified by "INFO-H-888", entitled "Advanced Data Warehouses" of 5 credits, taught in the fifth year in the Department of Computer Science, having as prerequisite the course "Advanced Databases", as professor Angelina Jolie, and for the moment no students enrolled or that took the course in the pasts.

```
INSERT INTO Course
VALUES ('INFO-H-888', 'Advanced Data Warehouses', 5, 5, 'Computer Science',
    TSetRefCourses(),
    TSetRefCourses((SELECT REF(c) FROM Course c WHERE c.CourseName='Advanced Databases')),
    TSetRefCourseTaken(), TSetRefStudents(),
    (SELECT TREAT(REF(f) AS REF TFaculty) FROM Person f WHERE f.name.lastName='Jolie') ) ;
```

Update the reverse `REF` attributes.

```
INSERT INTO TABLE (SELECT TREAT(VALUE(f) AS TFaculty).coursesTaughtRefs
     FROM Person f WHERE f.name.lastName = 'Jolie')
(SELECT REF(c) FROM Course c WHERE c.courseName = 'Advanced Data Warehouses') ;
INSERT INTO TABLE(SELECT c.isPrerequisiteOfRefs
     FROM Course c WHERE c.courseName = 'Advanced Databases')
(SELECT REF(c) FROM Course c WHERE c.courseName = 'Advanced Data Warehouses') ;
```

11. Add a new course identified by "INFO-H-999", entitled "Data Management in the Cloud" of 5 credits, taught in the fifth year in the Department of Computer Science, having as prerequisite the course "Advanced Databases", as professor Angelina Jolie, and for the moment no students enrolled or that took the course in the pasts.

```
INSERT INTO Course
VALUES ('INFO-H-999', 'Data Management in the Cloud', 5, 5, 'Computer Science',
    TSetRefCourses(),
    TSetRefCourses((SELECT REF(c) FROM Course c WHERE c.CourseName='Advanced Databases')),
    TSetRefCourseTaken(), TSetRefStudents(),
    (SELECT TREAT(REF(f) AS REF TFaculty) FROM Person f WHERE f.name.lastName='Jolie') ) ;
```

Update the reverse `REF` attribute.

```
INSERT INTO TABLE (SELECT TREAT(VALUE(f) AS TFaculty).coursesTaughtRefs
     FROM Person f WHERE f.name.lastName = 'Jolie')
(SELECT REF(c) FROM Course c WHERE c.courseName = 'Data Management in the Cloud') ;
INSERT INTO TABLE(SELECT c.isPrerequisiteOfRefs
     FROM Course c WHERE c.courseName = 'Advanced Databases')
(SELECT REF(c) FROM Course c WHERE c.courseName = 'Data Management in the Cloud') ;
```

12. Add the course identified by "INFO-H-415" as being taken by the student Brad Pitt in 2010 with grades 18.5 and 19.

```
INSERT INTO CourseTaken (
    SELECT TGrades(18.5,19), '2010', TREAT( Ref(p) AS REF TStudent), REF(c)
    FROM Course c, Person p
    WHERE c.courseNo = 'INFO-H-415'
    AND p.name.lastName = 'Pitt' ) ;
```

update the reverse `REF` attribute on table `Person`.

```
INSERT INTO TABLE (SELECT TREAT(VALUE(s) AS TStudent).courseTakenRefs
     FROM Person s WHERE s.name.lastName = 'Pitt')
( SELECT REF(c) FROM CourseTaken c WHERE c.courseRef.courseNo = 'INFO-H-415'
  AND c.studentRef.name.lastName = 'Pitt' );
```

update the reverse `REF` attribute on table `Course`.

```
INSERT INTO TABLE (SELECT c.courseTakenRefs
     FROM Course c WHERE c.courseName = 'Advanced Databases')
( SELECT REF(c) FROM CourseTaken c WHERE c.courseRef.courseName = 'INFO-H-415'
  AND c.studentRef.name.lastName = 'Pitt' );
```

13. Add the course identified by "INFO-H-888" to the courses to which the student Brad Pitt is enrolled.

```
INSERT INTO TABLE (SELECT TREAT(VALUE(s) AS TStudent).courseEnrolledRefs
     FROM Person s WHERE s.name.lastName = 'Pitt')
(SELECT REF(c) FROM Course c WHERE c.courseNo = 'INFO-H-888');
```

update the reverse `REF` attribute on table `Course`.

```
INSERT INTO TABLE (SELECT c.enrolledStudentRefs
     FROM Course c WHERE c.courseNo = 'INFO-H-888')
(SELECT TREAT(REF(p) as REF TSTUDENT) FROM Person p WHERE p.name.lastName = 'Pitt');
```

14. Add the course identified by "INFO-H-999" to the courses to which the student Brad Pitt is enrolled.

```
INSERT INTO TABLE (SELECT TREAT(VALUE(s) AS TStudent).courseEnrolledRefs
     FROM Person s WHERE s.name.lastName = 'Pitt')
(SELECT REF(c) FROM Course c WHERE c.courseNo = 'INFO-H-999');
```

update the reverse `REF` attribute on table `Course`.

```
INSERT INTO TABLE (SELECT c.enrolledStudentRefs
     FROM Course c WHERE c.courseNo = 'INFO-H-999')
(SELECT TREAT(REF(p) as REF TSTUDENT) FROM Person p WHERE p.name.lastName = 'Pitt');
```

15. Add the courses identified by "INFO-H-888" and "INFO-H-999" to the courses to which the student Jeremy Irons is enrolled.

```
INSERT INTO TABLE (SELECT TREAT(VALUE(s) AS TStudent).courseEnrolledRefs
     FROM Person s WHERE s.name.lastName = 'Irons')
( SELECT REF(c) FROM Course c WHERE c.courseNo = 'INFO-H-888'
  OR c.courseNo = 'INFO-H-999');
```

update the reverse `REF` attribute on table `Course`.

```
INSERT INTO TABLE (SELECT c.enrolledStudentRefs
     FROM Course c WHERE c.courseNo = 'INFO-H-888')
(SELECT TREAT(REF(p) as REF TSTUDENT) FROM Person p WHERE p.name.lastName = 'Irons');
INSERT INTO TABLE (SELECT c.enrolledStudentRefs
     FROM Course c WHERE c.courseNo = 'INFO-H-999')
(SELECT TREAT(REF(p) as REF TSTUDENT) FROM Person p WHERE p.name.lastName = 'Irons');
```

16. Add the course identified by "INFO-H-888" to the courses to which the student Scarlett Johansson is enrolled.

```
INSERT INTO TABLE (SELECT TREAT(VALUE(s) AS TStudent).courseEnrolledRefs
     FROM Person s WHERE s.name.lastName = 'Johansson')
(SELECT REF(c) FROM Course c WHERE c.courseNo = 'INFO-H-888');
```

update the reverse `REF` attribute on table `Course`.

```
INSERT INTO TABLE (SELECT c.enrolledStudentRefs
     FROM Course c WHERE c.courseNo = 'INFO-H-888')
(SELECT TREAT(REF(p) as REF TSTUDENT) FROM Person p WHERE p.name.lastName = 'Johansson');
```

## Querying the Database

1. Give the names of the persons.

```
SELECT p.name.firstName, p.name.lastName
FROM Person p;
```

2. Give the names and student number of the students.

```
SELECT s.name.firstName, s.name.lastName, TREAT(VALUE(s) as TStudent).studentNo
FROM Person s
WHERE VALUE(s) IS OF (TStudent);
```

3. Give the names of the persons who are not students nor faculty.

```
SELECT p.name.firstName, p.name.lastName
FROM Person p
WHERE VALUE(p) IS OF (ONLY TPerson);
```

4. Find the name and address of the person whose SSN is 123456789.

```
SELECT p.name.firstName, p.name.lastName, p.address.street, p.address.streetNumber,
    p.address.city, p.address.zip, p.address.country
FROM Person p WHERE SSN = '123456789';
```

5. Find the couples of SSN of persons having the same name.

```
SELECT p1.SSN, p2.SSN FROM Person p1, Person p2
WHERE p1.name = p2.name and p1.SSN < p2.SSN
```

6. Find the value of the person Michelle Pfeiffer.

```
SELECT VALUE(p) FROM Person p WHERE p.name = Tname(Michelle', 'Pfeiffer') ;
```

   This query will give as result an typed object value (as for the insertion above).

7. Give the telephone numbers of the student Brad Pitt

```
SELECT p.telephones
FROM Person p
WHERE p.name = Tname('Brad','Pitt');
```

8. Give the name of persons having the telephone number '2222'.

```
SELECT p.name.firstName, p.name.lastName
FROM Person p
WHERE '2222' in (SELECT * FROM TABLE(p.telephones));
```

   or

```
SELECT p.name.firstName, p.name.lastName
FROM Person p, TABLE(p.telephones) x
WHERE x.COLUMN_VALUE = '2222'
```

9. Give the name of persons having no telephone number.

```
SELECT p.name.firstName, p.name.lastName
FROM Person p
WHERE NOT EXISTS (SELECT * FROM TABLE(p.telephones));
```

10. Find the couples of names of persons sharing at least one telephone number.

```
SELECT DISTINCT p1.name.firstName, p1.name.lastName, p2.name.firstName, p2.name.lastName
FROM Person p1, TABLE(p1.telephones) x, Person p2, TABLE(p2.telephones) y
WHERE p1.SSN < p2.SSN AND x.COLUMN_VALUE = y.COLUMN_VALUE
```

11. Find the couples of SSN of persons having exactly the same set of telephones numbers.

```
SELECT DISTINCT p1.name.firstName, p1.name.lastName, p2.name.firstName, p2.name.lastName
FROM Person p1, Person p2
WHERE p1.SSN < p2.SSN AND EXISTS ( SELECT * FROM TABLE(p1.telephones) )
AND NOT EXISTS (
    SELECT * FROM TABLE(p1.telephones) x WHERE NOT EXISTS (
        SELECT * FROM TABLE(p2.telephones) y WHERE x.COLUMN_VALUE = y.COLUMN_VALUE ) )
AND NOT EXISTS (
    SELECT * FROM TABLE(p2.telephones) x WHERE NOT EXISTS (
        SELECT * FROM TABLE(p1.telephones) y WHERE x.COLUMN_VALUE = y.COLUMN_VALUE ) )
```

12. Give the numbers and names of students who have successfully completed the course "Advanced Databases".

```
SELECT c.studentref.studentNo,c.studentref.name.firstName,c.studentref.name.lastName
FROM CourseTaken c
WHERE c.courseref.courseName = 'Advanced Databases' ;
```

or

```
SELECT value(r).studentRef.studentNo, value(r).studentRef.name.firstName,
    value(r).studentRef.name.lastName
FROM Course c, TABLE(c.courseTakenRefs) r
WHERE c.courseName = 'Advanced Databases' ;
```

or

```
SELECT TREAT(VALUE(p) AS TStudent).studentNo, p.name.firstName, p.name.lastName
FROM Person p, TABLE(TREAT(VALUE(p) AS TStudent).courseTakenRefs) c
WHERE VALUE(p) IS OF (TStudent) AND value(c).courseRef.courseName = 'Advanced Databases';
```

Function `TREAT(value AS aSubtype)` where `value` is a value of type `OBJECT` that has subtypes (including the subtype `aSubtype`), redeclares the value as subtype `aSubtype`. This allows access to attributes and methods of the subtype.

13. Give the prerequisites of the course "Advanced Data Warehouses".

```
SELECT value(p).courseName
FROM Course c, TABLE (c.hasPrerequisiteRefs) p
WHERE courseName = 'Advanced Data Warehouses';
```

14. Search inconsistencies in prerequisites: Courses that have as prerequisite a course of a subsequent year. Give the name of the course and the prerequisite.

```
SELECT c.courseName, value(p).courseName
FROM Course c, TABLE (c.hasPrerequisiteRefs) p
WHERE value(p).year > c.year;
```

15. Search inconsistencies in prerequisites: Students enrolled in courses that have as prerequisite a course they have not taken. Give the name of the student, the name of the enrolled course and the name of the missing prerequisite.

```
SELECT p.name.firstName, p.name.lastName, VALUE(c).courseName, VALUE(r).courseName
FROM Person p, TABLE(TREAT(VALUE(p) AS TStudent).courseEnrolledRefs) c,
    TABLE(VALUE(c).hasPrerequisiteRefs) r
WHERE NOT EXISTS (SELECT * FROM CourseTaken t WHERE t.studentRef = REF(p)
    AND t.courseRef.courseNo = VALUE(r).courseNo )
```

16. Give the couples of names of a student and one of his teachers.

```
SELECT p.name.firstName, p.name.lastName, VALUE(c).courseRef.professorRef.name.firstName,
    VALUE(c).courseRef.professorRef.name.lastName
FROM Person p, TABLE (TREAT(VALUE(p) AS TStudent).courseTakenRefs) c
WHERE VALUE(p) IS OF (TStudent) ;
```

17. Give the course number of the courses in which Brad Pitt is enrolled.

```
select value(c).courseNo
from Person s, TABLE ( TREAT(VALUE(s) AS TStudent).courseEnrolledRefs ) c
WHERE s.name = Tname('Brad','Pitt')
```

18. Names of students who are enrolled in at least one course in which Brad Pitt is enrolled.

```
SELECT p.name.firstName, p.name.lastName
FROM Person p, Person s, TABLE (TREAT(VALUE(p) AS TStudent).courseTakenRefs) cp,
    TABLE (TREAT(VALUE(s) AS TStudent).courseTakenRefs) cs
WHERE VALUE(p) IS OF (TStudent)
AND s.name = Tname('Brad','Pitt') AND  VALUE(cp) = VALUE(cs) ;
```

19. Names of students who are enrolled in (at least) all courses in which Brad Pitt is enrolled.

```
SELECT p.name.firstName, p.name.lastName
FROM Person p, Person s
WHERE VALUE(p) IS OF TStudiant
AND s.name = Tname('Brad','Pitt')
AND NOT EXISTS (SELECT * FROM TABLE (TREAT(VALUE(s) AS TStudent).coursesTaken) cs
    WHERE NOT EXISTS (SELECT * FROM TABLE (TREAT(VALUE(p) AS TEtudiant).coursesTaken) cp
WHERE cp = cs ) ) ;
```