

Oracle 11g : l'exemple « jouet » (du cours) complété et quelques spécificités d'Oracle

PRÉAMBULE	2
CRÉATION DE LA BASE DE DONNÉES	3
CRÉATION DES DAD	8
<i>Création du DAD sous SQL*Plus</i>	9
<i>Modification du fichier d'initialisation</i>	9
CRÉATION DES UTILISATEURS	9
CRÉATION DES RÉPERTOIRES	9
CRÉATION DES XSD	9
CRÉATION DES SÉQUENCES	9
CRÉATION DES TYPES	10
CRÉATION DES TABLES ET DES CONTRAINTES D'INTÉGRITÉ	11
CRÉATION DES MÉTADONNÉES POUR ORACLE SPATIAL	12
CRÉATION DES INDEX (DONT LES INDEX SPATIAUX)	12
CRÉATION DES VUES	12
CRÉATION DES CORPS DE TYPES	12
CRÉATION DES DÉCLENCHEURS.....	15
CRÉATION DES FONCTIONS ET PROCÉDURES	16
CRÉATION DES PAQUETAGES (ET CORPS DE PAQUETAGES)	19
<i>Illustration de la surcharge, des exceptions et des curseurs gérés par FOR</i>	19
<i>Vérification des contraintes d'intégrité a posteriori</i>	20
CRÉATION DES DROITS	22
CRÉATION DES SYNONYMES	23
CRÉATION DE L'APPLICATION	23
PAGE D'ACCUEIL.....	23
TOUTES LES INFORMATIONS D'UNE TABLE (NON OBJET)	24
TOUTES LES INFORMATIONS SUR UN ÉTUDIANT	24
CRÉATION DES DONNÉES	28
INSERTION DES DONNÉES	28
VÉRIFICATION DE CONTRAINTES D'INTÉGRITÉ	39
INTERROGATIONS ET MISES À JOUR	41
VÉRIFICATION DES DONNÉES.....	41
REQUÊTES D'INTERROGATION RELATIONELLES	42
UTILISATION SIMPLE D'UN CURSEUR EN PL/SQL.....	49
RÉINITIALISATION D'UNE SÉQUENCE	50
INFORMATIONS GÉNÉRALES SUR UN OBJET COMPLEXE.....	50
ACCÈS AUX DONNÉES ET UTILISATION DES MÉTHODES (ICI DANS SELECT, WHERE, JOIN).....	53
APPEL D'UNE PROCÉDURE.....	55
UTILISATION DE MULTISET : RELATION PLATE TRANSFORMÉE EN NF ²	55
INFORMATIONS SUR LES COLLECTIONS IMBRIQUÉES EN NF ²	56
INFORMATIONS SUR LES COLLECTIONS IMBRIQUÉES EN RELATION PLATE	57
INFORMATIONS SUR LES COLLECTIONS IMBRIQUÉES AVEC UN CURSEUR	59
UTILISATION D'UNE VUE.....	61
NOMBRE D'ÉLÉMENTS DANS LES COLLECTIONS IMBRIQUÉES	61
MODIFICATIONS SIMPLES	62
MISES À JOUR DES ÉLÉMENTS DES COLLECTIONS IMBRIQUÉES	63
<i>Mises à jour des prénoms d'un étudiant</i>	63
<i>Mises à jour des téléphones d'un étudiant</i>	64
<i>Mises à jour des voitures possédées par un étudiant</i>	64
<i>Mises à jour des diplômes obtenus par un étudiant</i>	65
ILLUSTRATION DE SURCHARGE, EXCEPTIONS, CURSEUR GÉRÉ PAR FOR	65
UTILISATION DU PAQUETAGE DES CONTRAINTES D'INTÉGRITÉ VIOLÉES	66
XML	68
<i>Affichage au format XML de données classiques (c.-à-d. non dans un XMLTYPE)</i>	68

<i>Extraction de données XML (c.-à-d. contenues dans un XMLTYPE)</i>	71
<i>Mises à jour de données XML (c.-à-d. contenues dans un XMLTYPE)</i>	77
ORACLE SPATIAL	78
<i>Informations sur un index spatial (et plus précisément de type R-Tree)</i>	78
<i>Requêtes spatiales</i>	79
UTILISATION DE L'APPLICATION	86
PAGE D'ACCUEIL	87
TOUTES LES INFORMATIONS SUR UNE TABLE.....	88
<i>Départements français</i>	88
<i>Informations spatiales sur les étudiants</i>	89
INFORMATIONS SUR UN ÉTUDIANT	90
ANNEXES	93
INTERROGATION DU DICTIONNAIRE DE DONNÉES	93
<i>Informations générales sur le SGBD</i>	93
<i>Informations générales sur les XSD</i>	96
<i>Informations générales sur les objets de cette base de données</i>	97
SUPPRESSION DE LA BASE DE DONNÉES	107
SUPPRESSION DES DONNÉES	109
IMAGES	109
FICHER FICHE_PHILOSOPHE.XSD	109
LES POSITIONS GÉOGRAPHIQUES	113
LES FIGURES GÉOMÉTRIQUES	114
<i>Code PL/SQL créant les instructions à insérer</i>	114
<i>Balise Canvas d'HTML5 pré-programmée en JavaScript où insérer les instructions créées</i>	118
<i>Visualisation des figures géométriques avec un navigateur Web (supportant la balise Canvas)</i>	124

Préambule

L'objectif de ce document consiste principalement d'illustrer quelques-uns des éléments du cours de bases de données à travers un exemple implanté en PL/SQL avec la version 11g du SGBD Oracle mais aussi de présenter quelques spécificités d'Oracle. Ainsi, l'exemple « jouet » a été transformé notamment pour intégrer des extensions relationnelles-objet, des aspects Web (architecture à deux niveaux (*two-tier architecture*) seulement) et des données XML ; de plus, Oracle Spatial est utilisé.

Les données gérées sont : les étudiants (identifiant, nom, prénoms, téléphones, adresse (trois lignes, code postal, ville, site Web), département de naissance, pseudonyme, photographie (et sa signature), *curriculum vitae* (source de la fiche d'information, années et lieux de naissance et de décès, écoles/traditions, principaux intérêts, idées remarquables, œuvres principales, influences (par et sur)), position géographique, figure géométrique), leurs voitures (immatriculation (chiffres, lettres, département), couleur), les diplômes (intitulés abrégé et complet) qu'ils ont obtenus (année d'obtention) et les départements français (code, nom).

Le code permet d'illustrer la gestion des : utilisateurs, répertoires, séquences, types (et corps de types), tables, contraintes d'intégrité, vues, index (dont les index spatiaux), déclencheurs, fonctions et procédures, paquetages (et corps de paquetages), droits, synonymes. Outre les types de données classiques (NUMBER, CHAR, VARCHAR), d'autres sont utilisés : UDT (*User Defined Type*), VARRAY (*Variable-size array*), TABLE OF, NESTED TABLE, REF (*reference*), URITYPE (*Uniform Resource Identifier Type*), BLOB (*Binary Large Object*), ORDIMAGE (et ORDIMAGESIGNATURE), XMLTYPE (*Extensible Markup Language Type*), MDSYS.SDO_GEOMETRY (pour Oracle Spatial). Plusieurs paquetages sont également utilisés : DBMS_LOB (*DBMS (DataBase Management System) Large Object*), DBMS_EPG (*DBMS Embedded PL/SQL Gateway*), DBMS_XDB (*DBMS XML DataBase*), DBMS_OUTPUT, HTP (*HyperText Procedures*), OWA (*Oracle Web Agent*), OWA_UTIL (*OWA Utility subprograms*), URIFACTORY (*URI factory*), WPG_DOCLOAD.

Le diagramme de classes ci-dessous illustre (partiellement) ces commentaires.

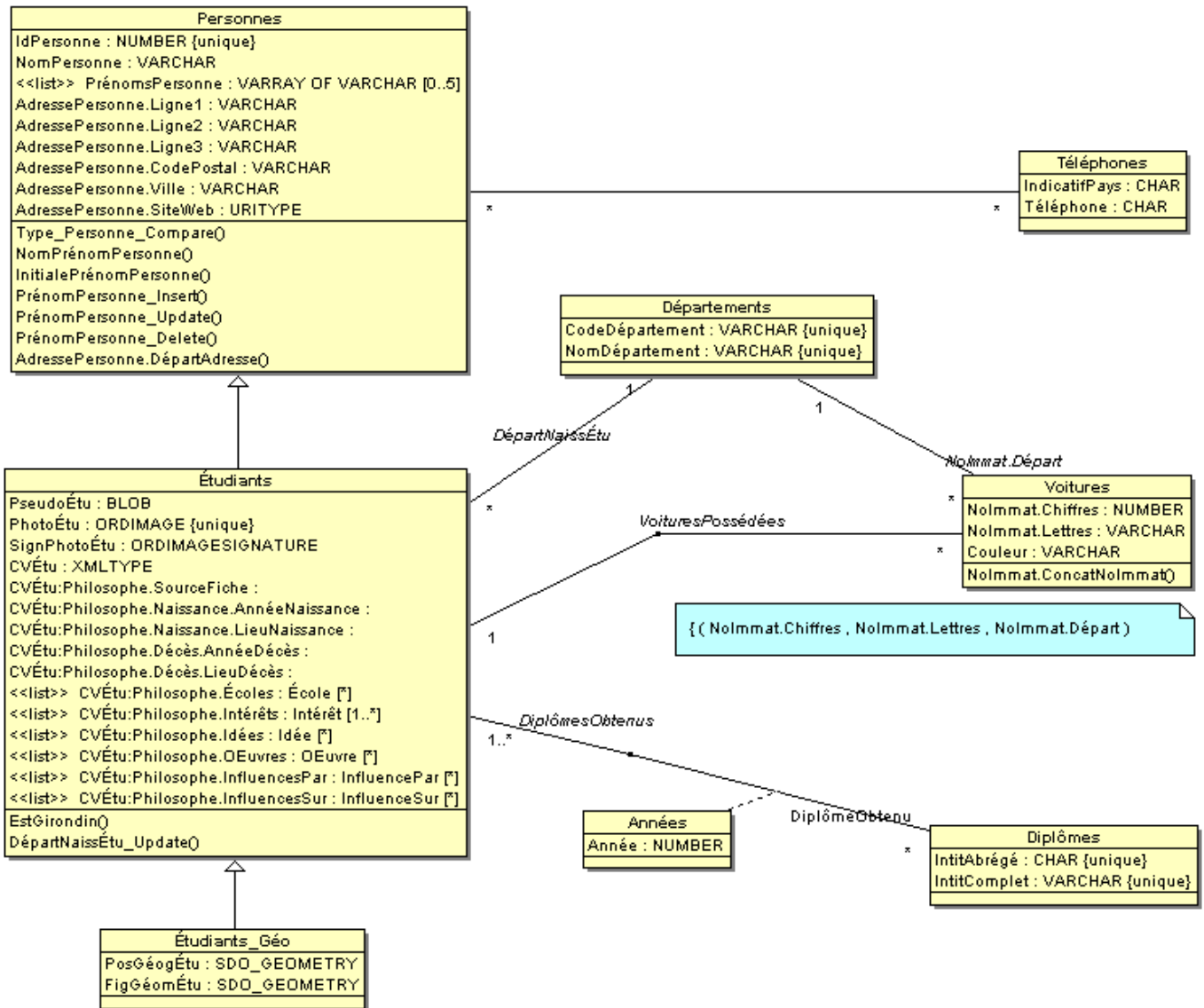


Diagramme réalisé avec  **BOUML**

Pour plus d'informations, consultez <http://www.oracle.com>, le site d'**ORACLE**, et par exemple la documentation sur la gestion de données multimédia (*Oracle® Multimedia Reference 11g Release 1*, http://download.oracle.com/docs/cd/B28359_01/appdev.111/b28414/toc.htm) ou sur XML (*Oracle® XML DB Developer's Guide 11g Release 1*, http://download.oracle.com/docs/cd/B28359_01/appdev.111/b28369/toc.htm) ou sur Oracle Spatial (*Oracle® Spatial Developer's Guide 11g Release 2*, http://download.oracle.com/docs/cd/E11882_01/appdev.112/e11830/sdo_intro.htm).

Création de la base de données

Il s'agit de créer les DAD (*Database Access Descriptor*), utilisateurs, répertoires, XSD (schémas XML), séquences, types, tables et contraintes d'intégrité, index, vues, corps de types, déclencheurs, fonctions et procédures, paquetages (et corps de paquetages), droits, synonymes.

Voici quelques informations (tables et droits et contraintes d'intégrité, vues, index, paquetages, procédures, fonctions, déclencheurs, types, séquences, répertoires, XSD, utilisateurs) sur cette base de données une fois connecté par l'outil Oracle SQL Developer :

- + Tables
- + Views
- + Indexes
- + Packages
- + Procedures
- + Functions
- + Triggers
- + Types
- + Sequences
- + Materialized Views
- + Materialized Views Logs
- + Synonyms
- + Public Synonyms
- + Database Links
- + Public Database Links
- + Directories
- + XML Schemas
- + Recycle Bin
- + Other Users

Tables

- DEPARTEMENTS
 - CODEDEPARTEMENT
 - NOMDEPARTEMENT
- DIPLOMES
 - INITABREGE
 - INITCOMPLET
- ETUDIANTS
 - IDPERSONNE
 - NOMPERSONNE
 - PRENOMSPERSONNE
 - TELEPHONESPERSONNE
 - ADRESSEPERSONNE
 - DEPARTNAISSETU
 - PSEUDOETU
 - PHOTOETU
 - SIGNPHOTOETU
 - CVETU
 - VOITURESPOSSEDEES
 - DIPLOMESOBTENUS
- ETUDIANTS_GEO
 - IDPERSONNE
 - POS GEOGETU
 - FIGGEOGETU
- MDRT_119C7\$
 - TABLE_DIPLOMESOBTENUS
 - TABLE_TELEPHONESPERSONNE
 - TABLE_VOITURESPOSSEDEES

Columns	Data	Constraints	Grants	Statistics	Column Statistics	Triggers	Dependenc
Actions...							
Privilege	Grantee	Grantable	Grantor	Object Name			
SELECT	ETD	NO	GUIBERT	DEPARTEMENTS			
Privilege	Grantee	Grantable	Grantor	Object Name			
SELECT	ETD	NO	GUIBERT	DIPLOMES			
Privilege	Grantee	Grantable	Grantor	Object Name			
SELECT	ETD	NO	GUIBERT	ETUDIANTS			
Privilege	Grantee	Grantable	Grantor	Object Name			
SELECT	ETD	NO	GUIBERT	ETUDIANTS_GEO			
Privilege	Grantee	Grantable	Grantor	Object Name			
SELECT	ETD	NO	GUIBERT	TABLE_DIPLOMESOBTENUS			
Privilege	Grantee	Grantable	Grantor	Object Name			
SELECT	ETD	NO	GUIBERT	TABLE_TELEPHONESPERSONNE			
Privilege	Grantee	Grantable	Grantor	Object Name			
SELECT	ETD	NO	GUIBERT	TABLE_VOITURESPOSSEDEES			

DEPARTEMENTS												
Columns	Data	Constraints	Grants	Statistics	Column Statistics	Triggers	Dependencies	Details	Partitions	Indexes	SQL	
Actions...												
Constraint Name	Constraint Type	Search Condition	Status	Deferrable	Validated	Generated	Index Name					
CLEPRIMAIRE_DEPART...	Primary_Key	(null)	ENABLED	NOT DEFERRABLE	VALIDATED	USER NAME	CLEPRIMAIRE_DEPART...					
SYS_C009977	Check	"CODEDEPARTEMENT" I...	ENABLED	NOT DEFERRABLE	VALIDATED	GENERATED NAME	(null)					
SYS_C009978	Check	"NOMDEPARTEMENT" IS...	ENABLED	NOT DEFERRABLE	VALIDATED	GENERATED NAME	(null)					
UNICITE_NOMDEPARTE...	Unique	(null)	ENABLED	NOT DEFERRABLE	VALIDATED	USER NAME	UNICITE_NOMDEPARTE...					

DIPLOMES												
Columns	Data	Constraints	Grants	Statistics	Column Statistics	Triggers	Dependencies	Details	Partitions	Indexes	SQL	
Actions...												
Constraint Name	Constraint Type	Search Condition	Status	Deferrable	Validated	Generated	Index Name					
CLEPRIMAIRE_DIPLOMES	Primary_Key	(null)	ENABLED	NOT DEFERRABLE	VALIDATED	USER NAME	CLEPRIMAIRE_DIPLOMES					
EXISTE_INITITABREGE	Check	InititAbrege IS NOT NULL	ENABLED	NOT DEFERRABLE	VALIDATED	USER NAME	(null)					
EXISTE_INITITCOMPLET	Check	InititCompleet IS NOT NULL	ENABLED	NOT DEFERRABLE	VALIDATED	USER NAME	(null)					
SYS_C009753	Unique	(null)	ENABLED	NOT DEFERRABLE	VALIDATED	GENERATED NAME	SYS_C009753					
UNICITE_INITITCOMPLET	Unique	(null)	ENABLED	NOT DEFERRABLE	VALIDATED	USER NAME	UNICITE_INITITCOMPLET					

ETUDIANTS												
Columns	Data	Constraints	Grants	Statistics	Column Statistics	Triggers	Dependencies	Details	Partitions	Indexes	SQL	
Actions...												
Constraint Name	Constraint Type	Search Condition	Status	Deferrable	Validated	Generated	Index Name					
CLEPRIMAIRE_ETUDIANTS	Primary_Key	(null)	ENABLED	NOT DEFERRABLE	VALIDATED	USER NAME	CLEPRIMAIRE_ETUDIAN...					
CONTRAINTE_IDPERSONNEPOSITIF	Check	IdPersonne > 0	ENABLED	NOT DEFERRABLE	VALIDATED	USER NAME	(null)					
EXISTE_CODEPOSTALADRPERSO...	Check	AdressePersonne.Cod...	ENABLED	NOT DEFERRABLE	VALIDATED	USER NAME	(null)					
EXISTE_LIGNE1ADRPERSO...	Check	AdressePersonne.Lign...	ENABLED	NOT DEFERRABLE	VALIDATED	USER NAME	(null)					
EXISTE_VILLEADRPERSO...	Check	AdressePersonne.Ville...	ENABLED	NOT DEFERRABLE	VALIDATED	USER NAME	(null)					
REF_ETUDIANTS_DEPARTEMENTS	Foreign_Key	(null)	ENABLED	NOT DEFERRABLE	VALIDATED	USER NAME	(null)					
SYS_C009775	Check	"IDPERSONNE" IS NOT ...	ENABLED	NOT DEFERRABLE	VALIDATED	GENERATED NAME	(null)					
SYS_C009776	Check	"NOMPERSONNE" IS NO...	ENABLED	NOT DEFERRABLE	VALIDATED	GENERATED NAME	(null)					
SYS_C009777	Check	"DEPARTNAISSETU" IS ...	ENABLED	NOT DEFERRABLE	VALIDATED	GENERATED NAME	(null)					
SYS_C009783	Unique	(null)	ENABLED	NOT DEFERRABLE	VALIDATED	GENERATED NAME	SYS_C009783					
SYS_C009784	Unique	(null)	ENABLED	NOT DEFERRABLE	VALIDATED	GENERATED NAME	SYS_C009784					
SYS_C009785	Unique	(null)	ENABLED	NOT DEFERRABLE	VALIDATED	GENERATED NAME	SYS_C009785					
SYS_C009786	Unique	(null)	ENABLED	NOT DEFERRABLE	VALIDATED	GENERATED NAME	SYS_C009786					

ETUDIANTS_GEO												
Columns	Data	Constraints	Grants	Statistics	Column Statistics	Triggers	Dependencies	Details	Partitions	Indexes	SQL	
Actions...												
Constraint Name	Constraint Type	Search Condition	Status	Deferrable	Validated	Generated	Index Name					
SYS_C009968	Check	"IDPERSONNE" IS NOT ...	ENABLED	NOT DEFERRABLE	VALIDATED	GENERATED NAME	(null)					
SYS_C009969	Primary_Key	(null)	ENABLED	NOT DEFERRABLE	VALIDATED	GENERATED NAME	SYS_C009969					
SYS_C009970	Foreign_Key	(null)	ENABLED	NOT DEFERRABLE	VALIDATED	GENERATED NAME	(null)					

TABLE_VOITURESPOSSEDEES												
Columns	Data	Constraints	Grants	Statistics	Column Statistics	Triggers	Dependencies	Details	Partitions	Indexes	SQL	
Actions...												
Constraint Name	Constraint Type	Search Condition	Status	Deferrable	Validated	Generated	Index Name					
CONTRAINTE_LISTECOULEURS	Check	Couleur IS NULL OR Couleur IN ('rouge', 'jaune', 'orange')	ENABLED	NOT DEFERRABLE	VALIDATED	USER NAME	(null)					
CONTRAINTE_NOIMMATCHIFFRESBORN	Check	Noimmat.Chiffres BETWEEN 1 AND 9999	ENABLED	NOT DEFERRABLE	VALIDATED	USER NAME	(null)					
CONTRAINTE_NOIMMATLETTRESMAJUS	Check	Noimmat.Lettres = UPPER(Noimmat.Lettres)	ENABLED	NOT DEFERRABLE	VALIDATED	USER NAME	(null)					
EXISTE_NOIMMATCHIFFRES	Check	Noimmat.Chiffres IS NOT NULL	ENABLED	NOT DEFERRABLE	VALIDATED	USER NAME	(null)					
EXISTE_NOIMMATLETTRES	Check	Noimmat.Lettres IS NOT NULL	ENABLED	NOT DEFERRABLE	VALIDATED	USER NAME	(null)					
UNICITE_NOIMMAT	Unique	(null)	ENABLED	NOT DEFERRABLE	VALIDATED	USER NAME	UNICITE_NOIMMAT					

Views	
VUE_VOITURESETUDIANTS	
NOIMMATCHIFFRES	
NOIMMATLETTRES	
NOIMMATDEPART	
COULEUR	
IDPERSONNE	
NOMPERSONNE	
DEPARTNAISSETU	

Indexes	
CLEPRIMAIRE_DEPARTEMENTS	
CLEPRIMAIRE_DIPLOMES	
CLEPRIMAIRE_ETUDIANTS	
INDEX_NOMETU	
INDEX_RTREE_FIGGEOMETU	
SYS_C009753	
SYS_C009783	
SYS_C009784	
SYS_C009785	
SYS_C009786	
SYS_C009969	
UNICITE_IDPERSONNE_ANNEE	
UNICITE_INITITCOMPLET	
UNICITE_NOIMMAT	
UNICITE_NOMDEPARTEMENT	

Index Owner	Index Name	Table Owner	Table Name	Column Name	Column Position	Descend
GUIBERT	CLEPRIMAIRE_DEPARTEMENTS	GUIBERT	DEPARTEMENTS	CODEDEPARTEMENT	1	ASC
Index Owner	Index Name	Table Owner	Table Name	Column Name	Column Position	Descend
GUIBERT	CLEPRIMAIRE_DIPLOMES	GUIBERT	DIPLOMES	INITABREGE	1	ASC
Index Owner	Index Name	Table Owner	Table Name	Column Name	Column Position	Descend
GUIBERT	CLEPRIMAIRE_ETUDIANTS	GUIBERT	ETUDIANTS	IDPERSONNE	1	ASC
Index Owner	Index Name	Table Owner	Table Name	Column Name	Column Position	Descend
GUIBERT	INDEX_NOMETU	GUIBERT	ETUDIANTS	NOMPERSONNE	1	ASC
Index Owner	Index Name	Table Owner	Table Name	Column Name	Column Position	Descend
GUIBERT	INDEX_RTREE_FIGGEOMETU	GUIBERT	ETUDIANTS_GEO	FIGGEOMETU	1	ASC
Index Owner	Index Name	Table Owner	Table Name	Column Name	Column Position	Descend
GUIBERT	SYS_C009753	GUIBERT	DIPLOMES	SYS_NC_OID\$	1	ASC
Index Owner	Index Name	Table Owner	Table Name	Column Name	Column Position	Descend
GUIBERT	SYS_C009783	GUIBERT	ETUDIANTS	DIPLOMESOBTENUS	1	ASC
Index Owner	Index Name	Table Owner	Table Name	Column Name	Column Position	Descend
GUIBERT	SYS_C009784	GUIBERT	ETUDIANTS	VOITURESPOSSEDEES	1	ASC
Index Owner	Index Name	Table Owner	Table Name	Column Name	Column Position	Descend
GUIBERT	SYS_C009785	GUIBERT	ETUDIANTS	TELEPHONESPERSONNE	1	ASC
Index Owner	Index Name	Table Owner	Table Name	Column Name	Column Position	Descend
GUIBERT	SYS_C009786	GUIBERT	ETUDIANTS	SYS_NC_OID\$	1	ASC
Index Owner	Index Name	Table Owner	Table Name	Column Name	Column Position	Descend
GUIBERT	SYS_C009969	GUIBERT	ETUDIANTS_GEO	IDPERSONNE	1	ASC
Index Owner	Index Name	Table Owner	Table Name	Column Name	Column Position	Descend
GUIBERT	UNICITE_IDPERSONNE_ANNEE	GUIBERT	TABLE_DIPLOMESOBTENUS	NESTED_TABLE_ID	1	ASC
GUIBERT	UNICITE_IDPERSONNE_ANNEE	GUIBERT	TABLE_DIPLOMESOBTENUS	ANNEE	2	ASC
Index Owner	Index Name	Table Owner	Table Name	Column Name	Column Position	Descend
GUIBERT	UNICITE_INTITCOMPLET	GUIBERT	DIPLOMES	INTITCOMPLET	1	ASC
Index Owner	Index Name	Table Owner	Table Name	Column Name	Column Position	Descend
GUIBERT	UNICITE_NOIMMAT	GUIBERT	TABLE_VOITURESPOSSEDEES	"NOIMMAT"."CHIFFRES"	1	ASC
GUIBERT	UNICITE_NOIMMAT	GUIBERT	TABLE_VOITURESPOSSEDEES	"NOIMMAT"."LETTRES"	2	ASC
GUIBERT	UNICITE_NOIMMAT	GUIBERT	TABLE_VOITURESPOSSEDEES	"NOIMMAT"."DEPART"	3	ASC
Index Owner	Index Name	Table Owner	Table Name	Column Name	Column Position	Descend
GUIBERT	UNICITE_NOMDEPARTEMENT	GUIBERT	DEPARTEMENTS	NOMDEPARTEMENT	1	ASC

DEPARTEMENTS

Columns Data Constraints Grants Statistics Column Statistics Triggers Dependencies Details Partitions Indexes SQL

Actions...

Index Owner	Index Name	Uniqueness	Temporary	Partitioned	Join Index	Columns
GUIBERT	UNICITE_NOMDEPARTEMENT	UNIQUE	N	NO	NO	NOMDEPARTEMENT
GUIBERT	CLEPRIMAIRE_DEPARTEMENTS	UNIQUE	N	NO	NO	CODEDEPARTEMENT

DIPLOMES

Columns Data Constraints Grants Statistics Column Statistics Triggers Dependencies Details Partitions Indexes SQL

Actions...

Index Owner	Index Name	Uniqueness	Temporary	Partitioned	Join Index	Columns
GUIBERT	SYS_C009753	UNIQUE	N	NO	NO	SYS_NC_OID\$
GUIBERT	CLEPRIMAIRE_DIPLOMES	UNIQUE	N	NO	NO	INITABREGE
GUIBERT	UNICITE_INTITCOMPLET	UNIQUE	N	NO	NO	INTITCOMPLET

ETUDIANTS

Columns Data Constraints Grants Statistics Column Statistics Triggers Dependencies Details Partitions Indexes SQL

Actions...

Index Owner	Index Name	Uniqueness	Temporary	Partitioned	Join Index	Columns
GUIBERT	SYS_C009783	UNIQUE	N	NO	NO	DIPLOMESOBTENUS
GUIBERT	SYS_C009784	UNIQUE	N	NO	NO	VOITURESPOSSEDEES
GUIBERT	SYS_C009785	UNIQUE	N	NO	NO	TELEPHONESPERSONNE
GUIBERT	SYS_C009786	UNIQUE	N	NO	NO	SYS_NC_OID\$
GUIBERT	INDEX_NOMETU	NONUNIQUE	N	NO	NO	NOMPERSONNE
GUIBERT	CLEPRIMAIRE_ETUDIANTS	UNIQUE	N	NO	NO	IDPERSONNE

Index Owner	Index Name	Uniqueness	Temporary	Partitioned	Join Index	Columns
GUIBERT	SYS_C009969	UNIQUE	N	NO	NO	IDPERSONNE
GUIBERT	INDEX_RTREE_FIGGEOMETU	NONUNIQUE	N	NO	NO	FIGGEOMETU

Index Owner	Index Name	Uniqueness	Temporary	Partitioned	Join Index	Columns
GUIBERT	UNICITE_IDPERSONNE_ANNEE	UNIQUE	N	NO	NO	NESTED_TABLE_ID, ANNEE

Index Owner	Index Name	Uniqueness	Temporary	Partitioned	Join Index	Columns
GUIBERT	UNICITE_NOIMMAT	UNIQUE	N	NO	NO	"NOIMMAT"."CHIFFRES", "NOIMMAT"."LETTRES", "NOIMMAT"."DEPART"

Packages

- CIVOLEES
 - CIVOLEES Body
 - Default_NoImmatDepart_violee
 - Default_Couleur_violee
 - Ref_ImmatVoiture_Dpts_violee
 - Ref_Etudiants_Dipl_violee
 - Unicite_IdPersonne_Dipl_violee
 - Default_NoImmatDepart_violee
 - Default_Couleur_violee
 - Ref_ImmatVoiture_Dpts_violee
 - Ref_Etudiants_Dipl_violee
 - Unicite_IdPersonne_Dipl_violee
- NBDILOBTETD
 - NBDILOBTETD Body
 - NombreDiplomes
 - NombreDiplomes(NUMBER)
 - NombreDiplomes(Diplomes.IntitAbrege%TYPE, NUMBER)
 - NombreDiplomes
 - NombreDiplomes(NUMBER)
 - NombreDiplomes(Diplomes.IntitAbrege%TYPE, NUMBER)

Types

- TYPE_ADRESSE
 - TYPE_ADRESSE Body
- TYPE_DIPLOME
- TYPE_DIPLOMEOBTENU
- TYPE_DIPLOMESOBTENUS
- TYPE_ETUDIANT
 - TYPE_ETUDIANT Body
- TYPE_IMMATVOITURE
 - TYPE_IMMATVOITURE Body
- TYPE_IMMATVOITURES
- TYPE_PERSONNE
 - TYPE_PERSONNE Body
- TYPE_PRENOMS
- TYPE_TELEPHONE
- TYPE_TELEPHONES
- TYPE_VARRAY_VARCHAR
- TYPE_VOITURE
- TYPE_VOITURES

Procedures

- AFFECTEPROPPHOTOSETUDIANTS
- COMPARESIGNPHOTOSETUDIANTS
 - D
- IMPORTEPHOTOSETUDIANTS
- INSEREPSEUDOETUDIANTS
 - IDP
 - FICPSEUDO

Functions

- ESTCODEDEPARTEMENT
 - CD
 - NUMBER

Triggers

- DECLEN_AVINSERTUPDATE_ETU_GEO
- DECLEN_AVINSERTUPDATE_ETUDIANT

Création du DAD sous SQL*Plus

```
-- connexion avec le compte SYS
CONNECT / AS SYSDBA
-- configuration du numéro du port pour HTTP
CALL DBMS_XDB.SETHTTPPORT(1158) ; -- et non le 8080 par défaut
ALTER SYSTEM REGISTER ;
-- création d'un DAD pour le compte ETD
DECLARE
    nd VARCHAR2(30) := 'DAD' ; -- nom du DAD
BEGIN
    -- création du DAD avec un chemin virtuel
    DBMS_EPG.CREATE_DAD(DAD_NAME=>nd,PATH=>'/'||nd||'/*') ;
    -- affectation des attributs (cf. aussi document-table-name, document-path, etc.)
    DBMS_EPG.SET_DAD_ATTRIBUTE(
        DAD_NAME=>nd,ATTR_NAME=>'default-page',ATTR_VALUE=>'home') ;
    DBMS_EPG.SET_DAD_ATTRIBUTE(
        DAD_NAME=>nd,ATTR_NAME=>'database-username',ATTR_VALUE=>'ETD') ;
    DBMS_EPG.AUTHORIZE_DAD(DAD_NAME=>nd,USER=>'GUIBERT') ;
    DBMS_EPG.AUTHORIZE_DAD(DAD_NAME=>nd,USER=>'ETD') ;
END ;
-- déconnexion
QUIT
```

Modification du fichier d'initialisation

```
# variables (local_listener voire dispatchers) à affecter dans le fichier init.ora
local_listener="(ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1158))"
```

Création des utilisateurs

```
-- création du compte ETD/ETD
CREATE USER ETD IDENTIFIED BY ETD ;
-- ETD va être l'utilisateur du DAD
ALTER USER ETD IDENTIFIED BY ETD ACCOUNT UNLOCK ;
-- déverrouillage du compte anonyme pour le DAD
ALTER USER ANONYMOUS ACCOUNT UNLOCK ;
-- déverrouillage du compte XDB pour XML DB
ALTER USER XDB ACCOUNT UNLOCK ;
```

Création des répertoires

```
-- création nom logique répertoire où sont les informations étudiants (images et XSD)
CREATE OR REPLACE DIRECTORY Rep_Etudiants AS
    'D:\Travail\Enseignement\TD\TD BD\scripts création bases\Oracle\JouetCoursBD RO';
N.B. : les images se trouvent en annexe.
```

Création des XSD

```
-- création du XSD (XML Schema i. e. grammaire des documents XML)
BEGIN
    DBMS_XMLSCHEMA.REGISTERSCHEMA(
        SCHEMAURL=>'fiche_philosophe.xsd', -- nom du schéma
        SCHEMADOC=>BFILENAME(UPPER('Rep_Etudiants'),'fiche_philosophe.xsd'),
        LOCAL=>FALSE, -- PUBLIC i. e. visible par tous
        GENTYPES=>FALSE,
        GENTABLES=>FALSE,
        FORCE=>FALSE, -- pour ne pas s'affranchir d'éventuelles erreurs
        OPTIONS=>DBMS_XMLSCHEMA.REGISTER_BINARYXML) ;
END ;
N.B. : le fichier fiche_philosophe.xsd se trouve en annexe.
```

Création des séquences

```
-- séquence pour les identifiants des personnes
CREATE SEQUENCE Sequence_IdPersonne
    START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 99 CYCLE NOCACHE ;
```

Création des types

```
-- type de tableau de chaînes de caractères
CREATE OR REPLACE TYPE Type_VARRAY_VARCHAR AS VARRAY(25) OF VARCHAR(70) ;
-- type des prénoms
CREATE OR REPLACE TYPE Type_Prenoms AS VARRAY(5) OF VARCHAR(30) ; -- doublons possible
-- type de téléphone
CREATE OR REPLACE TYPE Type_Telephone AS OBJECT (
    IndicatifPays CHAR(6) , -- cf. recommandation UIT-T E.164 (1/2/2004)
    Telephone CHAR(9) ) ; -- le n° sans le 1er chiffre (zéro)
-- type des téléphones
CREATE OR REPLACE TYPE Type_Telephones AS TABLE OF Type_Telephone ;
-- type d'adresse
CREATE OR REPLACE TYPE Type_Adresse AS OBJECT (
    Ligne1 VARCHAR(80) ,
    Ligne2 VARCHAR(80) ,
    Ligne3 VARCHAR(80) ,
    CodePostal VARCHAR(5) ,
    Ville VARCHAR(25) ,
    SiteWeb URITYPE ,
    MEMBER FUNCTION DepartAdresse RETURN VARCHAR ,
    PRAGMA RESTRICT_REFERENCES(DepartAdresse,WNDS) ) ; -- ne met pas à jour la BD
-- type d'immatriculation de voiture
CREATE OR REPLACE TYPE Type_ImmatVoiture AS OBJECT (
    Chiffres NUMBER(4) ,
    Lettres VARCHAR(3) ,
    Depart VARCHAR(3) ,
    MEMBER FUNCTION ConcatNoImmat(separ IN VARCHAR) RETURN VARCHAR ,
    PRAGMA RESTRICT_REFERENCES(ConcatNoImmat,WNDS) ) ;
-- type d'immatriculations de voitures
CREATE OR REPLACE TYPE Type_ImmatVoitures AS TABLE OF Type_ImmatVoiture ;
-- type de voiture
CREATE OR REPLACE TYPE Type_Voiture AS OBJECT (
    NoImmat Type_ImmatVoiture ,
    Couleur VARCHAR(10) ) ;
-- type des voitures
CREATE OR REPLACE TYPE Type_Voitures AS TABLE OF Type_Voiture ;
-- type de diplôme
CREATE OR REPLACE TYPE Type_Diplome AS OBJECT (
    IntitAbrege CHAR(5) ,
    IntitCompleto VARCHAR(80) ) ;
-- type de diplôme obtenu
CREATE OR REPLACE TYPE Type_DiplomeObtenu AS OBJECT (
    DiplomeObtenu REF Type_Diplome , -- pointeur logique vers 1 objet d'1 table objet
    Annee NUMBER(4) ) ;
-- type des diplômes obtenus
CREATE OR REPLACE TYPE Type_DiplomesObtenus AS TABLE OF Type_DiplomeObtenu ;
-- type de personne
CREATE OR REPLACE TYPE Type_Personne AS OBJECT (
    IdPersonne NUMBER(2) ,
    NomPersonne VARCHAR(30) ,
    PrenomsPersonne Type_Prenoms ,
    TelephonesPersonne Type_Telephones ,
    AdressePersonne Type_Adresse ,
    ORDER MEMBER FUNCTION Type_Personne_Compare(p IN Type_Personne) RETURN INTEGER ,
    PRAGMA RESTRICT_REFERENCES(Type_Personne_Compare,WNDS) ,
    MEMBER FUNCTION NomPrenomPersonne RETURN VARCHAR ,
    PRAGMA RESTRICT_REFERENCES(NomPrenomPersonne,WNDS) ,
    MEMBER FUNCTION InitialePrenomPersonne RETURN VARCHAR ,
    PRAGMA RESTRICT_REFERENCES(InitialePrenomPersonne,WNDS) ,
    MEMBER PROCEDURE PrenomPersonne_Insert(p IN VARCHAR) ,
    MEMBER PROCEDURE PrenomPersonne_Update(p_old IN VARCHAR,p_new IN VARCHAR) ,
    MEMBER PROCEDURE PrenomPersonne_Delete(p IN VARCHAR)
)
NOT FINAL ;
```

```

-- type d'étudiant
CREATE OR REPLACE TYPE Type_Etudiant UNDER Type_Personne ( -- héritage de type
  DepartNaissEtu VARCHAR(3) ,
  PseudoEtu BLOB ,
  PhotoEtu ORDSYS.ORDIMAGE ,
  SignPhotoEtu ORDSYS.ORDIMAGESIGNATURE , -- pour les comparaisons des images
  CVEtu XMLTYPE ,
  VoituresPossedees Type_Voitures ,
  DiplomesObtenus Type_DiplomesObtenus ,
  MEMBER FUNCTION EstGirondin RETURN NUMBER ,
  PRAGMA RESTRICT_REFERENCES(EstGirondin,WNDS) ,
  MEMBER PROCEDURE DepartNaissEtu_Update(d IN VARCHAR)
)
INSTANTIABLE
FINAL ;

```

Création des tables et des contraintes d'intégrité

Les contraintes d'intégrité implantées ici sont : clés primaires, référentielles, d'unicité, existentielles, de valeurs par défaut, etc.

```

-- table des départements
CREATE TABLE Departements (
  CodeDepartement VARCHAR(3) NOT NULL ,
  NomDepartement VARCHAR(25) NOT NULL ,
  CONSTRAINT ClePrimaire_Departements PRIMARY KEY ( CodeDepartement ) ,
  CONSTRAINT Unicite_NomDepartement UNIQUE ( NomDepartement ) ) ;
-- table des diplômes
CREATE TABLE Diplomes OF Type_Diplome (
  CONSTRAINT ClePrimaire_Diplomes PRIMARY KEY ( IntitAbrege ) ,
  CONSTRAINT Existe_IntitAbrege CHECK ( IntitAbrege IS NOT NULL ) ,
  CONSTRAINT Unicite_IntitCompleet UNIQUE ( IntitCompleet ) ,
  CONSTRAINT Existe_IntitCompleet CHECK ( IntitCompleet IS NOT NULL ) ) ;
-- table des étudiants
CREATE TABLE Etudiants OF Type_Etudiant (
  CONSTRAINT ClePrimaire_Etudiants PRIMARY KEY ( IdPersonne ) ,
  CONSTRAINT Contrainte_IdPersonnePositif CHECK ( IdPersonne > 0 ) ,
  CONSTRAINT Existe_IdPersonne IdPersonne NOT NULL ,
  CONSTRAINT Existe_NomPersonne NomPersonne NOT NULL ,
  CONSTRAINT Existe_LigneAdrPersonne
  CHECK ( AdressePersonne.Ligne1 IS NOT NULL ) ,
  CONSTRAINT Existe_CodePostalAdrPersonne
  CHECK ( AdressePersonne.CodePostal IS NOT NULL ) ,
  CONSTRAINT Existe_VilleAdrPersonne CHECK ( AdressePersonne.Ville IS NOT NULL ) ,
  CONSTRAINT Ref_Etudiants_Departements
  FOREIGN KEY ( DepartNaissEtu ) REFERENCES Departements ( CodeDepartement ) ,
  CONSTRAINT Existe_DepartNaissEtu DepartNaissEtu NOT NULL
)
NESTED TABLE TelephonesPersonne STORE AS Table_TelephonesPersonne ,
XMLTYPE COLUMN CVEtu STORE AS BINARY XML
  XMLSCHEMA "fiche_philosophe.xsd" ELEMENT "Philosophe" ,
NESTED TABLE VoituresPossedees STORE AS Table_VoituresPossedees ,
NESTED TABLE DiplomesObtenus STORE AS Table_DiplomesObtenus ;
ALTER TABLE Etudiants MODIFY (
  CONSTRAINT Defaut_DepartNaissEtu DepartNaissEtu DEFAULT '33' ) ;
ALTER TABLE Table_VoituresPossedees ADD (
  CONSTRAINT Unicite_NoImmat
  UNIQUE ( NoImmat.Chiffres , NoImmat.Lettres , NoImmat.Depart ) ,
  CONSTRAINT Existe_NoImmatChiffres CHECK ( NoImmat.Chiffres IS NOT NULL ) ,
  CONSTRAINT Contrainte_NoImmatChiffresBorn
  CHECK ( NoImmat.Chiffres BETWEEN 1 AND 9999 ) ,
  CONSTRAINT Existe_NoImmatLettres CHECK ( NoImmat.Lettres IS NOT NULL ) ,
  CONSTRAINT Contrainte_NoImmatLettresMajus
  CHECK ( NoImmat.Lettres = UPPER(NoImmat.Lettres) ) ,
  CONSTRAINT Contrainte_ListeCouleurs
  CHECK ( Couleur IS NULL OR Couleur IN ( 'rouge' , 'jaune' , 'orange' ) ) ) ;

```

```

--ALTER TABLE Table_VoituresPossedees DISABLE CONSTRAINT Unicite_NoImmat ;
--ALTER TABLE Table_VoituresPossedees ENABLE CONSTRAINT Unicite_NoImmat ;
--ALTER TABLE Table_DiplomesObtenus ADD ( SCOPE FOR ( DiplomeObtenu ) IS Diplomes ) ;
CREATE UNIQUE INDEX Unicite_IdPersonne_Annee
  ON Table_DiplomesObtenus ( NESTED_TABLE_ID , Annee ) ;
-- table des informations spatiales sur les étudiants
CREATE TABLE Etudiants_Geo (
  IdPersonne NUMBER(2) PRIMARY KEY NOT NULL REFERENCES Etudiants ( IdPersonne ) ,
  PosGeogEtu MDSYS.SDO_GEOMETRY ,
  FigGeomEtu MDSYS.SDO_GEOMETRY ) ;

```

Création des métadonnées pour Oracle Spatial

```

INSERT INTO USER_SDO_GEOM_METADATA VALUES (
  'ETUDIANTS_GEO' , -- nom de la table contenant l'attribut MDSYS.SDO_GEOMETRY
  'FIGGEOMETU' , -- nom de l'attribut MDSYS.SDO_GEOMETRY : figure géométrique étds
  MDSYS.SDO_DIM_ARRAY ( -- grille [0;20]*[0;10] avec une tolérance au 100ème près
    MDSYS.SDO_DIM_ELEMENT ( 'abscisse (X)' , 0 , 20 , 0.01 ) ,
    MDSYS.SDO_DIM_ELEMENT ( 'ordonnées (Y)' , 0 , 10 , 0.01 ) ) ,
  NULL ) ;
COMMIT ;

```

Création des index (dont les index spatiaux)

```

-- index sur le nom des étudiants
CREATE INDEX Index_NomEtu ON Etudiants ( NomPersonne ASC ) ;
-- index QuadRree sur la figure géométrique des étudiants
--CREATE INDEX Index_QuadTree_FigGeomEtu ON Etudiants_Geo ( FigGeomEtu )
--  INDEXTYPE IS MDSYS.SPATIAL_INDEX PARAMETERS('sdo_level=4') ;
--  ou ('sdo_numtiles=100')
-- index R-tree sur la figure géométrique des étudiants
CREATE INDEX Index_RTree_FigGeomEtu ON Etudiants_Geo ( FigGeomEtu )
  INDEXTYPE IS MDSYS.SPATIAL_INDEX ; -- ('sdo_index_dims=2') par défaut

```

Création des vues

```

-- vue sur les voitures des étudiants
CREATE OR REPLACE VIEW Vue_VoituresEtudiants AS
  SELECT V.NoImmat.Chiffres AS NoImmatChiffres ,
         V.NoImmat.Lettres AS NoImmatLettres ,
         V.NoImmat.Depart AS NoImmatDepart , V.Couleur ,
         IdPersonne , NomPersonne , DepartNaissEtu
  FROM Etudiants E , TABLE ( E.VoituresPossedees ) V ;

```

Création des corps de types

```

-- corps de type d'adresse
CREATE OR REPLACE TYPE BODY Type_Adresse IS
  -- calcul du département de l'adresse à partir du code postal
  MEMBER FUNCTION DepartAdresse
    RETURN VARCHAR
  IS
  BEGIN
    IF SUBSTR(SELF.CodePostal,1,2) = '97' THEN
      RETURN(SUBSTR(SELF.CodePostal,1,3)) ; -- départements d'outre-mer
    ELSE
      RETURN(SUBSTR(SELF.CodePostal,1,2)) ; -- métropole et Corse
    END IF ;
  END ;
END ;
-- corps de type d'immatriculation de voiture
CREATE OR REPLACE TYPE BODY Type_ImmatVoiture IS
  -- concatène le numéro d'immatriculation de la voiture
  MEMBER FUNCTION ConcatNoImmat (
    separ IN VARCHAR ) -- séparateur
    RETURN VARCHAR
  IS

```

```

        c CHAR(1) ; -- caractère de séparation
BEGIN
    IF separ IS NULL OR LENGTH(separ) = 0 THEN
        c := ' ' ; -- un espace par défaut
    ELSE
        c := SUBSTR(separ,1,1) ; -- 1er caractère chaîne non vide en paramètre
    END IF ;
    RETURN LPAD(CAST(SELF.Chiffres AS VARCHAR),4,'0') || c ||
        UPPER(TRIM(SELF.Lettres)) || c || SELF.Depart ;
END ;
END ;
-- corps de type de personne
CREATE OR REPLACE TYPE BODY Type_Personne IS
    -- comparaison de deux personnes
ORDER MEMBER FUNCTION Type_Personne_Compare (
    p IN Type_Personne ) -- personne à comparer
    RETURN INTEGER
IS
BEGIN
    IF SELF.IdPersonne = p.IdPersonne THEN
        RETURN 0 ; -- égalité entre les deux personnes
    ELSIF SELF.IdPersonne < p.IdPersonne THEN
        RETURN -1 ; -- "personne courante" plus petite que celle en paramètre
    ELSE
        RETURN +1 ; -- "personne courante" plus grande que celle en paramètre
    END IF ;
END ;
-- nom et 1er des prénoms de la personne
MEMBER FUNCTION NomPrenomPersonne
    RETURN VARCHAR
IS
BEGIN
    IF SELF.PrenomsPersonne.COUNT = 0 THEN
        RETURN(TRIM(SELF.NomPersonne)) ; -- que le nom d'une personne sans prénom
    ELSE
        RETURN(TRIM(SELF.NomPersonne) || ' ' || TRIM(SELF.PrenomsPersonne(1))) ;
    END IF ;
END ;
-- initiale du 1er des prénoms de la personne
MEMBER FUNCTION InitialePrenomPersonne
    RETURN VARCHAR
IS
    ip VARCHAR(45) ; -- initiale du prénom
    p VARCHAR(30) ; -- prénom
    i INTEGER ; -- indice de p
BEGIN
    IF SELF.PrenomsPersonne.COUNT = 0 THEN
        ip := NULL ; -- personne sans prénom
    ELSE
        p := TRIM(SELF.PrenomsPersonne(1)) ;
        IF p IS NULL OR LENGTH(p) = 0 THEN
            ip := NULL ; -- personne sans réellement de prénom
        ELSE
            ip := SUBSTR(p,1,1) || '.' ; -- initiale du 1er prénom
            i := 2 ;
            WHILE i < LENGTH(p) LOOP
                IF SUBSTR(p,i,1) IN ( '-' , ' ' ) AND
                    UPPER(SUBSTR(p,i+1,1)) BETWEEN 'A' AND 'Z' THEN
                    ip := ip || SUBSTR(p,i,2) || '.' ; -- composé ou multiple
                    i := i + 2 ;
                ELSE
                    i := i + 1 ;
                END IF ;
            END LOOP ;
        END IF ;
    END IF ;
END ;

```

```

        END IF ;
        RETURN(ip) ;
    END ;
-- insertion d'un prénom de la personne
MEMBER PROCEDURE PrenomPersonne_Insert (
    p IN VARCHAR ) -- prénom à insérer
IS
BEGIN
    IF SELF.PrenomsPersonne.COUNT < SELF.PrenomsPersonne.LIMIT THEN
        SELF.PrenomsPersonne.EXTEND() ; -- ajoute un prénom NULL
        SELF.PrenomsPersonne(SELF.PrenomsPersonne.COUNT) := TRIM(p) ; --l'affecte
    END IF ;
END ;
-- modification d'un prénom de la personne
MEMBER PROCEDURE PrenomPersonne_Update (
    p_old IN VARCHAR , -- ancien prénom à modifier
    p_new IN VARCHAR ) -- nouveau prénom à modifier
IS
    i INTEGER ; -- indice des prénoms
BEGIN
    FOR i IN 1..SELF.PrenomsPersonne.COUNT LOOP
        IF TRIM(SELF.PrenomsPersonne(i)) = TRIM(p_old) THEN
            SELF.PrenomsPersonne(i) := TRIM(p_new) ; -- modifie le prénom
        END IF ;
    END LOOP ;
END ;
-- suppression (de toutes les occurrences) d'un prénom de la personne
MEMBER PROCEDURE PrenomPersonne_Delete (
    p IN VARCHAR ) -- prénom à supprimer
IS
    i INTEGER ; -- indice des prénoms
BEGIN
    i := SELF.PrenomsPersonne.FIRST ;
    WHILE i <= SELF.PrenomsPersonne.LAST LOOP
        IF TRIM(SELF.PrenomsPersonne(i)) = TRIM(p) THEN
            SELF.PrenomsPersonne(i) :=
                SELF.PrenomsPersonne(SELF.PrenomsPersonne.LAST) ;
            -- remplace le prénom à supprimer par le dernier prénom
            SELF.PrenomsPersonne.TRIM(1) ; -- supprime le dernier prénom
        ELSE
            i := i + 1 ;
        END IF ;
    END LOOP ;
END ;
END ;
-- corps de type d'étudiant
CREATE OR REPLACE TYPE BODY Type_Etudiant IS
    -- indique si l'étudiant est girondin
    MEMBER FUNCTION EstGirondin
        RETURN NUMBER
    IS
    BEGIN
        IF SELF.DepartNaissEtu = '33' THEN
            RETURN(1) ; -- est girondin
        ELSE
            RETURN(0) ; -- n'est pas girondin
        END IF ;
    END ;
-- modification du département de naissance d'un étudiant
MEMBER PROCEDURE DepartNaissEtu_Update (
    d IN VARCHAR ) -- département de naissance à modifier
IS
BEGIN
    UPDATE Etudiants E
        SET E.DepartNaissEtu = d

```

```

WHERE VALUE(E) = SELF ; -- requiert Type_Personne_Compare()
END ;
END ;

```

Création des déclencheurs

```

-- déclenchement avant l'insertion ou la modification d'étudiants
CREATE OR REPLACE TRIGGER Declen_AvInsertUpdate_Etudiant
BEFORE
INSERT OR UPDATE
ON Etudiants
FOR EACH ROW
DECLARE
ivp INTEGER ; -- indice des voitures possédées par l'étudiant
ido INTEGER ; -- indice des diplômes obtenus par l'étudiant
jdo INTEGER ; -- indice des diplômes obtenus par l'étudiant
BEGIN
-- contrainte que le début du code postal doit correspondre à un département
IF EstCodeDepartement(:NEW.AdressePersonne.DepartAdresse()) = 0 THEN
RAISE_APPLICATION_ERROR(-20001,'Contrainte d'intégrité référentielle
Etudiants.AdressePersonne.CodePostal /
Departements.CodeDepartement violée pour '||
:NEW.AdressePersonne.DepartAdresse() ;
END IF ;
-- contraintes sur les voitures possédées
-- (valeurs par défaut et contrainte d'intégrité référentielle du département de
-- l'immatriculation sur les départements)
IF :NEW.VoituresPossedees IS NOT NULL AND
:NEW.VoituresPossedees.FIRST IS NOT NULL THEN
FOR ivp IN :NEW.VoituresPossedees.FIRST..:NEW.VoituresPossedees.LAST LOOP
IF :NEW.VoituresPossedees.EXISTS(ivp) THEN
-- CONSTRAINT Defaut_NoImmatDepart Depart DEFAULT '33'
IF :NEW.VoituresPossedees(ivp).NoImmat.Depart IS NULL THEN
:NEW.VoituresPossedees(ivp).NoImmat.Depart := '33' ;
END IF ;
-- CONSTRAINT Ref_ImmatVoiture_Departements FOREIGN KEY
-- ( Depart ) REFERENCES Departements ( CodeDepartement )
IF EstCodeDepartement(:NEW.VoituresPossedees(ivp).NoImmat.Depart) = 0
THEN
RAISE_APPLICATION_ERROR(-20002,'Contrainte d'intégrité
référentielle Etudiants.
VoituresPossedees[.NoImmat.Depart /
Departements.CodeDepartement violée
pour '||:NEW.VoituresPossedees(ivp).
NoImmat.Depart||' (voiture n° '||
CAST(ivp AS VARCHAR)||')' ) ;
END IF ;
-- CONSTRAINT Defaut_Couleur CHECK ( Couleur DEFAULT 'rouge' )
IF :NEW.VoituresPossedees(ivp).Couleur IS NULL THEN
:NEW.VoituresPossedees(ivp).Couleur := 'rouge' ;
END IF ;
END IF ;
END LOOP ;
END IF ;
-- contrainte d'unicité sur les diplômes obtenus par les étudiants
IF :NEW.DiplomesObtenus IS NOT NULL AND :NEW.DiplomesObtenus.FIRST IS NOT NULL
THEN
FOR ido IN :NEW.DiplomesObtenus.FIRST..:NEW.DiplomesObtenus.LAST-1 LOOP
IF :NEW.DiplomesObtenus.EXISTS(ido) THEN
FOR jdo IN ido+1..:NEW.DiplomesObtenus.LAST LOOP
IF :NEW.DiplomesObtenus.EXISTS(jdo) THEN
-- CONSTRAINT Unicite_IdPersonne_DiplomeObtenu
-- UNIQUE ( IdPersonne , DiplomeObtenu )
IF :NEW.DiplomesObtenus(ido).DiplomeObtenu =
:NEW.DiplomesObtenus(jdo).DiplomeObtenu THEN

```



```

        RAISE_APPLICATION_ERROR(-20003,'Contrainte d''unicité (
        Etudiants.IdPersonne , Etudiants.
        DiplomesObtenus.DiplomeObtenu )
        violée pour (diplôme n° '||
        CAST(ido AS VARCHAR)||' et '||
        CAST(jdo AS VARCHAR)||')') ;
    END IF ;
END IF ;
END LOOP ;
END IF ;
END LOOP ;
END IF ;
END ;
-- déclenchement avant l'insertion ou la modification des informations spatiales étlds
CREATE OR REPLACE TRIGGER Declen_AvInsertUpdate_Etu_Geo
BEFORE
INSERT OR UPDATE
ON Etudiants_Geo
FOR EACH ROW
DECLARE
    c NUMBER ; -- nb. cas d'incompatibilité figure géométrique avec métadonnées
BEGIN
    -- posi° géographique, SRID 8307 (WGS 84) : -90≤latitude≤90 et -180≤longitude≤180
    IF :NEW.PosGeogEtu.SDO_SRID = 8307 THEN
        IF :NEW.PosGeogEtu.SDO_POINT.X < -90 OR :NEW.PosGeogEtu.SDO_POINT.X > 90 THEN
            RAISE_APPLICATION_ERROR(-20004,'La latitude, dans la référence spatiale '
            ||:NEW.PosGeogEtu.SDO_SRID||', '||
            :NEW.PosGeogEtu.SDO_POINT.X||
            ' doit être comprise entre -90° et +90°') ;
        ELSIF :NEW.PosGeogEtu.SDO_POINT.Y < -180 OR
            :NEW.PosGeogEtu.SDO_POINT.Y > 180 THEN
            RAISE_APPLICATION_ERROR(-20005,
            'La longitude, dans la référence spatiale '||
            :NEW.PosGeogEtu.SDO_SRID||', '||
            :NEW.PosGeogEtu.SDO_POINT.Y||
            ' doit être comprise entre -180° et +180°') ;
        END IF ;
    END IF ;
    -- la figure géométrique doit être compatible avec les métadonnées
    SELECT COUNT(*)
    INTO c
    FROM USER_SDO_GEOM_METADATA M
    WHERE M.TABLE_NAME = 'ETUDIANTS_GEO' AND M.COLUMN_NAME = 'FIGGEOMETU' AND
        SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT(:NEW.FigGeomEtu,M.DIMINFO) <>
        'TRUE' ;
    IF c > 0 THEN
        RAISE_APPLICATION_ERROR(-20006,'La figure géométrique n'est pas correcte ou
        ne respecte pas les contraintes définies sur
        l'attribut (dans USER_SDO_GEOM_METADATA)') ;
    END IF ;
END ;

```

Création des fonctions et procédures

```

-- indique si un code du département en paramètre est trouvé parmi les départements
CREATE OR REPLACE FUNCTION EstCodeDepartement (
    cd IN VARCHAR ) -- code du département à rechercher
RETURN NUMBER
IS
    trouv NUMBER(1) ; -- le code du département est trouvé parmi les départements
BEGIN
    SELECT COUNT(*) INTO trouv FROM Departements WHERE CodeDepartement = cd ;
    IF trouv = 0 THEN
        RETURN(0) ; -- n'est pas un département
    ELSE

```

```

        RETURN(1) ; -- est un département
    END IF ;
END ;
-- insère (et valide définitivement) le pseudonyme des étudiants
CREATE OR REPLACE PROCEDURE InserePseudoEtudiants (
    idp IN Etudiants.IdPersonne%TYPE , -- identifiant de la personne
    ficpseudo IN VARCHAR ) -- fichier contenant le pseudonyme à insérer
IS
    pseudo BLOB ; -- pseudonyme de l'étudiant
    f BFILE ; -- pointeur de fichier en entrée
    psd PLS_INTEGER := 1 ; -- position source de départ
    pdd PLS_INTEGER := 1 ; -- position destination de départ
    nr VARCHAR(15) ; -- nom du répertoire où doivent se trouver les informations
    cr VARCHAR2(4000) ; -- chemin du répertoire où doivent se trouver informations
BEGIN
    nr := UPPER('Rep_Etudiants') ;
    IF ficpseudo IS NOT NULL THEN
        f := BFILENAME(nr, ficpseudo) ;
        IF DBMS_LOB.FILEEXISTS(f) = 1 THEN
            SELECT PseudoEtu
                INTO pseudo
                FROM Etudiants
                WHERE IdPersonne = idp
                FOR UPDATE ;
            DBMS_LOB.FILEOPEN(f, DBMS_LOB.FILE_READONLY) ;
            DBMS_LOB.LOADBLOBFROMFILE(pseudo, f, DBMS_LOB.LOBMAXSIZE, psd, pdd) ;
            DBMS_LOB.FILECLOSE(f) ;
            UPDATE Etudiants SET PseudoEtu = pseudo WHERE IdPersonne = idp ;
            COMMIT ;
            DBMS_OUTPUT.PUT_LINE('Pseudonyme inséré') ;
        ELSE
            SELECT DIRECTORY_PATH
                INTO cr
                FROM ALL_DIRECTORIES
                WHERE DIRECTORY_NAME = nr ;
            DBMS_OUTPUT.PUT_LINE('Le fichier ' || ficpseudo ||
                ' n''existe pas dans le répertoire ' || cr) ;
        END IF ;
    ELSE
        DBMS_OUTPUT.PUT_LINE('Aucun fichier n''a été précisé') ;
    END IF ;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Aucun étudiant n''a le n° ' || CAST(idp AS VARCHAR)) ;
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20000, 'Erreur non gérée de message ' || SQLERRM ||
            ' et de code ' || CAST(SQLCODE AS VARCHAR)) ;
END ;
-- importe localement les photographies des étudiants dans la base de données
CREATE OR REPLACE PROCEDURE ImportePhotosEtudiants
IS
    photo ORDSYS.ORDIMAGE ; -- photographie de l'étudiant à importer
    ctx RAW(4000) := NULL ; -- contexte
BEGIN
    FOR Curs_Etd IN ( SELECT IdPersonne , PhotoEtu FROM Etudiants ) LOOP
        IF NOT Curs_Etd.PhotoEtu.ISLOCAL THEN
            SELECT PhotoEtu
                INTO photo
                FROM Etudiants E
                WHERE E.IdPersonne = Curs_Etd.IdPersonne
                FOR UPDATE ;
            photo.IMPORTFROM(ctx, photo.SOURCE.SRCTYPE, photo.SOURCE.SRCLOCATION,
                photo.SOURCE.SRCNAME) ;
            UPDATE Etudiants E
                SET E.PhotoEtu = photo

```

```

        WHERE E.IdPersonne = Curs_Etd.IdPersonne ;
    END IF;
END LOOP ;
COMMIT ;
DBMS_OUTPUT.PUT_LINE('Photographies importées (localement) dans la base de
                        données') ;
END ;
-- affecte les propriétés des photographies des étudiants et génère leurs signatures
CREATE OR REPLACE PROCEDURE AffectePropPhotosEtudiants
IS
    photo ORDSYS.ORDIMAGE ; -- photographie de étudiant dont recherche les propriétés
    signphoto ORDSYS.ORDIMAGESIGNATURE ; -- signature de la photographie de étudiant
BEGIN
    FOR Curs_Etd IN ( SELECT IdPersonne , SignPhotoEtu , PhotoEtu FROM Etudiants )
        LOOP
            SELECT PhotoEtu , SignPhotoEtu
                INTO photo , signphoto
                FROM Etudiants E
                WHERE E.IdPersonne = Curs_Etd.IdPersonne
                FOR UPDATE ;
            photo.SETPROPERTIES() ;
            signphoto.GENERATESIGNATURE(photo) ;
            UPDATE Etudiants E
                SET E.PhotoEtu = photo
                WHERE E.IdPersonne = Curs_Etd.IdPersonne ;
        END LOOP ;
    COMMIT ;
    DBMS_OUTPUT.PUT_LINE('Propriétés des photographies affectées et signatures
                        générées') ;
END ;
-- compare deux à deux les signatures des photographies des étudiants
-- N. B. : signature = (couleur, texture, forme) chaque zone image + fond image
-- N. B. : score = valeur de comparaison de 2 images de 0 (du + similaire) à 100
CREATE OR REPLACE PROCEDURE CompareSignPhotosEtudiants (
    d IN FLOAT ) -- distance (entre 0 et 100) maximale entre deux scores
IS
    c VARCHAR(50) ; -- commande évaluation score comparaison photographies étudiants
BEGIN
    c := 'color=0.0,texture=0.5,shape=1.0,location=0.0' ;
    -- scores des comparaisons deux à deux des photographies des étudiants
    DBMS_OUTPUT.PUT_LINE('Les scores des comparaisons deux à deux des photographies
                        des étudiants sont les suivants (avec ''||c||'') :') ;
    FOR Curs_2Etds IN ( SELECT E1.IdPersonne IdPersonnel ,
                            E1.SignPhotoEtu SignPhotoEtu1 ,
                            E2.IdPersonne IdPersonne2 ,
                            E2.SignPhotoEtu SignPhotoEtu2
                        FROM Etudiants E1 , Etudiants E2
                        WHERE E1.IdPersonne < E2.IdPersonne
                        ORDER BY E1.IdPersonne , E2.IdPersonne ) LOOP
        DBMS_OUTPUT.PUT_LINE(' > ''||CAST(ORDSYS.ORDIMAGESIGNATURE.EVALUATESCORE(
                            Curs_2Etds.SignPhotoEtu1,Curs_2Etds.SignPhotoEtu2,c) AS
                            VARCHAR)||' % entre les étudiants n° ''||CAST(
                            Curs_2Etds.IdPersonnel AS VARCHAR)||' et ''||
                            CAST(Curs_2Etds.IdPersonne2 AS VARCHAR)) ;
    END LOOP ;
    -- les photographies des étudiants les plus similaires
    DBMS_OUTPUT.PUT_LINE('Les photographies des étudiants les plus similaires sont
                        les suivantes (avec ''||c||' et distantes de ''||
                        CAST(d AS VARCHAR)||'% au maximum) :') ;
    FOR Curs_2Etds IN ( SELECT E1.IdPersonne IdPersonnel ,
                            E2.IdPersonne IdPersonne2 ,
                            ORDSYS.ORDIMAGESIGNATURE.EVALUATESCORE(
                            E1.SignPhotoEtu,E2.SignPhotoEtu,c) Score
                        FROM Etudiants E1 , Etudiants E2
                        WHERE E1.IdPersonne < E2.IdPersonne AND ORDSYS.IMGSIMILAR(

```

```

        E1.SignPhotoEtu,E2.SignPhotoEtu,c,d) = 1
        ORDER BY Score , E1.IdPersonne , E2.IdPersonne ) LOOP
DBMS_OUTPUT.PUT_LINE(' > ' || CAST(Curs_2Etds.Score AS VARCHAR) ||
        ' % entre les étudiants n° ' ||
        CAST(Curs_2Etds.IdPersonnel AS VARCHAR) ||
        ' et ' || CAST(Curs_2Etds.IdPersonne2 AS VARCHAR)) ;
END LOOP ;
END ;

```

Création des paquetages (et corps de paquetages)

Deux paquetages sont ici présentés : le premier illustre la surcharge, les exceptions et les curseurs gérés par FOR tandis que le second vérifie des contraintes d'intégrité *a posteriori*.

Illustration de la surcharge, des exceptions et des curseurs gérés par FOR

```

-- affiche le nombre de diplômes obtenus par les étudiants
-- soit total, soit d'une année, soit d'un diplôme et d'une année
CREATE OR REPLACE PACKAGE NbdDiplObtEtd IS
  PROCEDURE NombreDiplomes ;
  PROCEDURE NombreDiplomes (
    a IN NUMBER ) ;
  PROCEDURE NombreDiplomes (
    iad IN Diplomes.IntitAbrege%TYPE ,
    a IN NUMBER ) ;
END ;
-- corps du paquetage du nombre de diplômes obtenus par les étudiants
CREATE OR REPLACE PACKAGE BODY NbdDiplObtEtd IS
  -- nombre total de diplômes obtenus par les étudiants
  PROCEDURE NombreDiplomes
  IS
    nbd INTEGER ; -- nombre de diplômes obtenus par les étudiants
  BEGIN
    SELECT SUM(CARDINALITY(DiplomesObtenus)) INTO nbd FROM Etudiants ;
    DBMS_OUTPUT.PUT_LINE('Nombre de diplômes obtenus par les étudiants : ' ||
      CAST(nbd AS VARCHAR)) ;
  END ;
  -- nombre diplômes obtenus par étudiants pour 1 année, en interrompant la liste
  PROCEDURE NombreDiplomes (
    a IN NUMBER ) -- année d'obtention d'un diplôme
  IS
    nbd INTEGER ; -- nombre de diplômes obtenus par les étudiants
    nbdmax INTEGER ; -- nombre diplômes obtenus par étudiants maximum à afficher
    idp Etudiants.IdPersonne%TYPE ; -- identifiant de la personne
    iad Diplomes.IntitAbrege%TYPE ; -- intitulé abrégé du diplôme
    CURSOR Curs_DO IS
      SELECT E.IdPersonne , Deref(EDO.DiplomeObtenu).IntitAbrege
      FROM Etudiants E , TABLE ( E.DiplomesObtenus ) EDO
      WHERE EDO.Annee = a
      ORDER BY E.IdPersonne ;
    Exception_ArretAffichage EXCEPTION ;
  BEGIN
    nbdmax := 2 ;
    nbd := 0 ;
    OPEN Curs_DO ;
    LOOP
      FETCH Curs_DO INTO idp , iad ;
      EXIT WHEN Curs_DO%NOTFOUND ;
      nbd := nbd + 1 ;
      DBMS_OUTPUT.PUT_LINE(CAST(nbd AS VARCHAR) || ' : diplôme intitulé ' || iad ||
        ' obtenu l''année ' || CAST(a AS VARCHAR) ||
        ' par l''étudiant n° ' || CAST(idp AS VARCHAR)) ;
      IF nbd >= nbdmax THEN
        CLOSE Curs_DO ;
        RAISE Exception_ArretAffichage ; -- on génère l'exception ad hoc
      END IF ;
    END LOOP ;
  END ;

```

```

END LOOP ;
CLOSE Curs_DO ;
IF nbd = 0 THEN
    DBMS_OUTPUT.PUT_LINE('Aucun diplôme n''a été obtenu l''année '||
                          CAST(a AS VARCHAR)||' par les étudiants') ;
ELSE
    DBMS_OUTPUT.PUT_LINE('Nombre de diplômes obtenus l''année '||
                          CAST(a AS VARCHAR)||
                          ' par les étudiants : '||CAST(nbd AS VARCHAR)) ;
END IF ;
-- produit automatiquement exception NO_DATA_FOUND (aucune ligne retournable)
SELECT IdPersonne INTO idp FROM Etudiants WHERE 0 = 1 ;
EXCEPTION
WHEN Exception_ArretAffichage THEN
    DBMS_OUTPUT.PUT_LINE('L''affichage de la liste a été interrompu suite
                          au '||CAST(nbdmax AS VARCHAR)||'ème étudiant') ;
WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Aucun étudiant ne peut vérifier le prédicat
                          (toujours faux i. e. contradiction)') ;
WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20000,'Erreur non gérée de message '||SQLERRM||
                              ' et de code '||CAST(SQLCODE AS VARCHAR)) ;
END ;
-- nombre de diplômes obtenus par les étudiants pour un diplôme et une année
PROCEDURE NombreDiplomes (
    iad IN Diplomes.IntitAbrege%TYPE , -- intitulé abrégé du diplôme
    a IN NUMBER ) -- année d'obtention d'un diplôme
IS
    nbd INTEGER ; -- nombre de diplômes obtenus par les étudiants
    CURSOR Curs_DO IS
        SELECT E.IdPersonne
        FROM Etudiants E , TABLE ( E.DiplomesObtenus ) EDO
        WHERE Deref(EDO.DiplomeObtenu).IntitAbrege = iad AND EDO.Annee = a
        ORDER BY E.IdPersonne ;
BEGIN
    nbd := 0 ;
    FOR Enreg_DO IN Curs_DO LOOP -- curseur tq ouverture, lectures, fermeture
        nbd := nbd + 1 ;
        DBMS_OUTPUT.PUT_LINE(CAST(Curs_DO%ROWCOUNT AS VARCHAR)||
                              ' : diplôme intitulé '||iad||
                              ' obtenu l''année '||CAST(a AS VARCHAR)||
                              ' par l''étudiant n° '||
                              CAST(Enreg_DO.IdPersonne AS VARCHAR)) ;
    END LOOP ;
    DBMS_OUTPUT.PUT_LINE('Nombre de diplômes intitulé '||iad||
                          ' obtenus l''année '||CAST(a AS VARCHAR)||
                          ' par les étudiants : '||CAST(nbd AS VARCHAR)) ;
END ;
END ;

```

Vérification des contraintes d'intégrité *a posteriori*

```

-- contraintes d'intégrité violées
CREATE OR REPLACE PACKAGE Civirolees IS
    PROCEDURE Defaut_NoImmatDepart_violee ;
    PROCEDURE Defaut_Couleur_violee ;
    PROCEDURE Ref_ImmatVoiture_Dpts_violee ;
    PROCEDURE Ref_Etudiants_Dipl_violee ;
    PROCEDURE Unicite_IdPersonne_Dipl_violee ;
END ;
-- corps du paquetage des contraintes d'intégrité violées
CREATE OR REPLACE PACKAGE BODY Civirolees IS
    -- affiche les données violant la contrainte d'intégrité de valeur par défaut des
    -- départements du numéro d'immatriculation des voitures possédées par les
    -- étudiants : CONSTRAINT Defaut_NoImmatDepart NoImmat.Depart DEFAULT '33'

```

```

PROCEDURE Defaut_NoImmatDepart_violee IS
  idp Etudiants.IdPersonne%TYPE ; -- identifiant de la personne
  noic NUMBER(4) ; -- Etudiants.VoituresPossedees.NoImmat.Chiffres%TYPE ;
  noil VARCHAR(3) ; -- Etudiants.VoituresPossedees.NoImmat.Lettres%TYPE ;
  CURSOR Curs_VPsansDpt IS
    SELECT IdPersonne , EVP.NoImmat.Chiffres , EVP.NoImmat.Lettres
    FROM Etudiants E , TABLE ( E.VoituresPossedees ) EVP
    WHERE EVP.NoImmat.Depart IS NULL ;
BEGIN
  OPEN Curs_VPsansDpt ;
  LOOP
    FETCH Curs_VPsansDpt INTO idp , noic , noil ;
    EXIT WHEN Curs_VPsansDpt%NOTFOUND ;
    DBMS_OUTPUT.PUT_LINE('Pas de valeur pour le département du numéro
      d'immatriculation de la voiture ' ||
      CAST(noic AS VARCHAR) || ' ' || noil ||
      ' possédée par l''étudiant n° ' ||
      CAST(idp AS VARCHAR)) ;
  END LOOP ;
  CLOSE Curs_VPsansDpt ;
END ;
-- affiche les données violant la contrainte d'intégrité de valeur par défaut des
-- départements du numéro d'immatriculation des voitures possédées par les
-- étudiants : CONSTRAINT Defaut_Couleur Couleur DEFAULT 'rouge'
PROCEDURE Defaut_Couleur_violee IS
  idp Etudiants.IdPersonne%TYPE ; -- identifiant de la personne
  noi VARCHAR(12) ; -- numéro d'immatriculation
  CURSOR Curs_VPsansCouleur IS
    SELECT IdPersonne , EVP.NoImmat.ConcatNoImmat('_')
    FROM Etudiants E , TABLE ( E.VoituresPossedees ) EVP
    WHERE EVP.Couleur IS NULL ;
BEGIN
  OPEN Curs_VPsansCouleur ;
  LOOP
    FETCH Curs_VPsansCouleur INTO idp , noi ;
    EXIT WHEN Curs_VPsansCouleur%NOTFOUND ;
    DBMS_OUTPUT.PUT_LINE('Pas de valeur pour la couleur de la voiture de
      numéro d'immatriculation ' || noi || ' possédée par
      l''étudiant n° ' || CAST(idp AS VARCHAR)) ;
  END LOOP ;
  CLOSE Curs_VPsansCouleur ;
END ;
-- affiche les données violant la contrainte d'intégrité référentielle des
-- départements du numéro d'immatriculation des voitures possédées par les
-- étudiants : CONSTRAINT Ref_ImmatVoiture_Departements FOREIGN KEY ( Depart )
-- REFERENCES Departements ( CodeDepartement )
PROCEDURE Ref_ImmatVoiture_Dpts_violee IS
  idp Etudiants.IdPersonne%TYPE ; -- identifiant de la personne
  noi VARCHAR(12) ; -- numéro d'immatriculation
  CURSOR Curs_pb_refDptVP IS
    SELECT IdPersonne , EVP.NoImmat.ConcatNoImmat('_')
    FROM Etudiants E , TABLE ( E.VoituresPossedees ) EVP , Departements D
    WHERE EVP.NoImmat.Depart = D.CodeDepartement (+) AND - jointure externe
      D.CodeDepartement IS NULL ;
BEGIN
  OPEN Curs_pb_refDptVP ;
  LOOP
    FETCH Curs_pb_refDptVP INTO idp , noi ;
    EXIT WHEN Curs_pb_refDptVP%NOTFOUND ;
    DBMS_OUTPUT.PUT_LINE('Le département de la voiture de numéro
      d'immatriculation ' || noi || ' possédée par
      l''étudiant n° ' || CAST(idp AS VARCHAR) ||
      ' ne référence pas un département') ;
  END LOOP ;
  CLOSE Curs_pb_refDptVP ;

```

```

END ;
-- affiche les données violant la contrainte d'intégrité référentielle des
-- diplômes obtenus par les étudiants : CONSTRAINT Ref_Etudiants_Diplomes
-- FOREIGN KEY ( DiplomeObtenu ) REFERENCES Diplomes
PROCEDURE Ref_Etudiants_Dipl_violee IS
  idp Etudiants.IdPersonne%TYPE ; -- identifiant de la personne
  ado NUMBER(4) ; -- année d'obtention du diplôme
  CURSOR Curs_pb_refIntitDO IS
    SELECT IdPersonne , EDO.Annee
    FROM Etudiants E , TABLE ( E.DiplomesObtenus ) EDO
    WHERE EDO.DiplomeObtenu IS DANGLING OR EDO.DiplomeObtenu IS NULL ;
BEGIN
  OPEN Curs_pb_refIntitDO ;
  LOOP
    FETCH Curs_pb_refIntitDO INTO idp , ado ;
    EXIT WHEN Curs_pb_refIntitDO%NOTFOUND ;
    DBMS_OUTPUT.PUT_LINE('Pas de valeur pour la référence à l''intitulé
                          abrégé du diplôme de l''année '||
                          CAST(ado AS VARCHAR)||' de l''étudiant n° '||
                          CAST(idp AS VARCHAR)) ;
  END LOOP ;
  CLOSE Curs_pb_refIntitDO ;
END ;
-- affiche les données violant la contrainte d'unicité des diplômes obtenus par
-- les étudiants : CONSTRAINT Unicite_IdPersonne_DiplomeObtenu UNIQUE
-- ( IdPersonne , DiplomeObtenu )
PROCEDURE Unicite_IdPersonne_Dipl_violee IS
  idp Etudiants.IdPersonne%TYPE ; -- identifiant de la personne
  iad Diplomes.IntitAbrege%TYPE ; -- intitulé abrégé du diplôme
  nba NUMBER(2) ; -- nombre années différentes pour une personne et un diplôme
  CURSOR Curs_pb_UniqIdpDO IS
    SELECT IdPersonne , Deref(EDO.DiplomeObtenu).IntitAbrege , COUNT(*)
    FROM Etudiants E , TABLE ( E.DiplomesObtenus ) EDO
    WHERE EDO.DiplomeObtenu IS NOT DANGLING AND -- référence pendante suite à
                                                -- suppress. objet référencé
          EDO.DiplomeObtenu IS NOT NULL
    GROUP BY IdPersonne , Deref(EDO.DiplomeObtenu).IntitAbrege
    HAVING COUNT(*) >= 2 ;
BEGIN
  OPEN Curs_pb_UniqIdpDO ;
  LOOP
    FETCH Curs_pb_UniqIdpDO INTO idp , iad , nba ;
    EXIT WHEN Curs_pb_UniqIdpDO%NOTFOUND ;
    DBMS_OUTPUT.PUT_LINE('Il y a '||CAST(nba AS VARCHAR)||' années
                          différentes pour la référence à l''intitulé abrégé
                          du diplôme de l''année '||iad||
                          ' de l''étudiant n° '||CAST(idp AS VARCHAR)) ;
  END LOOP ;
  CLOSE Curs_pb_UniqIdpDO ;
END ;
END ;

```

Création des droits

```

-- droits généraux pour le compte ETD
GRANT CONNECT , RESOURCE TO ETD ;
GRANT EXECUTE ANY TYPE TO ETD ;
GRANT CREATE ANY PROCEDURE TO ETD ;
GRANT EXECUTE ANY PROCEDURE TO ETD ;
GRANT SELECT ANY SEQUENCE TO ETD ;
GRANT SELECT_CATALOG_ROLE TO ETD ;
GRANT SELECT ANY TABLE TO ETD ;
-- droits liés à cette application pour le compte ETD
GRANT READ ON DIRECTORY Rep_Etudiants TO ETD ;
GRANT EXECUTE ON Type_VARRAY_VARCHAR TO ETD ;

```



```
GRANT SELECT ON Departements TO ETD ;
GRANT SELECT ON Diplomes TO ETD ;
GRANT SELECT ON Etudiants_Geo TO ETD ;
GRANT SELECT ON Etudiants TO ETD ;
```

Création des synonymes

```
-- synonymes publics (tables et vues)
CREATE PUBLIC SYNONYM Departements FOR GUIBERT.Departements ;
CREATE PUBLIC SYNONYM Diplomes FOR GUIBERT.Diplomes ;
CREATE PUBLIC SYNONYM Etudiants_Geo FOR GUIBERT.Etudiants_Geo ;
CREATE PUBLIC SYNONYM Etudiants FOR GUIBERT.Etudiants ;
CREATE PUBLIC SYNONYM Table_TelephonesPersonne FOR GUIBERT.Table_TelephonesPersonne ;
CREATE PUBLIC SYNONYM Table_VoituresPossedees FOR GUIBERT.Table_VoituresPossedees ;
CREATE PUBLIC SYNONYM Table_DiplomesObtenus FOR GUIBERT.Table_DiplomesObtenus ;
CREATE PUBLIC SYNONYM Vue_VoituresEtudiants FOR GUIBERT.Vue_VoituresEtudiants ;
```

Création de l'application

L'application, créée par le compte ETD, présente des informations de la base de données dans des pages Web à partir de <http://localhost:1158/DAD/home>.

Page d'accueil

```
CREATE OR REPLACE PROCEDURE ETD.home IS
BEGIN
    HTP.HTMLOPEN ;
    HTP.HEADOPEN ;
    HTP.HTITLE('Exemple "jouet"') ;
    HTP.HEADCLOSE ;
    HTP.BODYOPEN ;
    HTP.HEADER(2,'Bienvenue dans la gestion de l''exemple jouet (étudiants, voitures,
                diplômes, départements français)') ;
    -- affichage des tables de la BD
    HTP.BR() ;
    HTP.FORMOPEN(OWA_UTIL.GET_OWA_SERVICE_PATH||'AffTableNonObjetHTML','POST') ;
    HTP.FORMSELECTOPEN('nt','Sélectionnez l''une des tables (non objet) appartenant à
                        SCOTT ou GUIBERT de la base de données : ') ;
    FOR Curs_TablesNonObjet IN ( SELECT OWNER||'.'||TABLE_NAME tn
                                FROM ALL_TABLES
                                WHERE OWNER IN ( 'SCOTT' , 'GUIBERT' )
                                ORDER BY OWNER , TABLE_NAME ) LOOP
        HTP.FORMSELECTOPTION(Curs_TablesNonObjet.tn) ;
    END LOOP ;
    HTP.FORMSELECTCLOSE ;
    HTP.FORMSUBMIT(NULL,'Affichage de toutes les informations contenues dans la table
                    sélectionnée') ;
    HTP.FORMCLOSE ;
    -- liens d'accès à l'affichage des informations des étudiants
    HTP.BR() ;
    HTP.PRINT('Cliquez sur l''un des liens ci-dessous pour afficher les informations
              sur un étudiant : ') ;
    HTP.BR() ;
    FOR Curs_Etd IN ( SELECT IdPersonne , NomPersonne
                     FROM Etudiants
                     ORDER BY IdPersonne ) LOOP
        HTP.PRINT('<DD>') ;
        HTP.ANCHOR('AffEtuHTML?idp='||TO_CHAR(Curs_Etd.IdPersonne),
                  'Informations sur l''étudiant n° '||TO_CHAR(Curs_Etd.IdPersonne)||
                  ' ('||TO_CHAR(Curs_Etd.NomPersonne)||')') ;
    END LOOP ;
    -- horodate
    HTP.BR() ;
    HTP.BR() ;
    HTP.ITALIC('Nous sommes aujourd''hui le '||TO_CHAR(SYSDATE,'DD/MM/YYYY')||
              ' et il est '||TO_CHAR(SYSDATE,'HH24:MI:SS')||'.') ;
```

```

HTP.BODYCLOSE ;
HTP.HTMLCLOSE ;
END ;

```

Toutes les informations d'une table (non objet)

```

CREATE OR REPLACE PROCEDURE ETD.AffTableNonObjetHTML (
    nt VARCHAR DEFAULT 'Departements' )
IS
    b BOOLEAN ; -- retour de OWA_UTIL.TABLEPRINT ignoré
BEGIN
    b := OWA_UTIL.TABLEPRINT(nt, 'BORDER', OWA_UTIL.PRE_TABLE) ;
END ;

```

Toutes les informations sur un étudiant

```

-- toutes les informations d'un étudiant (dont le n° est passé en paramètre)
CREATE OR REPLACE PROCEDURE ETD.AffEtuHTML (
    idp IN Etudiants.IdPersonne%TYPE ) -- identifiant de la personne
IS
    L$NomPersonne Etudiants.NomPersonne%TYPE ;
    L$AdressePersonne Etudiants.AdressePersonne%TYPE ;
    CURSOR Curs_EPP IS
        SELECT COLUMN_VALUE PrenomPersonne
        FROM TABLE ( SELECT PrenomsPersonne
                      FROM Etudiants
                      WHERE IdPersonne = idp ) ;
    CURSOR Curs_ETP IS
        SELECT IndicatifPays , Telephone
        FROM TABLE ( SELECT TelephonesPersonne
                      FROM Etudiants
                      WHERE IdPersonne = idp )
        ORDER BY IndicatifPays , Telephone ;
    L$DepartNaissEtu Etudiants.DepartNaissEtu%TYPE ;
    L$PhotoEtu ORDSYS.ORDIMAGE ;
    nom VARCHAR2(100) ; -- nom récupéré du fichier contenant la photographie
    rep ALL_DIRECTORIES.DIRECTORY_PATH%TYPE ; -- répertoire d'origine images et XSD
    L$CVEtu XMLTYPE ;
    L$CVEtuSourceFiche VARCHAR2(40) ;
    L$CVEtuSourceFiche_URITYPE URITYPE ;
    CURSOR Curs_EVP IS
        SELECT EVP.NoImmat.ConcatNoImmat(' ') ConcatNoImmat , EVP.Couleur
        FROM TABLE ( SELECT VoituresPossedees
                      FROM Etudiants
                      WHERE IdPersonne = idp ) EVP
        ORDER BY EVP.NoImmat.Chiffres , EVP.NoImmat.Lettres , EVP.NoImmat.Depart ;
    CURSOR Curs_EDO IS
        SELECT EDO.Annee ,
               Deref(EDO.DiplomeObtenu).IntitAbrege IntitAbrege ,
               Deref(EDO.DiplomeObtenu).IntitCompleet IntitCompleet
        FROM TABLE ( SELECT DiplomesObtenus
                      FROM Etudiants
                      WHERE IdPersonne = idp ) EDO
        ORDER BY EDO.Annee , Deref(EDO.DiplomeObtenu).IntitAbrege ;
    nd Departements.NomDepartement%TYPE ; -- nom d'un département
    L$IdPersonne_Geo Etudiants_Geo.IdPersonne%TYPE ;
    L$PosGeogEtu MDSYS.SDO_GEOMETRY ;
    L$CS_NAME CS_SRS.CS_NAME%TYPE ;
    L$FigGeomEtu MDSYS.SDO_GEOMETRY ;
    t VARCHAR(4) := '<DD>' ; -- tabulation
BEGIN
    HTP.HTMLOPEN ;
    HTP.HEADOPEN ;
    HTP.HTITLE('Informations sur l''étudiant n° ' || CAST(idp AS VARCHAR)) ;
    HTP.HEADCLOSE ;
    HTP.BODYOPEN ;

```

```

SELECT NomPersonne , DepartNaissEtu , AdressePersonne , PhotoEtu ,
      CVEtu , EG.IdPersonne , PosGeogEtu , FigGeomEtu
INTO L$NomPersonne , L$DepartNaissEtu , L$AdressePersonne , L$PhotoEtu ,
      L$CVEtu , L$IdPersonne_Geo , L$PosGeogEtu , L$FigGeomEtu
FROM Etudiants E
LEFT OUTER JOIN Etudiants_Geo EG ON E.IdPersonne = EG.IdPersonne
WHERE E.IdPersonne = idp ;
-- exception NO_DATA_FOUND si l'étudiant n'existe pas
nom := SUBSTR(L$PhotoEtu.GETSOURCENAME(),1,
            INSTR(L$PhotoEtu.GETSOURCENAME(),'.')-1) ;

-- numéro
HTP.PRINT('Numéro : ') ;
HTP.BR() ;
HTP.PRINT(t||CAST(idp AS VARCHAR)) ;
HTP.BR() ;
-- nom
HTP.PRINT('Nom : ') ;
HTP.BR() ;
HTP.PRINT(t||L$NomPersonne) ;
HTP.BR() ;
-- prénoms
HTP.PRINT('Prénoms : ') ;
HTP.BR() ;
HTP.TABLEOPEN() ;
FOR EPP IN Curs_EPP LOOP
  HTP.TABLEROWOPEN ;
  HTP.TABLEDATA(CVALUE=>t) ;
  HTP.TABLEDATA(CVALUE=>EPP.PrenomPersonne) ;
  HTP.TABLEROWCLOSE ;
END LOOP ;
HTP.TABLECLOSE ;
-- téléphones
HTP.PRINT('Téléphones : ') ;
HTP.BR() ;
HTP.TABLEOPEN() ;
FOR ETP IN Curs_ETP LOOP
  HTP.TABLEROWOPEN ;
  HTP.TABLEDATA(CVALUE=>t) ;
  HTP.TABLEDATA(CVALUE=>ETP.IndicatifPays) ;
  HTP.TABLEDATA(CVALUE=>ETP.Telephone) ;
  HTP.TABLEROWCLOSE ;
END LOOP ;
HTP.TABLECLOSE ;
-- adresse
HTP.PRINT('Adresse : ') ;
HTP.BR() ;
HTP.PRINT(t||L$AdressePersonne.Ligne1) ;
HTP.BR() ;
HTP.PRINT(t||L$AdressePersonne.Ligne2) ;
HTP.BR() ;
IF L$AdressePersonne.Ligne3 IS NOT NULL THEN
  HTP.PRINT(t||L$AdressePersonne.Ligne3) ;
  HTP.BR() ;
END IF ;
HTP.PRINT(t||L$AdressePersonne.CodePostal||' '||L$AdressePersonne.Ville) ;
SELECT NomDepartement
      INTO nd
      FROM Departements
      WHERE CodeDepartement = L$AdressePersonne.DepartAdresse() ;
-- exception NO_DATA_FOUND si le code postal ne référence pas un département
HTP.PRINT(' ('||nd||')') ;
HTP.BR() ;
HTP.PRINT(t) ;
HTP.ANCHOR(L$AdressePersonne.SiteWeb.GETEXTERNALURL(),
          L$AdressePersonne.SiteWeb.GETEXTERNALURL()) ;

```

```

HTP.BR() ;
-- département de naissance
HTP.PRINT('Département de naissance : ') ;
HTP.BR() ;
HTP.PRINT(t||L$DepartNaissEtu) ;
SELECT NomDepartement
      INTO nd
      FROM Departements
      WHERE CodeDepartement = L$DepartNaissEtu ;
      -- exception NO_DATA_FOUND si la CI référentielle n'est pas respectée
HTP.PRINT(' ( '||nd||' )' ) ;
HTP.BR() ;
-- pseudonyme
HTP.PRINT('Pseudonyme : ') ;
HTP.BR() ;
HTP.PRINT(t) ;
HTP.IMG(CURL=>'AffPseudoEtuHTML?idp='||TO_CHAR(idp),
        CATTRIBUTES=>'title="'||nom||'"') ;
HTP.BR() ;
-- photographie
HTP.PRINT('Photographie : ') ;
HTP.BR() ;
HTP.PRINT(t) ;
HTP.IMG(CURL=>'AffPhotoEtuHTML?idp='||TO_CHAR(idp),
        CATTRIBUTES=>'title="'||nom||'"') ;
HTP.BR() ;
HTP.PRINT(t||'Fichier d'origine : '||L$PhotoEtu.GETSOURCENAME()) ;
HTP.BR() ;
SELECT DIRECTORY_PATH
      INTO rep
      FROM ALL_DIRECTORIES
      JOIN Etudiants E ON DIRECTORY_NAME = E.PhotoEtu.GETSOURCELOCATION()
      WHERE E.IdPersonne = idp ;
HTP.PRINT(t||'Répertoire d'origine : '||HTF.ESCAPE_SC(rep)) ;
HTP.BR() ;
HTP.PRINT(t||'Dernière date de mise à jour : '||
          TO_CHAR(L$PhotoEtu.GETUPDATETIME(), 'DD/MM/YYYY')) ;
HTP.BR() ;
HTP.PRINT(t||'Dimensions : hauteur='||TO_CHAR(L$PhotoEtu.GETHEIGHT())||
          ', largeur='||TO_CHAR(L$PhotoEtu.GETWIDTH())||' et taille='||
          TO_CHAR(L$PhotoEtu.GETCONTENTLENGTH())) ;
HTP.BR() ;
HTP.PRINT(t||'Type dans lequel l'image est stockée : '||
          L$PhotoEtu.GETFILEFORMAT()) ;
HTP.BR() ;
HTP.PRINT(t||'Type : '||L$PhotoEtu.GETCONTENTFORMAT()) ;
HTP.BR() ;
HTP.PRINT(t||'Algorithme de compression : '||L$PhotoEtu.GETCOMPRESSIONFORMAT()) ;
HTP.BR() ;
HTP.PRINT(t||'Type MIME (Multipurpose Internet Mail Extensions) : '||
          L$PhotoEtu.MIMETYPE) ;
HTP.BR() ;
-- curriculum vitæ
HTP.PRINT('Curriculum vitæ : ') ;
HTP.BR() ;
HTP.PRINT(t||'Source des informations : ') ;
SELECT EXTRACTVALUE(L$CVetu, '/Philosophe/SourceFiche')
      INTO L$CVetuSourceFiche
      FROM DUAL ;
L$CVetuSourceFiche_URITYPE := URIFACTORY.GETURI(L$CVetuSourceFiche);
HTP.ANCHOR(L$CVetuSourceFiche_URITYPE.GETEXTERNALURL(), L$CVetuSourceFiche) ;
HTP.BR() ;
AffNaissDecesCVetuHTML(t||'Naissance : ', 'Naissance', L$CVetu) ;
AffNaissDecesCVetuHTML(t||'Décès : ', 'Deces', L$CVetu) ;
AffItemCVetuHTML(t||'École/tradition : ', 'Ecoles', 'Ecole', L$CVetu) ;

```

```

AffItemCVEtuHTML(t||'Principaux intérêts : ','Interets','Interet',L$CVEtu) ;
AffItemCVEtuHTML(t||'Idées remarquables : ','Idees','Idee',L$CVEtu) ;
AffItemCVEtuHTML(t||'Œuvres principales : ','OEuvres','OEuvre',L$CVEtu) ;
AffItemCVEtuHTML(t||'Influencé par : ','InfluencesPar','InfluencePar',L$CVEtu) ;
AffItemCVEtuHTML(t||'A influencé : ','InfluencesSur','InfluenceSur',L$CVEtu) ;
-- voitures possédées
HTP.PRINT('Voitures : ') ;
HTP.BR() ;
HTP.TABLEOPEN() ;
FOR EVP IN Curs_EVP LOOP
    HTP.TABLEROWOPEN ;
    HTP.TABLEDATA(CVALUE=>t) ;
    HTP.TABLEDATA(CVALUE=>EVP.ConcatNoImmat) ;
    HTP.TABLEDATA(CVALUE=>EVP.Couleur) ;
    HTP.TABLEROWCLOSE ;
END LOOP ;
HTP.TABLECLOSE ;
-- diplômes obtenus
HTP.PRINT('Diplômes : ') ;
HTP.BR() ;
HTP.TABLEOPEN() ;
FOR EDO IN Curs_EDO LOOP
    HTP.TABLEROWOPEN ;
    HTP.TABLEDATA(CVALUE=>t) ;
    HTP.TABLEDATA(CVALUE=>EDO.Annee) ;
    HTP.TABLEDATA(CVALUE=>EDO.IntitAbrege) ;
    HTP.TABLEDATA(CVALUE=>EDO.IntitComple) ;
    HTP.TABLEROWCLOSE ;
END LOOP ;
HTP.TABLECLOSE ;
-- informations spatiales
IF L$IdPersonne_Geo IS NULL THEN
    HTP.PRINT('Aucune information spatiale') ;
ELSE
    -- position géographique
    HTP.PRINT('Position géographique : ') ;
    IF L$PosGeogEtu IS NULL THEN
        HTP.PRINT('(non renseignée)') ;
    ELSIF L$PosGeogEtu.SDO_GTYPE IS NULL OR L$PosGeogEtu.SDO_GTYPE <> 3001 OR
        L$PosGeogEtu.SDO_SRID IS NULL OR L$PosGeogEtu.SDO_SRID <> 8307 THEN
        HTP.PRINT('(de type ou de référence spatiale non renseigné/ée/és ou non
géré/ée/és)') ;
        HTP.BR() ;
        HTP.PRINT(t||'Type : '||TO_CHAR(L$PosGeogEtu.SDO_GTYPE)) ;
        HTP.BR() ;
        HTP.PRINT(t||'Référence spatiale : '||TO_CHAR(L$PosGeogEtu.SDO_SRID)) ;
    ELSE
        HTP.BR() ;
        HTP.PRINT(t||'Type : '||TO_CHAR(L$PosGeogEtu.SDO_GTYPE)||' "point 3D"') ;
        HTP.BR() ;
        HTP.PRINT(t||'Référence spatiale : '||TO_CHAR(L$PosGeogEtu.SDO_SRID)) ;
        SELECT CS_NAME
            INTO L$CS_NAME
            FROM CS_SRS
            WHERE SRID = L$PosGeogEtu.SDO_SRID ;
        HTP.PRINT(' "'||L$CS_NAME||'"') ;
        HTP.BR() ;
        HTP.PRINT(t||'Latitude : '||TO_CHAR(L$PosGeogEtu.SDO_POINT.X)||'°') ;
        HTP.BR() ;
        HTP.PRINT(t||'Longitude : '||TO_CHAR(L$PosGeogEtu.SDO_POINT.Y)||'°') ;
        HTP.BR() ;
        HTP.PRINT(t||'Altitude : '||TO_CHAR(L$PosGeogEtu.SDO_POINT.Z)||' m') ;
        HTP.BR() ;
        HTP.PRINT(t||'Cf. ') ;
        HTP.ANCHOR('http://maps.google.com/maps?q='||

```

```

        REPLACE(TO_CHAR(L$PosGeogEtu.SDO_POINT.X),',','.')||','||
        REPLACE(TO_CHAR(L$PosGeogEtu.SDO_POINT.Y),',','.'),
        'Google Maps') ;
HTP.PRINT(' ou ') ;
HTP.ANCHOR('http://toolserver.org/~geohack/geohack.php?language=fr' ||
        '&' || 'params=' ||
        REPLACE(TO_CHAR(ABS(L$PosGeogEtu.SDO_POINT.X)),',','.')||'_' ||
        TRANSLATE(REPLACE(TO_CHAR(SIGN(L$PosGeogEtu.SDO_POINT.X)),
        '-1','S'),'01','NN')||'_' ||
        REPLACE(TO_CHAR(ABS(L$PosGeogEtu.SDO_POINT.Y)),',','.')||'_' ||
        TRANSLATE(REPLACE(TO_CHAR(SIGN(L$PosGeogEtu.SDO_POINT.Y)),
        '-1','W'),'01','EE'),'GeoHack') ;

END IF ;
HTP.BR() ;
-- figure géométrique
HTP.PRINT('Figure géométrique : ') ;
IF L$FigGeomEtu IS NULL THEN
    HTP.PRINT('(non renseignée)') ;
ELSIF L$FigGeomEtu.SDO_GTYPE IS NULL THEN
    HTP.PRINT('(de type non renseigné)') ;
ELSE
    HTP.BR() ;
    HTP.PRINT(t || 'Type : ' || TO_CHAR(L$FigGeomEtu.SDO_GTYPE) || ' ' ||
        LTRIM(TO_CHAR(L$FigGeomEtu.GET_DIMS())) || ' dimension(s), ' ||
        LTRIM(TO_CHAR(L$FigGeomEtu.GET_LRS_DIM())) ||
        ' dimension(s) LRS, ') ;
    CASE L$FigGeomEtu.GET_GTYPE()
        WHEN 0 THEN HTP.PRINT('(inconnu)') ;
        WHEN 1 THEN HTP.PRINT('un point') ;
        WHEN 2 THEN HTP.PRINT('une ligne (de droites ou d''arcs de
            cercles)') ;
        WHEN 3 THEN HTP.PRINT('un polygone (avec ou sans trou)') ;
        WHEN 4 THEN HTP.PRINT('une collection') ;
        WHEN 5 THEN HTP.PRINT('un ou plusieurs points') ;
        WHEN 6 THEN HTP.PRINT('un ou plusieurs lignes (de droites ou d''arcs
            de cercles)') ;
        WHEN 7 THEN HTP.PRINT('un ou plusieurs polygones (avec ou sans
            trou)') ;
        ELSE HTP.PRINT('(actuellement non géré par Oracle)') ;
    END CASE ;
    END IF ;
    HTP.BR() ;
END IF ;
-- (fin)
HTP.BR() ;
HTP.BODYCLOSE ;
HTP.HTMLCLOSE ;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('L''étudiant de n° ' || CAST(idp AS VARCHAR) ||
            ' n''existe pas ou certaines de ses informations sont
            manquantes') ;
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20000,'Erreur non gérée de message ' || SQLERRM ||
            ' et de code ' || CAST(SQLCODE AS VARCHAR)) ;
END ;

```

Création des données

Les données vont tout d'abord être insérées ; ensuite, une insertion invalide permet de tester de nombreuses contraintes d'intégrité.

N. B. : la suppression des données se trouve en annexe.

Insertion des données

-- départements

```

INSERT INTO Departements VALUES ( '01' , 'Ain' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '02' , 'Aisne' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '03' , 'Allier' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '04' , 'Alpes de Haute Provence' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '05' , 'Hautes-Alpes' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '06' , 'Alpes-Maritimes' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '07' , 'Ardèche' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '08' , 'Ardennes' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '09' , 'Ariège' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '10' , 'Aube' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '11' , 'Aude' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '12' , 'Aveyron' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '13' , 'Bouches-du-Rhône' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '14' , 'Calvados' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '15' , 'Cantal' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '16' , 'Charente' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '17' , 'Charente-Maritime' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '18' , 'Cher' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '19' , 'Corrèze' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '2A' , 'Corse-du-Sud' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '2B' , 'Haute-Corse' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '21' , 'Côte-d'Or' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '22' , 'Côtes-d'Armor' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '23' , 'Creuse' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '24' , 'Dordogne' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '25' , 'Doubs' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '26' , 'Drôme' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '27' , 'Eure' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '28' , 'Eure-et-Loir' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '29' , 'Finistère' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '30' , 'Gard' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '31' , 'Haute-Garonne' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '32' , 'Gers' ) ;

```



```

COMMIT ;
INSERT INTO Departements VALUES ( '33' , 'Gironde' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '34' , 'Hérault' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '35' , 'Ille-et-Vilaine' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '36' , 'Indre' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '37' , 'Indre-et-Loire' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '38' , 'Isère' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '39' , 'Jura' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '40' , 'Landes' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '41' , 'Loir-et-Cher' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '42' , 'Loire' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '43' , 'Haute-Loire' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '44' , 'Loire-Atlantique' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '45' , 'Loiret' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '46' , 'Lot' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '47' , 'Lot-et-Garonne' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '48' , 'Lozère' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '49' , 'Maine-et-Loire' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '50' , 'Manche' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '51' , 'Marne' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '52' , 'Haute-Marne' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '53' , 'Mayenne' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '54' , 'Meurthe-et-Moselle' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '55' , 'Meuse' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '56' , 'Morbihan' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '57' , 'Moselle' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '58' , 'Nièvre' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '59' , 'Nord' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '60' , 'Oise' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '61' , 'Orne' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '62' , 'Pas-de-Calais' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '63' , 'Puy-de-Dôme' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '64' , 'Pyrénées-Atlantiques' ) ;
COMMIT ;

```

```

INSERT INTO Departements VALUES ( '65' , 'Hautes-Pyrénées' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '66' , 'Pyrénées-Orientales' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '67' , 'Bas-Rhin' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '68' , 'Haut-Rhin' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '69' , 'Rhône' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '70' , 'Haute-Saône' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '71' , 'Saône-et-Loire' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '72' , 'Sarthe' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '73' , 'Savoie' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '74' , 'Haute-Savoie' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '75' , 'Paris' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '76' , 'Seine-Maritime' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '77' , 'Seine-et-Marne' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '78' , 'Yvelines' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '79' , 'Deux-Sèvres' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '80' , 'Somme' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '81' , 'Tarn' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '82' , 'Tarn-et-Garonne' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '83' , 'Var' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '84' , 'Vaucluse' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '85' , 'Vendée' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '86' , 'Vienne' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '87' , 'Haute-Vienne' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '88' , 'Vosges' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '89' , 'Yonne' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '90' , 'Territoire de Belfort' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '91' , 'Essonne' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '92' , 'Hauts-de-Seine' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '93' , 'Seine-Saint-Denis' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '94' , 'Val-de-Marne' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '95' , 'Val-d'Oise' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '971' , 'Guadeloupe' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '972' , 'Martinique' ) ;

```

```

COMMIT ;
INSERT INTO Departements VALUES ( '973' , 'Guyane' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '974' , 'La Réunion' ) ;
COMMIT ;
INSERT INTO Departements VALUES ( '976' , 'Mayotte' ) ;
COMMIT ;
-- diplômes
INSERT INTO Diplomes VALUES ( 'DUT' , 'Diplôme Universitaire de Technologie' ) ;
COMMIT ;
INSERT INTO Diplomes VALUES ( 'BAC' , 'Baccalauréat' ) ;
COMMIT ;
INSERT INTO Diplomes VALUES ( 'MIAGe' , 'Maîtrise des Méthodes Informatiques
                                Appliquées à la Gestion des Entreprises' ) ;
COMMIT ;
INSERT INTO Diplomes VALUES ( 'DEUG' , 'Diplôme d'Études Universitaires Générales' );
COMMIT ;
-- étudiants
INSERT INTO Etudiants VALUES (
    5 , -- Sequence_IdPersonne.NEXTVAL
    'DURAND' ,
    Type_Prenoms ( 'Esther' , 'Eurielle' , 'Édouardine' ) ,
    Type_Telephones (
        Type_Telephone ( '+1' , '515151515' ) ,
        Type_Telephone ( '+1' , '525252525' ) ,
        Type_Telephone ( '+1-441' , '535353535' ) ,
        Type_Telephone ( '+1' , '545454545' ) ,
        Type_Telephone ( '+1' , '555555555' ) ,
        Type_Telephone ( '+1' , '565656565' ) ,
        Type_Telephone ( '+1' , '575757575' ) ,
        Type_Telephone ( '+1' , '585858585' ) ,
        Type_Telephone ( '+1' , '595959595' ) ) ,
    Type_Adresse ( 'Musée du Louvre' ,
        '99 rue de Rivoli' , NULL , '75001' , 'Paris' ,
        URIFACTORY.GETURI('http://www.louvre.fr') ) ,
    '33' ,
    EMPTY_BLOB() ,
    ORDSYS.ORDIMAGE.INIT('file',UPPER('Rep_Etudiants'),'Épicure.jpg') ,
    ORDSYS.ORDIMAGESIGNATURE.INIT() ,
    XMLTYPE.CREATEXML('
        <Philosophe>
            <SourceFiche>http://fr.wikipedia.org/wiki/Épicure</SourceFiche>
            <Naissance>
                <AnneeNaissance>fin -342 ou début -341</AnneeNaissance>
                <LieuNaissance>Athènes</LieuNaissance>
            </Naissance>
            <Deces>
                <AnneeDeces>-270</AnneeDeces>
                <LieuDeces></LieuDeces>
            </Deces>
            <Ecoles>
            </Ecoles>
            <Interets>
                <Interet>Physique</Interet>
                <Interet>Éthique</Interet>
                <Interet>Eudémonisme</Interet>
            </Interets>
            <Idees>
            </Idees>
            <OEuvres>
            </OEuvres>
            <InfluencesPar>
            </InfluencesPar>
            <InfluencesSur>
            </InfluencesSur>
    ')

```

```

        </Philosophe>
    ' ) ,
Type_Voitures (
    Type_Voiture ( Type_ImmatVoiture ( 3333 , 'BX' , '33' ) , 'rouge' ) ,
    Type_Voiture ( Type_ImmatVoiture ( 4040 , 'NT' , '40' ) , 'jaune' ) ) ,
Type_DiplomesObtenus (
    ( SELECT Type_DiplomeObtenu ( REF(D) , 1981 )
      FROM Diplomes D
      WHERE IntitAbrege = 'BAC' ) ,
    ( SELECT Type_DiplomeObtenu ( REF(D) , 1983 )
      FROM Diplomes D
      WHERE IntitAbrege = 'DUT' ) )
) ;
COMMIT ;
INSERT INTO Etudiants VALUES (
    4 , -- Sequence_IdPersonne.NEXTVAL
    'MARTIN' ,
    Type_Prenoms ( 'Aleyde' , 'Aldegonde' , 'Albertine' ) ,
    Type_Telephones (
        Type_Telephone ( '+262' , '414141414' ) ,
        Type_Telephone ( '+269' , '424242424' ) ,
        Type_Telephone ( '+248' , '434343434' ) ,
        Type_Telephone ( '+230' , '444444444' ) ) ,
    Type_Adresse ( 'Le Cabinet des Monnaies et Médailles' ,
        '10 rue Clovis-Hugues' , NULL , '13003' , 'Marseille' ,
    URIFACTORY.GETURI(
        'http://www.marseille.fr/sitevdm/jsp/site/Portal.jsp?page_id=282' ) ) ,
    '47' ,
    EMPTY_BLOB() ,
    ORDSYS.ORDIMAGE.INIT('file',UPPER('Rep_Etudiants'),'Aristote.jpg') ,
    ORDSYS.ORDIMAGESIGNATURE.INIT() ,
    XMLTYPE.CREATEXML( '
        <Philosophe>
            <SourceFiche>http://fr.wikipedia.org/wiki/Aristote</SourceFiche>
            <Naissance>
                <AnneeNaissance>-384</AnneeNaissance>
                <LieuNaissance>Stagire</LieuNaissance>
            </Naissance>
            <Deces>
                <AnneeDeces>-322</AnneeDeces>
                <LieuDeces>Chalcis</LieuDeces>
            </Deces>
            <Ecoles>
                <Ecole>fondateur du Lycée</Ecole>
                <Ecole>Péripatétisme</Ecole>
            </Ecoles>
            <Interets>
                <Interet>Physique</Interet>
                <Interet>Métaphysique</Interet>
                <Interet>Biologie</Interet>
                <Interet>Éthique</Interet>
                <Interet>Politique</Interet>
                <Interet>Langage</Interet>
                <Interet>Logique</Interet>
                <Interet>Poétique</Interet>
                <Interet>Rhétorique</Interet>
            </Interets>
            <Idees>
                <Idee>Syllogisme</Idee>
                <Idee>Puissance/Acte</Idee>
                <Idee>Matière/Forme</Idee>
                <Idee>Substance/Accident</Idee>
                <Idee>Catégorie</Idee>
            </Idees>
            <OEuvres>

```

```

        <Oeuvre>Catégories</Oeuvre>
        <Oeuvre>Métaphysique</Oeuvre>
        <Oeuvre>Physique</Oeuvre>
        <Oeuvre>Politiques</Oeuvre>
        <Oeuvre>Poétique</Oeuvre>
    </Oeuvres>
    <InfluencesPar>
        <InfluencePar>Homère</InfluencePar>
        <InfluencePar>Héraclite</InfluencePar>
        <InfluencePar>Parménide</InfluencePar>
        <InfluencePar>Anaxagore</InfluencePar>
        <InfluencePar>Empédocle</InfluencePar>
        <InfluencePar>Socrate</InfluencePar>
        <InfluencePar>Platon</InfluencePar>
    </InfluencesPar>
    <InfluencesSur>
        <InfluenceSur>Théophraste</InfluenceSur>
        <InfluenceSur>Ptolémée</InfluenceSur>
        <InfluenceSur>Horace</InfluenceSur>
        <InfluenceSur>Alexandre d' Aphrodise</InfluenceSur>
        <InfluenceSur>Néoplatonisme</InfluenceSur>
        <InfluenceSur>Boèce</InfluenceSur>
        <InfluenceSur>Péripatétisme</InfluenceSur>
        <InfluenceSur>Avicenne</InfluenceSur>
        <InfluenceSur>Averroès</InfluenceSur>
        <InfluenceSur>Maïmonide</InfluenceSur>
        <InfluenceSur>Thomas d' Aquin</InfluenceSur>
        <InfluenceSur>Guillaume d' Ockham</InfluenceSur>
        <InfluenceSur>Scolastique</InfluenceSur>
        <InfluenceSur>Leibniz</InfluenceSur>
        <InfluenceSur>Swedenborg</InfluenceSur>
        <InfluenceSur>Trendelenburg</InfluenceSur>
        <InfluenceSur>Schelling</InfluenceSur>
        <InfluenceSur>Marx</InfluenceSur>
        <InfluenceSur>Brentano</InfluenceSur>
        <InfluenceSur>Heidegger</InfluenceSur>
        <InfluenceSur>Arendt</InfluenceSur>
        <InfluenceSur>Ayn Rand</InfluenceSur>
        <InfluenceSur>Ricoeur</InfluenceSur>
    </InfluencesSur>
</Philosophe>
'),
Type_Voitures (
    Type_Voiture ( Type_ImmatVoiture ( 4747 , 'LA' , '47' ) , 'rouge' ) ) ,
Type_DiplomesObtenus (
    ( SELECT Type_DiplomeObtenu ( REF(D) , 1977 )
      FROM Diplomes D
      WHERE IntitAbrege = 'BAC' ) ,
    ( SELECT Type_DiplomeObtenu ( REF(D) , 1980 )
      FROM Diplomes D
      WHERE IntitAbrege = 'DEUG' ) ,
    ( SELECT Type_DiplomeObtenu ( REF(D) , 1982 )
      FROM Diplomes D
      WHERE IntitAbrege = 'MIAGE' ) )
);
COMMIT ;
INSERT INTO Etudiants VALUES (
    2 , -- Sequence_IdPersonne.NEXTVAL
    'LEROI' ,
    Type_Prenoms ( 'Saturnin' , 'Symphorien' , 'Samson' , 'Siméon' , 'Séraphin' ) ,
    Type_Telephones (
        Type_Telephone ( '+30' , '212121212' ) ,
        Type_Telephone ( '+33' , '222222222' ) ) ,
    Type_Adresse ( 'Musée d' Art Contemporain de Lyon' ,
        'Cité Internationale' , '81 quai Charles de Gaulle' , '69463' , 'Lyon' ,

```

```

        URIFACTORY.GETURI('http://www.moca-lyon.org') ) ,
'40' ,
EMPTY_BLOB() ,
ORDSYS.ORDIMAGE.INIT('file',UPPER('Rep_Etudiants'),'Socrate.jpg') ,
ORDSYS.ORDIMAGESIGNATURE.INIT() ,
XMLTYPE.CREATEXML('
    <Philosophe>
        <SourceFiche>http://fr.wikipedia.org/wiki/Socrate</SourceFiche>
        <Naissance>
            <AnneeNaissance>-470</AnneeNaissance>
            <LieuNaissance></LieuNaissance>
        </Naissance>
        <Deces>
            <AnneeDeces>-399</AnneeDeces>
            <LieuDeces>Athènes</LieuDeces>
        </Deces>
        <Ecoles>
        </Ecoles>
        <Interets>
            <Interet>Éthique</Interet>
        </Interets>
        <Idees>
            <Idee>Maïeutique</Idee>
            <Idee>Ironie socratique</Idee>
        </Idees>
        <Oeuvres>
        </Oeuvres>
        <InfluencesPar>
            <InfluencePar>Anaxagore de Clazomènes</InfluencePar>
            <InfluencePar>Prodicos</InfluencePar>
        </InfluencesPar>
        <InfluencesSur>
            <InfluenceSur>Platon</InfluenceSur>
            <InfluenceSur>Xénophon</InfluenceSur>
            <InfluenceSur>Antisthène</InfluenceSur>
            <InfluenceSur>les socratiques</InfluenceSur>
            <InfluenceSur>tous les philosophes occidentaux</InfluenceSur>
        </InfluencesSur>
    </Philosophe>
') ,
Type_Voitures ( ) , -- et non NULL
Type_DiplomesObtenus (
    ( SELECT Type_DiplomeObtenu ( REF(D) , 1980 )
      FROM Diplomes D
      WHERE IntitAbrege = 'BAC' ) ,
    ( SELECT Type_DiplomeObtenu ( REF(D) , 1982 )
      FROM Diplomes D
      WHERE IntitAbrege = 'DEUG' ) )
) ;
COMMIT ;
INSERT INTO Etudiants VALUES (
    7 , -- Sequence_IdPersonne.NEXTVAL
    'LEROI' ,
    Type_Prenoms ( 'Andoche' , 'Ambroise' , 'Alfred' , 'Anastase' , 'Aloysius' ) ,
    Type_Telephones (
        Type_Telephone ( '+49' , '717171717' ) ) ,
    Type_Adresse ( 'Musée d''Art Moderne et centre d''Art Contemporain de Toulouse' ,
        'Les Abattoirs' , '76 allées Charles-de-Fitte' , '31300' , 'Toulouse' ,
        URIFACTORY.GETURI('http://www.lesabattoirs.org') ) ,
    '33' ,
    EMPTY_BLOB() ,
    ORDSYS.ORDIMAGE.INIT('file',UPPER('Rep_Etudiants'),'Averroès.jpg') ,
    ORDSYS.ORDIMAGESIGNATURE.INIT() ,
    XMLTYPE.CREATEXML('
        <Philosophe>

```

```

    <SourceFiche>http://fr.wikipedia.org/wiki/Averroès</SourceFiche>
    <Naissance>
      <AnneeNaissance>1126</AnneeNaissance>
      <LieuNaissance>Cordoue</LieuNaissance>
    </Naissance>
    <Deces>
      <AnneeDeces>10 décembre 1198</AnneeDeces>
      <LieuDeces>Marrakech</LieuDeces>
    </Deces>
    <Ecoles>
    </Ecoles>
    <Interets>
      <Interet>Métaphysique</Interet>
      <Interet>Théologie</Interet>
      <Interet>Droit</Interet>
      <Interet>Médecine</Interet>
      <Interet>Politique</Interet>
      <Interet>Religion</Interet>
    </Interets>
    <Idees>
    </Idees>
    <OEuvres>
    </OEuvres>
    <InfluencesPar>
    </InfluencesPar>
    <InfluencesSur>
    </InfluencesSur>
  </Philosophe>
  ' ) ,
  Type_Voitures ( ) , -- et non NULL
  Type_DiplomesObtenus ( ) -- et non NULL
) ;
COMMIT ;
INSERT INTO Etudiants VALUES (
  3 , -- Sequence_IdPersonne.NEXTVAL
  'DUPOND' ,
  Type_Prenoms ( 'Philémon' , 'Placide' , 'Philomène' , 'Prosper' , 'Parfait' ) ,
  Type_Telephones ( ) , -- et non NULL
  Type_Adresse ( 'Musée National Picasso, La Guerre et La Paix' ,
    'Place de la libération' , NULL , '06220' , 'Vallauris' ,
    URIFACTORY.GETURI('http://www.musee-picasso-vallauris.fr') ) ,
  '17' ,
  EMPTY_BLOB() ,
  ORDSYS.ORDIMAGE.INIT('file',UPPER('Rep_Etudiants'),'Platon.jpg') ,
  ORDSYS.ORDIMAGESIGNATURE.INIT() ,
  XMLTYPE.CREATEXML('
    <Philosophe>
      <SourceFiche>http://fr.wikipedia.org/wiki/Platon</SourceFiche>
      <Naissance>
        <AnneeNaissance>vers -427</AnneeNaissance>
        <LieuNaissance>Athènes</LieuNaissance>
      </Naissance>
      <Deces>
        <AnneeDeces>vers -346</AnneeDeces>
        <LieuDeces>Athènes</LieuDeces>
      </Deces>
      <Ecoles>
      </Ecoles>
      <Interets>
        <Interet>Psychologie</Interet>
        <Interet>Politique</Interet>
        <Interet>Sophistique</Interet>
        <Interet>Théorie de la connaissance</Interet>
        <Interet>Métaphysique</Interet>
        <Interet>Langage</Interet>

```



```

        <Interet>Éthique</Interet>
    </Interets>
    <Idees>
</Idees>
    <OEuvres>
</OEuvres>
    <InfluencesPar>
</InfluencesPar>
    <InfluencesSur>
</InfluencesSur>
</Philosophe>
    ) ,
Type_Voitures ( ) , -- et non NULL
Type_DiplomesObtenus (
    ( SELECT Type_DiplomeObtenu ( REF(D) , 1981 )
      FROM Diplomes D
      WHERE IntitAbrege = 'BAC' ) ,
    ( SELECT Type_DiplomeObtenu ( REF(D) , 1983 )
      FROM Diplomes D
      WHERE IntitAbrege = 'DUT' ) ,
    ( SELECT Type_DiplomeObtenu ( REF(D) , 1985 )
      FROM Diplomes D
      WHERE IntitAbrege = 'MIAGe' ) )
) ;
COMMIT ;
-- insère (et valide définitivement) les pseudonymes des étudiants
EXECUTE InserePseudoEtudiants(5,'Épicure_Pseudo.jpg') ;
EXECUTE InserePseudoEtudiants(4,'Aristote_Pseudo.jpg') ;
EXECUTE InserePseudoEtudiants(2,'Socrate_Pseudo.jpg') ;
EXECUTE InserePseudoEtudiants(7,'Averroès_Pseudo.jpg') ;
EXECUTE InserePseudoEtudiants(3,'Platon_Pseudo.jpg') ;
-- affecte les propriétés des photographies des étudiants et génère leurs signatures
EXECUTE AffectePropPhotosEtudiants() ;
-- importe localement les photographies des étudiants dans la base de données
EXECUTE ImportePhotosEtudiants() ;
-- informations spatiales sur les étudiants
INSERT INTO Etudiants_Geo VALUES (
    5 ,
    MDSYS.SDO_GEOMETRY ( -- source du Rhin
        3001 , -- 3=3D , 0=pas de LRS , 01=point
        8307 , -- SRID="Longitude / Latitude (WGS 84)"
        MDSYS.SDO_POINT_TYPE ( 46.6325 , 8.672222 , 2346 ) , -- lat. , long., alt.
        NULL , -- c'est un point
        NULL ) , -- c'est un point
    MDSYS.SDO_GEOMETRY ( -- soleil
        2003 , -- 2=2D , 0=pas de LRS , 03=polygone
        NULL , -- pas de SRID
        NULL , -- ce n'est pas un point
        MDSYS.SDO_ELEM_INFO_ARRAY ( 1 , 1003 , 4 ) , -- 1=on commence par 1ère valeur
                                                    -- 1003=polygone extérieur
                                                    -- 4=cercle
        MDSYS.SDO_ORDINATE_ARRAY ( 3,8 , 2,9 , 1,8 ) ) -- 3 coordonnées pour cercle
) ;
COMMIT ;
INSERT INTO Etudiants_Geo VALUES (
    4 ,
    MDSYS.SDO_GEOMETRY ( -- source de la Loire
        3001 , -- 3=3D , 0=pas de LRS , 01=point
        8307 , -- SRID="Longitude / Latitude (WGS 84)"
        MDSYS.SDO_POINT_TYPE ( 44.843889 , 4.22 , 1408 ) , -- lat. , long., alt.
        NULL , -- c'est un point
        NULL ) , -- c'est un point
    MDSYS.SDO_GEOMETRY ( -- grande pyramide de Gizeh (c.-à-d. la pyramide de Khéops)
        2003 , -- 2=2D , 0=pas de LRS , 03=polygone
        NULL , -- pas de SRID

```

```

NULL , -- ce n'est pas un point
MDSYS.SDO_ELEM_INFO_ARRAY ( 1 , 1003 , 1 ) , -- 1=on commence par 1ère valeur
-- 1003=polygone extérieur
-- 1=lignes droites
MDSYS.SDO_ORDINATE_ARRAY ( 5,1 , 13,1 , 9,6 , 5,1 ) )
-- triangle = 1er point vers 2ème vers 3ème vers 1er (sens trigo.)
) ;
COMMIT ;
INSERT INTO Etudiants_Geo VALUES (
2 ,
MDSYS.SDO_GEOMETRY ( -- source de la Meuse
3001 , -- 3=3D , 0=pas de LRS , 01=point
8307 , -- SRID="Longitude / Latitude (WGS 84)"
MDSYS.SDO_POINT_TYPE ( 47.97435 , 5.633539 , 409 ) , -- lat. , long., alt.
NULL , -- c'est un point
NULL ) , -- c'est un point
MDSYS.SDO_GEOMETRY ( -- hauteur grande pyramide de Gizeh et angle droit avec sol
2002 , -- 2=2D , 0=pas de LRS , 02=lignes
NULL , -- pas de SRID
NULL , -- ce n'est pas un point
MDSYS.SDO_ELEM_INFO_ARRAY ( 1 , 2 , 1 ) , -- 1=on commence par 1ère valeur ,
-- 2=lignes , 1=lignes droites
MDSYS.SDO_ORDINATE_ARRAY ( 9,6 , 9,1 , 9.5,1 , 9.5,1.5 , 9,1.5 ) )
-- ligne brisée d'un 1er point vers 2ème vers 3ème vers 4ème vers 5ème
) ;
COMMIT ;
INSERT INTO Etudiants_Geo VALUES (
7 ,
MDSYS.SDO_GEOMETRY ( -- source du Rhône
3001 , -- 3=3D , 0=pas de LRS , 01=point
8307 , -- SRID="Longitude / Latitude (WGS 84)"
MDSYS.SDO_POINT_TYPE ( 46.602 , 8.376167 , 2250 ) , -- lat. , long., alt.
NULL , -- c'est un point
NULL ) , -- c'est un point
MDSYS.SDO_GEOMETRY ( -- ombre de la grande pyramide de Gizeh
2003 , -- 2=2D , 0=pas de LRS , 03=polygone
NULL , -- pas de SRID
NULL , -- ce n'est pas un point
MDSYS.SDO_ELEM_INFO_ARRAY ( 1 , 1003 , 1 ) , -- 1=on commence par 1ère valeur
-- 1003=polygone extérieur
-- 1=lignes droites
MDSYS.SDO_ORDINATE_ARRAY ( 9,6 , 13,1 , 19,1 , 9,6 ) )
-- triangle = 1er point vers 2ème vers 3ème vers 1er (sens trigo.)
) ;
COMMIT ;
INSERT INTO Etudiants_Geo VALUES (
3 ,
MDSYS.SDO_GEOMETRY ( -- source de la Seine
3001 , -- 3=3D , 0=pas de LRS , 01=point
8307 , -- SRID="Longitude / Latitude (WGS 84)"
MDSYS.SDO_POINT_TYPE ( 47.486183 , 4.717461 , 446 ) , -- lat. , long., alt.
NULL , -- c'est un point
NULL ) , -- c'est un point
MDSYS.SDO_GEOMETRY ( -- bâton d'un mètre utilisé par Thalès [de Milet]
2002 , -- 2=2D , 0=pas de LRS , 02=lignes
NULL , -- pas de SRID
NULL , -- ce n'est pas un point
MDSYS.SDO_ELEM_INFO_ARRAY ( 1 , 2 , 1 ) , -- 1=on commence par 1ère valeur ,
-- 2=lignes , 1=lignes droites
MDSYS.SDO_ORDINATE_ARRAY ( 17,1 , 17,2 ) ) -- ligne d'un point vers un autre
) ;
COMMIT ;

```

Vérification de contraintes d'intégrité

```
-- données invalides
INSERT INTO Etudiants VALUES (
  0 , -- 4 , -- Sequence_IdPersonne.NEXTVAL
  'ZIGOTO' ,
  Type_Prenoms ( 'Évariste' , 'Eusèbe' , 'Eustache' , 'Elfried' ,
    'Ernest-Edgar Évrard Élie-Éloi' ) ,
  Type_Telephones (
    Type_Telephone ( '+998' , '919191919' ) ) ,
  Type_Adresse ( 'Musée Jules Verne' ,
    '3 rue de l''Hermitage' , NULL , '44100' , 'Nantes' ,
    URIFACTORY.GETURI('http://www.nantes.fr/julesverne/acc_6.htm') ) ,
  '99' , ,
  EMPTY_BLOB() ,
  ORDSYS.ORDIMAGE.INIT('file',UPPER('Rep_Etudiants'),'Épictète.jpg') ,
  ORDSYS.ORDIMAGESIGNATURE.INIT() ,
  XMLTYPE.CREATEXML( '
    <Philosophe>
      <SourceFiche>http://fr.wikipedia.org/wiki/Épictète</SourceFiche>
      <Naissance>
        <AnneeNaissance></AnneeNaissance>
        <LieuNaissance></LieuNaissance>
      </Naissance>
      <Deces>
        <AnneeDeces></AnneeDeces>
        <LieuDeces></LieuDeces>
      </Deces>
      <Ecoles>
      </Ecoles>
      <Interets>
        <Interet></Interet>
      </Interets>
      <Idees>
      </Idees>
      <Oeuvres>
      </Oeuvres>
      <InfluencesPar>
      </InfluencesPar>
      <InfluencesSur>
      </InfluencesSur>
    </Philosophe>
  ' ) ,
  Type_Voitures (
    Type_Voiture ( Type_ImmatVoiture ( 9999 , 'az' , '99' ) , 'glauque' ) ,
    Type_Voiture ( Type_ImmatVoiture ( 4747 , 'LA' , '47' ) , 'ORANGE' ) ,
    Type_Voiture ( Type_ImmatVoiture ( 0000 , 'ZA' , NULL ) , NULL ) ) ,
  Type_DiplomesObtenus (
    ( SELECT Type_DiplomeObtenu ( REF(D) , 1981 )
      FROM Diplomes D
      WHERE IntitAbrege = 'BAC' ) ,
    ( SELECT Type_DiplomeObtenu ( REF(D) , 1981 )
      FROM Diplomes D
      WHERE IntitAbrege = 'DUT' ) ,
    ( SELECT Type_DiplomeObtenu ( REF(D) , 1985 )
      FROM Diplomes D
      WHERE IntitAbrege = 'DUT' ) )
) ;
-- messages d'erreur
-- ORA-02290: violation de contraintes (Contrainte_IdPersonnePositif) de vérification
-- ORA-00001: violation de contrainte unique (ClePrimaire_Etudiants)
-- ORA-02291: violation de contrainte d'intégrité (Ref_Etudiants_Departements) -
  clé parent introuvable
-- ORA-00001: violation de contrainte unique (Unicite_NoImmat)
-- ORA-02290: violation de contraintes (Contrainte_NoImmatChiffresBorn) de
```

```

vérification
-- ORA-02290: violation de contraintes (Contrainte_NoImmatLettresMajus) de
vérification
-- ORA-02290: violation de contraintes (Contrainte_ListeCouleurs) de vérification
-- ORA-20002: Contrainte d'intégrité référentielle
Etudiants.VoituresPossedees[.].NoImmat.Depart /
Departements.CodeDepartement violée pour 99 (voiture n° 1)
-- ORA-06512,ORA-04088: erreur lors d'exécution du déclencheur
Declen_AvInsertUpdate_Etudiant
-- ORA-20003: Contrainte d'unicité ( Etudiants.IdPersonne ,
Etudiants.DiplomesObtenus.DiplomeObtenu ) violée pour
(diplôme n° 2 et 3)
-- ORA-06512,ORA-04088: erreur lors d'exécution du déclencheur
Declen_AvInsertUpdate_Etudiant
-- ORA-00001: violation de contrainte unique (Unicite_IdPersonne_Anee)
-- données invalides
INSERT INTO Etudiants_Geo VALUES (
0 ,
MDSYS.SDO_GEOMETRY (
3001 , -- 3=3D , 0=pas de LRS , 01=point
8307 , -- SRID="Longitude / Latitude (WGS 84)"
MDSYS.SDO_POINT_TYPE ( -91 , 181 , 0 ) , -- lat.<-91° , long.>180° , alt.
NULL , -- c'est un point
NULL ) , -- c'est un point
MDSYS.SDO_GEOMETRY ( -- palmier
2003 , -- 2=2D , 0=pas de LRS , 03=polygone
NULL , -- pas de SRID
NULL , -- ce n'est pas un point
MDSYS.SDO_ELEM_INFO_ARRAY ( 1 , 1005 , 2 , -- 1=on commence par 1ère valeur
-- 1005=polygone composé extérieur
-- 2=2 composants
1 , 2 , 1 , -- (1er composant) :
-- 1=on commence par 1ère valeur
-- 2=lignes , 1=lignes droites
7 , 2 , 2 , -- (2nd composant) :
-- 7=on commence par 7ème valeur
-- 2=lignes
-- 2=lignes d'arcs de cercles
41 , 2003 , 3 ) , -- 41=on commence par 41ème val
-- 2003=polygone intérieur
-- 3=rectangle
MDSYS.SDO_ORDINATE_ARRAY ( 1,4 , 0,0 , 3,0 , 2,4 , 3,3 , 3,2 , 3,4 , 2,4.5 ,
4,4 , 4,3 , 4,5 , 1,5,5 , -1,5 , -1,3 , -1,4 ,
1,4.5 , 0,4 , 0,2 , 0,3 , 1,4 , -- polygone
-- extérieur
1,1 , 2,2 ) ) -- polygone intérieur
) ;
-- messages d'erreur
-- ORA-20004: La latitude, dans la référence spatiale 8307, -91 doit être comprise
entre -90° et +90°
-- ORA-04088: erreur lors d'exécution du déclencheur
Declen_AvInsertUpdate_Etu_Geo
-- ORA-20005: La longitude, dans la référence spatiale 8307, 181 doit être comprise
entre -180° et +180°
-- ORA-04088: erreur lors d'exécution du déclencheur
Declen_AvInsertUpdate_Etu_Geo
-- ORA-20006: La figure géométrique n'est pas correcte ou ne respecte pas les
contraintes définies sur l'attribut (dans USER_SDO_GEOM_METADATA)
-- ORA-04088: erreur lors d'exécution du déclencheur
Declen_AvInsertUpdate_Etu_Geo

```

Interrogations et mises à jour

Vérification des données

```
-- y a-t-il le bon nombre de données ?
SELECT COUNT(*) AS Nb , '= 101 ?' AS Egal , 'Departements' AS Tables
  FROM Departements
UNION
SELECT COUNT(*) , '= 5 ?' , 'Etudiants' FROM Etudiants
UNION
SELECT COUNT(*) , '= 21 ?' , 'Etudiants.PrenomsPersonne'
  FROM Etudiants E , TABLE ( E.PrenomsPersonne )
  -- FROM Etudiants E , TABLE ( SELECT PrenomsPersonne
  --                               FROM Etudiants EPP
  --                               WHERE EPP.IdPersonne = E.IdPersonne )
UNION
SELECT SUM(CARDINALITY(TelephonesPersonne)) , '= 16 ?' ,
  'Etudiants.TelephonesPersonne'
  FROM Etudiants
UNION
SELECT SUM(CARDINALITY(VoituresPossedees)) , '= 3 ?' , 'Etudiants.VoituresPossedees'
  FROM Etudiants
UNION
SELECT SUM(CARDINALITY(DiplomesObtenus)) , '= 10 ?' , 'Etudiants.DiplomesObtenus'
  FROM Etudiants
UNION
SELECT COUNT(*) , '= 5 ?' , 'Etudiants_Géo' FROM Etudiants_Geo
UNION
SELECT COUNT(*) , '= 4 ?' , 'Diplomes' FROM Diplomes
ORDER BY 3 ;
      NB EGAL      TABLES
-----
      101 = 101 ? Departements
        4 = 4 ?  Diplomes
        5 = 5 ?  Etudiants
       10 = 10 ? Etudiants.DiplomesObtenus
        5 = 5 ?  Etudiants_Géo
       21 = 21 ? Etudiants.PrenomsPersonne
       16 = 16 ? Etudiants.TelephonesPersonne
        3 = 3 ?  Etudiants.VoituresPossedees
-- les étudiants dont le document XML n'est pas valide
SELECT IdPersonne
FROM Etudiants E
WHERE E.CVEtu.ISSCHEMAVALID() = 0 ;
      aucune ligne sélectionnée
-- ne doit pas y avoir étudiants dt document XML ne contient pas certaines rubriques
SELECT COUNT(*)
FROM Etudiants
WHERE EXISTSNODE(CVEtu,'/Philosophe/SourceFiche') = 0 OR
      EXISTSNODE(CVEtu,'/Philosophe/Naissance') = 0 OR
      EXISTSNODE(CVEtu,'/Philosophe/Naissance/AnneeNaissance') = 0 OR
      EXISTSNODE(CVEtu,'/Philosophe/Naissance/LieuNaissance') = 0 OR
      EXISTSNODE(CVEtu,'/Philosophe/Deces') = 0 OR
      EXISTSNODE(CVEtu,'/Philosophe/Deces/AnneeDeces') = 0 OR
      EXISTSNODE(CVEtu,'/Philosophe/Deces/LieuDeces') = 0 OR
      EXISTSNODE(CVEtu,'/Philosophe/Ecoles') = 0 OR
      EXISTSNODE(CVEtu,'/Philosophe/Interets') = 0 OR
      EXISTSNODE(CVEtu,'/Philosophe/Interets/Interet[1]') = 0 OR
      EXISTSNODE(CVEtu,'/Philosophe/Idees') = 0 OR
      EXISTSNODE(CVEtu,'/Philosophe/OEuvres') = 0 OR
      EXISTSNODE(CVEtu,'/Philosophe/InfluencesPar') = 0 OR
      EXISTSNODE(CVEtu,'/Philosophe/InfluencesSur') = 0 ;
      COUNT(*)
      -----
              0
```

Requêtes d'interrogation relationnelles

-- les départements (code, nom, nombre d'étudiants qui y sont nés avec les plus petit
 -- et plus grand de leurs noms), triés sur le nom, ayant un nom de département
 -- renseigné, un code de département ne commençant pas par un 1, dont le nom de
 -- personne n'est ni Laurel ni Hardy, où sont nés au plus 3 étudiants et dont la
 -- moyenne des identifiants est au moins 2

```
SELECT CodeDepartement , NomDepartement ,
       COUNT(*) , MIN(NomPersonne) , MAX(NomPersonne)
FROM Departements
JOIN Etudiants ON CodeDepartement = DepartNaissEtu
WHERE NomDepartement IS NOT NULL AND SUBSTR(CodeDepartement,1,1) <> '1' AND
      NomPersonne NOT IN ( 'LAUREL' , 'HARDY' )
GROUP BY CodeDepartement , NomDepartement
HAVING COUNT(*) <= 3 OR AVG(IdPersonne) >= 2
ORDER BY NomDepartement ASC ;
      COD NOMDEPARTEMENT COUNT(*) MIN(NOMPERSONNE) MAX(NOMPERSONNE)
```

```
-----
33 Gironde                2 DURAND                LEROI
40 Landes                 1 LEROI                 LEROI
47 Lot-et-Garonne        1 MARTIN                MARTIN
```

-- les départements (code, nom, nom sauf pour la Gironde) et étudiants (identifiant,
 -- nom, deux fois l'indication s'il n'y a pas d'étudiant né dans ce département)
 -- qui y sont éventuellement nés, triés sur le code des départements et l'identifiant
 -- des étudiants, dont le département a vu naître au moins un étudiant ou dont le
 -- code de département est aquitain

```
SELECT CodeDepartement , NomDepartement , NULLIF(NomDepartement,'Gironde') ,
      IdPersonne , NomPersonne ,
      COALESCE(CAST(IdPersonne AS CHAR(1)),'(pas d'étd)') ,
      NVL(CAST(IdPersonne AS CHAR(1)),'(pas d'étd)')
```

```
FROM Departements
LEFT OUTER JOIN Etudiants ON CodeDepartement = DepartNaissEtu
WHERE IdPersonne IS NOT NULL OR CodeDepartement IN ('24','33','40','47','64')
ORDER BY CodeDepartement , IdPersonne ;
      COD NOMDEPARTEMENT      NULLIF(NOMDEPARTEMEN ID NOMPERSONNE COALESCE(CA NVL(CAST(ID
```

```
-----
17 Charente-Maritime      Charente-Maritime      3 DUPOND                3                3
24 Dordogne               Dordogne                (pas d'étd) (pas d'étd)
33 Gironde                5 DURAND                5                5
33 Gironde                7 LEROI                 7                7
40 Landes                 Landes                  2 LEROI                 2                2
47 Lot-et-Garonne        Lot-et-Garonne         4 MARTIN                4                4
64 Pyrénées-Atlantiques  Pyrénées-Atlantiques  (pas d'étd) (pas d'étd)
```

-- les départements (code, nom, nombre de lignes (>=1 même si aucun étudiant n'y est
 -- né), nombre d'étudiants qui y sont nés, tranche d'histogramme ici 1 pour les
 -- départements <= 95 et 1 + le mois actuel pour les départements >= 971, (1 + le
 -- mois actuel)-quantile, écart-type et variance des n° des étudiants), triés sur le
 -- code, dont le code est un nombre et dont le nom contient les lettres "e" et "n" et
 -- dont le nom est égal à celui-ci en mettant toutes les initiales en capitales et
 -- dont la longueur du nom est 6 ou 7 ou 17

```
SELECT CodeDepartement , NomDepartement , COUNT(*) , COUNT(IdPersonne) ,
      WIDTH_BUCKET(TO_NUMBER(CodeDepartement),0,1000,1+EXTRACT(MONTH FROM SYSDATE)),
      NTILE(1+EXTRACT(MONTH FROM SYSDATE)) OVER (ORDER BY CodeDepartement) Quantile,
      STDDEV(IdPersonne) Écart_type , VARIANCE(IdPersonne) Var
```

```
FROM Departements
LEFT OUTER JOIN Etudiants ON CodeDepartement = DepartNaissEtu
WHERE REGEXP_LIKE(CodeDepartement,'^[0-9]*$') AND INSTR(NomDepartement,'e') > 0 AND
      INSTR(NomDepartement,'n') > 0 AND NomDepartement = INITCAP(NomDepartement) AND
      LENGTH(NomDepartement) IN ( 6 , 7 , 17 )
```

```
GROUP BY CodeDepartement , NomDepartement
HAVING COUNT(*) <= 3
ORDER BY CodeDepartement ;
      COD NOMDEPARTEMENT      COUNT(*) COUNT(ID WIDTH_BUCKET QUANTILE ÉCART_TYPE VAR
```

```
-----
12 Aveyron                1                0                1                1
17 Charente-Maritime      1                1                1                1                0  0
```

33	Gironde	2	2	1	2	1,41421356	2
40	Landes	1	1	1	2	0	0
50	Manche	1	0	1	3		
53	Mayenne	1	0	1	4		
85	Vendée	1	0	1	5		
86	Vienne	1	0	1	6		
91	Essonne	1	0	1	7		
93	Seine-Saint-Denis	1	0	1	8		
973	Guyane	1	0	9	9		

-- intersection entre d'une part les premières lettres des noms des départements qui
-- ne sont pas des dernières lettres des noms des départements et d'autre part les
-- premières lettres des noms des étudiants qui ne sont pas des dernières lettres des
-- noms des étudiants

```
( SELECT SUBSTR(NomDepartement,1,1) -- A B C D E F G H I J L M N O P R S T V Y
FROM Departements
MINUS
SELECT UPPER(SUBSTR(NomDepartement,LENGTH(NomDepartement),1)) -- A D E L N R S T
FROM Departements
) -- B C F G H I J M O P V Y
INTERSECT
( SELECT SUBSTR(NomPersonne,1,1) FROM Etudiants -- D L M
MINUS
SELECT SUBSTR(NomPersonne,LENGTH(NomPersonne),1) FROM Etudiants -- D I N
) -- L M
ORDER BY 1 ;
S
-
M
```

-- les départements (code, nom)
-- dont la première lettre est aussi une première lettre des noms des étudiants et
-- dont la dernière lettre est aussi une dernière lettre des noms des étudiants

```
SELECT CodeDepartement , NomDepartement
FROM Departements
WHERE EXISTS ( SELECT SUBSTR(NomPersonne,LENGTH(NomPersonne),1)
FROM Etudiants
WHERE SUBSTR(NomPersonne,1,1) = SUBSTR(NomDepartement,1,1) ) AND
UPPER(SUBSTR(NomDepartement,LENGTH(NomDepartement),1)) IN (
SELECT SUBSTR(NomPersonne,LENGTH(NomPersonne),1)
FROM Etudiants
) ;
COD NOMDEPARTEMENT
--- -----
974 La Réunion
56 Morbihan
```

-- les numéros de ligne, départements (code, nom, région ou collectivité
-- territoriale) et étudiants (identifiant, nom) qui y sont éventuellement nés,
-- triés sur le code des départements et le nom des étudiants,
-- dont le code de département commence par un chiffre compris entre 2 et 4 suivi
-- d'un A ou B ou 0 ou 3 et dont le nom de département ne contient pas "et" et
-- est d'une longueur comprise entre 6 et 12 et est plus grand ou égal à 'C' dans
-- l'ordre alphabétique

```
SELECT ROW_NUMBER() OVER(ORDER BY CodeDepartement,NomPersonne) NuméroLigne ,
CodeDepartement , NomDepartement ,
CASE WHEN TRIM(CodeDepartement) IN ('24','33','40','47','64') THEN 'Aquitaine'
WHEN TRIM(CodeDepartement) IN ('2A','2B') THEN 'Corse'
ELSE '(autre)'
END Rég_CollTerr ,
IdPersonne , NomPersonne
FROM Departements
LEFT OUTER JOIN Etudiants ON CodeDepartement = DepartNaissEtu
WHERE REGEXP_LIKE(CodeDepartement, '^[2-4][AB03]') AND
NomDepartement NOT LIKE '%et%' AND LENGTH(NomDepartement) BETWEEN 6 AND 12 AND
NomDepartement >= 'C'
ORDER BY NuméroLigne ;
NUMÉROLIGNE COD NOMDEPARTEMENT RÉG_COLLTERR IDPERSONNE NOMPERSONNE
-----
```

```

1 2A Corse-du-Sud Corse
2 2B Haute-Corse Corse
3 23 Creuse (autre)
4 33 Gironde Aquitaine 5 DURAND
5 33 Gironde Aquitaine 7 LEROI
6 40 Landes Aquitaine 2 LEROI
7 43 Haute-Loire (autre)

```

```

-- les couples de départements où sont nés au moins un étudiant tels que
-- le nom du premier département est plus petit que le nom du second département,
-- avec soit le produit du nombre d'étudiants et des totaux pour chacune des deux
-- composantes du couple et un total général, soit les seuls totaux des deux
-- composantes et général

```

```

SELECT D1.NomDepartement , D2.NomDepartement , COUNT(*)
FROM Departements D1
JOIN Etudiants E1 ON D1.CodeDepartement = E1.DepartNaissEtu
LEFT OUTER JOIN Departements D2 ON D1.NomDepartement < D2.NomDepartement
JOIN Etudiants E2 ON D2.CodeDepartement = E2.DepartNaissEtu
GROUP BY CUBE ( D1.NomDepartement , D2.NomDepartement )
-- GROUP BY GROUPING SETS ( D1.NomDepartement , D2.NomDepartement , ( ) )
ORDER BY D1.NomDepartement ASC NULLS LAST , D2.NomDepartement ASC NULLS LAST ;
GROUP BY CUBE ( D1.NomDepartement , D2.NomDepartement )

```

```

NOMDEPARTEMENT NOMDEPARTEMENT COUNT(*)
-----

```

```

Charente-Maritime Gironde 2
Charente-Maritime Landes 1
Charente-Maritime Lot-et-Garonne 1
Charente-Maritime 4
Gironde Landes 2
Gironde Lot-et-Garonne 2
Gironde 4
Landes Lot-et-Garonne 1
Landes 1
Gironde 2
Landes 3
Lot-et-Garonne 4
9

```

```

GROUP BY GROUPING SETS ( D1.NomDepartement , D2.NomDepartement , ( ) )

```

```

NOMDEPARTEMENT NOMDEPARTEMENT COUNT(*)
-----

```

```

Charente-Maritime 4
Gironde 4
Landes 1
Gironde 2
Landes 3
Lot-et-Garonne 4
9

```

```

-- les départements et le nombre d'étudiants qui y sont nés, triés sur ce nombre en
-- ordre décroissant, en n'affichant que les r premières lignes (ou autant de plus
-- que d'éventuels ex-æquo)

```

```

ACCEPT r PROMPT 'Choisissez un rang maximal : '
SELECT DepartNaissEtu , NbEtdDuDepart
FROM ( SELECT DepartNaissEtu , NbEtdDuDepart ,
RANK() OVER(ORDER BY NbEtdDuDepart DESC) RangNbEtdDuDepart
FROM ( SELECT DepartNaissEtu , COUNT(*) NbEtdDuDepart
FROM Etudiants
GROUP BY DepartNaissEtu
ORDER BY NbEtdDuDepart DESC ) ) )

```

```

WHERE RangNbEtdDuDepart <= &r ;

```

```

r=0

```

```

aucune ligne sélectionnée

```

```

r=1

```

```

DEP NBETDDUDEPART
---

```

```

33 2

```

```

r≥2

```



```

DEP NBETDDUDEPART
--- -----
33          2
40          1
47          1
17          1
-- les rangs (avec ou sans trou pour les ex-æquo) des départements (de naissance des
-- étudiants), département, rang des noms des étudiants au sein des départements, nom
-- des étudiants, nombre total d'étudiants, nombre d'étudiants et nom minimal et
-- maximal des étudiants pour chaque département
SELECT RANK() OVER(ORDER BY DepartNaissEtu) ,
       DENSE_RANK() OVER(ORDER BY DepartNaissEtu) , DepartNaissEtu ,
       RANK() OVER(PARTITION BY DepartNaissEtu ORDER BY NomPersonne) , NomPersonne ,
       COUNT(*) OVER() , COUNT(*) OVER(PARTITION BY DepartNaissEtu) ,
       MIN(NomPersonne) OVER(PARTITION BY DepartNaissEtu) MinNom ,
       MAX(NomPersonne) OVER(PARTITION BY DepartNaissEtu) MaxNom
FROM Etudiants
ORDER BY DepartNaissEtu , NomPersonne ;
R R DEP R NOMPERSO C C MINNOM MAXNOM
- - - - -
1 1 17 1 DUPOND      5 1 DUPOND DUPOND
2 2 33 1 DURAND      5 2 DURAND LEROI
2 2 33 2 LEROI       5 2 DURAND LEROI
4 3 40 1 LEROI       5 1 LEROI  LEROI
5 4 47 1 MARTIN     5 1 MARTIN MARTIN
-- les départements où sont nés le moins d'étudiants (version 1)
SELECT CodeDepartement , NomDepartement
FROM Departements
JOIN Etudiants ON CodeDepartement = DepartNaissEtu
WHERE DepartNaissEtu IN (
  SELECT DepartNaissEtu
  FROM Etudiants
  GROUP BY DepartNaissEtu
  HAVING COUNT(*) = (
    SELECT MIN(Compte_DepartNaissEtu)
    FROM (
      SELECT DepartNaissEtu , COUNT(*) Compte_DepartNaissEtu
      FROM Etudiants
      GROUP BY DepartNaissEtu
    ) ) )
ORDER BY CodeDepartement ;
-- les départements où sont nés le moins d'étudiants (version 2)
WITH Comptes_DepartNaissEtu AS
( SELECT DepartNaissEtu , COUNT(*) Compte_DepartNaissEtu
  FROM Etudiants
  GROUP BY DepartNaissEtu
)
SELECT CodeDepartement , NomDepartement
FROM Departements
JOIN Etudiants ON CodeDepartement = DepartNaissEtu
WHERE DepartNaissEtu IN (
  SELECT DepartNaissEtu
  FROM Comptes_DepartNaissEtu
  WHERE Compte_DepartNaissEtu = ( SELECT MIN(Compte_DepartNaissEtu)
                                  FROM Comptes_DepartNaissEtu ) )
ORDER BY CodeDepartement ;
COD NOMDEPARTEMENT
--- -----
17 Charente-Maritime
40 Landes
47 Lot-et-Garonne
-- les identifiants de ligne (uniques pour une table) minimal et maximal des
-- étudiants (avec un indice donné à l'optimiseur pour accéder aux données grâce aux
-- identifiants de ligne)
SELECT /*+ROWID(Etudiants)*/ MIN(ROWID) , MAX(ROWID)
FROM Etudiants ;
MIN(ROWID)          MAX(ROWID)

```

```

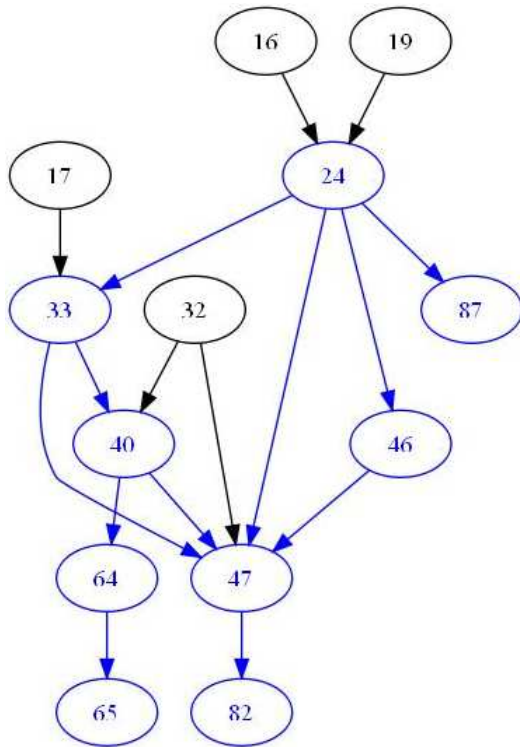
-----
AAARe9AAAEAAAAJbAAA AAARe9AAAEAAAAJcAAA
-- de nouveaux (créés temporairement pendant l'exécution de cette requête)
-- étudiants (identifiant, nom, département de naissance)
SELECT IdPersonne , NomPersonne , DepartNaissEtu
FROM Etudiants
MODEL RETURN UPDATED ROWS
  DIMENSION BY ( IdPersonne )
  MEASURES ( NomPersonne , DepartNaissEtu )
  RULES ( NomPersonne[1] = 'JOURLY' , DepartNaissEtu[1] = 24 ,
          NomPersonne[6] = 'RUNEN' , DepartNaissEtu[6] = 17 ,
          NomPersonne[8] = 'DUPENA' , DepartNaissEtu[8] = 47 ,
          NomPersonne[9] = 'BILMET' , DepartNaissEtu[9] = 33 ,
          NomPersonne[10] = 'LETOUR' , DepartNaissEtu[10] = 64 ,
          NomPersonne[11] = 'MONLIX' , DepartNaissEtu[11] = 40 )
ORDER BY IdPersonne ;
  IDPERSONNE NOMPERSOENNE DEP
-----
      1 JOURLY          24
      6 RUNEN           17
      8 DUPENA          47
      9 BILMET          33
     10 LETOUR          64
     11 MONLIX          40
-- les étudiants (identifiant, nom, département de naissance, partition sur le
-- département de naissance) en modifiant (temporairement pendant l'exécution de
-- cette requête) les noms de ceux dont l'identifiant est compris entre 3 et 5 et
-- en ajoutant deux nouveaux (créés temporairement pendant l'exécution de cette
-- requête) obtenus par le produit cartésien de leurs noms et des quatre départements
-- de naissance issus de la partition et dont le département de naissance est
-- soit répété (celui des quatre départements de naissance) soit fixé
SELECT IdPersonne , NomPersonne , DepartNaissEtu , Partition_DepartNaissEtu
FROM Etudiants
MODEL PARTITION BY ( DepartNaissEtu Partition_DepartNaissEtu )
  DIMENSION BY ( DepartNaissEtu , IdPersonne )
  MEASURES ( NomPersonne )
  RULES UPSERT ALL
    ( NomPersonne[ANY,IdPersonne BETWEEN 3 AND 5] =
      'Id2Nom : ' || TO_CHAR(CV(IdPersonne)) ,
      NomPersonne[ANY,13] = 'GAUCHER' ,
      NomPersonne[64,14] = 'BOURGARD' )
ORDER BY IdPersonne , Partition_DepartNaissEtu ;
  IDPERSONNE NOMPERSOENNE DEP PAR
-----
      2 LEROI           40  40
      3 Id2Nom : 3     17  17
      4 Id2Nom : 4     47  47
      5 Id2Nom : 5     33  33
      7 LEROI           33  33
     13 GAUCHER         17  17
     13 GAUCHER         33  33
     13 GAUCHER         40  40
     13 GAUCHER         47  47
     14 BOURGARD        64  17
     14 BOURGARD        64  33
     14 BOURGARD        64  40
     14 BOURGARD        64  47
-- les étudiants (identifiant, nom, département de naissance) ayant des informations
-- spatiales et d'identifiant au moins égal à 4, en effectuant (temporairement
-- pendant l'exécution de cette requête) un calcul sur leurs noms consistant à
-- concaténer le compteur d'itération au nom éventuel pour ceux nés en Gironde et
-- en créant autant de nouveaux étudiants girondins qu'il y avait d'étudiants non
-- girondins
SELECT IdPersonne , NomPersonne , Calculs_NomPersonne , DepartNaissEtu

```

```

FROM Etudiants
NATURAL JOIN Etudiants_Geo
WHERE IdPersonne >= 4
MODEL DIMENSION BY ( DepartNaissEtu , IdPersonne )
MEASURES ( NomPersonne , NomPersonne Calculs_NomPersonne )
RULES UPSERT ALL
ITERATE (6)
( Calculs_NomPersonne['33',ANY] =
  Calculs_NomPersonne[CV(),CV()]||'|', '|'|TO_CHAR(ITERATION_NUMBER) )
ORDER BY IdPersonne ;
IDPERSONNE  NOMPERSONNE  CALCULS_NOMPERSONNE  DEP
-----
4 MARTIN      MARTIN              47
4              , 0, 1, 2, 3, 4, 5  33
5 DURAND     DURAND, 0, 1, 2, 3, 4, 5 33
7 LEROI      LEROI, 0, 1, 2, 3, 4, 5 33
-- [requête indépendante des données contenues dans les tables]
-- à partir des départements aquitains et limitrophes, en allant du + petit code de
-- département vers le + grand constituant ainsi un graphe orienté acyclique,
-- tous les chemins partant du 24 en descendant
WITH DptsAquitEtLmtrph AS
( SELECT 16 De , 24 Vers FROM DUAL UNION
  SELECT 17 , 33 FROM DUAL UNION
  SELECT 19 , 24 FROM DUAL UNION
  SELECT 24 , 33 FROM DUAL UNION
  SELECT 24 , 46 FROM DUAL UNION
  SELECT 24 , 47 FROM DUAL UNION
  SELECT 24 , 87 FROM DUAL UNION
  SELECT 32 , 40 FROM DUAL UNION
  SELECT 32 , 47 FROM DUAL UNION
  SELECT 33 , 40 FROM DUAL UNION
  SELECT 33 , 47 FROM DUAL UNION
  SELECT 40 , 47 FROM DUAL UNION
  SELECT 40 , 64 FROM DUAL UNION
  SELECT 46 , 47 FROM DUAL UNION
  SELECT 47 , 82 FROM DUAL UNION
  SELECT 64 , 65 FROM DUAL )
SELECT De , Vers , LEVEL , LPAD(' ',2*(LEVEL-1))||Vers as Arbre ,
CONNECT_BY_ROOT(De) || SYS_CONNECT_BY_PATH(Vers,'>') as Chemin ,
CONNECT_BY_ISLEAF
FROM DptsAquitEtLmtrph
START WITH De = 24
CONNECT BY PRIOR Vers = De -- en descendant dans le DAG
ORDER SIBLINGS BY De , Vers ;
De Vers LEVEL Arbre Chemin CONNECT_BY_ISLEAF
-----
24 33 1 33 24>33 0
33 40 2 40 24>33>40 0
40 47 3 47 24>33>40>47 0
47 82 4 82 24>33>40>47>82 1
40 64 3 64 24>33>40>64 0
64 65 4 65 24>33>40>64>65 1
33 47 2 47 24>33>47 0
47 82 3 82 24>33>47>82 1
24 46 1 46 24>46 0
46 47 2 47 24>46>47 0
47 82 3 82 24>46>47>82 1
24 47 1 47 24>47 0
47 82 2 82 24>47>82 1
24 87 1 87 24>87 1

```



```

-- [requête indépendante des données contenues dans les tables]
-- à partir des départements aquitains et limitrophes, en allant du + petit code de
-- département vers le + grand constituant ainsi un graphe orienté acyclique,
-- tous les chemins partant du 47 en remontant

```

```
WITH DptsAquitEtLmtrph AS
```

```

( SELECT 16 De , 24 Vers FROM DUAL UNION
  SELECT 17 , 33 FROM DUAL UNION
  SELECT 19 , 24 FROM DUAL UNION
  SELECT 24 , 33 FROM DUAL UNION
  SELECT 24 , 46 FROM DUAL UNION
  SELECT 24 , 47 FROM DUAL UNION
  SELECT 24 , 87 FROM DUAL UNION
  SELECT 32 , 40 FROM DUAL UNION
  SELECT 32 , 47 FROM DUAL UNION
  SELECT 33 , 40 FROM DUAL UNION
  SELECT 33 , 47 FROM DUAL UNION
  SELECT 40 , 47 FROM DUAL UNION
  SELECT 40 , 64 FROM DUAL UNION
  SELECT 46 , 47 FROM DUAL UNION
  SELECT 47 , 82 FROM DUAL UNION
  SELECT 64 , 65 FROM DUAL )

```

```

SELECT Vers , De , LEVEL , LPAD(' ',2*(LEVEL-1))||De as Arbre ,
       CONNECT_BY_ROOT(Vers) || SYS_CONNECT_BY_PATH(De,'<') as Chemin ,
       CONNECT_BY_ISLEAF

```

```
FROM DptsAquitEtLmtrph
```

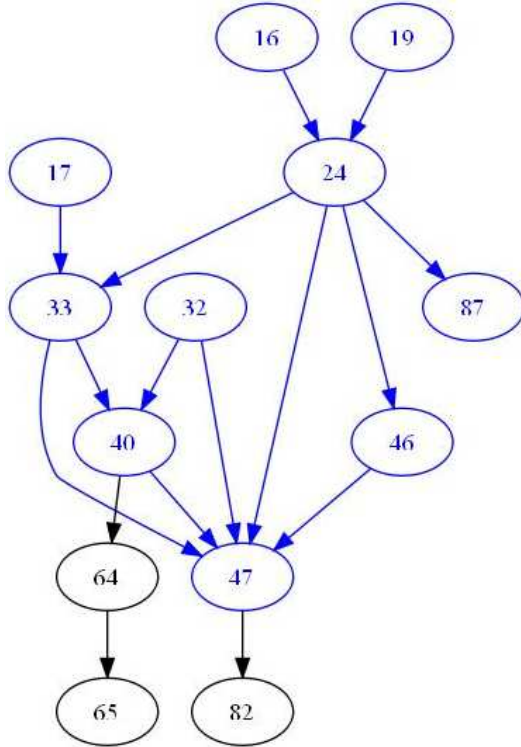
```
START WITH Vers = 47
```

```
CONNECT BY PRIOR De = Vers -- en remontant dans le DAG
```

```
ORDER SIBLINGS BY Vers , De ;
```

Vers	De	LEVEL	Arbre	Chemin	CONNECT_BY_ISLEAF
47	24	1	24	47<24	0
24	16	2	16	47<24<16	1
24	19	2	19	47<24<19	1
47	32	1	32	47<32	1
47	33	1	33	47<33	0
33	17	2	17	47<33<17	1
33	24	2	24	47<33<24	0
24	16	3	16	47<33<24<16	1
24	19	3	19	47<33<24<19	1
47	40	1	40	47<40	0

40	32	2	32	47<40<32	1
40	33	2	33	47<40<33	0
33	17	3	17	47<40<33<17	1
33	24	3	24	47<40<33<24	0
24	16	4	16	47<40<33<24<16	1
24	19	4	19	47<40<33<24<19	1
47	46	1	46	47<46	0
46	24	2	24	47<46<24	0
24	16	3	16	47<46<24<16	1
24	19	3	19	47<46<24<19	1



Utilisation simple d'un curseur en PL/SQL

```

-- nom d'un étudiant d'identifiant donné
DECLARE
  i Etudiants.IdPersonne%TYPE ; -- identifiant de l'étudiant
  n Etudiants.NomPersonne%TYPE ; -- nom de l'étudiant
BEGIN
  DBMS_OUTPUT.ENABLE(10000) ;
  i := 4 ; -- i := 1 ; -- i := 'azerty' ;
  SELECT NomPersonne INTO n FROM Etudiants WHERE IdPersonne = i ;
  DBMS_OUTPUT.PUT_LINE('Le nom de l''étudiant d''identifiant '||i||' est '||n||
    '.') ;

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Il n''y a pas d''étudiant d''identifiant '||i||'.') ;
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Erreur non prévue ; SQLCODE = ' || SQLCODE) ;
END ;

  i=4
      Le nom de l'étudiant d'identifiant 4 est MARTIN.
  i=1
      Il n'y a pas d'étudiant d'identifiant 1.
  i='azerty'
      Erreur non prévue ; SQLCODE = -6502
-- noms des étudiants nés dans un département donné
DECLARE
  d Etudiants.DepartNaissEtu%TYPE ; -- département de naissance des étudiants
  n Etudiants.NomPersonne%TYPE ; -- nom de l'étudiant
  CURSOR c IS SELECT NomPersonne

```

```

        FROM Etudiants
        WHERE DepartNaissEtu = d
        ORDER BY NomPersonne ;

BEGIN
    DBMS_OUTPUT.ENABLE(10000) ;
    d := '33' ; -- d := '000' ; -- d := 0 ;
    OPEN c ;
    DBMS_OUTPUT.PUT_LINE(' Département : ' || d) ;
    LOOP
        FETCH c INTO n ;
        EXIT WHEN c%NOTFOUND ;
        DBMS_OUTPUT.PUT_LINE(' > Étudiant : ' || n) ;
    END LOOP ;
    CLOSE c ;
END ;

```

```

    d='33'
        Département : 33
        > Étudiant : DURAND
        > Étudiant : LEROI
    d='000'
        Département : 000
    d=0
        Département : 0

```

Réinitialisation d'une séquence

-- affiche prochaine valeur de la séquence après avoir incrémenté sa valeur courante

```
SELECT Sequence_IdPersonne.NEXTVAL FROM DUAL ;
```

```
    NEXTVAL
```

```
    -----
```

```
    4
```

-- affiche la valeur courante de la séquence (sans l'incrémenter)

```
SELECT Sequence_IdPersonne.CURRVAL FROM DUAL ;
```

```
    CURRVAL
```

```
    -----
```

```
    4
```

-- réinitialisation de séquence à 1 de plus que maximum des IdPersonne de Etudiants

```
DECLARE
```

```
    s NUMBER ; -- valeur de la séquence
```

```
    m NUMBER ; -- maximum des Etudiants.IdPersonne (0 si aucun étudiant)
```

```
BEGIN
```

```
    SELECT Sequence_IdPersonne.CURRVAL INTO s FROM DUAL ;
```

```
    s := s - 1 ;
```

```
    EXECUTE IMMEDIATE
```

```
        'ALTER SEQUENCE Sequence_IdPersonne INCREMENT BY -' || CAST(s AS VARCHAR) ;
```

```
    SELECT Sequence_IdPersonne.NEXTVAL INTO s FROM DUAL ; -- affecte 1
```

```
    EXECUTE IMMEDIATE 'ALTER SEQUENCE Sequence_IdPersonne INCREMENT BY 1' ;
```

```
    SELECT COALESCE(MAX(IdPersonne),0) INTO m FROM Etudiants ;
```

```
    IF m > 0 THEN
```

```
        EXECUTE IMMEDIATE
```

```
            'ALTER SEQUENCE Sequence_IdPersonne INCREMENT BY ' || CAST(m AS VARCHAR);
```

```
        SELECT Sequence_IdPersonne.NEXTVAL INTO s FROM DUAL ; -- affecte 1 + m
```

```
        EXECUTE IMMEDIATE 'ALTER SEQUENCE Sequence_IdPersonne INCREMENT BY 1' ;
```

```
    END IF ;
```

```
END ;
```

```
SELECT Sequence_IdPersonne.CURRVAL FROM DUAL ;
```

```
    CURRVAL
```

```
    -----
```

```
    8
```

Informations générales sur un objet complexe

-- attributs des objets

```
SELECT * FROM Etudiants WHERE IdPersonne = 4 ;
```

```
    IDPERSONNE NOMPERSO
```

```
    -----
```

PRENOMSPERSONNE

TELEPHONESPERSONNE(INDICATIFPAYS, TELEPHONE)

ADRESSEPERSONNE(LIGNE1, LIGNE2, LIGNE3, CODEPOSTAL, VILLE, SITEWEB(URL))

DEP

PSEUDOETU

PHOTOETU(SOURCE(LOCALDATA, SRCTYPE, SRCLOCATION, SRCNAME, UPDATETIME, LOCAL), HE

SIGNPHOTOETU(SIGNATURE)

CVETU

VOITURESPOSSEDEES(NOIMMAT(CHIFFRES, LETTRES, DEPART), COULEUR)

DIPLOMESOBTENUS(DIPLOMEOBTENU, ANNEE)

4 MARTIN

TYPE_PRENOMS('Aleyde', 'Aldegonde', 'Albertine')

TYPE_TELEPHONES(TYPE_TELEPHONE('+262 ', '414141414'), TYPE_TELEPHONE('+269 ', '424242424'), TYPE_TELEPHONE('+248 ', '434343434'), TYPE_TELEPHONE('+230 ', '444444444'))

TYPE_ADRESSE('Le Cabinet des Monnaies et Médailles', '10 rue Clovis-Hugues', NULL, '13003', 'Marseille', HTTPURITYPE('www.marseille.fr/sitevdm/jsp/site/Portal.jsp?page_id=282'))

47

FFD8FFE000104A46494600010101006000600000FFE1001645786966000049492A00080000000000
[...]

1C939A28A00FFFD9

ORDIMAGE(ORDSOURCE('FFD8FFE000104A46494600010101004800480000FFDB0043000604050605
[...]

351418C290798C8208FE', 'file', 'REP_ETUDIANTS', 'Aristote.jpg', '26/08/09', 1), 267, 200, 16001, 'JFIF', '24BITRGB', 'JPEG', 'image/jpeg')

ORDIMAGESIGNATURE('425B01006203009850470FD095400800620D000073020000F803000089010
[...]

228')

<Philosophe>

<SourceFiche><http://fr.wikipedia.org/wiki/Aristote></SourceFiche>

<Naissance>

<AnneeNaissance>-384</AnneeNaissance>

<LieuNaissance>Stagire</LieuNaissance>

</Naissance>

<Deces>

<AnneeDeces>-322</AnneeDeces>

<LieuDeces>Chalcis</LieuDeces>

</Deces>

<Ecoles>

<Ecole>fondateur du Lycée</Ecole>

<Ecole>Péripatétisme</Ecole>

</Ecoles>

<Interets>

<Interet>Physique</Interet>

<Interet>Métaphysique</Interet>

<Interet>Biologie</Interet>

<Interet>Éthique</Interet>

<Interet>Politique</Interet>

<Interet>Langage</Interet>

<Interet>Logique</Interet>

<Interet>Poétique</Interet>

<Interet>Rhétorique</Interet>

</Interets>

<Idees>

<Idee>Syllogisme</Idee>

<Idee>Puissance/Acte</Idee>

<Idee>Matière/Forme</Idee>

<Idee>Substance/Accident</Idee>

<Idee>Catégorie</Idee>

</Idees>

<Oeuvres>


```

DEREF(REF(E))(IDPERSONNE, NOMPERSO, PRENOMSPERSO, TELEPHONEPERSONNE(INDIC
-----
TYPE_ETUDIANT(4, 'MARTIN', TYPE_PRENOMS('Aleyde', 'Aldegonde', 'Albertine'), TYP
[...])
82)))
-- OID
SELECT REF(E) FROM Etudiants E WHERE IdPersonne = 4 ;
REF(E)
-----
000028020913064C5048AD493695AF64D7D07F354C674ADD1C9C8447318C14A212F168B0E6010002
5B0001
-- adresses physiques
SELECT ROWID FROM Etudiants WHERE IdPersonne = 4 ;
ROWID
-----
AAARe9AAEAAAAJbAAB

```

Accès aux données et utilisation des méthodes (ici dans SELECT, WHERE, JOIN)

```

-- les étudiants girondins ou non
SELECT IdPersonne , NomPersonne , DepartNaissEtu , E.EstGirondin() ,
      DECODE(E.EstGirondin(),NULL,NULL,0,'N'est pas girondin','Est girondin')
FROM Etudiants E ;
IDPERSONNE NOMPERSO DEP E.ESTGIRONDIN() DECODE(E.ESTGIROND
-----
      4 MARTIN      47      0 N'est pas girondin
      2 LEROI       40      0 N'est pas girondin
      7 LEROI       33      1 Est girondin
      5 DURAND      33      1 Est girondin
      3 DUPOND      17      0 N'est pas girondin

```

```

-- les étudiants non girondins, avec le code et nom du département de naissance
SELECT IdPersonne , NomPersonne , DepartNaissEtu , NomDepartement
FROM Etudiants E
JOIN Departements ON DepartNaissEtu = CodeDepartement
WHERE E.EstGirondin() = 0 ;
IDPERSONNE NOMPERSO DEP NOMDEPARTEMENT
-----
      4 MARTIN      47 Lot-et-Garonne
      2 LEROI       40 Landes
      3 DUPOND      17 Charente-Maritime

```

```

-- l'adresse des étudiants
SELECT IdPersonne , NomPersonne , DepartNaissEtu ,
      E.AdressePersonne.Ligne1 , E.AdressePersonne.DepartAdresse() ,
      E.AdressePersonne.SiteWeb , E.AdressePersonne.SiteWeb.GETURL()
FROM Etudiants E
WHERE IdPersonne = 4 ;
IDPERSONNE NOMPERSO DEP
-----
ADRESSEPERSONNE.LIGNE1
-----
E.ADRESSEPERSONNE.DEPARTADRESSE()
-----
ADRESSEPERSONNE.SITEWEB(URL)
-----
E.ADRESSEPERSONNE.SITEWEB.GETURL()
-----

```

```

      4 MARTIN      47
Le Cabinet des Monnaies et Médailles
13
HTTPPURITYPE('www.marseille.fr/sitevdm/jsp/site/Portal.jsp?page_id=282')
http://www.marseille.fr/sitevdm/jsp/site/Portal.jsp?page_id=282

```

```

-- les départements de l'adresse et de naissance des étudiants
SELECT IdPersonne , NomPersonne , DA.NomDepartement , DN.NomDepartement
FROM Etudiants E
JOIN Departements DA ON E.AdressePersonne.DepartAdresse() = DA.CodeDepartement
JOIN Departements DN ON DepartNaissEtu = DN.CodeDepartement ;

```

```

IDPERSONNE  NOMPERSONNE  NOMDEPARTEMENT  NOMDEPARTEMENT
-----
4 MARTIN      Bouches-du-Rhône Lot-et-Garonne
2 LEROI       Rhône             Landes
7 LEROI       Haute-Garonne    Gironde
5 DURAND      Paris             Gironde
3 DUPOND      Alpes-Maritimes  Charente-Maritime
-- les longueurs (en octet) du champ stockant les pseudonymes des étudiants
SELECT IdPersonne , NomPersonne , DBMS_LOB.GETLENGTH(E.PseudoEtu)
FROM Etudiants E ;

```

```

IDPERSONNE  NOMPERSONNE  DBMS_LOB.GETLENGTH(E.PSEUDOETU)
-----
4 MARTIN      1288
2 LEROI       1201
7 LEROI       2535
5 DURAND      1334
3 DUPOND      1051

```

```

-- les informations sur les photographies des étudiants
SELECT IdPersonne , NomPersonne , -- E.PhotoEtu.SOURCE.LOCALDATA ,
E.PhotoEtu.SOURCE.SRCTYPE , E.PhotoEtu.SOURCE.SRCLLOCATION ,
E.PhotoEtu.SOURCE.SRCNAME , E.PhotoEtu.SOURCE.UPDATETIME ,
E.PhotoEtu.SOURCE.LOCAL ,
E.PhotoEtu.HEIGHT , E.PhotoEtu.WIDTH , E.PhotoEtu.CONTENTLENGTH ,
E.PhotoEtu.FILEFORMAT , E.PhotoEtu.CONTENTFORMAT ,
E.PhotoEtu.COMPRESSIONFORMAT , E.PhotoEtu.MIMETYPE
FROM Etudiants E
WHERE IdPersonne = 4 ;

```

```

IDPERSONNE  NOMPERSONNE
-----
PHOTOETU.SOURCE.SRCTYPE
-----
PHOTOETU.SOURCE.SRCLLOCATION
-----
PHOTOETU.SOURCE.SRCNAME
-----
PHOTOETU PHOTOETU.SOURCE.LOCAL PHOTOETU.HEIGHT PHOTOETU.WIDTH
-----
PHOTOETU.CONTENTLENGTH
-----
PHOTOETU.FILEFORMAT
-----
PHOTOETU.CONTENTFORMAT
-----
PHOTOETU.COMPRESSIONFORMAT
-----
PHOTOETU.MIMETYPE
-----

```

```

4 MARTIN
file
REP_ETUDIANTS
Aristote.jpg
26/08/09          1          267          200
                16001
JFIF
24BITRGB
JPEG
image/jpeg

```

```

SELECT IdPersonne , NomPersonne ,
E.PhotoEtu.GETSOURCELOCATION() ,
E.PhotoEtu.GETSOURCEINFORMATION() ,
E.PhotoEtu.GETHEIGHT() , E.PhotoEtu.GETWIDTH() ,
E.PhotoEtu.GETCONTENTLENGTH() , E.PhotoEtu.GETFILEFORMAT() ,
E.PhotoEtu.GETCONTENTFORMAT() , E.PhotoEtu.GETCOMPRESSIONFORMAT()
FROM Etudiants E

```

```

WHERE IdPersonne = 4 ;
IDPERSONNE NOMPERSO
-----
E.PHOTOETU.GETSOURCETYPE( )
-----
E.PHOTOETU.GETSOURCELOCATION( )
-----
E.PHOTOETU.GETSOURCENAME( )
-----
E.PHOTOE
-----
E.PHOTOETU.SOURCE.GETSOURCEINFORMATION( )
-----
E.PHOTOETU.GETHEIGHT( ) E.PHOTOETU.GETWIDTH( ) E.PHOTOETU.GETCONTENTLENGTH( )
-----
E.PHOTOETU.GETFILEFORMAT( )
-----
E.PHOTOETU.GETCONTENTFORMAT( )
-----
E.PHOTOETU.GETCOMPRESSIONFORMAT( )
-----
4 MARTIN
file
REP_ETUDIANTS
Aristote.jpg
26/08/09
file://REP_ETUDIANTS/Aristote.jpg
267 200 16001
JFIF
24BITRGB
JPEG

```

Appel d'une procédure

```

-- compare deux à deux les signatures des photographies des étudiants
EXECUTE CompareSignPhotosEtudiants(30) ;
    Les scores des comparaisons deux à deux des photographies des étudiants sont
    les suivants (avec color=0.0,texture=0.5,shape=1.0,location=0.0) :
    > 25,7206 % entre les étudiants n° 2 et 3
    > 26,3497 % entre les étudiants n° 2 et 4
    > 38,3417 % entre les étudiants n° 2 et 5
    > 62,5071 % entre les étudiants n° 2 et 7
    > 34,5488 % entre les étudiants n° 3 et 4
    > 27,4646 % entre les étudiants n° 3 et 5
    > 28,919 % entre les étudiants n° 3 et 7
    > 27,3621 % entre les étudiants n° 4 et 5
    > 28,2179 % entre les étudiants n° 4 et 7
    > 12,8742 % entre les étudiants n° 5 et 7
    Les photographies des étudiants les plus similaires sont les suivantes (avec
    color=0.0,texture=0.5,shape=1.0,location=0.0 et distantes de 30% au maximum) :
    > 12,8742 % entre les étudiants n° 5 et 7
    > 25,7206 % entre les étudiants n° 2 et 3
    > 26,3497 % entre les étudiants n° 2 et 4
    > 27,3621 % entre les étudiants n° 4 et 5
    > 27,4646 % entre les étudiants n° 3 et 5
    > 28,2179 % entre les étudiants n° 4 et 7
    > 28,919 % entre les étudiants n° 3 et 7

```

Utilisation de **MULTISET** : relation plate transformée en **NF²**

```

-- convertit les identifiant, début nom et département naissance en immatriculation
SELECT CAST( MULTISET( SELECT Type_ImmatVoiture(IdPersonne , SUBSTR(NomPersonne,1,3),
                                                                    DepartNaissEtu)
                        FROM Etudiants )
          AS Type_ImmatVoitures)
FROM DUAL ;

```

```

CAST(MULTISET( SELECTTYPE_IMMATVOITURE( IDPERSONNE ,SUBSTR(NOMPERSONNE,1,3) ,DEPART
-----
TYPE_IMMATVOITURES(TYPE_IMMATVOITURE(4, 'MAR', '47'), TYPE_IMMATVOITURE(2, 'LER
', '40'), TYPE_IMMATVOITURE(7, 'LER', '33'), TYPE_IMMATVOITURE(5, 'DUR', '33'),
TYPE_IMMATVOITURE(3, 'DUP', '17'))
-- idem, pour chaque étudiant
SELECT E.IdPersonne , E.NomPersonne , E.DepartNaissEtu ,
      CAST( MULTISET( SELECT Type_ImmatVoiture(IdPersonne , SUBSTR(NomPersonne,1,3),
                          DepartNaissEtu)
                          FROM Etudiants EV
                          WHERE EV.IdPersonne = E.IdPersonne )
      AS Type_ImmatVoitures)
FROM Etudiants E ;
IDPERSONNE NOMPERSONNE DEP CAST(MULTISET(SELECTTYPE_IMMATVOITURE( IDPERSONNE ,SUBSTR(NOM
-----
4 MARTIN 47 TYPE_IMMATVOITURES(TYPE_IMMATVOITURE(4, 'MAR', '47'))
2 LEROI 40 TYPE_IMMATVOITURES(TYPE_IMMATVOITURE(2, 'LER', '40'))
7 LEROI 33 TYPE_IMMATVOITURES(TYPE_IMMATVOITURE(7, 'LER', '33'))
5 DURAND 33 TYPE_IMMATVOITURES(TYPE_IMMATVOITURE(5, 'DUR', '33'))
3 DUPOND 17 TYPE_IMMATVOITURES(TYPE_IMMATVOITURE(3, 'DUP', '17'))
-- les étudiants et leurs voitures
SELECT E.IdPersonne , E.NomPersonne ,
      CAST( MULTISET( SELECT EVPI.NoImmat.ConcatNoImmat(' ')
      FROM TABLE ( SELECT VoituresPossedees
      FROM Etudiants EVP
      WHERE EVP.IdPersonne = E.IdPersonne ) EVPI )
      AS Type_VARRAY_VARCHAR )
FROM Etudiants E ;
IDPERSONNE NOMPERSONNE CAST(MULTISET(SELECT EVPI.NOIMMAT.CONCATNOIMMAT((' ')FROM
-----
4 MARTIN TYPE_VARRAY_VARCHAR('4747 LA 47')
2 LEROI TYPE_VARRAY_VARCHAR()
7 LEROI TYPE_VARRAY_VARCHAR()
5 DURAND TYPE_VARRAY_VARCHAR('3333 BX 33', '4040 NT 40')
3 DUPOND TYPE_VARRAY_VARCHAR()

```

Informations sur les collections imbriquées en NF²

```

-- nom et prénoms, téléphones, voitures possédées et diplômes obtenus des étudiants
SELECT IdPersonne , NomPersonne ,
      E.InitialePrenomPersonne() , E.NomPrenomPersonne() ,
      PrenomsPersonne , TelephonesPersonne , VoituresPossedees , DiplomesObtenus
FROM Etudiants E
WHERE IdPersonne = 4 ;
IDPERSONNE NOMPERSONNE E.INITIALEPRENOMPERSONNE() E.NOMPRENOMPERSONNE()
-----
PRENOMSPERSONNE
-----
TELEPHONESPERSONNE(INDICATIFPAYS, TELEPHONE)
-----
VOITURESPOSSEDEES(NOIMMAT(CHIFFRES, LETTRES, DEPART), COULEUR)
-----
DIPLOMESOBTENUS(DIPLOMEOBTENU, ANNEE)
-----
4 MARTIN A. MARTIN Aleyde
TYPE_PRENOMS('Aleyde', 'Aldegonde', 'Albertine')
TYPE_TELEPHONES(TYPE_TELEPHONE('+262 ', '414141414'), TYPE_TELEPHONE('+269 ',
'424242424'), TYPE_TELEPHONE('+248 ', '434343434'), TYPE_TELEPHONE('+230 ',
'444444444'))
TYPE_VOITURES(TYPE_VOITURE(TYPE_IMMATVOITURE(4747, 'LA', '47'), 'rouge'))
TYPE_DIPLOMESOBTENUS(TYPE_DIPLOMEOBTENU(00002202083590848D959C475394B4BAC664D5A
6C8C79207B4E5AA45C5AE4A0E7106AC0AC1, 1977), TYPE_DIPLOMEOBTENU(0000220208DDB47A
51BC564564A483597034FBEE4EC79207B4E5AA45C5AE4A0E7106AC0AC1, 1980), TYPE_DIPLOME
OBTENU(0000220208094C9F5511D245218CBDD7197FCD2C87C79207B4E5AA45C5AE4A0E7106AC0A
C1, 1982))

```

Informations sur les collections imbriquées en relation plate

```
-- nom et prénoms des étudiants (version 1)
SELECT IdPersonne , COLUMN_VALUE , PrenomsPersonne
FROM Etudiants E , TABLE ( E.PrenomsPersonne )
WHERE IdPersonne = 4 ;
-- nom et prénoms des étudiants (version 2)
SELECT IdPersonne , COLUMN_VALUE , PrenomsPersonne
FROM Etudiants E , TABLE ( SELECT PrenomsPersonne
                             FROM Etudiants EPP
                             WHERE EPP.IdPersonne = E.IdPersonne )
WHERE IdPersonne = 4 ;
      IDPERSONNE COLUMN_VALUE PRENOMSPERSONNE
-----
      4 Aleyde          TYPE_PRENOMS('Aleyde', 'Aldegonde', 'Albertine')
      4 Aldegonde       TYPE_PRENOMS('Aleyde', 'Aldegonde', 'Albertine')
      4 Albertine       TYPE_PRENOMS('Aleyde', 'Aldegonde', 'Albertine')
-- téléphones des étudiants (version 1)
SELECT IdPersonne , ETP.IndicatifPays , ETP.Telephone , TelephonesPersonne
FROM Etudiants E , TABLE ( E.TelephonesPersonne ) ETP
WHERE IdPersonne = 4 ;
-- téléphones des étudiants (version 2)
SELECT IdPersonne , ETP.IndicatifPays , ETP.Telephone , TelephonesPersonne
FROM Etudiants E , TABLE ( SELECT TelephonesPersonne
                             FROM Etudiants ETP
                             WHERE ETP.IdPersonne = E.IdPersonne ) ETP
WHERE IdPersonne = 4 ;
      IDPERSONNE INDICA TELEPHONE
-----
      TELEPHONESPERSONNE(INDICATIFPAYS, TELEPHONE)
-----
      4 +262  414141414
TYPE_TELEPHONES(TYPE_TELEPHONE('+262  ', '414141414'), TYPE_TELEPHONE('+269  ',
'424242424'), TYPE_TELEPHONE('+248  ', '434343434'), TYPE_TELEPHONE('+230  ',
'444444444'))
      4 +269  424242424
TYPE_TELEPHONES(TYPE_TELEPHONE('+262  ', '414141414'), TYPE_TELEPHONE('+269  ',
'424242424'), TYPE_TELEPHONE('+248  ', '434343434'), TYPE_TELEPHONE('+230  ',
'444444444'))
      4 +248  434343434
TYPE_TELEPHONES(TYPE_TELEPHONE('+262  ', '414141414'), TYPE_TELEPHONE('+269  ',
'424242424'), TYPE_TELEPHONE('+248  ', '434343434'), TYPE_TELEPHONE('+230  ',
'444444444'))
      4 +230  444444444
TYPE_TELEPHONES(TYPE_TELEPHONE('+262  ', '414141414'), TYPE_TELEPHONE('+269  ',
'424242424'), TYPE_TELEPHONE('+248  ', '434343434'), TYPE_TELEPHONE('+230  ',
'444444444'))
-- voitures possédées par les étudiants (version 1)
SELECT IdPersonne , NomPersonne ,
      EVP.NoImmat.Chiffres , EVP.NoImmat.Lettres , EVP.NoImmat.Depart ,
      EVP.NoImmat.ConcatNoImmat(' ') , EVP.Couleur , VoituresPossedees
FROM Etudiants E , TABLE ( E.VoituresPossedees ) EVP ;
-- voitures possédées par les étudiants (version 2)
SELECT IdPersonne , NomPersonne ,
      EVP.NoImmat.Chiffres , EVP.NoImmat.Lettres , EVP.NoImmat.Depart ,
      EVP.NoImmat.ConcatNoImmat(' ') , EVP.Couleur , VoituresPossedees
FROM Etudiants E , TABLE ( SELECT VoituresPossedees
                             FROM Etudiants EVP
                             WHERE EVP.IdPersonne = E.IdPersonne ) EVP ;
      IDPERSONNE NOMPERSO     NOIMMAT.CHIFFRES NOI NOI
-----
      EVP.NOIMMAT.CONCATNOIMMAT(' ') COULEUR
-----
      VOITURESPOSSEDEES(NOIMMAT(CHIFFRES, LETTRES, DEPART), COULEUR)
-----
```

```

5 DURAND 3333 BX 33
3333 BX 33 rouge
TYPE_VOITURES(TYPE_VOITURE(TYPE_IMMATVOITURE(3333, 'BX', '33'), 'rouge'), TYPE_
VOITURE(TYPE_IMMATVOITURE(4040, 'NT', '40'), 'jaune'))
5 DURAND 4040 NT 40
4040 NT 40 jaune
TYPE_VOITURES(TYPE_VOITURE(TYPE_IMMATVOITURE(3333, 'BX', '33'), 'rouge'), TYPE_
VOITURE(TYPE_IMMATVOITURE(4040, 'NT', '40'), 'jaune'))
4 MARTIN 4747 LA 47
4747 LA 47 rouge
TYPE_VOITURES(TYPE_VOITURE(TYPE_IMMATVOITURE(4747, 'LA', '47'), 'rouge'))
-- voitures possédées par les étudiants, dont ceux n'ayant pas de voiture

```

```

SELECT IdPersonne , NomPersonne ,
EVP.NoImmat.ConcatNoImmat(' ') , EVP.Couleur
FROM Etudiants E , TABLE ( E.VoituresPossedees ) (+) EVP
ORDER BY IdPersonne , EVP.NoImmat.ConcatNoImmat(' ') ;
IDPERSONNE NOMPERSONNE EVP.NOIMMA COULEUR

```

```

-----
2 LEROI
3 DUPOND
4 MARTIN 4747 LA 47 rouge
5 DURAND 3333 BX 33 rouge
5 DURAND 4040 NT 40 jaune
7 LEROI

```

-- diplômes obtenus par les étudiants (version 1)

```

SELECT IdPersonne , NomPersonne ,
EDO.DiplomeObtenu , Deref(EDO.DiplomeObtenu) , EDO.Annee , DiplomesObtenus
FROM Etudiants E , TABLE ( E.DiplomesObtenus ) EDO
WHERE IdPersonne = 4 ;

```

-- diplômes obtenus par les étudiants (version 2)

```

SELECT IdPersonne , NomPersonne ,
EDO.DiplomeObtenu , Deref(EDO.DiplomeObtenu) , EDO.Annee , DiplomesObtenus
FROM Etudiants E , TABLE ( SELECT DiplomesObtenus
FROM Etudiants EDO
WHERE EDO.IdPersonne = E.IdPersonne ) EDO

```

WHERE IdPersonne = 4 ;

IDPERSONNE NOMPERSONNE

DIPLOMEOBTENU

Deref(EDO.DIPLOMEOBTENU)(INTITABREGE, INTITCOMPLET)

ANNEE

DIPLOMESOBTENUS(DIPLOMEOBTENU, ANNEE)

4 MARTIN

```

00002202083590848D959C475394B4BAC664D5A6C8C79207B4E5AA45C5AE4A0E7106AC0AC1
TYPE_DIPLOME('BAC ', 'Baccalauréat')
1977

```

```

TYPE_DIPLOMESOBTENUS(TYPE_DIPLOMEOBTENU(00002202083590848D959C475394B4BAC664D5A
6C8C79207B4E5AA45C5AE4A0E7106AC0AC1, 1977), TYPE_DIPLOMEOBTENU(0000220208DDB47A
51BC564564A483597034FBEE4EC79207B4E5AA45C5AE4A0E7106AC0AC1, 1980), TYPE_DIPLOME
OBTENU(0000220208094C9F5511D245218CBDD7197FCD2C87C79207B4E5AA45C5AE4A0E7106AC0A
C1, 1982))

```

4 MARTIN

```

0000220208DDB47A51BC564564A483597034FBEE4EC79207B4E5AA45C5AE4A0E7106AC0AC1
TYPE_DIPLOME('DEUG ', 'Diplôme d'Études Universitaires Générales')
1980

```

```

TYPE_DIPLOMESOBTENUS(TYPE_DIPLOMEOBTENU(00002202083590848D959C475394B4BAC664D5A
6C8C79207B4E5AA45C5AE4A0E7106AC0AC1, 1977), TYPE_DIPLOMEOBTENU(0000220208DDB47A
51BC564564A483597034FBEE4EC79207B4E5AA45C5AE4A0E7106AC0AC1, 1980), TYPE_DIPLOME
OBTENU(0000220208094C9F5511D245218CBDD7197FCD2C87C79207B4E5AA45C5AE4A0E7106AC0A
C1, 1982))

```

```

4 MARTIN
0000220208094C9F5511D245218CBDD7197FCD2C87C79207B4E5AA45C5AE4A0E7106AC0AC1
TYPE_DIPLOME('MIAGe', 'Maîtrise des Méthodes Informatiques Appliquées à la Gest
ion des Entreprises')
1982
TYPE_DIPLOMESOBTENUS(TYPE_DIPLOMEOBTENU(00002202083590848D959C475394B4BAC664D5A
6C8C79207B4E5AA45C5AE4A0E7106AC0AC1, 1977), TYPE_DIPLOMEOBTENU(0000220208DDB47A
51BC564564A483597034FBEE4EC79207B4E5AA45C5AE4A0E7106AC0AC1, 1980), TYPE_DIPLOME
OBTENU(0000220208094C9F5511D245218CBDD7197FCD2C87C79207B4E5AA45C5AE4A0E7106AC0A
C1, 1982))
-- diplômes obtenus par les étudiants autres que ceux de n° 3 et 4,
-- dont ceux n'ayant pas de diplôme
SELECT IdPersonne , NomPersonne ,
      DEREf(EDO.DiplomeObtenu).IntitAbrege , DEREf(EDO.DiplomeObtenu).IntitCompleT ,
      EDO.Annee
FROM Etudiants E , TABLE ( E.DiplomesObtenus ) (+) EDO
WHERE IdPersonne NOT IN ( 3 , 4 )
ORDER BY IdPersonne , DEREf(EDO.DiplomeObtenu).IntitAbrege ;
IDPERSONNE  NOMPERSO
NNE  DEREf  DEREf(EDO.DIPLOMEOBTENU).INTITCOMPLET  ANNEE
-----
2 LEROI      BAC      Baccalauréat  1980
2 LEROI      DEUG     Diplôme d'Études Universitaires Générales 1982
5 DURAND     BAC      Baccalauréat  1981
5 DURAND     DUT      Diplôme Universitaire de Technologie 1983
7 LEROI

```

Informations sur les collections imbriquées avec un curseur

```

-- nom et prénoms des étudiants
SELECT IdPersonne , NomPersonne , CURSOR ( SELECT * FROM TABLE(PrenomsPersonne) )
FROM Etudiants
ORDER BY NomPersonne , IdPersonne ;
IDPERSONNE  NOMPERSO
NNE  CURSOR(SELECT*FROMTA
-----
3 DUPOND  CURSOR STATEMENT : 3
CURSOR STATEMENT : 3
COLUMN_VALUE
-----
Philémon
Placide
Philomène
Prosper
Parfait
5 ligne(s) sélectionnée(s).
5 DURAND  CURSOR STATEMENT : 3
CURSOR STATEMENT : 3
COLUMN_VALUE
-----
Esther
Eurielle
Édouardine
3 ligne(s) sélectionnée(s).
2 LEROI  CURSOR STATEMENT : 3
CURSOR STATEMENT : 3
COLUMN_VALUE
-----
Saturnin
Symphorien
Samson
Siméon
Séraphin
5 ligne(s) sélectionnée(s).
7 LEROI  CURSOR STATEMENT : 3
CURSOR STATEMENT : 3
COLUMN_VALUE

```

```

-----
Andoche
Ambroise
Alfred
Anastase
Aloysius
5 ligne(s) sélectionnée(s).
      4 MARTIN                                CURSOR STATEMENT : 3
CURSOR STATEMENT : 3
COLUMN_VALUE
-----

Aleyde
Aldegonde
Albertine
3 ligne(s) sélectionnée(s).
5 ligne(s) sélectionnée(s).
-- téléphones des étudiants
SELECT IdPersonne , NomPersonne , CURSOR ( SELECT * FROM TABLE(TelephonesPersonne) )
FROM Etudiants
WHERE IdPersonne = 4 ;
      IDPERSONNE NOMPERSONNE CURSOR(SELECT*FROMTA
-----
      4 MARTIN                                CURSOR STATEMENT : 3
CURSOR STATEMENT : 3
INDICA TELEPHONE
-----
+262  414141414
+269  424242424
+248  434343434
+230  444444444
4 ligne(s) sélectionnée(s).
1 ligne sélectionnée.
-- voitures possédées par les étudiants
SELECT IdPersonne , NomPersonne , CURSOR ( SELECT * FROM TABLE(VoituresPossedees) )
FROM Etudiants
WHERE IdPersonne = 4 ;
      IDPERSONNE NOMPERSONNE CURSOR(SELECT*FROMTA
-----
      4 MARTIN                                CURSOR STATEMENT : 3
CURSOR STATEMENT : 3
NOIMMAT(CHIFFRES, LETTRES, DEPART) COULEUR
-----
TYPE_IMMATVOITURE(4747, 'LA', '47') rouge
1 ligne sélectionnée.
1 ligne sélectionnée.
-- diplômes obtenus par les étudiants
SELECT IdPersonne , NomPersonne , CURSOR ( SELECT * FROM TABLE(DiplomesObtenus) )
FROM Etudiants
WHERE IdPersonne = 4 ;
      IDPERSONNE NOMPERSONNE CURSOR(SELECT*FROMTA
-----
      4 MARTIN                                CURSOR STATEMENT : 3
CURSOR STATEMENT : 3
DIPLOMEOBTENU
-----
ANNEE
-----
00002202083590848D959C475394B4BAC664D5A6C8C79207B4E5AA45C5AE4A0E7106AC0AC1
1977
0000220208DDB47A51BC564564A483597034FBEE4EC79207B4E5AA45C5AE4A0E7106AC0AC1
1980
0000220208094C9F5511D245218CBDD7197FCD2C87C79207B4E5AA45C5AE4A0E7106AC0AC1
1982
3 ligne(s) sélectionnée(s).
1 ligne sélectionnée.

```



```

SELECT IdPersonne , NomPersonne ,
      CURSOR ( SELECT DEREf(EDO.DiplomeObtenu) , EDO.Annee FROM
TABLE(DiplomesObtenus) EDO )
FROM Etudiants
WHERE IdPersonne = 4;
IDPERSONNE NOMPERSO NNE CURSOR(SELECTDEREF(E
-----
          4 MARTIN          CURSOR STATEMENT : 3
CURSOR STATEMENT : 3
DEREF(EDO.DIPLOMEOBTENU)(INTITABREGE, INTITCOMPLET)          ANNEE
-----
TYPE_DIPLOME('BAC ', 'Baccalauréat')          1977
TYPE_DIPLOME('DEUG ', 'Diplôme d'Études Universitaires Générales')          1980
TYPE_DIPLOME('MIAGE', 'Maîtrise des Méthodes Informatiques Appliquées à
la Gestion des Entreprises')          1982
3 ligne(s) sélectionnée(s).
1 ligne sélectionnée.

```

Utilisation d'une vue

```

-- toutes les voitures des étudiants
SELECT * FROM Vue_VoituresEtudiants ;
NOIMMATCHIFFRES NOI NOI COULEUR IDPERSONNE NOMPERSO NNE DEP
-----
          3333 BX  33  rouge          5 DURAND          33
          4040 NT  40  jaune          5 DURAND          33
          4747 LA  47  rouge          4 MARTIN          47

```

Nombre d'éléments dans les collections imbriquées

```

-- nombre de prénoms des étudiants (version 1)
SELECT IdPersonne , NomPersonne , COUNT(*)
FROM Etudiants E , TABLE ( SELECT PrenomsPersonne
FROM Etudiants EPP
WHERE EPP.IdPersonne = E.IdPersonne )
GROUP BY IdPersonne , NomPersonne ;
-- nombre de prénoms des étudiants (version 2)
SELECT IdPersonne , NomPersonne , COUNT(*)
FROM Etudiants E , TABLE ( E.PrenomsPersonne )
GROUP BY IdPersonne , NomPersonne ;
-- nombre de prénoms des étudiants (version 3)
SELECT IdPersonne , NomPersonne , ( SELECT COUNT(*) FROM TABLE(PrenomsPersonne) )
FROM Etudiants ;
IDPERSONNE NOMPERSO NNE C
-----
          4 MARTIN          3
          2 LEROI          5
          7 LEROI          5
          3 DUPOND          5
          5 DURAND          3
-- nombre de téléphones, voitures possédées et diplômes obtenus par les étudiants
SELECT IdPersonne , NomPersonne ,
      CARDINALITY(TelephonesPersonne) ,
      CARDINALITY(VoituresPossedees) ,
      CARDINALITY(DiplomesObtenus)
FROM Etudiants ;
IDPERSONNE NOMPERSO NNE CARDINALITY(TELEP CARDINALITY(VOITU CARDINALITY(DIPLO
-----
          4 MARTIN          4          1          3
          2 LEROI          2          0          2
          7 LEROI          1          0          0
          5 DURAND          9          2          2
          3 DUPOND          0          0          3

```

Modifications simples

```
-- modification de département de naissance, directement
UPDATE Etudiants SET DepartNaissEtu = '973' WHERE IdPersonne = 4 ;
    DEP
    ---
    973
-- modification de département de naissance, avec la méthode
DECLARE
    etd Type_Etudiant ; -- l'étudiant à modifier (département de naissance)
BEGIN
    SELECT VALUE(E) INTO etd FROM Etudiants E WHERE IdPersonne = 4 ;
    etd.DepartNaissEtu_Update('01') ;
END ;
    DEP
    ---
    01
-- modification de la première ligne d'adresse (directement)
UPDATE Etudiants E
    SET E.AdressePersonne.Ligne1 = 'Monnaies et Médailles'
    WHERE IdPersonne = 4 ;
    ADRESSEPERSONNE.LIGNE1
    -----
    Monnaies et Médailles
-- suppression globale des prénoms
UPDATE Etudiants SET PrenomsPersonne = Type_Prenoms ( ) WHERE IdPersonne = 4 ;
    PRENOMSPERSONNE
    -----
    TYPE_PRENOMS()
-- modification globale des prénoms
UPDATE Etudiants
    SET PrenomsPersonne = Type_Prenoms ( 'Aleyde', 'Aldegonde', 'Albertine' )
    WHERE IdPersonne = 4 ;
    PRENOMSPERSONNE
    -----
    TYPE_PRENOMS('Aleyde', 'Aldegonde', 'Albertine')
-- suppression globale des téléphones
UPDATE Etudiants SET TelephonesPersonne = Type_Telephones ( ) WHERE IdPersonne = 4 ;
    TELEPHONESPERSONNE(INDICATIFPAYS, TELEPHONE)
    -----
    TYPE_TELEPHONES()
-- modification globale des téléphones
UPDATE Etudiants
    SET TelephonesPersonne = Type_Telephones (
        Type_Telephone ( '+262' , '414141414' ) ,
        Type_Telephone ( '+269' , '424242424' ) ,
        Type_Telephone ( '+248' , '434343434' ) ,
        Type_Telephone ( '+230' , '444444444' ) )
    WHERE IdPersonne = 4 ;
    TELEPHONESPERSONNE(INDICATIFPAYS, TELEPHONE)
    -----
    TYPE_TELEPHONES(TYPE_TELEPHONE('+262' , '414141414'), TYPE_TELEPHONE('+269' ,
    '424242424'), TYPE_TELEPHONE('+248' , '434343434'), TYPE_TELEPHONE('+230' ,
    '444444444'))
-- suppression globale des voitures possédées
UPDATE Etudiants SET VoituresPossedees = Type_Voitures ( ) WHERE IdPersonne = 4 ;
    VOITURESPOSSEDEES(NOIMMAT(CHIFFRES, LETTRES, DEPART), COULEUR)
    -----
    TYPE_VOITURES()
-- modification globale des voitures possédées
UPDATE Etudiants
    SET VoituresPossedees = Type_Voitures (
        Type_Voiture ( Type_ImmatVoiture ( 4747 , 'LA' , '47' ) , 'rouge' ) )
    WHERE IdPersonne = 4 ;
    VOITURESPOSSEDEES(NOIMMAT(CHIFFRES, LETTRES, DEPART), COULEUR)
```

```

-----
TYPE_VOITURES(TYPE_VOITURE(TYPE_IMMATVOITURE(4747, 'LA', '47'), 'rouge'))
-- suppression globale des diplômes obtenus
UPDATE Etudiants SET DiplomesObtenus = Type_DiplomesObtenus ( ) WHERE IdPersonne = 4;
DIPLOMESOBTENUS(DIPLOMEOBTENU, ANNEE)
-----
TYPE_DIPLOMESOBTENUS()
-- modification globale des diplômes obtenus
UPDATE Etudiants
SET DiplomesObtenus = Type_DiplomesObtenus (
  ( SELECT Type_DiplomeObtenu ( REF(D) , 1977 )
    FROM Diplomes D
    WHERE IntitAbrege = 'BAC' ) ,
  ( SELECT Type_DiplomeObtenu ( REF(D) , 1980 )
    FROM Diplomes D
    WHERE IntitAbrege = 'DEUG' ) ,
  ( SELECT Type_DiplomeObtenu ( REF(D) , 1982 )
    FROM Diplomes D
    WHERE IntitAbrege = 'MIAGE' ) )
WHERE IdPersonne = 4 ;
DIPLOMESOBTENUS(DIPLOMEOBTENU, ANNEE)
-----
TYPE_DIPLOMESOBTENUS(TYPE_DIPLOMEOBTENU(00002202083590848D959C475394B4BAC664D5A
6C8C79207B4E5AA45C5AE4A0E7106AC0AC1, 1977), TYPE_DIPLOMEOBTENU(0000220208DDB47A
51BC564564A483597034FBEE4EC79207B4E5AA45C5AE4A0E7106AC0AC1, 1980), TYPE_DIPLOME
OBTENU(0000220208094C9F5511D245218CBDD7197FCD2C87C79207B4E5AA45C5AE4A0E7106AC0A
C1, 1982))
-- annulation
ROLLBACK ;

```

Mises à jour des éléments des collections imbriquées

Mises à jour des prénoms d'un étudiant

```

-- insertion de prénom
DECLARE
  i NUMBER(2) ; -- identifiant de l'étudiant à modifier (IdPersonne%TYPE)
  e Type_Etudiant ; -- l'étudiant à modifier (prénom à insérer)
BEGIN
  i := 4 ;
  SELECT VALUE(Etu) INTO e FROM Etudiants Etu WHERE IdPersonne = i ;
  e.PrenomPersonne_Insert('Anastasia') ;
  UPDATE Etudiants SET PrenomsPersonne = e.PrenomsPersonne WHERE IdPersonne = i ;
END ;
PRENOMSPERSONNE
-----
TYPE_PRENOMS('Aleyde', 'Aldegonde', 'Albertine', 'Anastasia')
-- modification de prénom
DECLARE
  i NUMBER(2) ; -- identifiant de l'étudiant à modifier (IdPersonne%TYPE)
  e Type_Etudiant ; -- l'étudiant à modifier (prénom à modifier)
BEGIN
  i := 4 ;
  SELECT VALUE(Etu) INTO e FROM Etudiants Etu WHERE IdPersonne = i ;
  e.PrenomPersonne_Update('Anastasia', 'Arabelle') ;
  UPDATE Etudiants SET PrenomsPersonne = e.PrenomsPersonne WHERE IdPersonne = i ;
END ;
PRENOMSPERSONNE
-----
TYPE_PRENOMS('Aleyde', 'Aldegonde', 'Albertine', 'Arabelle')
-- suppression de prénom
DECLARE
  i NUMBER(2) ; -- identifiant de l'étudiant à modifier (IdPersonne%TYPE)
  e Type_Etudiant ; -- l'étudiant à modifier (prénom à supprimer)
BEGIN

```

```

i := 4 ;
SELECT VALUE(Etu) INTO e FROM Etudiants Etu WHERE IdPersonne = i ;
e.PrenomPersonne_Delete('Arabelle') ;
UPDATE Etudiants SET PrenomsPersonne = e.PrenomsPersonne WHERE IdPersonne = i ;
END ;
PRENOMSPERSONNE
-----
TYPE_PRENOMS('Aleyde', 'Aldegonde', 'Albertine')
-- annulation
ROLLBACK ;

```

Mises à jour des téléphones d'un étudiant

```

-- insertion de téléphone
INSERT
INTO TABLE ( SELECT TelephonesPersonne FROM Etudiants WHERE IdPersonne = 4 )
VALUES ( Type_Telephone ( '+1-340' , '404040405' ) ) ;
TELEPHONESPERSONNE(INDICATIFPAYS, TELEPHONE)
-----
TYPE_TELEPHONES(TYPE_TELEPHONE('+262 ', '414141414'), TYPE_TELEPHONE('+269 ',
'424242424'), TYPE_TELEPHONE('+248 ', '434343434'), TYPE_TELEPHONE('+230 ',
'444444444'), TYPE_TELEPHONE('+1-340', '404040405'))
-- modification de téléphone
UPDATE
TABLE ( SELECT TelephonesPersonne FROM Etudiants WHERE IdPersonne = 4 )
SET IndicatifPays = '+408' , Telephone = '494949495'
WHERE IndicatifPays = '+1-340' AND Telephone = '404040405' ;
TELEPHONESPERSONNE(INDICATIFPAYS, TELEPHONE)
-----
TYPE_TELEPHONES(TYPE_TELEPHONE('+262 ', '414141414'), TYPE_TELEPHONE('+269 ',
'424242424'), TYPE_TELEPHONE('+248 ', '434343434'), TYPE_TELEPHONE('+230 ',
'444444444'), TYPE_TELEPHONE('+408 ', '494949495'))
-- suppression de téléphone
DELETE
FROM TABLE ( SELECT TelephonesPersonne FROM Etudiants WHERE IdPersonne = 4 )
WHERE IndicatifPays = '+408' AND Telephone = '494949495' ;
TELEPHONESPERSONNE(INDICATIFPAYS, TELEPHONE)
-----
TYPE_TELEPHONES(TYPE_TELEPHONE('+262 ', '414141414'), TYPE_TELEPHONE('+269 ',
'424242424'), TYPE_TELEPHONE('+248 ', '434343434'), TYPE_TELEPHONE('+230 ',
'444444444'))
-- annulation
ROLLBACK ;

```

Mises à jour des voitures possédées par un étudiant

```

-- insertion de voiture possédée
INSERT
INTO TABLE ( SELECT VoituresPossedees FROM Etudiants WHERE IdPersonne = 4 )
VALUES ( Type_Voiture ( Type_ImmatVoiture ( 0971 , 'BT' , '971' ) , 'orange' ) ) ;
VOITURESPOSSEDEES(NOIMMAT(CHIFFRES, LETTRES, DEPART), COULEUR)
-----
TYPE_VOITURES(TYPE_VOITURE(TYPE_IMMATVOITURE(4747, 'LA', '47'), 'rouge'), TYPE_
VOITURE(TYPE_IMMATVOITURE(971, 'BT', '971'), 'orange'))
-- modification de voiture possédée
UPDATE
TABLE ( SELECT VoituresPossedees FROM Etudiants WHERE IdPersonne = 4 ) EVP
SET EVP.NoImmat.Chiffres = 0972 ,
EVP.NoImmat.Lettres = 'SP' ,
EVP.NoImmat.Depart = '972' ,
Couleur = 'jaune'
WHERE EVP.NoImmat.Chiffres = 0971 AND
EVP.NoImmat.Lettres = 'BT' AND
EVP.NoImmat.Depart = '971' ;
VOITURESPOSSEDEES(NOIMMAT(CHIFFRES, LETTRES, DEPART), COULEUR)
-----

```

```

TYPE_VOITURES(TYPE_VOITURE(TYPE_IMMATVOITURE(4747, 'LA', '47'), 'rouge'), TYPE_
VOITURE(TYPE_IMMATVOITURE(972, 'SP', '972'), 'jaune'))
-- suppression de voiture possédée
DELETE
FROM TABLE ( SELECT VoituresPossedees FROM Etudiants WHERE IdPersonne = 4 ) EVP
WHERE EVP.NoImmat.Chiffres = 0972 AND
      EVP.NoImmat.Lettres = 'SP' AND
      EVP.NoImmat.Depart = '972' ;
VOITURESPOSSEDEES(NOIMMAT(CHIFFRES, LETTRES, DEPART), COULEUR)
-----
TYPE_VOITURES(TYPE_VOITURE(TYPE_IMMATVOITURE(4747, 'LA', '47'), 'rouge'))
-- annulation
ROLLBACK ;

```

Mises à jour des diplômes obtenus par un étudiant

```

-- insertion de diplôme obtenu
INSERT
INTO TABLE ( SELECT DiplomesObtenus FROM Etudiants WHERE IdPersonne = 4 )
VALUES ( ( SELECT Type_DiplomeObtenu ( REF(D) , 2009 )
          FROM Diplomes D
          WHERE IntitAbrege = 'MIAGe' ) ) ;
DIPLOMESOBTENUS(DIPLOMEOBTENU, ANNEE)
-----
TYPE_DIPLOMESOBTENUS(TYPE_DIPLOMEOBTENU(00002202083590848D959C475394B4BAC664D5A
6C8C79207B4E5AA45C5AE4A0E7106AC0AC1, 1977), TYPE_DIPLOMEOBTENU(0000220208DDB47A
51BC564564A483597034FBEE4EC79207B4E5AA45C5AE4A0E7106AC0AC1, 1980), TYPE_DIPLOME
OBTENU(0000220208094C9F5511D245218CBDD7197FCD2C87C79207B4E5AA45C5AE4A0E7106AC0A
C1, 1982), TYPE_DIPLOMEOBTENU(0000220208094C9F5511D245218CBDD7197FCD2C87C79207B
4E5AA45C5AE4A0E7106AC0AC1, 2009))
-- modification de diplôme obtenu
UPDATE
TABLE ( SELECT DiplomesObtenus FROM Etudiants WHERE IdPersonne = 4 ) EDO
SET EDO.DiplomeObtenu = ( SELECT REF(D)
                        FROM Diplomes D
                        WHERE IntitAbrege = 'DEUG' ) ,
      EDO.Annee = 2008
WHERE Deref(EDO.DiplomeObtenu).IntitAbrege = 'MIAGe' AND EDO.Annee = 2009 ;
DIPLOMESOBTENUS(DIPLOMEOBTENU, ANNEE)
-----
TYPE_DIPLOMESOBTENUS(TYPE_DIPLOMEOBTENU(00002202083590848D959C475394B4BAC664D5A
6C8C79207B4E5AA45C5AE4A0E7106AC0AC1, 1977), TYPE_DIPLOMEOBTENU(0000220208DDB47A
51BC564564A483597034FBEE4EC79207B4E5AA45C5AE4A0E7106AC0AC1, 1980), TYPE_DIPLOME
OBTENU(0000220208094C9F5511D245218CBDD7197FCD2C87C79207B4E5AA45C5AE4A0E7106AC0A
C1, 1982), TYPE_DIPLOMEOBTENU(0000220208DDB47A51BC564564A483597034FBEE4EC79207B
4E5AA45C5AE4A0E7106AC0AC1, 2008))
-- suppression de diplôme obtenu
DELETE
FROM TABLE ( SELECT DiplomesObtenus FROM Etudiants WHERE IdPersonne = 4 ) EDO
WHERE Deref(EDO.DiplomeObtenu).IntitAbrege = 'DEUG' AND EDO.Annee = 2008 ;
DIPLOMESOBTENUS(DIPLOMEOBTENU, ANNEE)
-----
TYPE_DIPLOMESOBTENUS(TYPE_DIPLOMEOBTENU(00002202083590848D959C475394B4BAC664D5A
6C8C79207B4E5AA45C5AE4A0E7106AC0AC1, 1977), TYPE_DIPLOMEOBTENU(0000220208DDB47A
51BC564564A483597034FBEE4EC79207B4E5AA45C5AE4A0E7106AC0AC1, 1980), TYPE_DIPLOME
OBTENU(0000220208094C9F5511D245218CBDD7197FCD2C87C79207B4E5AA45C5AE4A0E7106AC0A
C1, 1982))
-- annulation
ROLLBACK ;

```

Illustration de surcharge, exceptions, curseur géré par FOR

```

EXECUTE NbDiplObtEtd.NombreDiplomes ;
      Nombre de diplômes obtenus par les étudiants : 10
EXECUTE NbDiplObtEtd.NombreDiplomes(1977) ;
      1 : diplôme intitulé BAC obtenu l'année 1977 par l'étudiant n° 4

```

```

    Nombre de diplômes obtenus l'année 1977 par les étudiants : 1
    Aucun étudiant ne peut vérifier le prédicat (toujours faux i. e. contradiction)
EXECUTE NbDiplObtEtd.NombreDiplomes(1981) ;
    1 : diplôme intitulé BAC obtenu l'année 1981 par l'étudiant n° 3
    2 : diplôme intitulé BAC obtenu l'année 1981 par l'étudiant n° 5
    L'affichage de la liste a été interrompu suite au 2ème étudiant
EXECUTE NbDiplObtEtd.NombreDiplomes(2009) ;
    Aucun diplôme n'a été obtenu l'année 2009 par les étudiants
    Aucun étudiant ne peut vérifier le prédicat (toujours faux i. e. contradiction)
EXECUTE NbDiplObtEtd.NombreDiplomes('BAC',1981) ;
    1 : diplôme intitulé BAC obtenu l'année 1981 par l'étudiant n° 3
    2 : diplôme intitulé BAC obtenu l'année 1981 par l'étudiant n° 5
    Nombre de diplômes intitulé BAC obtenus l'année 1981 par les étudiants : 2
EXECUTE NbDiplObtEtd.NombreDiplomes('MIAGE',2009) ;
    Nombre de diplômes intitulé MIAGE obtenus l'année 2009 par les étudiants : 0

```

Utilisation du paquetage des contraintes d'intégrité violées

```

INSERT INTO Etudiants VALUES (
    9 , -- Sequence_IdPersonne.NEXTVAL
    'ZIGOTO' ,
    Type_Prenoms ( 'Évariste' , 'Eusèbe' , 'Eustache' , 'Elfried' ,
                  'Ernest-Edgar Évrard Élie-Éloi' ) ,
    Type_Telephones (
        Type_Telephone ( '+998' , '919191919' ) ) ,
    Type_Adresse ( 'Musée Jules Verne' ,
                  '3 rue de l''Hermitage' , NULL , '44100' , 'Nantes' ,
                  URIFACTORY.GETURI('http://www.nantes.fr/julesverne/acc_6.htm') ) ,
    '90' ,
    EMPTY_BLOB() ,
    ORDSYS.ORDIMAGE.INIT('file',UPPER('Rep_Etudiants'),'Épictète.jpg') ,
    ORDSYS.ORDIMAGESIGNATURE.INIT() ,
    XMLTYPE.CREATEXML('
        <Philosophe>
            <SourceFiche>http://fr.wikipedia.org/wiki/Épictète</SourceFiche>
            <Naissance>
                <AnneeNaissance></AnneeNaissance>
                <LieuNaissance></LieuNaissance>
            </Naissance>
            <Deces>
                <AnneeDeces></AnneeDeces>
                <LieuDeces></LieuDeces>
            </Deces>
            <Ecoles>
            </Ecoles>
            <Interets>
                <Interet></Interet>
            </Interets>
            <Idees>
            </Idees>
            <OEuvres>
            </OEuvres>
            <InfluencesPar>
            </InfluencesPar>
            <InfluencesSur>
            </InfluencesSur>
        </Philosophe>
    ') ,
    Type_Voitures (
        Type_Voiture ( Type_ImmatVoiture ( 9001 , 'AZ' , '90' ) , 'orange' ) ,
        Type_Voiture ( Type_ImmatVoiture ( 9002 , 'BY' , '90' ) , 'orange' ) ,
        Type_Voiture ( Type_ImmatVoiture ( 9003 , 'CX' , NULL ) , NULL ) ) ,
    Type_DiplomesObtenus (
        ( SELECT Type_DiplomeObtenu ( REF(D) , 1981 )
          FROM Diplomes D

```

```

        WHERE IntitAbrege = 'BAC' ) ,
    ( SELECT Type_DiplomeObtenu ( REF(D) , 1983 )
      FROM Diplomes D
        WHERE IntitAbrege = 'DUT' ) ,
    ( SELECT Type_DiplomeObtenu ( REF(D) , 1985 )
      FROM Diplomes D
        WHERE IntitAbrege = 'MIAGe' ) )
) ;
INSERT
  INTO TABLE ( SELECT VoituresPossedees FROM Etudiants WHERE IdPersonne = 9 )
  VALUES ( Type_Voiture ( Type_ImmatVoiture ( 9004 , 'DW' , NULL ) , NULL ) ) ;
EXECUTE Civirolees.Default_NoImmatDepart_violee ;
  Pas de valeur pour le département du numéro d'immatriculation de la voiture
  9004 DW possédée par l'étudiant n° 9
EXECUTE Civirolees.Default_Couleur_violee ;
  Pas de valeur pour la couleur de la voiture de numéro d'immatriculation
  9004_DW_ possédée par l'étudiant n° 9
INSERT
  INTO TABLE ( SELECT VoituresPossedees FROM Etudiants WHERE IdPersonne = 9 )
  VALUES ( Type_Voiture ( Type_ImmatVoiture ( 9005 , 'EV' , '9Z' ) , NULL ) ) ;
EXECUTE Civirolees.Ref_ImmatVoiture_Dpts_violee ;
  Le département de la voiture de numéro d'immatriculation 9004_DW_ possédée par
  l'étudiant n° 9 ne référence pas un département
  Le département de la voiture de numéro d'immatriculation 9005_EV_9Z possédée
  par l'étudiant n° 9 ne référence pas un département
INSERT
  INTO TABLE ( SELECT DiplomesObtenus FROM Etudiants WHERE IdPersonne = 9 )
  VALUES ( Type_DiplomeObtenu ( NULL , 1984 ) ) ;
EXECUTE Civirolees.Ref_Etudiants_Dipl_violee ;
  Pas de valeur pour la référence à l'intitulé abrégé du diplôme de l'année 1984
  de l'étudiant n° 9
INSERT INTO Diplomes VALUES ( 'Lic' , 'Licence' ) ;
INSERT
  INTO TABLE ( SELECT DiplomesObtenus FROM Etudiants WHERE IdPersonne = 9 )
  VALUES ( ( SELECT Type_DiplomeObtenu ( REF(D) , 1986 )
            FROM Diplomes D
              WHERE IntitAbrege = 'Lic' ) ) ;
DELETE FROM Diplomes WHERE IntitAbrege = 'Lic' ;
EXECUTE Civirolees.Ref_Etudiants_Dipl_violee ;
  Pas de valeur pour la référence à l'intitulé abrégé du diplôme de l'année 1984
  de l'étudiant n° 9
  Pas de valeur pour la référence à l'intitulé abrégé du diplôme de l'année 1986
  de l'étudiant n° 9
INSERT
  INTO TABLE ( SELECT DiplomesObtenus FROM Etudiants WHERE IdPersonne = 9 )
  VALUES ( ( SELECT Type_DiplomeObtenu ( REF(D) , 1982 )
            FROM Diplomes D
              WHERE IntitAbrege = 'BAC' ) ) ;
EXECUTE Civirolees.Unicite_IdPersonne_Dipl_violee ;
  Il y a 2 années différentes pour la référence à l'intitulé abrégé du diplôme de
  l'année DEUG de l'étudiant n° 9
SELECT IdPersonne , NomPersonne , Deref(EDO.DiplomeObtenu) , EDO.Annee
FROM Etudiants E , TABLE ( E.DiplomesObtenus ) EDO
WHERE IdPersonne = 9 ;
  IDPERSONNE  NOMPERSONNE  Deref(EDO.DIPLOMEOBTENU)(INTITABREGE, INTITCOMPLET ANNEE
  -----
    9 ZIGOTO      TYPE_DIPLOME('BAC  ', 'Baccalauréat')          1981
    9 ZIGOTO      TYPE_DIPLOME('BAC  ', 'Baccalauréat')          1982
    9 ZIGOTO      TYPE_DIPLOME('DUT  ', 'Diplôme Universitaire de Te
    chnologie')                                     1983
    9 ZIGOTO                                     1984
    9 ZIGOTO      TYPE_DIPLOME('MIAGe', 'Maîtrise des Méthodes Infor
    matiques Appliquées à la Gestion des Entreprises') 1985
    9 ZIGOTO                                     1986
-- annulation

```

ROLLBACK ;

XML

Affichage au format XML de données classiques (c.-à-d. non dans un XMLTYPE)

```
-- identifiant et nom des étudiants au format XML
SELECT XMLELEMENT("Etudiant",
                 XMLELEMENT("IdPersonne", IdPersonne), XMLELEMENT("NomPersonne", NomPersonne))
FROM Etudiants ;
-----
XMLELEMENT("ETUDIANT", XMLELEMENT("IDPERSONNE", IDPERSONNE), XMLELEMENT("NOMPERSONNE", NOMP
-----
<Etudiant><IdPersonne>4</IdPersonne><NomPersonne>MARTIN</NomPersonne></Etudiant>
<Etudiant><IdPersonne>2</IdPersonne><NomPersonne>LEROI</NomPersonne></Etudiant>
<Etudiant><IdPersonne>7</IdPersonne><NomPersonne>LEROI</NomPersonne></Etudiant>
<Etudiant><IdPersonne>5</IdPersonne><NomPersonne>DURAND</NomPersonne></Etudiant>
<Etudiant><IdPersonne>3</IdPersonne><NomPersonne>DUPOND</NomPersonne></Etudiant>
SELECT XMLELEMENT("Etudiant", XMLFOREST(IdPersonne, NomPersonne))
FROM Etudiants ;
-----
XMLELEMENT("ETUDIANT", XMLFOREST(IDPERSONNE, NOMPERSONNE))
-----
<Etudiant><IdPersonne>4</IdPersonne><NomPersonne>MARTIN</NomPersonne></Etudiant>
<Etudiant><IdPersonne>2</IdPersonne><NomPersonne>LEROI</NomPersonne></Etudiant>
<Etudiant><IdPersonne>7</IdPersonne><NomPersonne>LEROI</NomPersonne></Etudiant>
<Etudiant><IdPersonne>5</IdPersonne><NomPersonne>DURAND</NomPersonne></Etudiant>
<Etudiant><IdPersonne>3</IdPersonne><NomPersonne>DUPOND</NomPersonne></Etudiant>
SELECT VALUE(E).GETCLOBVAL()
FROM TABLE ( XMLSEQUENCE ( CURSOR ( SELECT IdPersonne , NomPersonne
                                     FROM Etudiants ) ) ) E ;
-----
VALUE(E).GETCLOBVAL()
-----
<ROW>
  <IDPERSONNE>4</IDPERSONNE>
  <NOMPERSONNE>MARTIN</NOMPERSONNE>
</ROW>
<ROW>
  <IDPERSONNE>2</IDPERSONNE>
  <NOMPERSONNE>LEROI</NOMPERSONNE>
</ROW>
<ROW>
  <IDPERSONNE>7</IDPERSONNE>
  <NOMPERSONNE>LEROI</NOMPERSONNE>
</ROW>
<ROW>
  <IDPERSONNE>5</IDPERSONNE>
  <NOMPERSONNE>DURAND</NOMPERSONNE>
</ROW>
<ROW>
  <IDPERSONNE>3</IDPERSONNE>
  <NOMPERSONNE>DUPOND</NOMPERSONNE>
</ROW>
5 ligne(s) sélectionnée(s).
-- les départements et les étudiants nés dans ce département au format XML
SELECT XMLELEMENT("Departement",
                 XMLELEMENT("CodeDepartement", D.CodeDepartement),
                 XMLELEMENT("NomDepartement", D.NomDepartement),
                 XMLELEMENT("Etudiants",
                             XMLAGG(XMLELEMENT("Etudiant",
                                                 XMLELEMENT("IdPersonne", E.IdPersonne),
                                                 XMLELEMENT("NomPersonne", E.NomPersonne))
                                     ORDER BY E.IdPersonne)))
FROM Departements D
JOIN Etudiants E ON CodeDepartement = DepartNaissEtu
WHERE TRIM(D.CodeDepartement) IN ( '33' , '47' )
GROUP BY D.CodeDepartement , D.NomDepartement ;
-----
XMLELEMENT("DEPARTEMENT", XMLELEMENT("CODEDEPARTEMENT", D.CODEDEPARTEMENT), XMLELE
```



```

-----
<Departement>
  <CodeDepartement>33</CodeDepartement>
  <NomDepartement>Gironde</NomDepartement>
  <Etudiants>
    <Etudiant>
      <IdPersonne>5</IdPersonne>
      <NomPersonne>DURAND</NomPersonne>
    </Etudiant>
    <Etudiant>
      <IdPersonne>7</IdPersonne>
      <NomPersonne>LEROI</NomPersonne>
    </Etudiant>
  </Etudiants>
</Departement>
<Departement>
  <CodeDepartement>47</CodeDepartement>
  <NomDepartement>Lot-et-Garonne</NomDepartement>
  <Etudiants>
    <Etudiant>
      <IdPersonne>4</IdPersonne>
      <NomPersonne>MARTIN</NomPersonne>
    </Etudiant>
  </Etudiants>
</Departement>
2 ligne(s) sélectionnée(s)
-- les étudiants (identifiant, nom, prénoms, téléphones, adresse, code et nom du
-- département de naissance, voitures, diplômes) non girondins et dont le nom ne
-- commence pas et ne finit pas par un 'D' au format XML
SELECT DBMS_XMLGEN.GETXML(
  'SELECT IdPersonne , NomPersonne , PrenomsPersonne , TelephonesPersonne ,
    AdressePersonne , DepartNaissEtu , NomDepartement ,
    VoituresPossedees , DiplomesObtenus
  FROM Etudiants E
  JOIN Departements ON CodeDepartement = DepartNaissEtu
  WHERE E.EstGirondin() <> 1 AND
    NomPersonne NOT LIKE ( ' || ''' || 'D%D' || ''' )
  ORDER BY IdPersonne
  ' )
FROM DUAL ;
DBMS_XMLGEN.GETXML( 'SELECT IdPersonne , NomPersonne , PrenomsPersonne , TelephonesPers
-----
<?xml version="1.0"?>
<ROWSET>
  <ROW>
    <IdPersonne>2</IdPersonne>
    <NomPersonne>LEROI</NomPersonne>
    <PrenomsPersonne>
      <VARCHAR2>Saturnin</VARCHAR2>
      <VARCHAR2>Symphorien</VARCHAR2>
      <VARCHAR2>Samson</VARCHAR2>
      <VARCHAR2>Siméon</VARCHAR2>
      <VARCHAR2>Séraphin</VARCHAR2>
    </PrenomsPersonne>
    <TelephonesPersonne>
      <Type_Telephone>
        <IndicatifPays>+30 </IndicatifPays>
        <Telephone>212121212</Telephone>
      </Type_Telephone>
      <Type_Telephone>
        <IndicatifPays>+33 </IndicatifPays>
        <Telephone>222222222</Telephone>
      </Type_Telephone>
    </TelephonesPersonne>
    <AdressePersonne>

```

```

<Ligne1>Musée d'Art Contemporain de Lyon</Ligne1>
<Ligne2>Cité Internationale</Ligne2>
<Ligne3>81 quai Charles de Gaulle</Ligne3>
<CodePostal>69463</CodePostal>
<Ville>Lyon</Ville>
<SiteWeb>
  <URL>www.moca-lyon.org</URL>
</SiteWeb>
</AdressePersonne>
<DepartNaissEtu>40</DepartNaissEtu>
<NomDepartement>Landes</NomDepartement>
<VoituresPossedees/>
<DiplomesObtenus>
  <Type_DiplomeObtenu>
    <DiplomeObtenu>00002202083590848D959C475394B4BAC664D5A6C8C79207B4E5AA45
      C5AE4A0E7106AC0AC1</DiplomeObtenu>
    <Annee>1980</Annee>
  </Type_DiplomeObtenu>
  <Type_DiplomeObtenu>
    <DiplomeObtenu>0000220208DDB47A51BC564564A483597034FBEE4EC79207B4E5AA45
      C5AE4A0E7106AC0AC1</DiplomeObtenu>
    <Annee>1982</Annee>
  </Type_DiplomeObtenu>
</DiplomesObtenus>
</ROW>
<ROW>
  <IdPersonne>4</IdPersonne>
  <NomPersonne>MARTIN</NomPersonne>
  <PrenomsPersonne>
    <VARCHAR2>Aleyde</VARCHAR2>
    <VARCHAR2>Aldegonde</VARCHAR2>
    <VARCHAR2>Albertine</VARCHAR2>
  </PrenomsPersonne>
  <TelephonesPersonne>
    <Type_Telephone>
      <IndicatifPays>+262 </IndicatifPays>
      <Telephone>414141414</Telephone>
    </Type_Telephone>
    <Type_Telephone>
      <IndicatifPays>+269 </IndicatifPays>
      <Telephone>424242424</Telephone>
    </Type_Telephone>
    <Type_Telephone>
      <IndicatifPays>+248 </IndicatifPays>
      <Telephone>434343434</Telephone>
    </Type_Telephone>
    <Type_Telephone>
      <IndicatifPays>+230 </IndicatifPays>
      <Telephone>444444444</Telephone>
    </Type_Telephone>
  </TelephonesPersonne>
  <AdressePersonne>
    <Ligne1>Le Cabinet des Monnaies et Médailles</Ligne1>
    <Ligne2>10 rue Clovis-Hugues</Ligne2>
    <CodePostal>13003</CodePostal>
    <Ville>Marseille</Ville>
    <SiteWeb>
      <URL>www.marseille.fr/sitevdm/jsp/site/Portal.jsp?page_id=282</URL>
    </SiteWeb>
  </AdressePersonne>
  <DepartNaissEtu>47</DepartNaissEtu>
  <NomDepartement>Lot-et-Garonne</NomDepartement>
  <VoituresPossedees>
    <Type_Voiture>
      <NoImmat>

```

```

        <Chiffres>4747</Chiffres>
        <Lettres>LA</Lettres>
        <Depart>47</Depart>
    </NoImmat>
    <Couleur>rouge</Couleur>
</Type_Voiture>
</VoituresPossedees>
<DiplomesObtenus>
    <Type_DiplomeObtenu>
        <DiplomeObtenu>00002202083590848D959C475394B4BAC664D5A6C8C79207B4E5AA45
            C5AE4A0E7106AC0AC1</DiplomeObtenu>
        <Annee>1977</Annee>
    </Type_DiplomeObtenu>
    <Type_DiplomeObtenu>
        <DiplomeObtenu>0000220208DDB47A51BC564564A483597034FBEE4EC79207B4E5AA45
            C5AE4A0E7106AC0AC1</DiplomeObtenu>
        <Annee>1980</Annee>
    </Type_DiplomeObtenu>
    <Type_DiplomeObtenu>
        <DiplomeObtenu>0000220208094C9F5511D245218CBDD7197FCD2C87C79207B4E5AA45
            C5AE4A0E7106AC0AC1</DiplomeObtenu>
        <Annee>1982</Annee>
    </Type_DiplomeObtenu>
</DiplomesObtenus>
</ROW>
</ROWSET>

```

Extraction de données XML (c.-à-d. contenues dans un XMLTYPE)

-- année et lieu naissance ainsi que les intérêts du curriculum vitæ des étudiants

```

SELECT IdPersonne , NomPersonne ,
       EXTRACTVALUE(CVETu, '/Philosophe/Naissance/AnneeNaissance' ) ,
       EXTRACTVALUE(CVETu, '/Philosophe/Naissance/LieuNaissance' ) ,
       EXTRACTVALUE(CVETu, '/Philosophe/Interets/Interet[1]')
FROM Etudiants
WHERE IdPersonne IN ( 4 , 7 ) ;
-----
IDPERSONNE  NOMPERSO  EXTRACTVALUE(CVETU  EXTRACTVALUE(CVETU  EXTRACTVALUE(CVETU
-----
          4  MARTIN          -384          Stagire          Physique
          7  LEROI           1126          Cordoue          Métaphysique

```

-- les intérêts (philosophiques) du curriculum vitæ des étudiants

```

SELECT IdPersonne , NomPersonne , EXTRACT(CVETu, '/Philosophe/Interets/Interet')
FROM Etudiants
WHERE IdPersonne IN ( 4 , 7 ) ;
-----
IDPERSONNE  NOMPERSO  EXTRACT(CVETU, '/PHILOSOPHE/INTERETS/INTERET')
-----
          4  MARTIN          <Interet>Physique</Interet>
          4  MARTIN          <Interet>Métaphysique</Interet>
          4  MARTIN          <Interet>Biologie</Interet>
          4  MARTIN          <Interet>Éthique</Interet>
          4  MARTIN          <Interet>Politique</Interet>
          4  MARTIN          <Interet>Langage</Interet>
          4  MARTIN          <Interet>Logique</Interet>
          4  MARTIN          <Interet>Poétique</Interet>
          4  MARTIN          <Interet>Rhétorique</Interet>
          7  LEROI          <Interet>Métaphysique</Interet>
          7  LEROI          <Interet>Théologie</Interet>
          7  LEROI          <Interet>Droit</Interet>
          7  LEROI          <Interet>Médecine</Interet>
          7  LEROI          <Interet>Politique</Interet>
          7  LEROI          <Interet>Religion</Interet>

```

2 ligne(s) sélectionnée(s).

-- les sources d'information du curriculum vitæ des étudiants

```

SELECT IdPersonne , NomPersonne ,
       EXTRACT(CVETu, '/Philosophe/SourceFiche' ) ,

```

```

EXTRACTVALUE(CVetu, '/Philosophe/SourceFiche' ) ,
XMLQUERY(' $o/Philosophe/SourceFiche/text()'
  PASSING BY VALUE CVetu AS "o" RETURNING CONTENT) ,
XMLCAST(XMLQUERY(' $o/Philosophe/SourceFiche/text()'
  PASSING BY VALUE CVetu AS "o" RETURNING CONTENT)
  AS VARCHAR2(23))
FROM Etudiants
WHERE IdPersonne IN ( 4 , 7 ) ;
IDPERSONNE NOMPERSONNE
-----
EXTRACT(CVETU, '/PHILOSOPHE/SOURCEFICHE' )
-----
EXTRACTVALUE(CVETU, '/PHILOSOPHE/SOURCEFICHE' )
-----
XMLQUERY(' $O/PHILOSOPHE/SOURCEFICHE/TEXT()' PASSINGBYVALUECVETUAS"O" RETURNINGCON
-----
XMLCAST(XMLQUERY(' $O/PH
-----
      4 MARTIN
<SourceFiche>http://fr.wikipedia.org/wiki/Aristote</SourceFiche>
http://fr.wikipedia.org/wiki/Aristote
http://fr.wikipedia.org/wiki/Aristote
http://fr.wikipedia.org
      7 LEROI
<SourceFiche>http://fr.wikipedia.org/wiki/Averroès</SourceFiche>
http://fr.wikipedia.org/wiki/Averroès
http://fr.wikipedia.org/wiki/Averroès
http://fr.wikipedia.org
-- l'année et le lieu de naissance du curriculum vitæ des étudiants
SELECT IdPersonne , NomPersonne , N.*
FROM Etudiants E , TABLE ( XMLSEQUENCE(EXTRACT(E.CVetu, '/Philosophe/Naissance')) ) N
WHERE IdPersonne IN ( 4 , 7 ) ;
IDPERSONNE NOMPERSONNE COLUMN_VALUE
-----
      4 MARTIN      <Naissance>
                    <AnneeNaissance>-384</AnneeNaissance>
                    <LieuNaissance>Stagire</LieuNaissance>
                    </Naissance>
      7 LEROI      <Naissance>
                    <AnneeNaissance>1126</AnneeNaissance>
                    <LieuNaissance>Cordoue</LieuNaissance>
                    </Naissance>
-- les intérêts (philosophiques) du curriculum vitæ des étudiants
SELECT IdPersonne , NomPersonne , I.*
FROM Etudiants E ,
TABLE ( XMLSEQUENCE(EXTRACT(E.CVetu, '/Philosophe/Interets/Interet')) ) I
WHERE IdPersonne IN ( 4 , 7 ) ;
IDPERSONNE NOMPERSONNE COLUMN_VALUE
-----
      4 MARTIN      <Interet>Physique</Interet>
      4 MARTIN      <Interet>Métaphysique</Interet>
      4 MARTIN      <Interet>Biologie</Interet>
      4 MARTIN      <Interet>Éthique</Interet>
      4 MARTIN      <Interet>Politique</Interet>
      4 MARTIN      <Interet>Langage</Interet>
      4 MARTIN      <Interet>Logique</Interet>
      4 MARTIN      <Interet>Poétique</Interet>
      4 MARTIN      <Interet>Rhétorique</Interet>
      7 LEROI      <Interet>Métaphysique</Interet>
      7 LEROI      <Interet>Théologie</Interet>
      7 LEROI      <Interet>Droit</Interet>
      7 LEROI      <Interet>Médecine</Interet>
      7 LEROI      <Interet>Politique</Interet>
      7 LEROI      <Interet>Religion</Interet>
SELECT IdPersonne , NomPersonne , EXTRACTVALUE(CV.COLUMN_VALUE, '/Interet') Interet

```

```

FROM Etudiants , XMLTABLE('/Philosophe/Interets/Interet' PASSING CVetu) CV
WHERE IdPersonne IN ( 4 , 7 )
ORDER BY IdPersonne , Interet ;
IDPERSONNE NOMPERSONNE Interet
-----
4 MARTIN Biologie
4 MARTIN Éthique
4 MARTIN Langage
4 MARTIN Logique
4 MARTIN Métaphysique
4 MARTIN Physique
4 MARTIN Poétique
4 MARTIN Politique
4 MARTIN Rhétorique
7 LEROI Droit
7 LEROI Médecine
7 LEROI Métaphysique
7 LEROI Politique
7 LEROI Religion
7 LEROI Théologie
-- le nombre d'intérêts (philosophiques) du curriculum vitæ des étudiants
SELECT IdPersonne , NomPersonne , COUNT(EXTRACTVALUE(CV.COLUMN_VALUE,'/Interet'))
FROM Etudiants , XMLTABLE('/Philosophe/Interets/Interet' PASSING CVetu) CV
GROUP BY IdPersonne , NomPersonne ;
IDPERSONNE NOMPERSONNE COUNT(EXTRACTVALUE(CV.COLUMN_VALUE,'/INTERET'))
-----
5 DURAND 3
4 MARTIN 9
3 DUPOND 7
2 LEROI 1
7 LEROI 6
-- intérêts curriculum vitæ des étudiants avec nombre étudiants qui s'y intéressent
SELECT EXTRACTVALUE(CV.COLUMN_VALUE,'/Interet') Interet , COUNT(*)
FROM Etudiants , XMLTABLE('/Philosophe/Interets/Interet' PASSING CVetu) CV
GROUP BY EXTRACTVALUE(CV.COLUMN_VALUE,'/Interet')
ORDER BY Interet ;
Interet COUNT(*)
-----
Biologie 1
Droit 1
Éthique 4
Eudémonisme 1
Langage 2
Logique 1
Médecine 1
Métaphysique 3
Physique 2
Poétique 1
Politique 3
Psychologie 1
Religion 1
Rhétorique 1
Sophistique 1
Théologie 1
Théorie de la connaissance 1
-- les intérêts (philosophiques) du curriculum vitæ des étudiants
SELECT E.IdPersonne , E.NomPersonne ,
CAST( MULTISET( SELECT EXTRACTVALUE(CV.COLUMN_VALUE,'/Interet')
FROM Etudiants ECV ,
XMLTABLE('/Philosophe/Interets/Interet' PASSING CVetu) CV
WHERE ECV.IdPersonne = E.IdPersonne )
AS Type_VARRAY_VARCHAR )
FROM Etudiants E
WHERE IdPersonne IN ( 4 , 7 ) ;
IDPERSONNE NOMPERSONNE CAST(MULTISET(SELECT EXTRACTVALUE(CV.COLUMN_VALUE,'/INTER

```

```

-----
      4 MARTIN      TYPE_VARRAY_VARCHAR('Physique', 'Métaphysique',
                    'Biologie', 'Éthique', 'Politique', 'Langage',
                    'Logique', 'Poétique', 'Rhétorique')
      7 LEROI      TYPE_VARRAY_VARCHAR('Métaphysique', 'Théologie',
                    'Droit', 'Médecine', 'Politique', 'Religion')
-- les sources d'information et le lieu de naissance du curriculum vitæ des étudiants
SELECT IdPersonne , NomPersonne , CV.SourceFiche , N.LieuNaissance
FROM Etudiants ,
XMLTABLE('/Philosophe' PASSING CVetu
         COLUMNS SourceFiche VARCHAR2(40) PATH 'SourceFiche' , -- URITYPE
                  Naissance XMLTYPE PATH 'Naissance' ) CV ,
XMLTABLE('/Naissance' PASSING CV.Naissance
         COLUMNS LieuNaissance VARCHAR2(10) PATH 'LieuNaissance' ) N ;
IDPERSONNE NOMPERSOENNE SOURCEFICHE                                LIEUNAISSANCE
-----
      4 MARTIN      http://fr.wikipedia.org/wiki/Aristote Stagire
      2 LEROI      http://fr.wikipedia.org/wiki/Socrate
      7 LEROI      http://fr.wikipedia.org/wiki/Averroès Cordoue
      5 DURAND     http://fr.wikipedia.org/wiki/Épicure Athènes
      3 DUPOND     http://fr.wikipedia.org/wiki/Platon Athènes
-- le nombre d'étudiants qui s'intéressent à l'éthique selon leur curriculum vitæ
SELECT COUNT(*)
FROM Etudiants
WHERE XMLEXISTS('$o/Philosophe/Interets/Interet[text()="Éthique"]'
              PASSING BY VALUE CVetu AS "o") ;
COUNT(*)
-----
      4
-- étudiants n'ayant qu'un seul intérêt (philosophique) selon leur curriculum vitæ
SELECT IdPersonne , NomPersonne
FROM Etudiants
WHERE EXISTSNODE(CVetu, '/Philosophe/Interets/Interet[2]') = 0 ;
IDPERSONNE NOMPERSOENNE
-----
      2 LEROI
-- idées du curriculum vitæ étudiants influencés par et ayant eu influence sur autres
SELECT IdPersonne , NomPersonne , EXTRACTVALUE(CV_Idees.COLUMN_VALUE, '/Idee')
FROM Etudiants , XMLTABLE('/Philosophe/Idees/Idee' PASSING CVetu) CV_Idees
WHERE EXISTSNODE(CVetu, '/Philosophe/Idees/Idee') = 1 AND
      EXISTSNODE(CVetu, '/Philosophe/InfluencesPar/InfluencePar') = 1 AND
      EXISTSNODE(CVetu, '/Philosophe/InfluencesSur/InfluenceSur') = 1 ;
IDPERSONNE NOMPERSOENNE EXTRACTVALUE(CV_IDEES.COLUMN_VALUE, '/IDEE')
-----
      4 MARTIN      Syllogisme
      4 MARTIN      Puissance/Acte
      4 MARTIN      Matière/Forme
      4 MARTIN      Substance/Accident
      4 MARTIN      Catégorie
      2 LEROI      Maïeutique
      2 LEROI      Ironie socratique
-- qui influence et a été influencé par qui
SELECT EXTRACTVALUE(CV_IP.COLUMN_VALUE, '/InfluencePar') AS Influence_De ,
      SUBSTR(EXTRACTVALUE(CVetu, '/Philosophe/SourceFiche'),
            LENGTH('http://fr.wikipedia.org/wiki/')+1) AS Influence_Vers
FROM Etudiants ,
XMLTABLE('/Philosophe/InfluencesPar/InfluencePar' PASSING CVetu) CV_IP
UNION
SELECT SUBSTR(EXTRACTVALUE(CVetu, '/Philosophe/SourceFiche'),
            LENGTH('http://fr.wikipedia.org/wiki/')+1) ,
      EXTRACTVALUE(CV_IP.COLUMN_VALUE, '/InfluenceSur')
FROM Etudiants ,
XMLTABLE('/Philosophe/InfluencesSur/InfluenceSur' PASSING CVetu) CV_IP
ORDER BY Influence_De , Influence_Vers ;
Influence_De              Influence_Vers

```



```

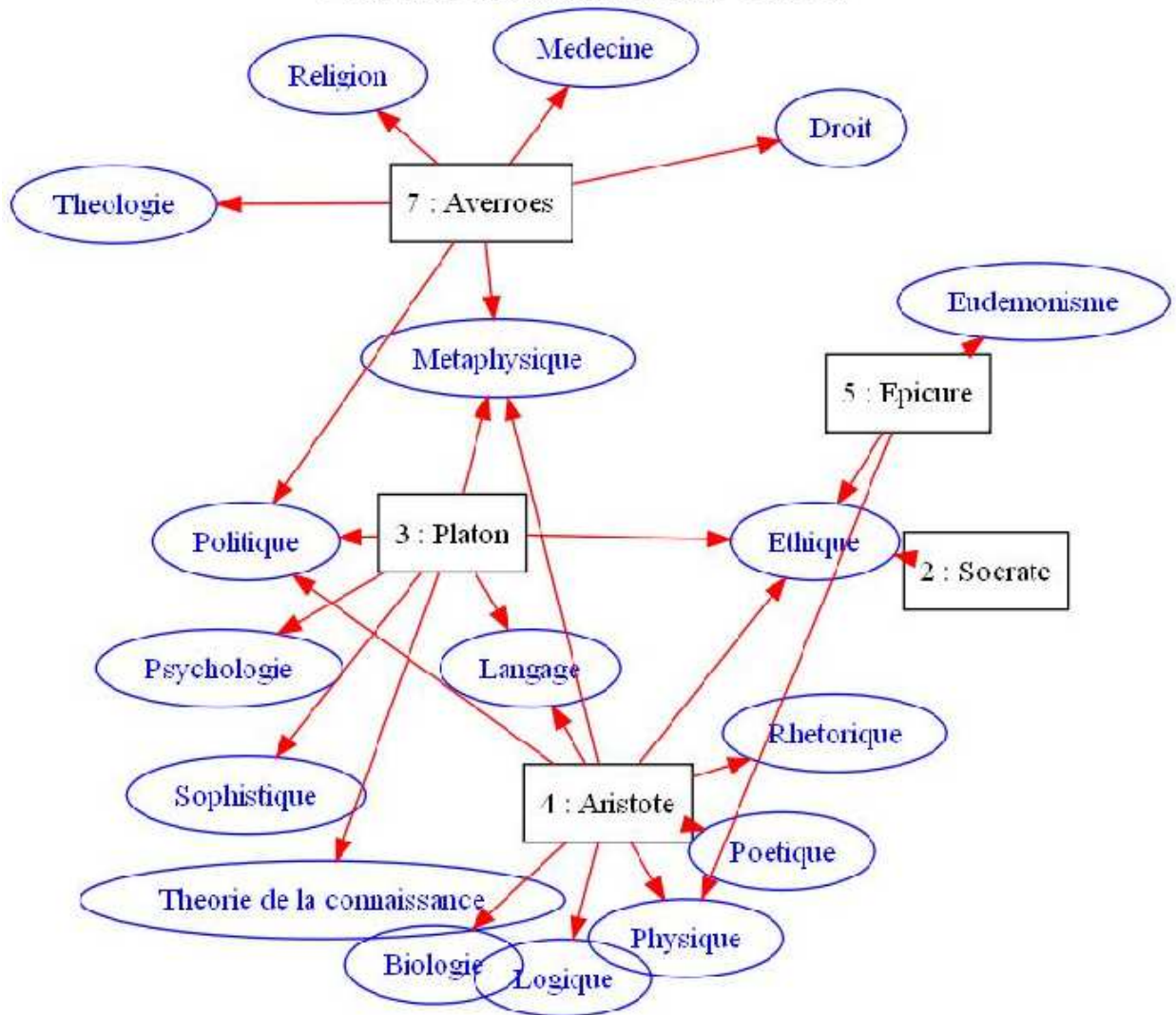
                                "' -> '||c.Interet||'" ;') ;
END LOOP ;
DBMS_OUTPUT.PUT_LINE('}') ;
END ;
digraph G {
  graph [labelloc="t", label="Liens entre etudiants philosophes et interets"] ;
  node [color=blue, fontcolor=blue, shape=ellipse] ;
  edge [color=red] ;
  "2 : Socrate" [color=black, fontcolor=black, shape=box] ;
  "2 : Socrate" -> "Ethique" ;
  "3 : Platon" [color=black, fontcolor=black, shape=box] ;
  "3 : Platon" -> "Ethique" ;
  "3 : Platon" -> "Langage" ;
  "3 : Platon" -> "Metaphysique" ;
  "3 : Platon" -> "Politique" ;
  "3 : Platon" -> "Psychologie" ;
  "3 : Platon" -> "Sophistique" ;
  "3 : Platon" -> "Theorie de la connaissance" ;
  "4 : Aristote" [color=black, fontcolor=black, shape=box] ;
  "4 : Aristote" -> "Biologie" ;
  "4 : Aristote" -> "Ethique" ;
  "4 : Aristote" -> "Langage" ;
  "4 : Aristote" -> "Logique" ;
  "4 : Aristote" -> "Metaphysique" ;
  "4 : Aristote" -> "Physique" ;
  "4 : Aristote" -> "Poetique" ;
  "4 : Aristote" -> "Politique" ;
  "4 : Aristote" -> "Rhetorique" ;
  "5 : Epicure" [color=black, fontcolor=black, shape=box] ;
  "5 : Epicure" -> "Ethique" ;
  "5 : Epicure" -> "Eudemonisme" ;
  "5 : Epicure" -> "Physique" ;
  "7 : Averroes" [color=black, fontcolor=black, shape=box] ;
  "7 : Averroes" -> "Droit" ;
  "7 : Averroes" -> "Medecine" ;
  "7 : Averroes" -> "Metaphysique" ;
  "7 : Averroes" -> "Politique" ;
  "7 : Averroes" -> "Religion" ;
  "7 : Averroes" -> "Theologie" ;
}

```




Voici le résultat de Graphviz obtenu avec l'algorithme *twopi* (les sommets sont placés sur des cercles concentriques en fonction de leur distance à une racine donnée) :

Liens entre étudiants philosophes et interets



Mises à jour de données XML (c.-à-d. contenues dans un XMLTYPE)

```

-- modification du lieu de naissance du curriculum vitæ d'un étudiant
UPDATE Etudiants
  SET CVetu = UPDATEXML(CVetu, '/Philosophe/Naissance/LieuNaissance/text()',
                        'Gradignan')
WHERE IdPersonne = 4 ;
-----
  4 MARTIN      Gradignan

-- annulation
ROLLBACK ;
-- ajout d'un intérêt (philosophique) au curriculum vitæ d'un étudiant
UPDATE Etudiants
  SET CVetu = APPENDCHILDXML(CVetu, '/Philosophe/Interets',
                             XMLTYPE('<Interet>Pédagogie</Interet>'))
WHERE IdPersonne = 4 ;
-----
  4 MARTIN      <Interet>Physique</Interet>
                <Interet>Métaphysique</Interet>
                <Interet>Biologie</Interet>
  
```

```

        <Interet>Éthique</Interet>
        <Interet>Politique</Interet>
        <Interet>Langage</Interet>
        <Interet>Logique</Interet>
        <Interet>Poétique</Interet>
        <Interet>Rhétorique</Interet>
        <Interet>Pédagogie</Interet>
-- suppression cinquième des intérêts (philosophiques) curriculum vitæ d'un étudiant
UPDATE Etudiants
SET CVetu = DELETXML(CVetu, '/Philosophe/Interets/Interet[5]')
WHERE IdPersonne = 4 ;
IDPERSONNE NOMPERSONNE EXTRACT(E.CVETU, '/PHILOSOPHE/INTERETS/INTERET')
-----
4 MARTIN      <Interet>Physique</Interet>
               <Interet>Métaphysique</Interet>
               <Interet>Biologie</Interet>
               <Interet>Éthique</Interet>
               <Interet>Langage</Interet>
               <Interet>Logique</Interet>
               <Interet>Poétique</Interet>
               <Interet>Rhétorique</Interet>
               <Interet>Pédagogie</Interet>
-- insertion en cinquième dans intérêts (philosophiques) curriculum vitæ un étudiant
UPDATE Etudiants
SET CVetu = INSERTCHILDXMLBEFORE(CVetu, '/Philosophe/Interets', 'Interet[5]',
                                   XMLTYPE('<Interet>Politique</Interet>'))
WHERE IdPersonne = 4 ;
IDPERSONNE NOMPERSONNE EXTRACT(E.CVETU, '/PHILOSOPHE/INTERETS/INTERET')
-----
4 MARTIN      <Interet>Physique</Interet>
               <Interet>Métaphysique</Interet>
               <Interet>Biologie</Interet>
               <Interet>Éthique</Interet>
               <Interet>Politique</Interet>
               <Interet>Langage</Interet>
               <Interet>Logique</Interet>
               <Interet>Poétique</Interet>
               <Interet>Rhétorique</Interet>
               <Interet>Pédagogie</Interet>

-- annulation
ROLLBACK ;

```

Oracle Spatial

Informations sur un index spatial (et plus précisément de type R-Tree)

```

-- informations sur l'index via USER_SDO_INDEX_INFO
SELECT TABLE_OWNER , TABLE_NAME , COLUMN_NAME ,
       SDO_INDEX_TYPE , SDO_INDEX_TABLE , SDO_INDEX_STATUS
       -- c.-à-d. * sauf INDEX_NAME
FROM USER_SDO_INDEX_INFO
WHERE INDEX_NAME = 'INDEX_RTREE_FIGGEOMETU' ;
TABLE_OWNER TABLE_NAME COLUMN_NAME SDO_INDEX_TYPE SDO_INDEX_TABLE SDO_INDEX_STATUS
-----
GUIBERT      ETUDIANTS_GEO FIGGEOMETU RTREE MDRT_119C7$ VALID
-- quelques informations sur l'index (de type R-Tree) via USER_SDO_INDEX_METADATA
SELECT SDO_INDEX_OWNER INDEX_OWNER , SDO_TSNAME TSNAME ,
       SDO_COLUMN_NAME COLUMN_NAME , SDO_INDEX_TYPE INDEX_TYPE ,
       SDO_INDEX_TABLE INDEX_TABLE , SDO_INDEX_STATUS STATUS
FROM USER_SDO_INDEX_METADATA
WHERE SDO_INDEX_NAME = 'INDEX_RTREE_FIGGEOMETU' ;
INDEX_OWNER TSNAME COLUMN_NAME INDEX_TYPE INDEX_TABLE STATUS
-----
GUIBERT      GUIBERT "FIGGEOMETU" RTREE MDRT_119C7$ VALID
SELECT SDO_RTREE_HEIGHT HEIGHT , SDO_RTREE_NUM_NODES NUM_NODES ,

```

```

SDO_RTREE_DIMENSIONALITY DIMENSIONALITY , SDO_RTREE_FANOUT FANOUT ,
SDO_RTREE_ROOT ROOT , SDO_RTREE_SEQ_NAME SEQ_NAME , SDO_INDEX_DIMS DIMS ,
SDO_RTREE_QUALITY QUALITY
FROM USER_SDO_INDEX_METADATA
WHERE SDO_INDEX_NAME = 'INDEX_RTREE_FIGGEOMETU' ;
HEIGHT NUM_NODES DIMENSIONALITY FANOUT ROOT SEQ_NAME DIMS QUALITY
-----
2 1 2 34 AAARnIAAEAAAANPAAD MDRS_119C7$ 2
-- nombre de mégaoctets nécessaires à l'index
SELECT SDO_TUNE.ESTIMATE_RTREE_INDEX_SIZE('GUIBERT','ETUDIANTS_GEO','FIGGEOMETU')
FROM DUAL ;
SDO_TUNE.ESTIMATE_RTREE_INDEX_SIZE('GUIBERT','ETUDIANTS_GEO','FIGGEOMETU')
-----
1
-- nombre de mégaoctets estimés nécessaires à l'index pour 100000 éléments
-- géométriques, des blocs de 2048 octets, 10% d'espace libre minimum (dans chaque
-- bloc), 2 dimensions, cas non géodésique
SELECT SDO_TUNE.ESTIMATE_RTREE_INDEX_SIZE(100000,2048,10,2,0) FROM DUAL ;
SDO_TUNE.ESTIMATE_RTREE_INDEX_SIZE(100000,2048,10,2,0)
-----
9
-- mesure la qualité ou la qualité moyenne de la dégradation de l'index
SELECT SDO_TUNE.QUALITY_DEGRADATION('GUIBERT','INDEX_RTREE_FIGGEOMETU') FROM DUAL ;
SDO_TUNE.QUALITY_DEGRADATION('GUIBERT','INDEX_RTREE_FIGGEOMETU')
-----
-2.492228
-- plus petit rectangle englobant tous éléments selon l'index (de type R-Tree) via
-- USER_SDO_INDEX_METADATA
SELECT SDO_ROOT_MBR
FROM USER_SDO_INDEX_METADATA
WHERE SDO_INDEX_NAME = 'INDEX_RTREE_FIGGEOMETU' ;
SDO_ROOT_MBR(SDO_GTYPE,SDO_SRID,SDO_POINT(X,Y,Z),SDO_ELEM_INFO,SDO_ORDINATES)
-----
SDO_GEOMETRY(2003,NULL,NULL,SDO_ELEM_INFO_ARRAY(1,1003,3),SDO_ORDINATE_ARRAY(1,1,19,9))
-- plus petit rectangle englobant tous les éléments géométriques
SELECT SDO_TUNE.EXTENT_OF('ETUDIANTS_GEO','FIGGEOMETU') FROM DUAL ;
SDO_TUNE.EXTENT_OF('ETUDIANTS_GEO','FIGGEOMETU')(SDO_GTYPE,SDO_SRID,SDO_POINT(X,Y,Z),
-----
SDO_GEOMETRY(2003,NULL,NULL,SDO_ELEM_INFO_ARRAY(1,1003,3),SDO_ORDINATE_ARRAY(1,1,19,9))
-- statistiques sur les valeurs prises par l'attribut MDSYS.SDO_GEOMETRY
CALL SDO_TUNE.MIX_INFO('ETUDIANTS_GEO','FIGGEOMETU') ;
Total number of geometries: 5
Point geometries: 0 (0%)
Curvestring geometries: 2 (40%)
Polygon geometries: 3 (60%)
Complex geometries: 0 (0%)
-- largeur et hauteur du plus petit rectangle moyen de Etudiants_Geo.FigGeomEtu
DECLARE
c VARCHAR2(32) := 'ETUDIANTS_GEO' ; -- nom table contenant attribut SDO_GEOMETRY
a VARCHAR2(32) := 'FIGGEOMETU' ; -- nom de l'attribut MDSYS.SDO_GEOMETRY
l NUMBER ; -- largeur du plus petit rectangle moyen
h NUMBER ; -- hauteur du plus petit rectangle moyen
BEGIN
SDO_TUNE.AVERAGE_MBR(c,a,l,h) ;
DBMS_OUTPUT.PUT_LINE('Le plus petit rectangle moyen de '||c||'.'||a||
' a une largeur de '||l||' et une hauteur de '||h);
END ;
Le plus petit rectangle moyen de ETUDIANTS_GEO.FIGGEOMETU a une largeur de 4.1 et une
hauteur de 3.6

```

Requêtes spatiales

```

-- figures géométriques compatibles avec contraintes définies sur attribut (v. 1) ?
SELECT IdPersonne ID ,
SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT(FigGeomEtu,M.DIMINFO) Valide
FROM Etudiants_Geo , USER_SDO_GEOM_METADATA M
WHERE M.TABLE_NAME = 'ETUDIANTS_GEO' AND M.COLUMN_NAME = 'FIGGEOMETU'

```

```

ORDER BY IdPersonne ;
-- figures géométriques compatibles avec contraintes définies sur attribut (v. 2) ?
SELECT IdPersonne ID ,
       SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT(FigGeomEtu,
       (SELECT DIMINFO
        FROM USER_SDO_GEOM_METADATA
        WHERE TABLE_NAME = 'ETUDIANTS_GEO' AND COLUMN_NAME = 'FIGGEOMETU'))
       Valide
FROM Etudiants_Geo
ORDER BY IdPersonne ;
      ID VALIDE
-- -----
      2 TRUE
      3 TRUE
      4 TRUE
      5 TRUE
      7 TRUE
-- les dimensions et types des positions géographiques des étudiants
SELECT IdPersonne ID , EG_PG.PosGeogEtu.GET_DIMS() DIMS ,
       EG_PG.PosGeogEtu.GET_GTYPE() GTYPE , EG_PG.PosGeogEtu Géo
FROM Etudiants_Geo EG_PG
ORDER BY ID ;
      ID DIMS GTYPE GÉO(SDO_GTYPE,SDO_SRID,SDO_POINT(X,Y,Z),SDO_ELEM_INFO,SDO_ORDINATES)
-----
      2      3      1 SDO_GEOMETRY(3001,8307,SDO_POINT_TYPE(47.97435,5.633539,409),NULL,NULL)
      3      3      1 SDO_GEOMETRY(3001,8307,SDO_POINT_TYPE(47.486183,4.717461,446),NULL,NULL)
      4      3      1 SDO_GEOMETRY(3001,8307,SDO_POINT_TYPE(44.843889,4.22,1408),NULL,NULL)
      5      3      1 SDO_GEOMETRY(3001,8307,SDO_POINT_TYPE(46.6325,8.672222,2346),NULL,NULL)
      7      3      1 SDO_GEOMETRY(3001,8307,SDO_POINT_TYPE(46.602,8.376167,2250),NULL,NULL)
-- les dimensions et types des figures géométriques des étudiants
SELECT IdPersonne , EG_FG.FigGeomEtu.GET_DIMS() ,
       EG_FG.FigGeomEtu.GET_GTYPE() T , EG_FG.FigGeomEtu
FROM Etudiants_Geo EG_FG
ORDER BY 1 ;
      ID DIMS T FIGGEOMETU(SDO_GTYPE,SDO_SRID,SDO_POINT(X,Y,Z),SDO_ELEM_INFO,SDO_ORDINATES)
-----
      2      2      2 SDO_GEOMETRY(2002,NULL,NULL,SDO_ELEM_INFO_ARRAY(1,2,1),
                                SDO_ORDINATE_ARRAY(9,6,9,1,9.5,1,9.5,1.5,9,1.5))
      3      2      2 SDO_GEOMETRY(2002,NULL,NULL,SDO_ELEM_INFO_ARRAY(1,2,1),
                                SDO_ORDINATE_ARRAY(17,1,17,2))
      4      2      3 SDO_GEOMETRY(2003,NULL,NULL,SDO_ELEM_INFO_ARRAY(1,1003,1),
                                SDO_ORDINATE_ARRAY(5,1,13,1,9,6,5,1))
      5      2      3 SDO_GEOMETRY(2003,NULL,NULL,SDO_ELEM_INFO_ARRAY(1,1003,4),
                                SDO_ORDINATE_ARRAY(3,8,2,9,1,8))
      7      2      3 SDO_GEOMETRY(2003,NULL,NULL,SDO_ELEM_INFO_ARRAY(1,1003,1),
                                SDO_ORDINATE_ARRAY(9,6,13,1,19,1,9,6))
      5 ligne(s) sélectionnée(s).
-- SRID et positions géographiques (latitude, longitude et altitude) des étudiants
SELECT IdPersonne ID , NomPersonne Nom , EG.PosGeogEtu.SDO_SRID SRID ,
       EG.PosGeogEtu.SDO_POINT.X Latitude , EG.PosGeogEtu.SDO_POINT.Y Longitude ,
       EG.PosGeogEtu.SDO_POINT.Z Altitude
FROM Etudiants_Geo EG
NATURAL JOIN Etudiants
ORDER BY IdPersonne ;
      ID NOM      SRID  LATITUDE  LONGITUDE  ALTITUDE
-- -----
      2 LEROI  8307  47.97435  5.633539   409
      3 DUPOND 8307  47.486183 4.717461   446
      4 MARTIN 8307  44.843889 4.22       1408
      5 DURAND 8307  46.6325   8.672222  2346
      7 LEROI  8307  46.602    8.376167  2250
-- deux systèmes géodésiques (mondial et français) :
-- WGS 84 (World Geodetic System 1984) = système géodésique mondial associé au GPS
-- RGF93 (Réseau Géodésique Français 1993) = système géodésique officiel en France
SELECT SRID , CS_NAME , WKTEXT
FROM CS_SRS
WHERE SRID IN ( 8307 , 2154 )

```

```

ORDER BY SRID DESC ;
SRID CS_NAME WKTEXT
-----
8307 Longitude / Latitude (WGS 84) GEOGCS [ "Longitude / Latitude (WGS 84)", DATUM
["WGS 84", SPHEROID ["WGS 84", 6378137,
298.257223563]], PRIMEM [ "Greenwich", 0.000000 ],
UNIT ["Decimal Degree", 0.01745329251994330]]
2154 RGF93 / Lambert-93 PROJCS["RGF93 / Lambert-93", GEOGCS [ "RGF93", DATUM
["Reseau Geodesique Francais 1993 (EPSG ID 6171)",
SPHEROID ["GRS 1980 (EPSG ID 7019)", 6378137,
298.257222101]], PRIMEM [ "Greenwich", 0.000000 ],
UNIT ["Decimal Degree", 0.01745329251994328]],
PROJECTION ["Lambert Conformal Conic"], PARAMETER
["Latitude_Of_Origin", 46.5], PARAMETER
["Central_Meridian", 3], PARAMETER
["Standard_Parallel_1", 49], PARAMETER
["Standard_Parallel_2", 44], PARAMETER
["False_Easting", 700000], PARAMETER
["False_Northing", 6600000], UNIT ["Meter", 1]]

2 ligne(s) sélectionnée(s).
-- positions géographiques étudiants de n° 4 ou 2 ou 3 dans référence spatiale 2154
SELECT IdPersonne ID , NomPersonne Nom , SDO_CS.TRANSFORM(EG.PosGeogEtu,2154)
FROM Etudiants_Geo EG
NATURAL JOIN Etudiants
WHERE IdPersonne IN ( 4 , 2 , 3 )
ORDER BY IdPersonne ;
ID NOM SDO_CS.TRANSFORM(EG.POSGEOGETU,2154)(SDO_GTYPE, SDO_SRID, SDO_POINT(X, Y, Z)
-----
2 LEROI SDO_GEOMETRY(3001,2154,SDO_POINT_TYPE(6604339.06,3435220.54,409),NULL,NULL)
3 DUPOND SDO_GEOMETRY(3001,2154,SDO_POINT_TYPE(6615279.01,3291156.64,446),NULL,NULL)
4 MARTIN SDO_GEOMETRY(3001,2154,SDO_POINT_TYPE(6333920.62,3038314.92,1408),NULL,NULL)
-- transforma° posi° géographique dpt Info. IUT Bx 1 référence spatiale 8307 en 2154
SELECT SDO_CS.TRANSFORM(MDSYS.SDO_GEOMETRY(3001,8307,
MDSYS.SDO_POINT_TYPE(44.79115,-0.6087,33),
NULL,NULL),2154)
FROM DUAL ;
SDO_CS.TRANSFORM(MDSYS.SDO_GEOMETRY(3001,8307,MDSYS.SDO_POINT_TYPE(44.79115,-0.6087,33)
-----
SDO_GEOMETRY(3001,2154,SDO_POINT_TYPE(6680211.24,2431533.92,33),NULL,NULL)
-- aires, centres gravité, longueurs/périmètres figures géométriques tolérance attr.
SELECT IdPersonne ID , SDO_GEOM.SDO_AREA(FigGeomEtu,M.DIMINFO) Aire ,
SDO_GEOM.SDO_CENTROID(FigGeomEtu,M.DIMINFO) CentreGrav ,
SDO_GEOM.SDO_LENGTH(FigGeomEtu,M.DIMINFO) Lg_OU_Périm
FROM Etudiants_Geo , USER_SDO_GEOM_METADATA M
WHERE M.TABLE_NAME = 'ETUDIANTS_GEO' AND M.COLUMN_NAME = 'FIGGEOMETU'
ORDER BY IdPersonne ;
ID AIRE CENTREGRAV(SDO_GTYPE,SDO_SRID,SDO_POINT(X,Y,Z),SDO_ELEM_INFO, LG_OU_PÉRIM
-----
2 0 6.5
3 0 1
4 20 SDO_GEOMETRY(2001,NULL,SDO_POINT_TYPE(9,2.66666667,
NULL),NULL,NULL) 20.8062485
5 3.14159265 SDO_GEOMETRY(2001,NULL,SDO_POINT_TYPE(2,8,
NULL),NULL,NULL) 6.28318531
7 15 SDO_GEOMETRY(2001,NULL,SDO_POINT_TYPE(13.66666667,2.66666667,
NULL),NULL,NULL) 23.5834641

5 ligne(s) sélectionnée(s).
-- rectangles englobants des figures géométriques des étudiants
SELECT IdPersonne ID , FigGeomEtu , SDO_GEOM.SDO_MBR(FigGeomEtu)
FROM Etudiants_Geo
ORDER BY IdPersonne ;
ID SDO_GEOM.SDO_MBR(FIGGEOMETU)(SDO_GTYPE,SDO_SRID,SDO_POINT(X,Y,Z),SDO_ELEM_INFO,SDO_O
-----
2 SDO_GEOMETRY(2003,NULL,NULL,SDO_ELEM_INFO_ARRAY(1,1003,3),
SDO_ORDINATE_ARRAY(9,1,9.5,6))
3 SDO_GEOMETRY(2002,NULL,NULL,SDO_ELEM_INFO_ARRAY(1,2,1),
SDO_ORDINATE_ARRAY(17,1,17,2))
4 SDO_GEOMETRY(2003,NULL,NULL,SDO_ELEM_INFO_ARRAY(1,1003,3),
SDO_ORDINATE_ARRAY(5,1,13,6))
5 SDO_GEOMETRY(2003,NULL,NULL,SDO_ELEM_INFO_ARRAY(1,1003,3),

```

```

                SDO_ORDINATE_ARRAY(1,7,3,9))
7 SDO_GEOMETRY(2003,NULL,NULL,SDO_ELEM_INFO_ARRAY(1,1003,3),
                SDO_ORDINATE_ARRAY(9,1,19,6))
5 ligne(s) sélectionnée(s).
-- enveloppes convexes des figures géométriques étudiants à tolérance sur attribut
SELECT IdPersonne ID , FigGeomEtu , SDO_GEOM.SDO_CONVEXHULL(FigGeomEtu,M.DIMINFO)
FROM Etudiants_Geo , USER_SDO_GEOM_METADATA M
WHERE M.TABLE_NAME = 'ETUDIANTS_GEO' AND M.COLUMN_NAME = 'FIGGEOMETU'
ORDER BY IdPersonne ;
ID SDO_GEOM.SDO_CONVEXHULL(FIGGEOMETU,M.DIMINFO)(SDO_GTYPE,SDO_SRID,SDO_POINT(X,Y,Z),SD
-----
2 SDO_GEOMETRY(2003,NULL,NULL,SDO_ELEM_INFO_ARRAY(1,1003,1),
                SDO_ORDINATE_ARRAY(9.5,1,9.5,1.5,9,6,9,1.5,9,1,9.5,1))
3
4 SDO_GEOMETRY(2003,NULL,NULL,SDO_ELEM_INFO_ARRAY(1,1003,1),
                SDO_ORDINATE_ARRAY(13,1,9,6,5,1,13,1))
5 SDO_GEOMETRY(2003,NULL,NULL,SDO_ELEM_INFO_ARRAY(1,1003,1),
                SDO_ORDINATE_ARRAY(3,7,3,9,2,9,1,9,1,7,3,7))
7 SDO_GEOMETRY(2003,NULL,NULL,SDO_ELEM_INFO_ARRAY(1,1003,1),
                SDO_ORDINATE_ARRAY(19,1,9,6,13,1,19,1))
5 ligne(s) sélectionnée(s).
-- distances (2 à 2) entre toutes figures géométriques étds à tolérance sur attribut
SELECT EG1.IdPersonne ID1 , EG2.IdPersonne ID2 ,
       SDO_GEOM.SDO_DISTANCE(EG1.FigGeomEtu,M.DIMINFO,EG2.FigGeomEtu,M.DIMINFO) Dist
FROM Etudiants_Geo EG1 , Etudiants_Geo EG2 , USER_SDO_GEOM_METADATA M
WHERE EG1.IdPersonne < EG2.IdPersonne AND
      M.TABLE_NAME = 'ETUDIANTS_GEO' AND M.COLUMN_NAME = 'FIGGEOMETU'
ORDER BY EG1.IdPersonne , EG2.IdPersonne ;
ID1 ID2          DIST
--- ---  -----
2 3             7.5
2 4              0
2 5 6.28010989
2 7              0
3 4              4
3 5 15.1554944
3 7              0
4 5 5.71547176
4 7              0
5 7 6.28010989
-- métadonnées sur les figures géométriques des étudiants (collection imbriquée)
SELECT M.DIMINFO
FROM USER_SDO_GEOM_METADATA M
WHERE M.TABLE_NAME = 'ETUDIANTS_GEO' AND M.COLUMN_NAME = 'FIGGEOMETU' ;
DIMINFO(SDO_DIMNAME, SDO_LB, SDO_UB, SDO_TOLERANCE)
-----
SDO_DIM_ARRAY(SDO_DIM_ELEMENT('abscisse (X)',0,20,,.01),
              SDO_DIM_ELEMENT('ordonnées (Y)',0,10,,.01))
1 ligne sélectionnée.
-- métadonnées sur figures géométriques étudiants (collec° imbriquée en rela° plate)
SELECT M.SDO_DIMNAME , SDO_LB , SDO_UB , SDO_TOLERANCE
FROM TABLE ( SELECT DIMINFO
              FROM USER_SDO_GEOM_METADATA M
              WHERE M.TABLE_NAME = 'ETUDIANTS_GEO' AND
                    M.COLUMN_NAME = 'FIGGEOMETU' ) M ;
SDO_DIMNAME    SDO_LB SDO_UB SDO_TOLERANCE
-----
abscisse (X)      0      20          .01
ordonnées (Y)    0      10          .01
-- coordonnées de l'origine des métadonnées sur les figures géométriques étudiants
SELECT X.SDO_LB X_min , (X.SDO_LB+X.SDO_UB)/2 X_moy , X.SDO_UB X_max ,
       Y.SDO_LB Y_min , (Y.SDO_LB+Y.SDO_UB)/2 Y_moy , Y.SDO_UB Y_max
FROM TABLE ( SELECT DIMINFO
              FROM USER_SDO_GEOM_METADATA M
              WHERE M.TABLE_NAME = 'ETUDIANTS_GEO' AND
                    M.COLUMN_NAME = 'FIGGEOMETU' ) X ,
TABLE ( SELECT DIMINFO

```

```

FROM USER_SDO_GEOM_METADATA M
WHERE M.TABLE_NAME = 'ETUDIANTS_GEO' AND
      M.COLUMN_NAME = 'FIGGEOMETU' ) Y
WHERE X.SDO_DIMNAME = 'abscisse (X)' AND Y.SDO_DIMNAME = 'ordonnées (Y)' ;
X_MIN X_MOY X_MAX Y_MIN Y_MOY Y_MAX
-----
      0      10      20      0      5      10
-- distance plus proche de toutes figures géométriques étds avec (10,5) 100ème près
SELECT IdPersonne ID ,
      SDO_GEOM.SDO_DISTANCE(EG.FigGeomEtu,
      MDSYS.SDO_GEOMETRY(2001,NULL,
      MDSYS.SDO_POINT_TYPE(10,5,NULL),
      NULL,NULL),0.01) Distance_moy

FROM Etudiants_Geo EG
ORDER BY IdPersonne ;
ID DISTANCE_MOY
-- -----
  2          1
  3  7.61577311
  4  0.156173762
  5  7.54400375
  7          0
-- dist. + proche toutes posi°s géographiques étds avec dpt Info. IUT Bx 1 à 1 m près
SELECT IdPersonne ID ,
      TO_CHAR(SDO_GEOM.SDO_DISTANCE(PosGeogEtu,
      MDSYS.SDO_GEOMETRY(3001,8307,MDSYS.SDO_POINT_TYPE(44.79115,-0.6087,33),
      NULL,NULL),1),'99G999G999D99') Distance_IUT

FROM Etudiants_Geo
ORDER BY IdPersonne ;
ID DISTANCE_IUT
-- -----
  2  775 660,09
  3  660 822,50
  4  533 970,72
  5  1 046 431,23
  7  1 013 671,90
-- toutes figures géométriques étds en interaction avec figure géométrique étd n° 4
SELECT IdPersonne ID , NomPersonne NOM
FROM Etudiants_Geo EG
NATURAL JOIN Etudiants E
WHERE SDO_FILTER(FigGeomEtu,(SELECT FigGeomEtu
      FROM Etudiants_Geo
      WHERE IdPersonne = 4),'querytype=WINDOW') = 'TRUE'

ORDER BY IdPersonne ;
ID NOM
-- -----
  2 LEROI
  4 MARTIN
  7 LEROI
-- toutes figures géométriques étds sauf 4 interaction avec figure géométrique étd 4
SELECT EG4.IdPersonne ID4 , E4.NomPersonne NOM4 ,
      EGsauf4.IdPersonne ID , Esauf4.NomPersonne NOM
FROM Etudiants_Geo EG4 JOIN Etudiants E4 ON EG4.IdPersonne = E4.IdPersonne ,
      Etudiants_Geo EGsauf4
      JOIN Etudiants Esauf4 ON EGsauf4.IdPersonne = Esauf4.IdPersonne
WHERE EG4.IdPersonne = 4 AND EGsauf4.IdPersonne <> EG4.IdPersonne AND
      SDO_FILTER(EG4.FigGeomEtu,EGsauf4.FigGeomEtu,'querytype=WINDOW') = 'TRUE'

ORDER BY EGsauf4.IdPersonne ;
ID4 NOM4 ID NOM
-- -----
  4 MARTIN 2 LEROI
  4 MARTIN 7 LEROI
-- toutes figures géométriques étudiants en interaction avec le rectangle [8;9]*[0;7]
SELECT IdPersonne ID , NomPersonne NOM
FROM Etudiants_Geo

```

```

NATURAL JOIN Etudiants
WHERE SDO_FILTER(FigGeomEtu,
                MDSYS.SDO_GEOMETRY(2003,NULL,NULL,
                                   MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,3),
                                   MDSYS.SDO_ORDINATE_ARRAY(8,0,9,7)),
                'querytype=WINDOW') = 'TRUE'
ORDER BY IdPersonne ;
ID NOM
-- -----
2 LEROI
4 MARTIN
7 LEROI
-- les 2 figures géométriques étds les + proches (sans elle-même) de celle étd n° 5
SELECT IdPersonne ID , NomPersonne NOM
FROM Etudiants_Geo
NATURAL JOIN Etudiants
WHERE IdPersonne <> 5 AND
      SDO_NN(FigGeomEtu,(SELECT FigGeomEtu
                        FROM Etudiants_Geo
                        WHERE IdPersonne = 5),'sdo_num_res=3') = 'TRUE'
ORDER BY IdPersonne ;
ID NOM
-- -----
2 LEROI
4 MARTIN
-- figures géométriques, des polygones 2D, sont + proches à distance 0<=d<=5 de (9,7)
DECLARE
d NUMBER ; -- distance
x NUMBER := 9 ; -- abscisse du point de comparaison
y NUMBER := 7 ; -- ordonnée du point de comparaison
c NUMBER ; -- curseur sur les étudiants (figures géométriques)
BEGIN
FOR d IN 0..5 LOOP
DBMS_OUTPUT.PUT_LINE('Étudiants dont figures géométriques, polygones 2D,
                    sont à distance au plus '||TO_CHAR(d)||' de ('||
                    TO_CHAR(x)||','||TO_CHAR(y)||')') ;
FOR c IN ( SELECT EGPolyg2D.IdPersonne , MDSYS.SDO_NN_DISTANCE(382) Distance
          -- 382 arbitraire mais identique SDO_NN_DISTANCE() et SDO_NN()
          FROM Etudiants_Geo EGPolyg2D
          WHERE EGPolyg2D.FigGeomEtu.GET_GTYPE() = 03 AND -- 03=polygones 2D
                SDO_NN(EGPolyg2D.FigGeomEtu,
                      MDSYS.SDO_GEOMETRY(2001,NULL,
                                          MDSYS.SDO_POINT_TYPE(x,y,NULL),
                                          NULL,NULL),
                      'sdo_num_res='||TO_CHAR(d),382) = 'TRUE'
          ORDER BY IdPersonne ) LOOP
DBMS_OUTPUT.PUT_LINE(' > '||'Étudiant n° '||c.IdPersonne||
                    ' à distance '||c.Distance) ;
END LOOP ;
END LOOP ;
END ;
Étudiants dont figures géométriques, polygones 2D, sont à distance au plus 0 de (9,7)
Étudiants dont figures géométriques, polygones 2D, sont à distance au plus 1 de (9,7)
> Étudiant n° 4 à distance 1
Étudiants dont figures géométriques, polygones 2D, sont à distance au plus 2 de (9,7)
> Étudiant n° 4 à distance 1
> Étudiant n° 7 à distance 1
Étudiants les figures géométriques, polygones 2D, sont à distance au plus 3 de (9,7)
> Étudiant n° 4 à distance 1
> Étudiant n° 7 à distance 1
Étudiants les figures géométriques, polygones 2D, sont à distance au plus 4 de (9,7)
> Étudiant n° 4 à distance 1
> Étudiant n° 5 à distance 6,07106781186548
> Étudiant n° 7 à distance 1
Étudiants les figures géométriques, polygones 2D, sont à distance au plus 5 de (9,7)
> Étudiant n° 4 à distance 1
> Étudiant n° 5 à distance 6,07106781186548

```



```

> Étudiant n° 7 à distance 1
-- les figures géométriques des étudiants qui recouvrent une autre
SELECT EG1.IdPersonne ID , 'recouvre' , EG2.IdPersonne ID
FROM Etudiants_Geo EG1 , Etudiants_Geo EG2
WHERE SDO_RELATE(EG1.FigGeomEtu,EG2.FigGeomEtu,'mask=COVERS') = 'TRUE'
ORDER BY EG1.IdPersonne , EG2.IdPersonne ;
ID 'RECOUVR ID
-- -----
4 recouvre 2
7 recouvre 3
-- les figures géométriques des étudiants sauf n° 5 qui sont disjointes ou accolées
SELECT EG1.IdPersonne ID , 'disjointe ou accolée' , EG2.IdPersonne ID
FROM Etudiants_Geo EG1 , Etudiants_Geo EG2
WHERE EG1.IdPersonne <> 5 AND EG2.IdPersonne <> 5 AND
EG1.IdPersonne < EG2.IdPersonne AND -- pour casser la symétrie
SDO_RELATE(EG1.FigGeomEtu,EG2.FigGeomEtu,'mask=DISJOINT+TOUCH') = 'TRUE'
ORDER BY EG1.IdPersonne , EG2.IdPersonne ;
ID 'DISJOINTEOUACCOLÉE' ID
-- -----
2 disjointe ou accolée 3
2 disjointe ou accolée 7
3 disjointe ou accolée 4
4 disjointe ou accolée 7
-- les figures géométriques des étudiants qui interagissent
SELECT EG1.IdPersonne ID , 'interagit' , EG2.IdPersonne ID
FROM Etudiants_Geo EG1 , Etudiants_Geo EG2
WHERE EG1.IdPersonne < EG2.IdPersonne AND -- pour casser la symétrie
SDO_RELATE(EG1.FigGeomEtu,EG2.FigGeomEtu,
'mask=ANYINTERACT querytype=JOIN') = 'TRUE'
ORDER BY EG1.IdPersonne , EG2.IdPersonne ;
ID 'INTERAGI ID
-- -----
2 interagit 4
2 interagit 7
3 interagit 7
4 interagit 7
-- les figures géométriques des étudiants à distance d'au plus 8.5 de (0,0)
SELECT IdPersonne ID
FROM Etudiants_Geo
WHERE SDO_WITHIN_DISTANCE(FigGeomEtu,
MDSYS.SDO_GEOMETRY(2001,NULL,
MDSYS.SDO_POINT_TYPE(0,0,NULL),
NULL,NULL),
'distance=8.5') = 'TRUE'
ORDER BY IdPersonne ;
ID
--
4
5
-- figure géométrique de la zone entourant à 1.0 celle étd n° 3 avec tolérance 10ème
SELECT EG3engl.FigGeomEtu_engl
FROM ( SELECT SDO_GEOM.SDO_BUFFER(EG3.FigGeomEtu,1.0,0.1) FigGeomEtu_engl
FROM Etudiants_Geo EG3
WHERE EG3.IdPersonne = 3 ) EG3engl ;
FIGGEOMETU(SDO_GTYPE,SDO_SRID,SDO_POINT(X,Y,Z),SDO_ELEM_INFO,SDO_ORDINATES)
-----
SDO_GEOMETRY(2003,NULL,NULL,SDO_ELEM_INFO_ARRAY(1,1005,4, 1,2,1, 3,2,2, 7,2,1, 9,2,2),
SDO_ORDINATE_ARRAY(18,1, 18,2, 17,3, 16,2, 16,1, 17,0, 18,1))
1 ligne sélectionnée.
N.B. : c'est un polygone extérieur (___05) composé de 4 éléments de type lignes (1___) droites ou arcs de cercles : 1,2,1
une ligne droite de (18,1) à (18,2), 3,2,2 une ligne d'arcs de cercles de (18,2) à (17,3) puis à (16,2),
7,2,1 une ligne droite de (16,2) à (16,1) et 9,2,2 une ligne d'arcs de cercles de (16,1) à (17,0) puis à
(18,1)
-- intersection, union, deux différences, différence symétrique, tolérance du 100ème,
-- entre figure géométrique étd 7 et zone entourant à 1.0 celle étd 3 tolérance 10ème
SELECT SDO_GEOM.SDO_INTERSECTION(EG7.FigGeomEtu,EG3_engl.FigGeomEtu_engl,0.01) ,

```

```

SDO_GEOM.SDO_UNION(EG7.FigGeomEtu,EG3_engl.FigGeomEtu_engl,0.01) ,
SDO_GEOM.SDO_DIFFERENCE(EG7.FigGeomEtu,EG3_engl.FigGeomEtu_engl,0.01) ,
SDO_GEOM.SDO_DIFFERENCE(EG3_engl.FigGeomEtu_engl,EG7.FigGeomEtu,0.01) ,
SDO_GEOM.SDO_XOR(EG7.FigGeomEtu,EG3_engl.FigGeomEtu_engl,0.01)
FROM Etudiants_Geo EG7 ,
( SELECT SDO_GEOM.SDO_BUFFER(EG3.FigGeomEtu,1.0,0.1) FigGeomEtu_engl
FROM Etudiants_Geo EG3
WHERE EG3.IdPersonne = 3 ) EG3_engl
WHERE EG7.IdPersonne = 7 ;
SDO_GEOM.SDO_INTERSECTION(EG7.FIGGEOMETU,EG3_ENGL.FIGGEOMETU_ENGL,0.01)(SDO_GTYPE,SDO_S
SDO_GEOM.SDO_UNION(EG7.FIGGEOMETU,EG3_ENGL.FIGGEOMETU_ENGL,0.01)(SDO_GTYPE,SDO_SRID,SDO
SDO_GEOM.SDO_DIFFERENCE(EG7.FIGGEOMETU,EG3_ENGL.FIGGEOMETU_ENGL,0.01)(SDO_GTYPE,SDO_SRI
SDO_GEOM.SDO_DIFFERENCE(EG3_ENGL.FIGGEOMETU_ENGL,EG7.FIGGEOMETU,0.01)(SDO_GTYPE,SDO_SRI
SDO_GEOM.SDO_XOR(EG7.FIGGEOMETU,EG3_ENGL.FIGGEOMETU_ENGL,0.01)(SDO_GTYPE,SDO_SRID,SDO_P
-----
SDO_GEOMETRY(2003,NULL,NULL,SDO_ELEM_INFO_ARRAY(1,1005,2,1,2,1,9,2,2),
SDO_ORDINATE_ARRAY(16,2,16,1,18,1,18,1.5,16.1055728,2.4472136,16.026751,
2.22975292,16,2))
SDO_GEOMETRY(2003,NULL,NULL,
SDO_ELEM_INFO_ARRAY(1,1005,4,1,2,1,7,2,2,11,2,1,17,2,2),
SDO_ORDINATE_ARRAY(18,1,19,1,18,1.5,18,2,17.2297529,2.97324899,16.1055728,
2.4472136,9,6,13,1,16,1,17,0,18,1))
SDO_GEOMETRY(2007,NULL,NULL,
SDO_ELEM_INFO_ARRAY(1,1003,1,9,1005,3,9,2,1,15,2,2,19,2,1),
SDO_ORDINATE_ARRAY(18,1.5,18,1,19,1,18,1.5,9,6,13,1,16,1,16,2,16.026751,
2.22975292,16.1055728,2.4472136,9,6))
SDO_GEOMETRY(2007,NULL,NULL,
SDO_ELEM_INFO_ARRAY(1,1005,2,1,2,1,5,2,2,11,1005,2,11,2,2,15,2,1),
SDO_ORDINATE_ARRAY(16.1055728,2.4472136,18,1.5,18,2,17.2297529,2.97324899,
16.1055728,2.4472136,16,1,17,0,18,1,16,1))
SDO_GEOMETRY(2007,NULL,NULL,
SDO_ELEM_INFO_ARRAY(1,1003,1,9,1005,2,9,2,1,13,2,2,19,1005,2,19,2,2,
23,2,1,27,1005,3,27,2,1,33,2,2,37,2,1),
SDO_ORDINATE_ARRAY(18,1.5,18,1,19,1,18,1.5,16.1055728,2.4472136,18,1.5,18,
2,17.2297529,2.97324899,16.1055728,2.4472136,16,1,17,0,
18,1,16,1,9,6,13,1,16,1,16,2,16.026751,2.22975292,
16.1055728,2.4472136,9,6))
1 ligne sélectionnée.
-- aires, tolérance sur attribut, des
-- intersection, union, deux différences, différence symétrique, tolérance du 100ème,
-- entre figure géométrique étd 7 et zone entourant à 1.0 celle étd 3 tolérance 10ème
SELECT SDO_GEOM.SDO_AREA(SDO_GEOM.SDO_INTERSECTION(EG7.FigGeomEtu,
EG3_engl.FigGeomEtu_engl,0.01),M.DIMINFO) Aire_Intersection ,
SDO_GEOM.SDO_AREA(SDO_GEOM.SDO_UNION(EG7.FigGeomEtu,
EG3_engl.FigGeomEtu_engl,0.01),M.DIMINFO) Aire_Union ,
SDO_GEOM.SDO_AREA(SDO_GEOM.SDO_DIFFERENCE(EG7.FigGeomEtu,
EG3_engl.FigGeomEtu_engl,0.01),M.DIMINFO) Aire_Différence_1 ,
SDO_GEOM.SDO_AREA(SDO_GEOM.SDO_DIFFERENCE(EG3_engl.FigGeomEtu_engl,
EG7.FigGeomEtu,0.01),M.DIMINFO) Aire_Différence_2 ,
SDO_GEOM.SDO_AREA(SDO_GEOM.SDO_XOR(EG7.FigGeomEtu,
EG3_engl.FigGeomEtu_engl,0.01),M.DIMINFO) Aire_Différence_Symétr
FROM Etudiants_Geo EG7 ,
( SELECT SDO_GEOM.SDO_BUFFER(EG3.FigGeomEtu,1.0,0.1) FigGeomEtu_engl
FROM Etudiants_Geo EG3
WHERE EG3.IdPersonne = 3 ) EG3_engl,
USER_SDO_GEOM_METADATA M
WHERE EG7.IdPersonne = 7 AND
M.TABLE_NAME = 'ETUDIANTS_GEO' AND M.COLUMN_NAME = 'FIGGEOMETU' ;
AIRE_INTERSECTION AIRE_UNION AIRE_DIFFÉRENCE_1 AIRE_DIFFÉRENCE_2 AIRE_DIFFÉRENCE_SYMÉTR
-----
1,9818238 18,1597688 13,0181762 3,15976885 16,177945

```

Utilisation de l'application

Voici quelques captures d'écran de pages Web illustrant l'application : la page d'accueil, toutes les informations de tables et les informations sur un étudiant.

Page d'accueil



Exemple "jouet"

Bienvenue dans la gestion de l'exemple jouet (étudiants, voitures, diplômes, départements français)

Sélectionnez l'une des tables (non objet) appartenant à SCOTT ou GUIBERT de la base de données :

- GUIBERT.DEPARTEMENTS
- GUIBERT.DEPARTEMENTS
- GUIBERT.ETUDIANTS_GEO
- GUIBERT.MDRT_119C7\$
- SCOTT.BONUS
- SCOTT.DEPT
- SCOTT.EMP
- SCOTT.SALGRADE

Affichage de toutes les informations contenues dans la table sélectionnée

Cliquez sur l'un des liens ci-dessous pour afficher les informations sur un étudiant :

- [Informations sur l'étudiant n° 2 \(LEROI\)](#)
- [Informations sur l'étudiant n° 3 \(DUPOND\)](#)
- [Informations sur l'étudiant n° 4 \(MARTIN\)](#)
- [Informations sur l'étudiant n° 5 \(DURAND\)](#)
- [Informations sur l'étudiant n° 7 \(LEROI\)](#)

Nous sommes aujourd'hui le 13/07/2011 et il est 17:47:58.

Toutes les informations sur une table

Départements français



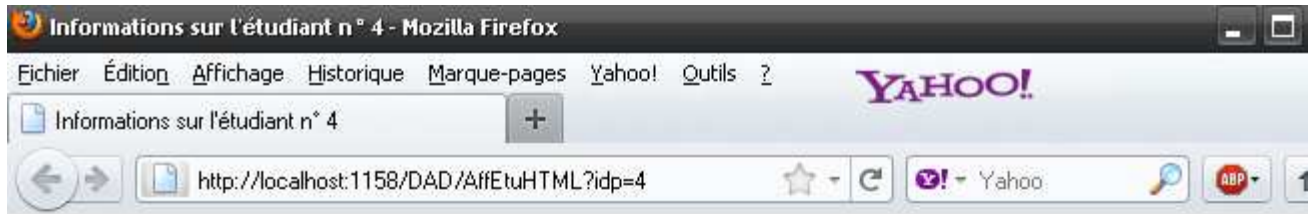
CODEDEPARTEMENT	NOMDEPARTEMENT		
		50	Manche
		51	Marne
01	Ain	52	Haute-Marne
02	Aisne	53	Mayenne
03	Allier	54	Meurthe-et-Moselle
04	Alpes de Haute Provence	55	Meuse
05	Hautes-Alpes	56	Morbihan
06	Alpes-Maritimes	57	Moselle
07	Ardèche	58	Nièvre
08	Ardennes	59	Nord
09	Ariège	60	Oise
10	Aube	61	Orne
11	Aude	62	Pas-de-Calais
12	Aveyron	63	Puy-de-Dôme
13	Bouches-du-Rhône	64	Pyrénées-Atlantiques
14	Calvados	65	Hautes-Pyrénées
15	Cantal	66	Pyrénées-Orientales
16	Charente	67	Bas-Rhin
17	Charente-Maritime	68	Haut-Rhin
18	Cher	69	Rhône
19	Corrèze	70	Haute-Saône
2A	Corse-du-Sud	71	Saône-et-Loire
2B	Haute-Corse	72	Sarthe
21	Côte-d'Or	73	Savoie
22	Côtes-d'Armor	74	Haute-Savoie
23	Creuse	75	Paris
24	Dordogne	76	Seine-Maritime
25	Doubs	77	Seine-et-Marne
26	Drôme	78	Yvelines
27	Eure	79	Deux-Sèvres
28	Eure-et-Loir	80	Somme
29	Finistère	81	Tarn
30	Gard	82	Tarn-et-Garonne
31	Haute-Garonne	83	Var
32	Gers	84	Vaucluse
33	Gironde	85	Vendée
34	Hérault	86	Vienne
35	Ille-et-Vilaine	87	Haute-Vienne
36	Indre	88	Vosges
37	Indre-et-Loire	89	Yonne
38	Isère	90	Territoire de Belfort
39	Jura	91	Essonne
40	Landes	92	Hauts-de-Seine
41	Loir-et-Cher	93	Seine-Saint-Denis
42	Loire	94	Val-de-Marne
43	Haute-Loire	95	Val-d'Oise
44	Loire-Atlantique	971	Guadeloupe
45	Loiret	972	Martinique
46	Lot	973	Guyane
47	Lot-et-Garonne	974	La Réunion
48	Lozère	976	Mayotte
49	Maine-et-Loire		

Informations spatiales sur les étudiants



IDPERSONNE	POSCEOGETU	FIGGEOMETU
4	Not Printable	Not Printable
2	Not Printable	Not Printable
7	Not Printable	Not Printable
3	Not Printable	Not Printable
5	Not Printable	Not Printable

Informations sur un étudiant



Informations sur l'étudiant n° 4

Numéro :

4

Nom :

MARTIN

Prénoms :

Aleyde

Aldegonde

Albertine

Téléphones :

+230 444444444

+248 434343434

+262 414141414

+269 424242424

Adresse :

Le Cabinet des Monnaies et Médailles

10 rue Clovis-Hugues

13003 Marseille (Bouches-du-Rhône)

http://www.marseille.fr/sitevdm/jsp/site/Portal.jsp?page_id=282

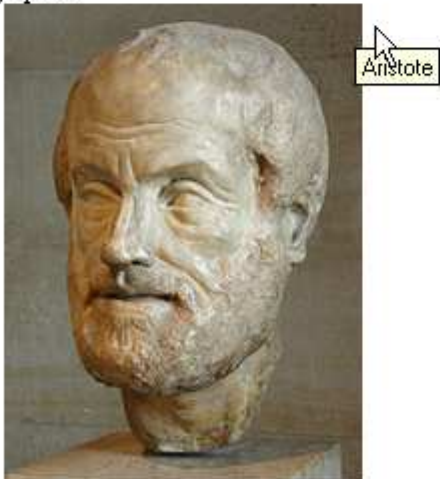
Département de naissance :

47 (Lot-et-Garonne)

Pseudonyme :

Αριστοτέλης

Photographie :



Fichier d'origine : Aristote.jpg

Répertoire d'origine : D:\Travail\Enseignement\TD\TD BD\scripts création bases\Oracle\Jouet\CoursBD RO

Dernière date de mise à jour : 13/07/2011

.../...

Dimensions : hauteur=267, largeur=200 et taille=16001
Type dans lequel l'image est stockée : JFIF
Type : 24BITRGB
Algorithme de compression : JPEG
Type MIME (Multipurpose Internet Mail Extensions) : image/jpeg

Curriculum vitæ :

Source des informations : <http://fr.wikipedia.org/wiki/Aristote>

Naissance : -384 (Stagire)

Décès : -322 (Chalcis)

École/tradition : fondateur du Lycée, Péripatétisme

Principaux intérêts : Physique, Métaphysique, Biologie, Éthique, Politique, Langage, Logique, Poétique,

Rhétorique

Idées remarquables : Syllogisme, Puissance/Acte, Matière/Forme, Substance/Accident, Catégorie

Ouvres principales : Catégories, Métaphysique, Physique, Politiques, Poétique

Influencé par : Homère, Héraclite, Parménide, Anaxagore, Empédocle, Socrate, Platon

A influencé : Théophraste, Ptolémée, Horace, Alexandre d'Aphrodise, Néoplatonisme, Boèce,

Péripatétisme, *Avicenne*, *Averroès*, Maïmonide, Thomas d'Aquin, Guillaume d'Ockham, Scolastique, Leibniz, Swedenborg, Trendelenburg, Schelling, Marx, Brentano, Heidegger, Arendt, Ayn Rand, Ricoeur

Voitures :

4747 LA 47 rouge

Diplômes :

1977 BAC Baccalauréat

1980 DEUG Diplôme d'Études Universitaires Générales

1982 MIAGe Maîtrise des Méthodes Informatiques Appliquées à la Gestion des Entreprises

Position géographique :

Type : 3001 "point 3D"

Référence spatiale : 8307 "Longitude / Latitude (WGS 84)"

Latitude : 44.843889°

Longitude : 4.22°

Altitude : 1408 m

Cf. [Google Maps](#) ou [GeoHack](#)

Figure géométrique :

Type : 2003 "2 dimension(s), 0 dimension(s) LRS, un polygone (avec ou sans trou)"

Informations sur l'étudiant n° 7

Numéro :

7

Nom :

LEROI

Prénoms :

Andoche

Ambroise

Alfred

Anastase

Aloysius

Téléphones :

+49 717171717

Adresse :

Musée d'Art Moderne et centre d'Art Contemporain de Toulouse

Les Abattoirs

76 allées Charles-de-Fitte

31300 Toulouse (Haute-Garonne)

<http://www.lesabattoirs.org>

Département de naissance :

33 (Gironde)

.../...

Pseudonyme :

أبو الوليد محمد بن احمد بن محمد بن احمد بن رشد

Photographie :



Fichier d'origine : Averroès.jpg

Répertoire d'origine : D:\Travail\Enseignement\TD\TD BD\scripts création bases\Oracle\JouetCoursBD RO

Dernière date de mise à jour : 13/07/2011

Dimensions : hauteur=157, largeur=200 et taille=10412

Type dans lequel l'image est stockée : JFIF

Type : 24BITRGB

Algorithme de compression : JPEG

Type MIME (Multipurpose Internet Mail Extensions) : image/jpeg

Curriculum vitæ :

Source des informations : <http://fr.wikipedia.org/wiki/Averroès>

Naissance : 1126 (Cordoue)

Décès : 10 décembre 1198 (Marrakech)

Principaux intérêts : Métaphysique, Théologie, Droit, Médecine, Politique, Religion

Voitures :

Diplômes :

Position géographique :

Type : 3001 "point 3D"

Référence spatiale : 8307 "Longitude / Latitude (WGS 84)"

Latitude : 46.602°

Longitude : 8.376167°

Altitude : 2250 m

Cf. [Google Maps](#) ou [GeoHack](#)

Figure géométrique :

Type : 2003 "2 dimension(s), 0 dimension(s) LRS, un polygone (avec ou sans trou)"

<http://maps.google.com/maps?q=46.602,8.376167>

Annexes

Interrogation du dictionnaire de données

L'interrogation du dictionnaire des données permet de retrouver des informations générales sur le SGBD, sur les XSD et sur les objets de cette base de données.

Informations générales sur le SGBD

```
-- informations générales (sur tous les paramètres)
SHOW PARAMETER ;
-- information sur un seul paramètre (ici, le nom de la base)
SHOW PARAMETER DB_NAME ;
      NAME      TYPE      VALUE
-----
      db_name string  OG
-- vérification de la version d'Oracle
SELECT STATUS , VERSION , COMP_NAME
FROM DBA_REGISTRY
WHERE COMP_NAME LIKE 'Oracle XML%' ;
      STATUS VERSION      COMP_NAME
-----
```

```

VALID 11.1.0.7.0 Oracle XML Database
-- vérification du port pour http
SELECT HTTP_PORT , HTTP_PROTOCOL FROM XDB.XDB$ROOT_INFO ;
      HTTP_PORT  HTTP_PROTOCOL
-----
      1158 tcp
-- affichage des configurations des DAD
DECLARE
  nDAD      DBMS_EPG.VARCHAR2_TABLE ; -- noms des DAD
  aDAD      DBMS_EPG.VARCHAR2_TABLE ; -- noms des attributs de DAD
  vDAD      DBMS_EPG.VARCHAR2_TABLE ; -- valeurs des attributs de DAD
  d         NUMBER ; -- n° du DAD
  a         NUMBER ; -- n° d'attribut de DAD
  m         NUMBER ; -- n° du "mapping" de DAD
  Autoris   NUMBER ; -- autorisations du DAD
BEGIN
  DBMS_EPG.GET_DAD_LIST(nDAD) ;
  FOR d IN 1..nDAD.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE('> DAD : '||nDAD(d)) ;
    DBMS_EPG.GET_ALL_DAD_ATTRIBUTES(nDAD(d),aDAD,vDAD) ;
    FOR a IN 1..vDAD.COUNT LOOP
      DBMS_OUTPUT.PUT_LINE('>      '||aDAD(a)||' : '||vDAD(a)) ;
    END LOOP ;
    DBMS_EPG.GET_ALL_DAD_MAPPINGS(nDAD(d),vDAD) ;
    FOR m IN 1..vDAD.COUNT LOOP
      DBMS_OUTPUT.PUT_LINE('>      mapping : '||vDAD(m)) ;
    END LOOP ;
    FOR Autoris IN ( SELECT Username
                     FROM DBA_EPG_DAD_AUTHORIZATION
                     WHERE DAD_name = nDAD(d) ) LOOP
      DBMS_OUTPUT.PUT_LINE('>      authorized : '||Autoris.Username) ;
    END LOOP ;
  END LOOP ;
END ;

> DAD : APEX
>   database-username : ANONYMOUS
>   default-page      : apex
>   document-table-name : wwv_flow_file_objects$
>   document-path     : docs
>   document-procedure : wwv_flow_file_mgr.process_download
>   nls-language      : american_america.al32utf8
>   request-validation-function : wwv_flow_epg_include_modules.authorize
>   mapping           : /apex/*
> DAD : DAD
>   database-username : ETD
>   default-page      : home
>   mapping           : /DAD/*
>   authorized       : GUIBERT
>   authorized       : ETD
-- vérification des jeux de caractères (de la base et de la session)
SHOW PARAMETER NLS_LANGUAGE ;
      NAME          TYPE          VALUE
-----
      nls_language string AMERICAN
SHOW PARAMETER NLS_TERRITORY ;
      NAME          TYPE          VALUE
-----
      nls_territory string AMERICA
SELECT *
FROM NLS_DATABASE_PARAMETERS
WHERE PARAMETER IN ( 'NLS_LANGUAGE' , 'NLS_TERRITORY' , 'NLS_CHARACTERSET' ,
                    'NLS_NCHAR_CHARACTERSET' ) ;
      PARAMETER          VALUE
-----
      NLS_LANGUAGE      AMERICAN

```

```

NLS_TERRITORY          AMERICA
NLS_CHARACTERSET       WE8MSWIN1252
NLS_NCHAR_CHARACTERSET AL16UTF16
SELECT *
FROM NLS_SESSION_PARAMETERS
WHERE PARAMETER IN ( 'NLS_LANGUAGE' , 'NLS_TERRITORY' , 'NLS_CHARACTERSET' ) ;
PARAMETER      VALUE
-----
NLS_LANGUAGE   FRENCH
NLS_TERRITORY  FRANCE
HOST ECHO %NLS_LANG%
          FRENCH_FRANCE.WE8PC850
-- utilisateur, date et heure, séquence, version
SHOW USER ;
          USER est "GUIBERT"
SELECT UID , USER , SYSDATE , SYSTIMESTAMP FROM DUAL ;
UID USER      SYSDATE  SYSTIMESTAMP
-----
          88 GUIBERT 12/07/11 12/07/11 17:53:44,906000 +02:00
SELECT Sequence_IdPersonne.CURRVAL , Sequence_IdPersonne.NEXTVAL FROM DUAL ;
CURRVAL NEXTVAL
-----
          12          12
SELECT * FROM V$VERSION ;
BANNER
-----
Oracle Database 11g Enterprise Edition Release 11.1.0.7.0 - Production
PL/SQL Release 11.1.0.7.0 - Production
CORE      11.1.0.7.0      Production
TNS for 32-bit Windows: Version 11.1.0.7.0 - Production
NLSRTL Version 11.1.0.7.0 - Production
-- utilisateurs créés simultanément ou postérieurement au compte GUIBERT
SELECT USERNAME , USER_ID , CREATED
FROM ALL_USERS
WHERE CREATED >= ( SELECT CREATED FROM ALL_USERS WHERE USERNAME = 'GUIBERT' )
ORDER BY USERNAME ;
USERNAME          USER_ID CREATED
-----
ETD                89      10/07/09
GUIBERT            88      09/07/09
SPATIAL_CSW_ADMIN_USR 91      05/07/11
SPATIAL_WFS_ADMIN_USR 90      05/07/11
-- commentaires sur quelques unes des vues du dictionnaire des données
SELECT *
FROM DICTIONARY
WHERE TABLE_NAME IN ( 'ALL_CONS_COLUMNS' , 'ALL_CONSTRAINTS' , 'ALL_DEPENDENCIES' ,
'ALL_DIRECTORIES' , 'ALL_IND_COLUMNS' , 'ALL_INDEXES' ,
'ALL_NESTED_TABLES' , 'ALL_OBJECT_TABLES' , 'ALL_OBJECTS' ,
'ALL_PROCEDURES' , 'ALL_REFS' , 'ALL_SEQUENCES' , 'ALL_SOURCE' ,
'ALL_SYNONYMS' , 'ALL_TAB_COLUMNS' , 'ALL_TABLES' ,
'ALL_TRIGGERS' , 'ALL_TYPE_METHODS' , 'ALL_TYPES' , 'ALL_USERS' ,
'ALL_VARRAYS' , 'ALL_VIEWS' , 'ALL_XML_TAB_COLS' ,
'ALL_XML_TABLES' ,
'DBA_EPG_DAD_AUTHORIZATION' , 'DBA_REGISTRY' , 'DBA_SYS_PRIVS' ,
'DBA_TAB_PRIVS' ,
'USER_SDO_GEOM_METADATA' , 'USER_SDO_INDEX_INFO' ,
'USER_SDO_INDEX_METADATA' )
ORDER BY TABLE_NAME ;
TABLE_NAME
-----
COMMENTS
-----
ALL_CONS_COLUMNS
Information about accessible columns in constraint definitions
ALL_CONSTRAINTS

```

Constraint definitions on accessible tables
 ALL_DEPENDENCIES
 Dependencies to and from objects accessible to the user
 ALL_DIRECTORIES
 Description of all directories accessible to the user
 ALL_IND_COLUMNS
 COLUMNS comprising INDEXes on accessible TABLES
 ALL_INDEXES
 Descriptions of indexes on tables accessible to the user
 ALL_NESTED_TABLES
 Description of nested tables in tables accessible to the user
 ALL_OBJECTS
 Objects accessible to the user
 ALL_OBJECT_TABLES
 Description of all object tables accessible to the user
 ALL_PROCEDURES
 Functions/procedures/packages/types/triggers available to the user
 ALL_REFS
 Description of REF columns contained in tables accessible to the user
 ALL_SEQUENCES
 Description of SEQUENCEs accessible to the user
 ALL_SOURCE
 Current source on stored objects that user is allowed to create
 ALL_SYNONYMS
 All synonyms for base objects accessible to the user and session
 ALL_TAB_COLUMNS
 Columns of user's tables, views and clusters
 ALL_TABLES
 Description of relational tables accessible to the user
 ALL_TRIGGERS
 Triggers accessible to the current user
 ALL_TYPE_METHODS
 Description of methods of types accessible to the user
 ALL_TYPES
 Description of types accessible to the user
 ALL_USERS
 Information about all users of the database
 ALL_VARRAYS
 Description of varrays in tables accessible to the user
 ALL_VIEWS
 Description of views accessible to the user
 ALL_XML_TAB_COLS
 Description of the all XMLType tables that the user has privileges on
 ALL_XML_TABLES
 Description of the all XMLType tables that the user has privileges on
 DBA_EPG_DAD_AUTHORIZATION
 DADs authorized to use different user's privileges
 DBA_REGISTRY

 DBA_SYS_PRIVS
 System privileges granted to users and roles
 DBA_TAB_PRIVS
 All grants on objects in the database

Informations générales sur les XSD

```

-- tables des XSD
SELECT XMLSCHEMA , TABLE_NAME , ELEMENT_NAME , STORAGE_TYPE
FROM ALL_XML_TABLES
WHERE OWNER = 'GUIBERT'
ORDER BY XMLSCHEMA , TABLE_NAME ;
-- colonnes des tables des XSD
SELECT XMLSCHEMA , TABLE_NAME , COLUMN_NAME , ELEMENT_NAME , STORAGE_TYPE
FROM ALL_XML_TAB_COLS
WHERE OWNER = 'GUIBERT'

```

```

ORDER BY XMLSCHEMA , TABLE_NAME , COLUMN_NAME ;
XMLSCHEMA          TABLE_NAME COLUMN_NAME          ELEMENT_NAME STORAGE_TYPE
-----
fiche_philosophes.xsd ETUDIANTS CVETU          Philosophe      BINARY

```

Informations générales sur les objets de cette base de données

```

-- objets (séquences, types, LOB , tables (dont celles imbriquées), contraintes
--      d'intégrité, index, vues, déclencheurs, fonctions et procédures,
--      paquetages, etc.)

```

```

SELECT OBJECT_TYPE , OBJECT_NAME
FROM ALL_OBJECTS
WHERE OWNER = 'GUIBERT'
ORDER BY OBJECT_TYPE , OBJECT_NAME ;
OBJECT_TYPE  OBJECT_NAME
-----
FUNCTION      ESTCODEDEPARTEMENT
INDEX         CLEPRIMAIRE_DEPARTEMENTS
INDEX         CLEPRIMAIRE_DIPLOMES
INDEX         CLEPRIMAIRE_ETUDIANTS
INDEX         INDEX_NOMETU
INDEX         INDEX_RTREE_FIGGEOMETU
INDEX         SYS_C009753
INDEX         SYS_C009783
INDEX         SYS_C009784
INDEX         SYS_C009785
INDEX         SYS_C009786
INDEX         SYS_C009969
INDEX         UNICITE_IDPERSONNE_ANNEE
INDEX         UNICITE_INTITCOMPLET
INDEX         UNICITE_NOIMMAT
INDEX         UNICITE_NOMDEPARTEMENT
LOB           SYS_LOB0000071613C00020$$$
LOB           SYS_LOB0000071613C00022$$$
LOB           SYS_LOB0000071613C00036$$$
LOB           SYS_LOB0000071613C00038$$$
LOB           SYS_LOB0000071901C00009$$$
LOB           SYS_LOB0000071901C00010$$$
LOB           SYS_LOB0000072121C00008$$$
LOB           SYS_LOB0000072121C00009$$$
LOB           SYS_LOB0000072121C00016$$$
LOB           SYS_LOB0000072121C00017$$$
LOB           SYS_LOB0000072136C00003$$$
PACKAGE       CIVIOLEES
PACKAGE       NBDIPLOBTETD
PACKAGE BODY  CIVIOLEES
PACKAGE BODY  NBDIPLOBTETD
PROCEDURE     AFFECTEPROPPHOTOSETUDIANTS
PROCEDURE     COMPARESIGNPHOTOSETUDIANTS
PROCEDURE     IMPORTEPHOTOSETUDIANTS
PROCEDURE     INSEREPEUDOETUDIANTS
SEQUENCE      MDRS_119C7$
SEQUENCE      SEQUENCE_IDPERSONNE
TABLE         DEPARTEMENTS
TABLE         DIPLOMES
TABLE         ETUDIANTS
TABLE         ETUDIANTS_GEO
TABLE         MDRT_119C7$
TABLE         TABLE_DIPLOMESOBTENUS
TABLE         TABLE_TELEPHONESPERSONNE
TABLE         TABLE_VOITURESPOSSEDEES
TRIGGER       DECLEN_AVINSERTUPDATE_ETUDIANT
TRIGGER       DECLEN_AVINSERTUPDATE_ETU_GEO
TYPE         TYPE_ADRESSE
TYPE         TYPE_DIPLOME

```

```

TYPE          TYPE_DIPLOMEOBTENU
TYPE          TYPE_DIPLOMESOBTENUS
TYPE          TYPE_ETUDIANT
TYPE          TYPE_IMMATVOITURE
TYPE          TYPE_IMMATVOITURES
TYPE          TYPE_PERSONNE
TYPE          TYPE_PRENOMS
TYPE          TYPE_TELEPHONE
TYPE          TYPE_TELEPHONES
TYPE          TYPE_VARRAY_VARCHAR
TYPE          TYPE_VOITURE
TYPE          TYPE_VOITURES
TYPE BODY     TYPE_ADRESSE
TYPE BODY     TYPE_ETUDIANT
TYPE BODY     TYPE_IMMATVOITURE
TYPE BODY     TYPE_PERSONNE
VIEW          VUE_VOITURESETUDIANTS

```

```

SELECT OBJECT_NAME , SUBOBJECT_NAME , STATUS
FROM ALL_OBJECTS
WHERE OWNER = 'GUIBERT' AND OBJECT_TYPE = 'LOB'
ORDER BY OBJECT_NAME ;

```

OBJECT_NAME	SUBOBJECT_NAME	STATUS
SYS_LOB0000071613C00020\$\$		VALID
SYS_LOB0000071613C00022\$\$		VALID
SYS_LOB0000071613C00036\$\$		VALID
SYS_LOB0000071613C00038\$\$		VALID
SYS_LOB0000071901C00009\$\$		VALID
SYS_LOB0000071901C00010\$\$		VALID
SYS_LOB0000072121C00008\$\$		VALID
SYS_LOB0000072121C00009\$\$		VALID
SYS_LOB0000072121C00016\$\$		VALID
SYS_LOB0000072121C00017\$\$		VALID
SYS_LOB0000072136C00003\$\$		VALID

```
-- répertoires
```

```

SELECT DIRECTORY_NAME , DIRECTORY_PATH
FROM ALL_DIRECTORIES
WHERE DIRECTORY_NAME = UPPER('Rep_Etudiants') ;

```

```

DIRECTORY_NAME
-----
DIRECTORY_PATH
-----

```

```
REP_ETUDIANTS
```

```
D:\Travail\Enseignement\TD\TD BD\scripts création bases\Oracle\JouetCoursBD RO
```

```
-- séquences
```

```

SELECT SEQUENCE_NAME , MIN_VALUE , MAX_VALUE , INCREMENT_BY
FROM ALL_SEQUENCES
WHERE SEQUENCE_OWNER = 'GUIBERT'
ORDER BY SEQUENCE_NAME ;

```

SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY
MDRS_119C7\$	1	1,0000E+27	1
SEQUENCE_IDPERSONNE	1	99	1

```
-- types
```

```

SELECT TYPE_NAME , TYPECODE , ATTRIBUTES , METHODS
FROM ALL_TYPES
WHERE OWNER = 'GUIBERT'
ORDER BY TYPE_NAME ;

```

TYPE_NAME	TYPECODE	ATTRIBUTES	METHODS
TYPE_ADRESSE	OBJECT	6	1
TYPE_DIPLOME	OBJECT	2	0
TYPE_DIPLOMEOBTENU	OBJECT	2	0
TYPE_DIPLOMESOBTENUS	COLLECTION	0	0
TYPE_ETUDIANT	OBJECT	12	8

TYPE_IMMATVOITURE	OBJECT	3	1
TYPE_IMMATVOITURES	COLLECTION	0	0
TYPE_PERSONNE	OBJECT	5	6
TYPE_PRENOMS	COLLECTION	0	0
TYPE_TELEPHONE	OBJECT	2	0
TYPE_TELEPHONES	COLLECTION	0	0
TYPE_VARRAY_VARCHAR	COLLECTION	0	0
TYPE_VOITURE	OBJECT	2	0
TYPE_VOITURES	COLLECTION	0	0

-- méthodes

```
SELECT TYPE_NAME , METHOD_NAME , METHOD_TYPE
FROM ALL_TYPE_METHODS
WHERE OWNER = 'GUIBERT'
ORDER BY TYPE_NAME , METHOD_NAME ;
```

TYPE_NAME	METHOD_NAME	METHOD
TYPE_ADRESSE	DEPARTADRESSE	PUBLIC
TYPE_ETUDIANT	DEPARTNAISSETU_UPDATE	PUBLIC
TYPE_ETUDIANT	ESTGIROINDIN	PUBLIC
TYPE_ETUDIANT	INITIALEPRENOMPERSONNE	PUBLIC
TYPE_ETUDIANT	NOMPRENOMPERSONNE	PUBLIC
TYPE_ETUDIANT	PRENOMPERSONNE_DELETE	PUBLIC
TYPE_ETUDIANT	PRENOMPERSONNE_INSERT	PUBLIC
TYPE_ETUDIANT	PRENOMPERSONNE_UPDATE	PUBLIC
TYPE_ETUDIANT	TYPE_PERSONNE_COMPARE	ORDER
TYPE_IMMATVOITURE	CONCATNOIMMAT	PUBLIC
TYPE_PERSONNE	INITIALEPRENOMPERSONNE	PUBLIC
TYPE_PERSONNE	NOMPRENOMPERSONNE	PUBLIC
TYPE_PERSONNE	PRENOMPERSONNE_DELETE	PUBLIC
TYPE_PERSONNE	PRENOMPERSONNE_INSERT	PUBLIC
TYPE_PERSONNE	PRENOMPERSONNE_UPDATE	PUBLIC
TYPE_PERSONNE	TYPE_PERSONNE_COMPARE	ORDER

-- dépendances entre types

```
SELECT TYPE , NAME , REFERENCED_NAME , REFERENCED_TYPE
FROM ALL_DEPENDENCIES
WHERE OWNER = 'GUIBERT'
ORDER BY TYPE , NAME ;
```

TYPE	NAME	REFERENCED_NAME	REFERENCED_TYPE
FUNCTION	ESTCODEDEPARTEMENT	STANDARD	PACKAGE
FUNCTION	ESTCODEDEPARTEMENT	DEPARTEMENTS	TABLE
FUNCTION	ESTCODEDEPARTEMENT	SYS_STUB_FOR_PURITY_ANALYSIS	PACKAGE
INDEX	INDEX_RTREE_FIGGEOMETU	SPATIAL_INDEX	INDEXTYPE
PACKAGE	NBDIPLOBTETD	DIPLOMES	TABLE
PACKAGE	NBDIPLOBTETD	STANDARD	PACKAGE
PACKAGE BODY	CIVIOLEES	DIPLOMES	TABLE
PACKAGE BODY	CIVIOLEES	DBMS_OUTPUT	SYNONYM
PACKAGE BODY	CIVIOLEES	STANDARD	PACKAGE
PACKAGE BODY	CIVIOLEES	CIVIOLEES	PACKAGE
PACKAGE BODY	CIVIOLEES	TYPE_IMMATVOITURE	TYPE
PACKAGE BODY	CIVIOLEES	DEPARTEMENTS	TABLE
PACKAGE BODY	CIVIOLEES	ETUDIANTS	TABLE
PACKAGE BODY	NBDIPLOBTETD	DIPLOMES	TABLE
PACKAGE BODY	NBDIPLOBTETD	STANDARD	PACKAGE
PACKAGE BODY	NBDIPLOBTETD	DBMS_STANDARD	PACKAGE
PACKAGE BODY	NBDIPLOBTETD	NBDIPLOBTETD	PACKAGE
PACKAGE BODY	NBDIPLOBTETD	DBMS_OUTPUT	SYNONYM
PACKAGE BODY	NBDIPLOBTETD	ETUDIANTS	TABLE
PROCEDURE	AFFECTEPROPPHOTOSETUDIANTS	SYS_STUB_FOR_PURITY_ANALYSIS	PACKAGE
PROCEDURE	AFFECTEPROPPHOTOSETUDIANTS	ETUDIANTS	TABLE
PROCEDURE	AFFECTEPROPPHOTOSETUDIANTS	STANDARD	PACKAGE
PROCEDURE	AFFECTEPROPPHOTOSETUDIANTS	ORDIMAGE	TYPE
PROCEDURE	AFFECTEPROPPHOTOSETUDIANTS	DBMS_OUTPUT	SYNONYM
PROCEDURE	AFFECTEPROPPHOTOSETUDIANTS	ORDIMAGESIGNATURE	TYPE
PROCEDURE	COMPARESIGNPHOTOSETUDIANTS	IMGSIMILAR	OPERATOR
PROCEDURE	COMPARESIGNPHOTOSETUDIANTS	DBMS_OUTPUT	SYNONYM
PROCEDURE	COMPARESIGNPHOTOSETUDIANTS	SYS_STUB_FOR_PURITY_ANALYSIS	PACKAGE
PROCEDURE	COMPARESIGNPHOTOSETUDIANTS	STANDARD	PACKAGE

PROCEDURE	COMPARESIGNPHOTOSETUDIANTS	ETUDIANTS	TABLE
PROCEDURE	COMPARESIGNPHOTOSETUDIANTS	ORDIMAGESIGNATURE	TYPE
PROCEDURE	IMPORTEPHOTOSETUDIANTS	ETUDIANTS	TABLE
PROCEDURE	IMPORTEPHOTOSETUDIANTS	SYS_STUB_FOR_PURITY_ANALYSIS	PACKAGE
PROCEDURE	IMPORTEPHOTOSETUDIANTS	STANDARD	PACKAGE
PROCEDURE	IMPORTEPHOTOSETUDIANTS	DBMS_OUTPUT	SYNONYM
PROCEDURE	IMPORTEPHOTOSETUDIANTS	ORDSOURCE	TYPE
PROCEDURE	IMPORTEPHOTOSETUDIANTS	ORDIMAGE	TYPE
PROCEDURE	INSEREPSEUDOETUDIANTS	DBMS_STANDARD	PACKAGE
PROCEDURE	INSEREPSEUDOETUDIANTS	STANDARD	PACKAGE
PROCEDURE	INSEREPSEUDOETUDIANTS	ETUDIANTS	TABLE
PROCEDURE	INSEREPSEUDOETUDIANTS	DBMS_LOB	SYNONYM
PROCEDURE	INSEREPSEUDOETUDIANTS	DBMS_OUTPUT	SYNONYM
PROCEDURE	INSEREPSEUDOETUDIANTS	ALL_DIRECTORIES	SYNONYM
PROCEDURE	INSEREPSEUDOETUDIANTS	SYS_STUB_FOR_PURITY_ANALYSIS	PACKAGE
TABLE	BIN\$gP63IRFdTEqFqnjSJW/LLg==\$0	SDO_GEOMETRY	TYPE
TABLE	BIN\$gP63IRFdTEqFqnjSJW/LLg==\$0	SDO_ELEM_INFO_ARRAY	TYPE
TABLE	BIN\$gP63IRFdTEqFqnjSJW/LLg==\$0	STANDARD	PACKAGE
TABLE	BIN\$gP63IRFdTEqFqnjSJW/LLg==\$0	SDO_ORDINATE_ARRAY	TYPE
TABLE	DIPLOMES	STANDARD	PACKAGE
TABLE	DIPLOMES	TYPE_DIPLOME	TYPE
TABLE	ETUDIANTS	STANDARD	PACKAGE
TABLE	ETUDIANTS	ORDIMAGESIGNATURE	TYPE
TABLE	ETUDIANTS	TYPE_VOITURES	TYPE
TABLE	ETUDIANTS	TYPE_TELEPHONES	TYPE
TABLE	ETUDIANTS	TYPE_PRENOMS	TYPE
TABLE	ETUDIANTS	TYPE_ETUDIANT	TYPE
TABLE	ETUDIANTS	TYPE_DIPLOMESOBTENUS	TYPE
TABLE	ETUDIANTS	TYPE_ADRESSE	TYPE
TABLE	ETUDIANTS	XMLTYPE	SYNONYM
TABLE	ETUDIANTS	ORDIMAGE	TYPE
TABLE	ETUDIANTS	DBURITYPE	TYPE
TABLE	ETUDIANTS	XDZ21Q4fEQR/KxdVylnpCB1Q==	XML SCHEMA
TABLE	ETUDIANTS	HTTPURITYPE	TYPE
TABLE	ETUDIANTS	URITYPE	SYNONYM
TABLE	ETUDIANTS	FTPURITYPE	TYPE
TABLE	ETUDIANTS	URITYPE	TYPE
TABLE	ETUDIANTS	XMLTYPE	TYPE
TABLE	ETUDIANTS	XDBURITYPE	TYPE
TABLE	ETUDIANTS_GEO	SDO_ELEM_INFO_ARRAY	TYPE
TABLE	ETUDIANTS_GEO	SDO_ORDINATE_ARRAY	TYPE
TABLE	ETUDIANTS_GEO	STANDARD	PACKAGE
TABLE	ETUDIANTS_GEO	SDO_GEOMETRY	TYPE
TABLE	TABLE_DIPLOMESOBTENUS	TYPE_DIPLOMEOBTENU	TYPE
TABLE	TABLE_DIPLOMESOBTENUS	STANDARD	PACKAGE
TABLE	TABLE_DIPLOMESOBTENUS	TYPE_DIPLOME	TYPE
TABLE	TABLE_TELEPHONESPERSONNE	TYPE_TELEPHONE	TYPE
TABLE	TABLE_TELEPHONESPERSONNE	STANDARD	PACKAGE
TABLE	TABLE_VOITURESPOSSEDEES	TYPE_IMMATVOITURE	TYPE
TABLE	TABLE_VOITURESPOSSEDEES	STANDARD	PACKAGE
TABLE	TABLE_VOITURESPOSSEDEES	TYPE_VOITURE	TYPE
TRIGGER	DECLEN_AVINSERTUPDATE_ETUDIANT	DBMS_STANDARD	PACKAGE
TRIGGER	DECLEN_AVINSERTUPDATE_ETUDIANT	TYPE_DIPLOMESOBTENUS	TYPE
TRIGGER	DECLEN_AVINSERTUPDATE_ETUDIANT	PLITBLM	SYNONYM
TRIGGER	DECLEN_AVINSERTUPDATE_ETUDIANT	TYPE_DIPLOME	TYPE
TRIGGER	DECLEN_AVINSERTUPDATE_ETUDIANT	TYPE_ADRESSE	TYPE
TRIGGER	DECLEN_AVINSERTUPDATE_ETUDIANT	TYPE_VOITURE	TYPE
TRIGGER	DECLEN_AVINSERTUPDATE_ETUDIANT	ETUDIANTS	TABLE
TRIGGER	DECLEN_AVINSERTUPDATE_ETUDIANT	ESTCODEDEPARTEMENT	FUNCTION
TRIGGER	DECLEN_AVINSERTUPDATE_ETUDIANT	TYPE_IMMATVOITURE	TYPE
TRIGGER	DECLEN_AVINSERTUPDATE_ETUDIANT	TYPE_VOITURES	TYPE
TRIGGER	DECLEN_AVINSERTUPDATE_ETUDIANT	STANDARD	PACKAGE
TRIGGER	DECLEN_AVINSERTUPDATE_ETUDIANT	TYPE_DIPLOMEOBTENU	TYPE
TRIGGER	DECLEN_AVINSERTUPDATE_ETU_GEO	STANDARD	PACKAGE
TRIGGER	DECLEN_AVINSERTUPDATE_ETU_GEO	USER_SDO_GEOM_METADATA	SYNONYM
TRIGGER	DECLEN_AVINSERTUPDATE_ETU_GEO	SDO_POINT_TYPE	TYPE
TRIGGER	DECLEN_AVINSERTUPDATE_ETU_GEO	DBMS_STANDARD	PACKAGE
TRIGGER	DECLEN_AVINSERTUPDATE_ETU_GEO	SDO_GEOM	PACKAGE
TRIGGER	DECLEN_AVINSERTUPDATE_ETU_GEO	ETUDIANTS_GEO	TABLE
TRIGGER	DECLEN_AVINSERTUPDATE_ETU_GEO	SDO_GEOM	SYNONYM
TRIGGER	DECLEN_AVINSERTUPDATE_ETU_GEO	SDO_GEOMETRY	TYPE
TRIGGER	DECLEN_AVINSERTUPDATE_ETU_GEO	SDO_DIM_ARRAY	TYPE


```

TABLE_DIPLOMESOBTENUS      TYPE_DIPLOMEOBTENU YES      DISABLED
TABLE_TELEPHONESPERSONNE  TYPE_TELEPHONE      YES      DISABLED
TABLE_VOITURESPOSSEDEES  TYPE_VOITURE        YES      DISABLED
-- tables imbriquées
SELECT PARENT_TABLE_NAME , PARENT_TABLE_COLUMN , TABLE_NAME
FROM ALL_NESTED_TABLES
WHERE OWNER = 'GUIBERT'
ORDER BY PARENT_TABLE_NAME , PARENT_TABLE_COLUMN ;
PARENT_TABLE_NAME PARENT_TABLE_COLUMN TABLE_NAME
-----
ETUDIANTS          DIPLOMESOBTENUS      TABLE_DIPLOMESOBTENUS
ETUDIANTS          TELEPHONESPERSONNE  TABLE_TELEPHONESPERSONNE
ETUDIANTS          VOITURESPOSSEDEES  TABLE_VOITURESPOSSEDEES
-- tableaux variables
SELECT PARENT_TABLE_NAME , PARENT_TABLE_COLUMN
FROM ALL_VARRAYS
WHERE OWNER = 'GUIBERT'
ORDER BY PARENT_TABLE_NAME , PARENT_TABLE_COLUMN ;
PARENT_TABLE_NAME          PARENT_TABLE_COLUMN
-----
BIN$gP63IRFdTEqFqnjSjW/LLg==$0 "FORME" ."SDO_ELEM_INFO"
BIN$gP63IRFdTEqFqnjSjW/LLg==$0 "FORME" ."SDO_ORDINATES"
ETUDIANTS                    PRENOMSPERSONNE
ETUDIANTS_GEO                "FIGGEOMETU" ."SDO_ELEM_INFO"
ETUDIANTS_GEO                "FIGGEOMETU" ."SDO_ORDINATES"
ETUDIANTS_GEO                "POS GEOGETU" ."SDO_ELEM_INFO"
ETUDIANTS_GEO                "POS GEOGETU" ."SDO_ORDINATES"
-- vues
SELECT VIEW_NAME FROM ALL_VIEWS WHERE OWNER = 'GUIBERT' ORDER BY VIEW_NAME ;
VIEW_NAME
-----
VUE_VOITURESETUDIANTS
-- colonnes des tables et vues
SELECT TABLE_NAME , COLUMN_ID , COLUMN_NAME , DATA_TYPE
FROM ALL_TAB_COLUMNS
WHERE OWNER = 'GUIBERT'
ORDER BY TABLE_NAME , COLUMN_ID ;
TABLE_NAME          COLUMN_ID COLUMN_NAME          DATA_TYPE
-----
DEPARTEMENTS        1 CODEDEPARTEMENT      VARCHAR2
DEPARTEMENTS        2 NOMDEPARTEMENT      VARCHAR2
DIPLOMES            1 INTITABREGE         CHAR
DIPLOMES            2 INTITCOMPLET        VARCHAR2
ETUDIANTS           1 IDPERSONNE          NUMBER
ETUDIANTS           2 NOMPERSONNE         VARCHAR2
ETUDIANTS           3 PRENOMSPERSONNE    TYPE_PRENOMS
ETUDIANTS           4 TELEPHONESPERSONNE TYPE_TELEPHONES
ETUDIANTS           5 ADRESSEPERSONNE    TYPE_ADRESSE
ETUDIANTS           6 DEPARTNAISSETU     VARCHAR2
ETUDIANTS           7 PSEUDOETU          BLOB
ETUDIANTS           8 PHOTOETU           ORDIMAGE
ETUDIANTS           9 SIGNPHOTOETU       ORDIMAGESIGNATURE
ETUDIANTS           10 CVETU              XMLTYPE
ETUDIANTS           11 VOITURESPOSSEDEES TYPE_VOITURES
ETUDIANTS           12 DIPLOMESOBTENUS    TYPE_DIPLOMESOBTENUS
ETUDIANTS_GEO       1 IDPERSONNE          NUMBER
ETUDIANTS_GEO       2 POSGEOGETU         SDO_GEOMETRY
ETUDIANTS_GEO       3 FIGGEOMETU         SDO_GEOMETRY
MDRT_119C7$         1 NODE_ID            NUMBER
MDRT_119C7$         2 NODE_LEVEL         NUMBER
MDRT_119C7$         3 INFO               BLOB
VUE_VOITURESETUDIANTS 1 NOIMMATCHIFFRES    NUMBER
VUE_VOITURESETUDIANTS 2 NOIMMATLETTRES     VARCHAR2
VUE_VOITURESETUDIANTS 3 NOIMMATDEPART      VARCHAR2
VUE_VOITURESETUDIANTS 4 COULEUR            VARCHAR2

```

VUE_VOITURESETUDIANTS	5 IDPERSONNE	NUMBER
VUE_VOITURESETUDIANTS	6 NOMPERSONNE	VARCHAR2
VUE_VOITURESETUDIANTS	7 DEPARTNAISSETU	VARCHAR2

-- contraintes d'intégrité
-- (clés primaires, référentielles, unicités, existentielles, autres)

```
SELECT TABLE_NAME , CONSTRAINT_NAME , CONSTRAINT_TYPE
FROM ALL_CONSTRAINTS
WHERE OWNER = 'GUIBERT'
ORDER BY TABLE_NAME , CONSTRAINT_NAME ;
```

TABLE_NAME	CONSTRAINT_NAME	C
-----	-----	-----
DEPARTEMENTS	CLEPRIMAIRE_DEPARTEMENTS	P
DEPARTEMENTS	SYS_C009977	C
DEPARTEMENTS	SYS_C009978	C
DEPARTEMENTS	UNICITE_NOMDEPARTEMENT	U
DIPLOMES	CLEPRIMAIRE_DIPLOMES	P
DIPLOMES	EXISTE_INTITABREGE	C
DIPLOMES	EXISTE_INTITCOMPLET	C
DIPLOMES	SYS_C009753	U
DIPLOMES	UNICITE_INTITCOMPLET	U
ETUDIANTS	CLEPRIMAIRE_ETUDIANTS	P
ETUDIANTS	CONTRAINTE_IDPERSONNEPOSITIF	C
ETUDIANTS	EXISTE_CODEPOSTALADRPERSO	C
ETUDIANTS	EXISTE_LIGNE1ADRPERSO	C
ETUDIANTS	EXISTE_VILLEADRPERSO	C
ETUDIANTS	REF_ETUDIANTS_DEPARTEMENTS	R
ETUDIANTS	SYS_C009775	C
ETUDIANTS	SYS_C009776	C
ETUDIANTS	SYS_C009777	C
ETUDIANTS	SYS_C009783	U
ETUDIANTS	SYS_C009784	U
ETUDIANTS	SYS_C009785	U
ETUDIANTS	SYS_C009786	U
ETUDIANTS_GEO	SYS_C009968	C
ETUDIANTS_GEO	SYS_C009969	P
ETUDIANTS_GEO	SYS_C009970	R
TABLE_VOITURESPOSSEDEES	CONTRAINTE_LISTECOULEURS	C
TABLE_VOITURESPOSSEDEES	CONTRAINTE_NOIMMATCHIFFRESBORN	C
TABLE_VOITURESPOSSEDEES	CONTRAINTE_NOIMMATLETTRESMAJUS	C
TABLE_VOITURESPOSSEDEES	EXISTE_NOIMMATCHIFFRES	C
TABLE_VOITURESPOSSEDEES	EXISTE_NOIMMATLETTRES	C
TABLE_VOITURESPOSSEDEES	UNICITE_NOIMMAT	U

```
SELECT TABLE_NAME , CONSTRAINT_NAME , R_CONSTRAINT_NAME
FROM ALL_CONSTRAINTS
WHERE OWNER = 'GUIBERT' AND CONSTRAINT_TYPE = 'R'
ORDER BY TABLE_NAME , CONSTRAINT_NAME ;
```

TABLE_NAME	CONSTRAINT_NAME	R_CONSTRAINT_NAME
-----	-----	-----
ETUDIANTS	REF_ETUDIANTS_DEPARTEMENTS	CLEPRIMAIRE_DEPARTEMENTS
ETUDIANTS_GEO	SYS_C009970	CLEPRIMAIRE_ETUDIANTS

```
SELECT TABLE_NAME , CONSTRAINT_NAME , SEARCH_CONDITION
FROM ALL_CONSTRAINTS
WHERE OWNER = 'GUIBERT' AND CONSTRAINT_TYPE = 'C'
ORDER BY TABLE_NAME , CONSTRAINT_NAME ;
```

TABLE_NAME	CONSTRAINT_NAME	SEARCH_CONDITION
-----	-----	-----
DEPARTEMENTS	SYS_C009977	"CODEDEPARTEMENT" IS NOT NULL
DEPARTEMENTS	SYS_C009978	"NOMDEPARTEMENT" IS NOT NULL
DIPLOMES	EXISTE_INTITABREGE	IntitAbrege IS NOT NULL
DIPLOMES	EXISTE_INTITCOMPLET	IntitCompleet IS NOT NULL
ETUDIANTS	CONTRAINTE_IDPERSONNEPOSITIF	IdPersonne > 0
ETUDIANTS	EXISTE_CODEPOSTALADRPERSO	AdressePersonne.CodePostal IS NOT NULL
ETUDIANTS	EXISTE_LIGNE1ADRPERSO	AdressePersonne.Ligne1 IS NOT NULL
ETUDIANTS	EXISTE_VILLEADRPERSO	AdressePersonne.Ville IS NOT NULL
ETUDIANTS	SYS_C009775	"IDPERSONNE" IS NOT NULL
ETUDIANTS	SYS_C009776	"NOMPERSONNE" IS NOT NULL
ETUDIANTS	SYS_C009777	"DEPARTNAISSETU" IS NOT NULL
ETUDIANTS_GEO	SYS_C009968	"IDPERSONNE" IS NOT NULL

```

TABLE_VOITURESPOSSEDEES  CONTRAINTE_LISTECOULEURS          Couleur IS NULL OR Couleur IN
                                                                    ( 'rouge' , 'jaune' , 'orange' )
TABLE_VOITURESPOSSEDEES  CONTRAINTE_NOIMMATCHIFFRESBORN      NoImmat.Chiffres BETWEEN 1 AND 9999
TABLE_VOITURESPOSSEDEES  CONTRAINTE_NOIMMATLETTRESMAJUS      NoImmat.Lettres = UPPER(NoImmat.Lettres)
TABLE_VOITURESPOSSEDEES  EXISTE_NOIMMATCHIFFRES          NoImmat.Chiffres IS NOT NULL
TABLE_VOITURESPOSSEDEES  EXISTE_NOIMMATLETTRES              NoImmat.Lettres IS NOT NULL
SELECT TABLE_NAME , CONSTRAINT_NAME , COLUMN_NAME
FROM ALL_CONS_COLUMNS
WHERE OWNER = 'GUIBERT'
ORDER BY TABLE_NAME , CONSTRAINT_NAME , COLUMN_NAME ;

```

TABLE_NAME	CONSTRAINT_NAME	COLUMN_NAME
DEPARTEMENTS	CLEPRIMAIRE_DEPARTEMENTS	CODEDEPARTEMENT
DEPARTEMENTS	SYS_C009977	CODEDEPARTEMENT
DEPARTEMENTS	SYS_C009978	NOMDEPARTEMENT
DEPARTEMENTS	UNICITE_NOMDEPARTEMENT	NOMDEPARTEMENT
DIPLOMES	CLEPRIMAIRE_DIPLOMES	INTITABREGE
DIPLOMES	EXISTE_INTITABREGE	INTITABREGE
DIPLOMES	EXISTE_INTITCOMPLET	INTITCOMPLET
DIPLOMES	SYS_C009753	SYS_NC_OID\$
DIPLOMES	UNICITE_INTITCOMPLET	INTITCOMPLET
ETUDIANTS	CLEPRIMAIRE_ETUDIANTS	IDPERSONNE
ETUDIANTS	CONTRAINTE_IDPERSONNEPOSITIF	IDPERSONNE
ETUDIANTS	EXISTE_CODEPOSTALADRPERSO	"ADRESSEPERSONNE"."CODEPOSTAL"
ETUDIANTS	EXISTE_LIGNE1ADRPERSO	"ADRESSEPERSONNE"."LIGNE1"
ETUDIANTS	EXISTE_VILLEADRPERSO	"ADRESSEPERSONNE"."VILLE"
ETUDIANTS	REF_ETUDIANTS_DEPARTEMENTS	DEPARTNAISSETU
ETUDIANTS	SYS_C009775	IDPERSONNE
ETUDIANTS	SYS_C009776	NOMPERSONNE
ETUDIANTS	SYS_C009777	DEPARTNAISSETU
ETUDIANTS	SYS_C009783	SYS_NC0004100042\$
ETUDIANTS	SYS_C009784	SYS_NC0003900040\$
ETUDIANTS	SYS_C009785	SYS_NC0000600007\$
ETUDIANTS	SYS_C009786	SYS_NC_OID\$
ETUDIANTS_GEO	SYS_C009968	IDPERSONNE
ETUDIANTS_GEO	SYS_C009969	IDPERSONNE
ETUDIANTS_GEO	SYS_C009970	IDPERSONNE
TABLE_VOITURESPOSSEDEES	CONTRAINTE_LISTECOULEURS	COULEUR
TABLE_VOITURESPOSSEDEES	CONTRAINTE_NOIMMATCHIFFRESBORN	"NOIMMAT"."CHIFFRES"
TABLE_VOITURESPOSSEDEES	CONTRAINTE_NOIMMATLETTRESMAJUS	"NOIMMAT"."LETTRES"
TABLE_VOITURESPOSSEDEES	EXISTE_NOIMMATCHIFFRES	"NOIMMAT"."CHIFFRES"
TABLE_VOITURESPOSSEDEES	EXISTE_NOIMMATLETTRES	"NOIMMAT"."LETTRES"
TABLE_VOITURESPOSSEDEES	UNICITE_NOIMMAT	"NOIMMAT"."CHIFFRES"
TABLE_VOITURESPOSSEDEES	UNICITE_NOIMMAT	"NOIMMAT"."DEPART"
TABLE_VOITURESPOSSEDEES	UNICITE_NOIMMAT	"NOIMMAT"."LETTRES"

```

SELECT CONSTRAINT_NAME , COLUMN_NAME
FROM ALL_CONSTRAINTS
NATURAL JOIN ALL_CONS_COLUMNS
WHERE OWNER = 'GUIBERT'
ORDER BY CONSTRAINT_NAME , COLUMN_NAME ;

```

CONSTRAINT_NAME	COLUMN_NAME
CLEPRIMAIRE_DEPARTEMENTS	CODEDEPARTEMENT
CLEPRIMAIRE_DIPLOMES	INTITABREGE
CLEPRIMAIRE_ETUDIANTS	IDPERSONNE
CONTRAINTE_IDPERSONNEPOSITIF	IDPERSONNE
CONTRAINTE_LISTECOULEURS	COULEUR
CONTRAINTE_NOIMMATCHIFFRESBORN	"NOIMMAT"."CHIFFRES"
CONTRAINTE_NOIMMATLETTRESMAJUS	"NOIMMAT"."LETTRES"
EXISTE_CODEPOSTALADRPERSO	"ADRESSEPERSONNE"."CODEPOSTAL"
EXISTE_INTITABREGE	INTITABREGE
EXISTE_INTITCOMPLET	INTITCOMPLET
EXISTE_LIGNE1ADRPERSO	"ADRESSEPERSONNE"."LIGNE1"
EXISTE_NOIMMATCHIFFRES	"NOIMMAT"."CHIFFRES"
EXISTE_NOIMMATLETTRES	"NOIMMAT"."LETTRES"
EXISTE_VILLEADRPERSO	"ADRESSEPERSONNE"."VILLE"
REF_ETUDIANTS_DEPARTEMENTS	DEPARTNAISSETU
SYS_C009753	SYS_NC_OID\$
SYS_C009775	IDPERSONNE
SYS_C009776	NOMPERSONNE

```

SYS_C009777          DEPARTNAISSETU
SYS_C009783          SYS_NC0004100042$
SYS_C009784          SYS_NC0003900040$
SYS_C009785          SYS_NC0000600007$
SYS_C009786          SYS_NC_OID$
SYS_C009968          IDPERSONNE
SYS_C009969          IDPERSONNE
SYS_C009970          IDPERSONNE
SYS_C009977          CODEDEPARTEMENT
SYS_C009978          NOMDEPARTEMENT
UNICITE_INTITCOMPLET INTITCOMPLET
UNICITE_NOIMMAT      "NOIMMAT" ."CHIFFRES"
UNICITE_NOIMMAT      "NOIMMAT" ."DEPART"
UNICITE_NOIMMAT      "NOIMMAT" ."LETTRES"
UNICITE_NOMDEPARTEMENT NOMDEPARTEMENT

```

-- index (dont clés primaires, unicités)

```

SELECT TABLE_NAME , INDEX_NAME , INDEX_TYPE , UNIQUENESS
FROM ALL_INDEXES
WHERE OWNER = 'GUIBERT'
ORDER BY TABLE_NAME , INDEX_NAME ;

```

TABLE_NAME	INDEX_NAME	INDEX_TYPE	UNIQUENESS
DEPARTEMENTS	CLEPRIMAIRE_DEPARTEMENTS	NORMAL	UNIQUE
DEPARTEMENTS	UNICITE_NOMDEPARTEMENT	NORMAL	UNIQUE
DIPLOMES	CLEPRIMAIRE_DIPLOMES	NORMAL	UNIQUE
DIPLOMES	SYS_C009753	NORMAL	UNIQUE
DIPLOMES	UNICITE_INTITCOMPLET	NORMAL	UNIQUE
ETUDIANTS	CLEPRIMAIRE_ETUDIANTS	NORMAL	UNIQUE
ETUDIANTS	INDEX_NOMETU	NORMAL	NONUNIQUE
ETUDIANTS	SYS_C009783	NORMAL	UNIQUE
ETUDIANTS	SYS_C009784	NORMAL	UNIQUE
ETUDIANTS	SYS_C009785	NORMAL	UNIQUE
ETUDIANTS	SYS_C009786	NORMAL	UNIQUE
ETUDIANTS_GEO	INDEX_RTREE_FIGGEOMETU	DOMAIN	NONUNIQUE
ETUDIANTS_GEO	SYS_C009969	NORMAL	UNIQUE
TABLE_DIPLOMESOBTENUS	UNICITE_IDPERSONNE_ANNEE	NORMAL	UNIQUE
TABLE_VOITURESPOSSEDEES	UNICITE_NOIMMAT	NORMAL	UNIQUE

```

SELECT TABLE_NAME , INDEX_NAME , COLUMN_NAME
FROM ALL_IND_COLUMNS
WHERE TABLE_OWNER = 'GUIBERT'
ORDER BY TABLE_NAME , INDEX_NAME , COLUMN_NAME ;

```

TABLE_NAME	INDEX_NAME	COLUMN_NAME
DEPARTEMENTS	CLEPRIMAIRE_DEPARTEMENTS	CODEDEPARTEMENT
DEPARTEMENTS	UNICITE_NOMDEPARTEMENT	NOMDEPARTEMENT
DIPLOMES	CLEPRIMAIRE_DIPLOMES	INTITABREGÉ
DIPLOMES	SYS_C009753	SYS_NC_OID\$
DIPLOMES	UNICITE_INTITCOMPLET	INTITCOMPLET
ETUDIANTS	CLEPRIMAIRE_ETUDIANTS	IDPERSONNE
ETUDIANTS	INDEX_NOMETU	NOMPERSONNE
ETUDIANTS	SYS_C009783	DIPLOMESOBTENUS
ETUDIANTS	SYS_C009784	VOITURESPOSSEDEES
ETUDIANTS	SYS_C009785	TELEPHONESPERSONNE
ETUDIANTS	SYS_C009786	SYS_NC_OID\$
ETUDIANTS_GEO	INDEX_RTREE_FIGGEOMETU	FIGGEOMETU
ETUDIANTS_GEO	SYS_C009969	IDPERSONNE
TABLE_DIPLOMESOBTENUS	UNICITE_IDPERSONNE_ANNEE	ANNEE
TABLE_DIPLOMESOBTENUS	UNICITE_IDPERSONNE_ANNEE	NESTED_TABLE_ID
TABLE_VOITURESPOSSEDEES	UNICITE_NOIMMAT	"NOIMMAT" ."CHIFFRES"
TABLE_VOITURESPOSSEDEES	UNICITE_NOIMMAT	"NOIMMAT" ."DEPART"
TABLE_VOITURESPOSSEDEES	UNICITE_NOIMMAT	"NOIMMAT" ."LETTRES"

-- déclencheurs

```

SELECT TABLE_NAME , TRIGGER_NAME , TRIGGER_TYPE , TRIGGERING_EVENT
FROM ALL_TRIGGERS
WHERE OWNER = 'GUIBERT'

```

```

ORDER BY TABLE_NAME , TRIGGER_NAME , TRIGGER_TYPE , TRIGGERING_EVENT ;
TABLE_NAME      TRIGGER_NAME      TRIGGER_TYPE      TRIGGERING_EVENT
-----
ETUDIANTS      DECLEN_AVINSERTUPDATE_ETUDIANT BEFORE EACH ROW INSERT OR UPDATE
ETUDIANTS_GEO DECLEN_AVINSERTUPDATE_ETU_GEO  BEFORE EACH ROW INSERT OR UPDATE
-- fonctions et procédures (indépendantes ou des paquetages)
SELECT OBJECT_TYPE , OBJECT_NAME , PROCEDURE_NAME
FROM ALL_PROCEDURES
WHERE OWNER = 'GUIBERT'
ORDER BY OBJECT_TYPE , OBJECT_NAME , PROCEDURE_NAME ;
OBJECT_TYPE      OBJECT_NAME      PROCEDURE_NAME
-----
FUNCTION          ESTCODEDEPARTEMENT
PACKAGE          CIVIOLEES      DEFAULT_COULEUR_VIOLEE
PACKAGE          CIVIOLEES      DEFAULT_NOIMMATDEPART_VIOLEE
PACKAGE          CIVIOLEES      REF_ETUDIANTS_DIPL_VIOLEE
PACKAGE          CIVIOLEES      REF_IMMATVOITURE_DPTS_VIOLEE
PACKAGE          CIVIOLEES      UNICITE_IDPERSONNE_DIPL_VIOLEE
PACKAGE          CIVIOLEES
PACKAGE          NBDIPLOBTETD      NOMBREDIPLOMES
PACKAGE          NBDIPLOBTETD      NOMBREDIPLOMES
PACKAGE          NBDIPLOBTETD      NOMBREDIPLOMES
PACKAGE          NBDIPLOBTETD
PROCEDURE        AFFECTEPROPPHOTOSETUDIANTS
PROCEDURE        COMPARESIGNPHOTOSETUDIANTS
PROCEDURE        IMPORTEPHOTOSETUDIANTS
PROCEDURE        INSEREPEUDOETUDIANTS
TRIGGER          DECLEN_AVINSERTUPDATE_ETUDIANT
TRIGGER          DECLEN_AVINSERTUPDATE_ETU_GEO
TYPE             TYPE_ADRESSE      DEPARTADRESSE
TYPE             TYPE_ETUDIANT      DEPARTNAISSETU_UPDATE
TYPE             TYPE_ETUDIANT      ESTGIRONDIN
TYPE             TYPE_IMMATVOITURE  CONCATNOIMMAT
TYPE             TYPE_PERSONNE      INITIALEPRENOMPERSONNE
TYPE             TYPE_PERSONNE      NOMPrenomPERSONNE
TYPE             TYPE_PERSONNE      PRENOMPERSONNE_DELETE
TYPE             TYPE_PERSONNE      PRENOMPERSONNE_INSERT
TYPE             TYPE_PERSONNE      PRENOMPERSONNE_UPDATE
TYPE             TYPE_PERSONNE      TYPE_PERSONNE_COMPARE
-- code source (type et corps de type, déclencheur, fonction et procédure,
-- paquetage et corps de paquetage, etc.)
SELECT NAME , TYPE , LINE , TEXT
FROM ALL_SOURCE
WHERE OWNER = 'GUIBERT'
ORDER BY NAME , TYPE , LINE ;
SELECT NAME , TYPE , LINE , TEXT
FROM ALL_SOURCE
WHERE OWNER = 'GUIBERT' AND NAME IN ( 'ESTCODEDEPARTEMENT' , 'TYPE_DIPLOME' )
ORDER BY NAME , TYPE , LINE ;
NAME              TYPE              LINE TEXT
-----
ESTCODEDEPARTEMENT FUNCTION        1 FUNCTION EstCodeDepartement (
ESTCODEDEPARTEMENT FUNCTION        2      cd IN VARCHAR ) -- code du départem[...]
ESTCODEDEPARTEMENT FUNCTION        3      RETURN NUMBER
ESTCODEDEPARTEMENT FUNCTION        4 IS
ESTCODEDEPARTEMENT FUNCTION        5      trouv NUMBER(1) ; -- le code du dép[...]
ESTCODEDEPARTEMENT FUNCTION        6 BEGIN
ESTCODEDEPARTEMENT FUNCTION        7      SELECT COUNT(*) INTO trouv
FROM Departements
WHERE CodeDepartement = cd ;
ESTCODEDEPARTEMENT FUNCTION        8      IF trouv = 0 THEN
ESTCODEDEPARTEMENT FUNCTION        9          RETURN(0) ; -- n'est pas un[...]
ESTCODEDEPARTEMENT FUNCTION       10      ELSE
ESTCODEDEPARTEMENT FUNCTION       11          RETURN(1) ; -- est un départ[...]
ESTCODEDEPARTEMENT FUNCTION       12      END IF ;

```

```

ESTCODEDEPARTEMENT FUNCTION 13 END ;
TYPE_DIPLOME TYPE 1 TYPE Type_Diplome AS OBJECT (
TYPE_DIPLOME TYPE 2 IntitAbrege CHAR(5) ,
TYPE_DIPLOME TYPE 3 IntitCompleet VARCHAR(80)
TYPE_DIPLOME TYPE 4 ) ;
-- droits
SELECT PRIVILEGE FROM DBA_SYS_PRIVS WHERE GRANTEE = 'GUIBERT' ORDER BY PRIVILEGE ;
PRIVILEGE
-----
UNLIMITED TABLESPACE
SELECT TABLE_NAME , PRIVILEGE , GRANTEE , GRANTOR
FROM DBA_TAB_PRIVS
WHERE OWNER = 'GUIBERT'
ORDER BY TABLE_NAME , PRIVILEGE , GRANTEE , GRANTOR ;
TABLE_NAME PRIVILEGE GRANTEE GRANTOR
-----
DEPARTEMENTS SELECT ETD GUIBERT
DIPLOMES SELECT ETD GUIBERT
ETUDIANTS SELECT ETD GUIBERT
ETUDIANTS_GEO SELECT ETD GUIBERT
TABLE_DIPLOMESOBTENUS SELECT ETD GUIBERT
TABLE_TELEPHONESPERSONNE SELECT ETD GUIBERT
TABLE_VOITURESPOSSEDEES SELECT ETD GUIBERT
TYPE_VARRAY_VARCHAR EXECUTE ETD GUIBERT
TYPE_VARRAY_VARCHAR EXECUTE PUBLIC GUIBERT
-- synonymes
SELECT SYNONYM_NAME , TABLE_NAME
FROM ALL_SYNONYMS
WHERE TABLE_OWNER = 'GUIBERT'
ORDER BY SYNONYM_NAME , TABLE_NAME ;
SYNONYM_NAME TABLE_NAME
-----
DEPARTEMENTS DEPARTEMENTS
DIPLOMES DIPLOMES
ETUDIANTS ETUDIANTS
ETUDIANTS_GEO ETUDIANTS_GEO
TABLE_DIPLOMESOBTENUS TABLE_DIPLOMESOBTENUS
TABLE_TELEPHONESPERSONNE TABLE_TELEPHONESPERSONNE
TABLE_VOITURESPOSSEDEES TABLE_VOITURESPOSSEDEES
VUE_VOITURESETUDIANTS VUE_VOITURESETUDIANTS

```

Suppression de la base de données

```

-- synonymes
DROP PUBLIC SYNONYM Departements ;
DROP PUBLIC SYNONYM Diplomes ;
DROP PUBLIC SYNONYM Etudiants_Geo ;
DROP PUBLIC SYNONYM Etudiants ;
DROP PUBLIC SYNONYM Table_TelephonesPersonne ;
DROP PUBLIC SYNONYM Table_VoituresPossedees ;
DROP PUBLIC SYNONYM Table_DiplomesObtenus ;
DROP PUBLIC SYNONYM Vue_VoituresEtudiants ;
-- droits
REVOKE READ ON DIRECTORY Rep_Etudiants FROM ETD ;
REVOKE EXECUTE ON Type_VARRAY_VARCHAR FROM PUBLIC ;
REVOKE EXECUTE ON Type_VARRAY_VARCHAR FROM ETD ;
REVOKE SELECT ON Departements FROM ETD ;
REVOKE SELECT ON Diplomes FROM ETD ;
REVOKE SELECT ON Etudiants_Geo FROM ETD ;
REVOKE SELECT ON Etudiants FROM ETD ;
-- corps de paquetages
DROP PACKAGE BODY NbDiplObtEtd ;
DROP PACKAGE BODY Civoilees ;
-- paquetages
DROP PACKAGE NbDiplObtEtd ;

```

```

DROP PACKAGE Civiolees ;
-- fonctions et procédures
DROP PROCEDURE CompareSignPhotosEtudiants ;
DROP PROCEDURE AffectePropPhotosEtudiants ;
DROP PROCEDURE InserePseudoEtudiants ;
DROP FUNCTION EstCodeDepartement ;
-- déclencheurs
DROP TRIGGER Declen_AvInsertUpdate_Etu_Geo ;
DROP TRIGGER Declen_AvInsertUpdate_Etudiant ;
-- corps de types
DROP TYPE BODY Type_Etudiant ;
DROP TYPE BODY Type_Personne ;
DROP TYPE BODY Type_Adresse ;
DROP TYPE BODY Type_ImmatVoiture ;
-- vues
DROP VIEW Vue_VoituresEtudiants ;
-- index
DROP INDEX Index_NomEtu ;
-- index spatiaux
--DROP INDEX Index_QuadTree_FigGeomEtu ;
DROP INDEX Index_RTree_FigGeomEtu ;
-- métadonnées pour Oracle Spatial
DELETE
    FROM USER_SDO_GEOM_METADATA
    WHERE TABLE_NAME = 'ETUDIANTS_GEO' AND COLUMN_NAME = 'FIGGEOMETU' ;
COMMIT ;
-- tables (et contraintes d'intégrité)
DROP TABLE Etudiants_Geo CASCADE CONSTRAINTS ;
DROP TABLE Etudiants CASCADE CONSTRAINTS ;
DROP TABLE Diplomes CASCADE CONSTRAINTS ;
DROP TABLE Departements CASCADE CONSTRAINTS ;
-- types
DROP TYPE Type_Etudiant ;
DROP TYPE Type_Personne ;
DROP TYPE Type_DiplomesObtenus ;
DROP TYPE Type_DiplomeObtenu ;
DROP TYPE Type_Diplome ;
DROP TYPE Type_Voitures ;
DROP TYPE Type_Voiture ;
DROP TYPE Type_ImmatVoitures ;
DROP TYPE Type_ImmatVoiture ;
DROP TYPE Type_Adresse ;
DROP TYPE Type_Telephones ;
DROP TYPE Type_Telephone ;
DROP TYPE Type_Prenoms ;
DROP TYPE Type_VARRAY_VARCHAR ;
-- séquences
DROP SEQUENCE Sequence_IdPersonne ;
-- répertoires
DROP DIRECTORY Rep_Etudiants ;
-- utilisateurs
DROP USER ETD CASCADE ;
-- XSD
BEGIN
DBMS_XMLSCHEMA.DELETESCHEMA('fiche_philosophe.xsd',
                            DBMS_XMLSCHEMA.DELETE_CASCADE_FORCE);
END ;
-- DAD
CONNECT / AS SYSDBA
DECLARE
    nd VARCHAR2(30) := 'DAD' ; -- nom du DAD
BEGIN
    EXEC DBMS_EPG.DROP_DAD(DAD_NAME=>nd) ;
END ;
QUIT

```


Suppression des données

```
DELETE FROM Etudiants_Geo ;
COMMIT ;
DELETE FROM Etudiants ;
COMMIT ;
DELETE FROM Diplomes ;
COMMIT ;
DELETE FROM Departements ;
COMMIT ;
```

Images



Aristote.jpg



Averroès.jpg



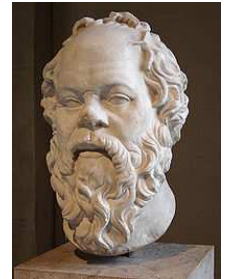
Épictète.jpg



Épikure.jpg



Platon.jpg



Socrate.jpg

```
Aristote_Pseudo.jpg   Ἀριστοτέλης
Averroès_Pseudo.jpg  أبو الوليد محمد بن احمد بن محمد بن احمد بن رشد
Épictète_Pseudo.jpg  Ἐπίκτητος
Épikure_Pseudo.jpg   Ἐπίκουρος
Platon_Pseudo.jpg    Πλάτων
Socrate_Pseudo.jpg   Σωκράτης
```

N. B. : cf. <http://fr.wikipedia.org/wiki/> pour obtenir davantage d'informations sur ces étudiants philosophes...

Fichier *fiche_philosophe.xsd*

```
<?xml
  version="1.0"
  encoding="UTF-8"
  ?>
<xsd:schema
  attributeFormDefault="unqualified"
  elementFormDefault="qualified"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xdb="http://xmlns.oracle.com/xdb"
  xdb:storeVarrayAsTable="false"
  version="1.0"
  >
<xsd:element
  name="Philosophe"
  type="PhilosopheTypeXML"
  />
<!-- Philosophe -->
<xsd:complexType
  name="PhilosopheTypeXML"
  >
  <xsd:sequence>
    <xsd:element
      name="SourceFiche"
      type="xsd:anyURI"
      xdb:SQLName="SourceFiche"
    />
    <xsd:element
      name="Naissance"
      type="NaissanceTypeXML"
    />
  </xsd:sequence>
</xsd:complexType>
```

```

        xdb:SQLName="Naissance"
    />
    <xsd:element
        name="Deces"
        type="DecesTypeXML"
        xdb:SQLName="Deces"
    />
    <xsd:element
        name="Ecoles"
        type="EcolesTypeXML"
        xdb:SQLName="Ecoles"
    />
    <xsd:element
        name="Interets"
        type="InteretsTypeXML"
        xdb:SQLName="Interets"
    />
    <xsd:element
        name="Idees"
        type="IdeesTypeXML"
        xdb:SQLName="Idees"
    />
    <xsd:element
        name="OEuvres"
        type="OEuvresTypeXML"
        xdb:SQLName="OEuvres"
    />
    <xsd:element
        name="InfluencesPar"
        type="InfluencesParTypeXML"
        xdb:SQLName="InfluencesPar"
    />
    <xsd:element
        name="InfluencesSur"
        type="InfluencesSurTypeXML"
        xdb:SQLName="InfluencesSur"
    />
</xsd:sequence>
</xsd:complexType>
<!-- Naissance -->
<xsd:complexType
    name="NaissanceTypeXML"
    >
    <xsd:sequence>
        <xsd:element
            name="AnneeNaissance"
            type="xsd:string"
            xdb:SQLName="AnneeNaissance"
        />
        <xsd:element
            name="LieuNaissance"
            type="xsd:string"
            xdb:SQLName="LieuNaissance"
        />
    </xsd:sequence>
</xsd:complexType>
<!-- Deces -->
<xsd:complexType
    name="DecesTypeXML"
    >
    <xsd:sequence>
        <xsd:element
            name="AnneeDeces"
            type="xsd:string"
            xdb:SQLName="AnneeDeces"

```

```

        />
        <xsd:element
            name="LieuDeces"
            type="xsd:string"
            xdb:SQLName="LieuDeces"
        />
    </xsd:sequence>
</xsd:complexType>
<!-- Ecoles -->
<xsd:complexType
    name="EcolesTypeXML"
    >
    <xsd:sequence>
        <xsd:element
            minOccurs="0"
            maxOccurs="unbounded"
            name="Ecole"
            type="xsd:string"
            xdb:SQLName="Ecole"
            xdb:maintainOrder="true"
        />
    </xsd:sequence>
</xsd:complexType>
<!-- Interets -->
<xsd:complexType
    name="InteretsTypeXML"
    >
    <xsd:sequence>
        <xsd:element
            minOccurs="1"
            maxOccurs="unbounded"
            name="Interet"
            type="xsd:string"
            xdb:SQLName="Interet"
            xdb:maintainOrder="true"
        />
    </xsd:sequence>
</xsd:complexType>
<!-- Idees -->
<xsd:complexType
    name="IdeesTypeXML"
    >
    <xsd:sequence>
        <xsd:element
            minOccurs="0"
            maxOccurs="unbounded"
            name="Idee"
            type="xsd:string"
            xdb:SQLName="Idee"
            xdb:maintainOrder="true"
        />
    </xsd:sequence>
</xsd:complexType>
<!-- OEuvres -->
<xsd:complexType
    name="OEuvresTypeXML"
    >
    <xsd:sequence>
        <xsd:element
            minOccurs="0"
            maxOccurs="unbounded"
            name="OEuvre"
            type="xsd:string"
            xdb:SQLName="OEuvre"
            xdb:maintainOrder="true"

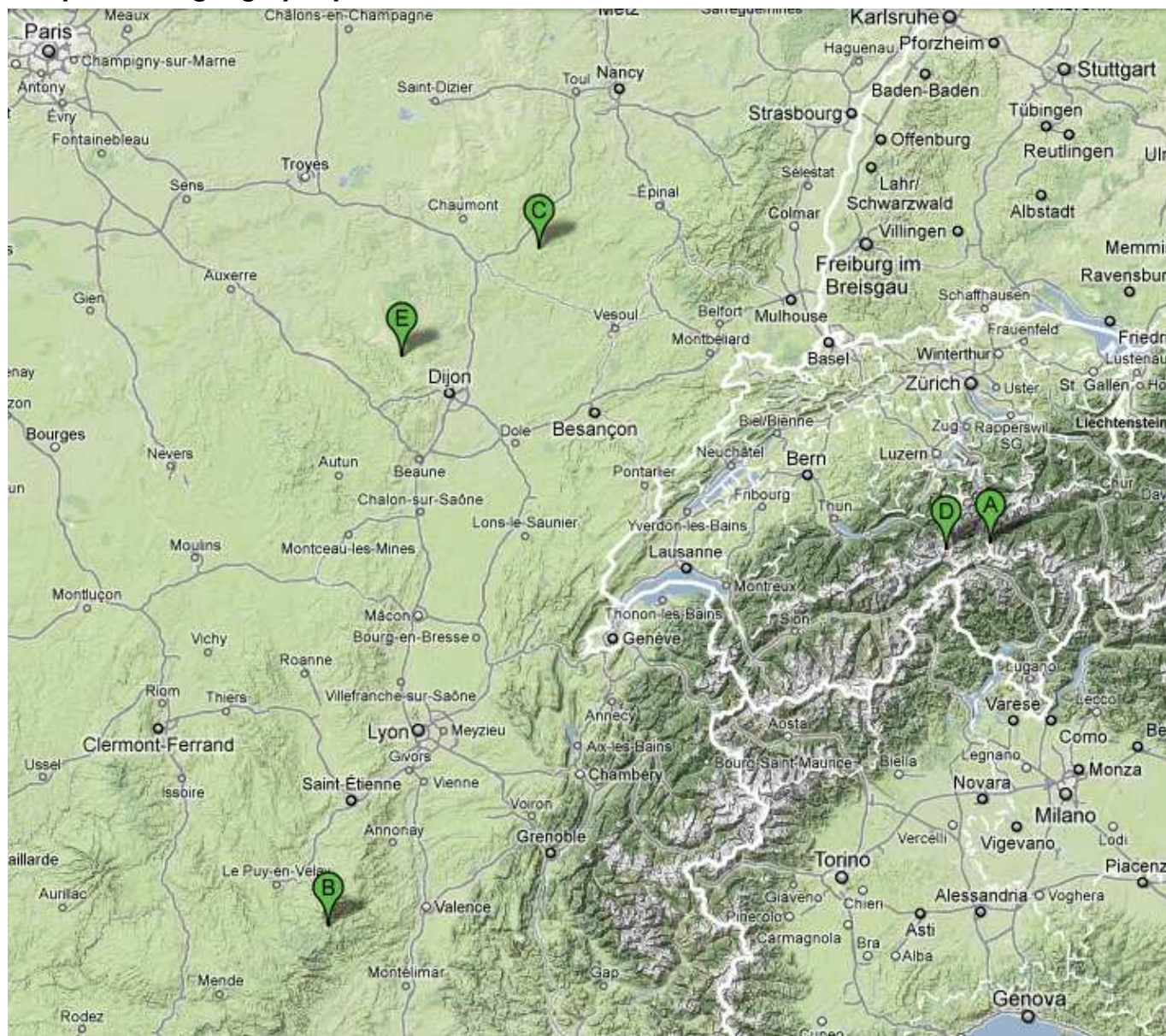
```

```

        />
    </xsd:sequence>
</xsd:complexType>
<!-- InfluencesPar -->
<xsd:complexType
    name="InfluencesParTypeXML"
    >
    <xsd:sequence>
        <xsd:element
            minOccurs="0"
            maxOccurs="unbounded"
            name="InfluencePar"
            type="xsd:string"
            xdb:SQLName="InfluencePar"
            xdb:maintainOrder="true"
            />
    </xsd:sequence>
</xsd:complexType>
<!-- InfluencesSur -->
<xsd:complexType
    name="InfluencesSurTypeXML"
    >
    <xsd:sequence>
        <xsd:element
            minOccurs="0"
            maxOccurs="unbounded"
            name="InfluenceSur"
            type="xsd:string"
            xdb:SQLName="InfluenceSur"
            xdb:maintainOrder="true"
            />
    </xsd:sequence>
</xsd:complexType>
</xsd:schema>

```

Les positions géographiques



	Fleuve	Longueur	Source	Latitude		Longitude		Altitude
				Coordonnées		Coordonnées		
				Sexagésimales	Décimales	Sexagésimales	Décimales	
A	Rhin	1320 km	Rheinquelle (Sud du col de l'Oberalp, Suisse)	46° 37' 57" N	46.6325°	8° 40' 20" E	8.672222°	2346 m
B	Loire	1013 km	mont Gerbier-de-Jonc (Sainte-Eulalie, Ardèche (07), France)	44° 50' 38" N	44.843889°	4° 13' 12" E	4.22°	1408 m
C	Meuse	950 km	Pouilly-en-Bassigny (Haute-Marne (52), France)	47° 58' 27.66" N	47.97435°	5° 38' 0.74" E	5.633539°	409 m
D	Rhône	812 km	Gletsch (Glacier du Rhône, Canton du Valais, Suisse)	46° 36' 7.2" N	46.602°	8° 22' 34.2" E	8.376167°	2250 m
E	Seine	777 km	plateau de Langres (Source-Seine, Côte-d'Or (21), France)	47° 29' 10.26" N	47.486183°	4° 43' 2.86" E	4.717461°	446 m

N. B. : cf. <http://fr.wikipedia.org/wiki/> et <http://toolserver.org/~geohack/>.

Les figures géométriques

Afin de visualiser simplement des figures géométriques avec un navigateur Web, voici par exemple un code PL/SQL qui crée, pour différents tuples valides d'un attribut spatial indexé, et seulement dans certains cas, des instructions à insérer dans une balise Canvas d'HTML5 pré-programmée en JavaScript.

Code PL/SQL créant les instructions à insérer

```
-- crée, pour un SDO_GEOMETRY, les appels à des procédures JavaScript dans Canvas d'HTML5
-- Principaux cas d'utilisation :
--   d = 2 (2D)
--   tt = 01 (orientation non traitée) et 05, 02 et 06, 03 et 07 c.-à-d.
--       point(s), segment(s) de droite(s) et d'arc(s) de cercle, polygone(s) composés de
--       segments de droites ou segments d'arc(s) de cercle ou rectangles ou cercles
--   etype,interpretation = 1,n (entier positif) 2,{1,2} 4,n (entier positif)
--                               {1,2}003,{1,2,3,4} {1,2}005,n (entier positif)
DECLARE
  c NUMBER ; -- curseur
  info VARCHAR2(250); -- information sur l'étudiant
  f Etudiants_Geo.FigGeomEtu%TYPE ; -- figure géométrique (SDO_GEOMETRY) de l'étudiant
  bornes_aff BOOLEAN ; -- lignes 4 bornes (et zoom et quadrillage) ont été affichées ?
  borne_inf_x NUMBER ; -- borne inférieure pour la première coordonnée
  borne_inf_y NUMBER ; -- borne inférieure pour la seconde coordonnée
  borne_sup_x NUMBER ; -- borne supérieure pour la première coordonnée
  borne_sup_y NUMBER ; -- borne supérieure pour la seconde coordonnée
  d NUMBER ; -- dimension (de dlts = SDO_GEOMETRY.SDO_GTYPE) de la figure géométrique de l'étd
  tt NUMBER ; -- type (de dlts = SDO_GEOMETRY.SDO_GTYPE) de la figure géométrique de l'étd
  i_ordinates NUMBER ; -- indice de SDO_GEOMETRY.SDO_ORDINATES
  min_ordinates_x NUMBER ; -- valeur minimale 1res coordonnées de SDO_GEOMETRY.SDO_ORDINATES
  min_ordinates_y NUMBER ; -- valeur minimale 2des coordonnées de SDO_GEOMETRY.SDO_ORDINATES
  i_triplet NUMBER ; -- indice de SDO_GEOMETRY.SDO_ELEM_INFO
  starting NUMBER ; -- starting_offset (SDO_STARTING_OFFSET) 1 élément figure géométrique étd
  etype NUMBER ; -- etype (SDO_ETYPE) 1 élément figure géométrique étd
  interpret NUMBER ; -- interpretation (SDO_INTERPRETATION) 1 élément figure géométrique étd
  nb_compo NUMBER ; -- nombre de composantes d'un élément composé de figure géométrique de étd
  i_compo NUMBER ; -- indice de composantes
  aff_1er_pt BOOLEAN ; -- indique si on doit afficher le 1er point (vrai pr tt=2, faux pr tt=3)
  coul_ext_ou_int CHAR(3) ; -- indique si couleur doit être celle de extérieur ou de intérieur
  txt_deb_pt VARCHAR2(70) := 'affiche_point(ctx,conversion_repere_x(canvas,borne_inf_x,' ;
  -- texte du début de l'instruction d'affichage d'un point
  txt_deb_drt VARCHAR2(70) := 'affiche_droite(ctx,conversion_repere_x(canvas,borne_inf_x,' ;
  -- texte du début de l'instruction d'affichage d'une droite
  txt_deb_arc VARCHAR2(70) :=
    'affiche_arc_cercle(ctx,conversion_repere_x(canvas,borne_inf_x,' ;
  -- texte du début de l'instruction d'affichage d'un arc de cercle
  txt_deb_rct VARCHAR2(70) :=
    'affiche_rectangle(ctx,conversion_repere_x(canvas,borne_inf_x,' ;
  -- texte du début de l'instruction d'affichage d'un rectangle
  txt_deb_crcl VARCHAR2(70) := 'affiche_cercle(ctx,conversion_repere_x(canvas,borne_inf_x,' ;
  -- texte du début de l'instruction d'affichage d'un cercle
  txt_separ_x VARCHAR2(70) := ',zoom),conversion_repere_x(canvas,borne_inf_x,' ;
  -- texte séparation précédant une première coordonnée d'une
  -- instruction d'affichage d'une figure géométrique
  txt_separ_y VARCHAR2(70) := ',zoom),conversion_repere_y(canvas,borne_inf_y,' ;
  -- texte séparation précédant une seconde coordonnée d'une
  -- instruction d'affichage d'une figure géométrique
  txt_fin1 VARCHAR2(70) := ',zoom),couleur_' ;
  -- 1re partie texte fin instruction affichage figure géométrique
  txt_fin2_pt VARCHAR2(70) := ',epaiss_point);' ;
  -- 2de partie texte fin instruction d'affichage d'un point
  txt_fin2_sgmt VARCHAR2(70) := ',epaiss_segment);' ;
  -- 2de partie texte fin instruction d'affichage d'un segment
  dern_ordinates NUMBER ; -- dernière seconde coordonnée à considérer
  i_segment NUMBER ; -- indice des segments (droites ou arcs de cercle) identiques répétées
BEGIN
  EXECUTE IMMEDIATE 'ALTER SESSION SET NLS_NUMERIC_CHARACTERS = ''.,'' ' ;
  -- remplacement de virgule par point pour séparateur entre partie entière et décimale
  DBMS_OUTPUT.ENABLE(50000) ;
  bornes_aff := FALSE ;
  -- récupération des figures géométriques valides et existantes de chaque étudiant avec les
  -- métadonnées (grille = première et seconde coordonnées inférieure et supérieure)
  FOR c IN ( SELECT TO_CHAR(EG_FG.IdPersonne) info , EG_FG.FigGeomEtu f ,
```

```

        X_DIMINFO.SDO_LB borne_inf_x , Y_DIMINFO.SDO_LB borne_inf_y ,
        X_DIMINFO.SDO_UB borne_sup_x , Y_DIMINFO.SDO_UB borne_sup_y
FROM Etudiants_Geo EG_FG ,
USER_SDO_GEOM_METADATA X , TABLE ( X.DIMINFO ) X_DIMINFO ,
USER_SDO_GEOM_METADATA Y , TABLE ( Y.DIMINFO ) Y_DIMINFO
WHERE X.TABLE_NAME = UPPER('Etudiants_Geo') AND
X.COLUMN_NAME = UPPER('FigGeomEtu') AND
X_DIMINFO.SDO_DIMNAME = 'abscisse (X)' AND
SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT(EG_FG.FigGeomEtu,X.DIMINFO) =
'TRUE' AND
Y.TABLE_NAME = UPPER('Etudiants_Geo') AND
Y.COLUMN_NAME = UPPER('FigGeomEtu') AND
Y_DIMINFO.SDO_DIMNAME = 'ordonnées (Y)' AND
SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT(EG_FG.FigGeomEtu,Y.DIMINFO) =
'TRUE' AND
EG_FG.FigGeomEtu IS NOT NULL
ORDER BY IdPersonne
) LOOP
-- si la figure géométrique existe et est en 2D et tt appartient à {1,2,3,5,6,7}
-- c.-à-d. point(s), segment(s) de droite(s) ou d'arc(s) de cercle, polygone(s)
IF c.f IS NOT NULL THEN
d := c.f.GET_DIMS();
tt := c.f.GET_GTYPE();
IF d = 2 AND tt IN ( 1 , 2 , 3 , 5 , 6 , 7 ) THEN
-- affichage éventuel des 4 bornes et zoom et quadrillage
IF NOT bornes_aff THEN
DBMS_OUTPUT.PUT_LINE('borne_inf_x = ' || c.borne_inf_x || ');');
DBMS_OUTPUT.PUT_LINE('borne_inf_y = ' || c.borne_inf_y || ');');
DBMS_OUTPUT.PUT_LINE('borne_sup_x = ' || c.borne_sup_x || ');');
DBMS_OUTPUT.PUT_LINE('borne_sup_y = ' || c.borne_sup_y || ');');
DBMS_OUTPUT.PUT_LINE('zoom = calcul_zoom(canvas,borne_inf_x,borne_inf_y,borne_sup_x,
borne_sup_y);');
DBMS_OUTPUT.PUT_LINE('affiche_quadrillage(canvas,ctx,borne_inf_x,borne_inf_y,
borne_sup_x,borne_sup_y,zoom,couleur_quadrill,
couleur_quadrill_zero,epaiss_quadrill,espacemt_quadrill);');
bornes_aff := TRUE ;
END IF ;
-- information sur l'étudiant
DBMS_OUTPUT.PUT_LINE('/// ' || c.info) ;
-- traitement de SDO_POINT
IF c.f.SDO_POINT IS NOT NULL THEN
-- affichage de l'information sur l'étudiant
DBMS_OUTPUT.PUT_LINE('affiche_texte(ctx,conversion_repere_x(canvas,borne_inf_x,' ||
c.f.SDO_POINT.X || txt_separ_y || c.f.SDO_POINT.Y || ',zoom),' ||
TRIM(SUBSTR(c.info,1,2)) || ',couleur_texte,police_texte);');
-- affichage du point
DBMS_OUTPUT.PUT_LINE(txt_deb_pt || c.f.SDO_POINT.X || txt_separ_y || c.f.SDO_POINT.Y ||
txt_fin1 || 'ext' || txt_fin2_pt) ;
END IF ;
-- traitement de SDO_ELEM_INFO (et SDO_ORDINATES)
IF c.f.SDO_ELEM_INFO IS NOT NULL THEN -- et donc c.f.SDO_ORDINATES IS NOT NULL
-- affichage de l'information sur l'étudiant
min_ordinates_x := c.borne_sup_x;
min_ordinates_y := c.borne_sup_y;
FOR i_ordinates IN 1..c.f.SDO_ORDINATES.COUNT/2 LOOP
IF c.f.SDO_ORDINATES(2*i_ordinates-1) < min_ordinates_x THEN
min_ordinates_x := c.f.SDO_ORDINATES(2*i_ordinates-1) ;
END IF ;
IF c.f.SDO_ORDINATES(2*i_ordinates) < min_ordinates_y THEN
min_ordinates_y := c.f.SDO_ORDINATES(2*i_ordinates) ;
END IF ;
END LOOP ;
DBMS_OUTPUT.PUT_LINE('affiche_texte(ctx,conversion_repere_x(canvas,borne_inf_x,' ||
min_ordinates_x || txt_separ_y || min_ordinates_y || ',zoom),' ||
TRIM(SUBSTR(c.info,1,2)) || ',couleur_texte,police_texte);');
-- traitement des triplets SDO_ELEM_INFO
i_triplet := 1 ;
WHILE 3 * i_triplet <= c.f.SDO_ELEM_INFO.COUNT LOOP
-- constitution du triplet
starting := c.f.SDO_ELEM_INFO(3*i_triplet-2) ;
etype := c.f.SDO_ELEM_INFO(3*i_triplet-1) ;
interpret := c.f.SDO_ELEM_INFO(3*i_triplet) ;

```

```

-- cas des point(s)
IF etype = 1 THEN
  nb_compo := interpret ;
  FOR i_compo IN 1..nb_compo LOOP -- on ne traite donc pas l'orientation
    DBMS_OUTPUT.PUT_LINE(txt_deb_pt||c.f.SDO_ORDINATES(starting+2*i_compo-2)||
      txt_separ_y||c.f.SDO_ORDINATES(starting+2*i_compo-1)||
      txt_fin1||'ext'||txt_fin2_pt) ;
  END LOOP ;
-- cas des segment(s) de droite(s) ou d'arc(s) de cercle et cas des polygone(s)
ELSIF etype IN ( 2 , 4 , 1003 , 2003 , 1005 , 2005 ) THEN
  -- afficher le premier point ?
  IF tt IN ( 2 , 6 ) THEN
    aff_ler_pt := TRUE ; -- pour segments, il faut afficher le premier point
  ELSE
    aff_ler_pt := FALSE ; -- pr polygones (et points), ne pas ré-afficher 1er point
  END IF ;
  -- couleur extérieure ou intérieure
  IF etype >= 2000 THEN
    coul_ext_ou_int := 'int' ; -- pour les etype=200{3,5}, couleur intérieure
  ELSE
    coul_ext_ou_int := 'ext' ; -- par défaut, c'est la couleur extérieure
  END IF ;
  -- figure simple ou composée
  IF etype IN ( 4 , 1005 , 2005 ) THEN
    nb_compo := interpret ; -- figure ayant a priori plusieurs composantes(s)
    -- positionnement sur le triplet suivant c.-à-d. sur 1re composante à traiter
    i_triplet := i_triplet + 1 ;
    starting := c.f.SDO_ELEM_INFO(3*i_triplet-2) ;
    etype := c.f.SDO_ELEM_INFO(3*i_triplet-1) ;
    interpret := c.f.SDO_ELEM_INFO(3*i_triplet) ;
  ELSE
    nb_compo := 1 ; -- figure simple (ayant une seule composante)
  END IF ;
  -- traitement de chacune des composantes (ou de l'unique composante)
  FOR i_compo IN 1..nb_compo LOOP
    -- calcul de dernière coordonnée à considérer, pour segments et polygones
    IF 3 * i_triplet = c.f.SDO_ELEM_INFO.COUNT THEN -- ou >=
      -- arrêt sur la toute dernière coordonnée
      dern_ordinates := c.f.SDO_ORDINATES.COUNT ;
    ELSIF i_compo < nb_compo THEN
      -- connexion des deux composantes successives (avec 1er point segment suiv.)
      dern_ordinates := c.f.SDO_ELEM_INFO(3*i_triplet+1) + 1 ;
    ELSE
      -- arrêt sur la toute dernière coordonnée de la dernière composante
      dern_ordinates := c.f.SDO_ELEM_INFO(3*i_triplet+1) - 1 ;
    END IF ;
    -- traitement d'une composante
    IF etype IN ( 2 , 1003 , 2003 ) AND interpret = 1 THEN
      -- ##### un (ou plusieurs) segment(s) de droite(s) #####
      -- affichage du premier point
      IF aff_ler_pt AND i_compo = 1 THEN
        DBMS_OUTPUT.PUT_LINE(txt_deb_pt||c.f.SDO_ORDINATES(starting)||txt_separ_y
          ||c.f.SDO_ORDINATES(starting+1)||txt_fin1||
          coul_ext_ou_int||txt_fin2_pt) ;
      END IF ;
      -- affichage de chaque droite avec son point de fin
      i_segment := starting + 3 ; -- on doit tracer une droite reliant au moins le
        -- point du début de composante à un autre point
      WHILE i_segment <= dern_ordinates LOOP
        DBMS_OUTPUT.PUT_LINE(txt_deb_drt||c.f.SDO_ORDINATES(i_segment-3)||
          txt_separ_y||c.f.SDO_ORDINATES(i_segment-2)||
          txt_separ_x||c.f.SDO_ORDINATES(i_segment-1)||
          txt_separ_y||c.f.SDO_ORDINATES(i_segment)||txt_fin1||
          coul_ext_ou_int||txt_fin2_sgmt) ;
        DBMS_OUTPUT.PUT_LINE(txt_deb_pt||c.f.SDO_ORDINATES(i_segment-1)||
          txt_separ_y||c.f.SDO_ORDINATES(i_segment)||txt_fin1||
          coul_ext_ou_int||txt_fin2_pt) ;
        i_segment := i_segment + 2 ;
      END LOOP ;
    ELSIF etype IN ( 2 , 1003 , 2003 ) AND interpret = 2 THEN
      -- ##### un (ou plusieurs) segment(s) d'arc(s) de cercle #####
      -- affichage du premier point

```



```

IF aff_ler_pt AND i_compo = 1 THEN
  DBMS_OUTPUT.PUT_LINE(txt_deb_pt||c.f.SDO_ORDINATES(starting)||txt_separ_y
    ||c.f.SDO_ORDINATES(starting+1)||txt_fin1||
    coul_ext_ou_int||txt_fin2_pt) ;
END IF ;
-- affichage de chaque arc de cercle avec ses 2 derniers points
i_segment := starting + 5 ; -- on doit tracer un arc de cercle reliant au -
-- le point début de composante à 2 autres pts
WHILE i_segment <= dern_ordinates LOOP
  DBMS_OUTPUT.PUT_LINE(txt_deb_arc||c.f.SDO_ORDINATES(i_segment-5)||
    txt_separ_y||c.f.SDO_ORDINATES(i_segment-4)||
    txt_separ_x||c.f.SDO_ORDINATES(i_segment-3)||
    txt_separ_y||c.f.SDO_ORDINATES(i_segment-2)||
    txt_separ_x||c.f.SDO_ORDINATES(i_segment-1)||
    txt_separ_y||c.f.SDO_ORDINATES(i_segment)||txt_fin1||
    coul_ext_ou_int||txt_fin2_sgmt) ;
  DBMS_OUTPUT.PUT_LINE(txt_deb_pt||c.f.SDO_ORDINATES(i_segment-3)||
    txt_separ_y||c.f.SDO_ORDINATES(i_segment-2)||
    txt_fin1||coul_ext_ou_int||txt_fin2_pt) ;
  DBMS_OUTPUT.PUT_LINE(txt_deb_pt||c.f.SDO_ORDINATES(i_segment-1)||
    txt_separ_y||c.f.SDO_ORDINATES(i_segment)||txt_fin1||
    coul_ext_ou_int||txt_fin2_pt) ;
  i_segment := i_segment + 4 ;
END LOOP ;
ELSIF etype IN ( 1003 , 2003 ) AND interpret = 3 THEN
  -- ##### un rectangle #####
  DBMS_OUTPUT.PUT_LINE(txt_deb_rct||c.f.SDO_ORDINATES(starting)||txt_separ_y||
    c.f.SDO_ORDINATES(starting+1)||txt_separ_x||
    c.f.SDO_ORDINATES(starting+2)||txt_separ_y||
    c.f.SDO_ORDINATES(starting+3)||txt_fin1||
    coul_ext_ou_int||txt_fin2_sgmt) ;
  DBMS_OUTPUT.PUT_LINE(txt_deb_pt||c.f.SDO_ORDINATES(starting)||txt_separ_y||
    c.f.SDO_ORDINATES(starting+1)||txt_fin1||
    coul_ext_ou_int||txt_fin2_pt) ;
  DBMS_OUTPUT.PUT_LINE(txt_deb_pt||c.f.SDO_ORDINATES(starting+2)||
    txt_separ_y||c.f.SDO_ORDINATES(starting+3)||txt_fin1||
    coul_ext_ou_int||txt_fin2_pt) ;
ELSIF etype IN ( 1003 , 2003 ) AND interpret = 4 THEN
  -- ##### un cercle #####
  DBMS_OUTPUT.PUT_LINE(txt_deb_crcl||c.f.SDO_ORDINATES(starting)||txt_separ_y
    ||c.f.SDO_ORDINATES(starting+1)||txt_separ_x||
    c.f.SDO_ORDINATES(starting+2)||txt_separ_y||
    c.f.SDO_ORDINATES(starting+3)||txt_separ_x||
    c.f.SDO_ORDINATES(starting+4)||txt_separ_y||
    c.f.SDO_ORDINATES(starting+5)||txt_fin1||
    coul_ext_ou_int||txt_fin2_sgmt) ;
  DBMS_OUTPUT.PUT_LINE(txt_deb_pt||c.f.SDO_ORDINATES(starting)||txt_separ_y
    ||c.f.SDO_ORDINATES(starting+1)||txt_fin1||
    coul_ext_ou_int||txt_fin2_pt) ;
  DBMS_OUTPUT.PUT_LINE(txt_deb_pt||c.f.SDO_ORDINATES(starting+2)||txt_separ_y
    ||c.f.SDO_ORDINATES(starting+3)||txt_fin1||
    coul_ext_ou_int||txt_fin2_pt) ;
  DBMS_OUTPUT.PUT_LINE(txt_deb_pt||c.f.SDO_ORDINATES(starting+4)||txt_separ_y
    ||c.f.SDO_ORDINATES(starting+5)||txt_fin1||
    coul_ext_ou_int||txt_fin2_pt) ;
ELSE
  NULL ;
END IF ;
-- positionnement sur le triplet suivant c.-à-d. sur la composante suivante
IF i_compo < nb_compo THEN
  i_triplet := i_triplet + 1 ;
  starting := c.f.SDO_ELEM_INFO(3*i_triplet-2) ;
  etype := c.f.SDO_ELEM_INFO(3*i_triplet-1) ;
  interpret := c.f.SDO_ELEM_INFO(3*i_triplet) ;
END IF ;
END LOOP ;
ELSE
  NULL ;
END IF ;
-- triplet suivant
i_triplet := i_triplet + 1 ;
END LOOP ;

```

```

        END IF ;
    END IF;
END IF;
END LOOP ;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Erreur non prévue ; SQLCODE = ' || SQLCODE) ;
END ;

```

Balise Canvas d'HTML5 pré-programmée en JavaScript où insérer les instructions créées

```

<!DOCTYPE HTML>
<html>
  <head>
  </head>
  <body>
    <canvas id="Spatial2Canvas" width="1000" height="300">
      Votre navigateur n'accepte pas la balise canvas.
    </canvas>
    <script>
      // *****
      // affiche des figures géométriques en deux dimensions
      // *****
      window.onload = function(){
        var canvas = document.getElementById("Spatial2Canvas");
        var ctx = canvas.getContext("2d");
        // variables communes
        var borne_inf_x; // borne inférieure 1re coordonnée issue métadonnées attribut spatial
        var borne_inf_y; // borne inférieure 2de coordonnée issue métadonnées attribut spatial
        var borne_sup_x; // borne supérieure 1re coordonnée issue métadonnées attribut spatial
        var borne_sup_y; // borne supérieure 2de coordonnée issue métadonnées attribut spatial
        var zoom; // zoom pour passer des dimensions issues des métadonnées de l'attribut
                    // spatial aux dimensions de la fenêtre graphique
        var couleur_ext = "black"; // couleur de l'objet ou de l'extérieur
        var couleur_int = "blue"; // couleur de l'intérieur
        var couleur_quadrill = "grey"; // couleur des traits du quadrillage
        var couleur_quadrill_zero = "red"; // couleur des abscisses et ordonnées quadrillage
        var couleur_texte = "green"; // couleur du texte
        var police_texte = "12pt Arial"; // police du texte
        var epaiss_segment = 2; // épaisseur d'un segment (droite ou arc de cercle)
        var epaiss_point = 3; // épaisseur d'un point
        var epaiss_quadrill = 1; // épaisseur des traits du quadrillage
        var espacemt_quadrill = 1; // espacement entre deux traits du quadrillage
        // instructions créées à partir des données d'Oracle Spatial
        borne_inf_x = 0; // -2 pour intégrer le palmier
        borne_inf_y = 0; // -1 pour intégrer le palmier
        borne_sup_x = 20;
        borne_sup_y = 10;
        zoom = calcul_zoom(canvas,borne_inf_x,borne_inf_y,borne_sup_x,borne_sup_y);
        affiche_quadrillage(canvas,ctx,borne_inf_x,borne_inf_y,borne_sup_x,borne_sup_y,zoom,
                           couleur_quadrill,couleur_quadrill_zero,epaiss_quadrill,espacemt_quadrill);
        // 2 : hauteur grande pyramide de Gizeh et angle droit avec sol
        affiche_texte(ctx,conversion_repere_x(canvas,borne_inf_x,9,zoom),
                     conversion_repere_y(canvas,borne_inf_y,0.25,zoom),"2",couleur_texte,
                     police_texte);
        affiche_point(ctx,conversion_repere_x(canvas,borne_inf_x,9,zoom),
                     conversion_repere_y(canvas,borne_inf_y,6,zoom),couleur_ext,epaiss_point);
        affiche_droite(ctx,conversion_repere_x(canvas,borne_inf_x,9,zoom),
                      conversion_repere_y(canvas,borne_inf_y,6,zoom),
                      conversion_repere_x(canvas,borne_inf_x,9,zoom),
                      conversion_repere_y(canvas,borne_inf_y,1,zoom),couleur_ext,epaiss_segment);
        affiche_point(ctx,conversion_repere_x(canvas,borne_inf_x,9,zoom),
                     conversion_repere_y(canvas,borne_inf_y,1,zoom),couleur_ext,epaiss_point);
        affiche_droite(ctx,conversion_repere_x(canvas,borne_inf_x,9,zoom),
                      conversion_repere_y(canvas,borne_inf_y,1,zoom),
                      conversion_repere_x(canvas,borne_inf_x,9.5,zoom),
                      conversion_repere_y(canvas,borne_inf_y,1,zoom),couleur_ext,epaiss_segment);
        affiche_point(ctx,conversion_repere_x(canvas,borne_inf_x,9.5,zoom),
                     conversion_repere_y(canvas,borne_inf_y,1,zoom),couleur_ext,epaiss_point);
        affiche_droite(ctx,conversion_repere_x(canvas,borne_inf_x,9.5,zoom),
                      conversion_repere_y(canvas,borne_inf_y,1,zoom),
                      conversion_repere_x(canvas,borne_inf_x,9.5,zoom),
                      conversion_repere_y(canvas,borne_inf_y,1,zoom),
                      conversion_repere_x(canvas,borne_inf_x,9.5,zoom),

```

```

conversion_repere_y(canvas, borne_inf_y, 1.5, zoom), couleur_ext, epaiss_segment);
affiche_point(ctx, conversion_repere_x(canvas, borne_inf_x, 9.5, zoom),
              conversion_repere_y(canvas, borne_inf_y, 1.5, zoom), couleur_ext, epaiss_point);
affiche_droite(ctx, conversion_repere_x(canvas, borne_inf_x, 9.5, zoom),
               conversion_repere_y(canvas, borne_inf_y, 1.5, zoom),
               conversion_repere_x(canvas, borne_inf_x, 9, zoom),
               conversion_repere_y(canvas, borne_inf_y, 1.5, zoom), couleur_ext, epaiss_segment);
affiche_point(ctx, conversion_repere_x(canvas, borne_inf_x, 9, zoom),
              conversion_repere_y(canvas, borne_inf_y, 1.5, zoom), couleur_ext, epaiss_point);
// 3 : bâton d'un mètre utilisé par Thalès [de Milet]
affiche_texte(ctx, conversion_repere_x(canvas, borne_inf_x, 17, zoom),
              conversion_repere_y(canvas, borne_inf_y, 0.25, zoom), "3", couleur_texte,
              police_texte);
affiche_point(ctx, conversion_repere_x(canvas, borne_inf_x, 17, zoom),
              conversion_repere_y(canvas, borne_inf_y, 1, zoom), couleur_ext, epaiss_point);
affiche_droite(ctx, conversion_repere_x(canvas, borne_inf_x, 17, zoom),
               conversion_repere_y(canvas, borne_inf_y, 1, zoom),
               conversion_repere_x(canvas, borne_inf_x, 17, zoom),
               conversion_repere_y(canvas, borne_inf_y, 2, zoom), couleur_ext, epaiss_segment);
affiche_point(ctx, conversion_repere_x(canvas, borne_inf_x, 17, zoom),
              conversion_repere_y(canvas, borne_inf_y, 2, zoom), couleur_ext, epaiss_point);
// 4 : grande pyramide de Gizeh (c.-à-d. la pyramide de Khéops)
affiche_texte(ctx, conversion_repere_x(canvas, borne_inf_x, 5, zoom),
              conversion_repere_y(canvas, borne_inf_y, 0.25, zoom), "4", couleur_texte,
              police_texte);
affiche_droite(ctx, conversion_repere_x(canvas, borne_inf_x, 5, zoom),
               conversion_repere_y(canvas, borne_inf_y, 1, zoom),
               conversion_repere_x(canvas, borne_inf_x, 13, zoom),
               conversion_repere_y(canvas, borne_inf_y, 1, zoom), couleur_ext, epaiss_segment);
affiche_point(ctx, conversion_repere_x(canvas, borne_inf_x, 13, zoom),
              conversion_repere_y(canvas, borne_inf_y, 1, zoom), couleur_ext, epaiss_point);
affiche_droite(ctx, conversion_repere_x(canvas, borne_inf_x, 13, zoom),
               conversion_repere_y(canvas, borne_inf_y, 1, zoom),
               conversion_repere_x(canvas, borne_inf_x, 9, zoom),
               conversion_repere_y(canvas, borne_inf_y, 6, zoom), couleur_ext, epaiss_segment);
affiche_point(ctx, conversion_repere_x(canvas, borne_inf_x, 9, zoom),
              conversion_repere_y(canvas, borne_inf_y, 6, zoom), couleur_ext, epaiss_point);
affiche_droite(ctx, conversion_repere_x(canvas, borne_inf_x, 9, zoom),
               conversion_repere_y(canvas, borne_inf_y, 6, zoom),
               conversion_repere_x(canvas, borne_inf_x, 5, zoom),
               conversion_repere_y(canvas, borne_inf_y, 1, zoom), couleur_ext, epaiss_segment);
affiche_point(ctx, conversion_repere_x(canvas, borne_inf_x, 5, zoom),
              conversion_repere_y(canvas, borne_inf_y, 1, zoom), couleur_ext, epaiss_point);
// 5 : soleil
affiche_texte(ctx, conversion_repere_x(canvas, borne_inf_x, 0.25, zoom),
              conversion_repere_y(canvas, borne_inf_y, 8, zoom), "5", couleur_texte, police_texte);
affiche_cercle(ctx, conversion_repere_x(canvas, borne_inf_x, 3, zoom),
               conversion_repere_y(canvas, borne_inf_y, 8, zoom),
               conversion_repere_x(canvas, borne_inf_x, 2, zoom),
               conversion_repere_y(canvas, borne_inf_y, 9, zoom),
               conversion_repere_x(canvas, borne_inf_x, 1, zoom),
               conversion_repere_y(canvas, borne_inf_y, 8, zoom), couleur_ext, epaiss_segment);
affiche_point(ctx, conversion_repere_x(canvas, borne_inf_x, 3, zoom),
              conversion_repere_y(canvas, borne_inf_y, 8, zoom), couleur_ext, epaiss_point);
affiche_point(ctx, conversion_repere_x(canvas, borne_inf_x, 2, zoom),
              conversion_repere_y(canvas, borne_inf_y, 9, zoom), couleur_ext, epaiss_point);
affiche_point(ctx, conversion_repere_x(canvas, borne_inf_x, 1, zoom),
              conversion_repere_y(canvas, borne_inf_y, 8, zoom), couleur_ext, epaiss_point);
// 7 : ombre de la grande pyramide de Gizeh
affiche_texte(ctx, conversion_repere_x(canvas, borne_inf_x, 13, zoom),
              conversion_repere_y(canvas, borne_inf_y, 0.25, zoom), "7", couleur_texte,
              police_texte);
affiche_droite(ctx, conversion_repere_x(canvas, borne_inf_x, 9, zoom),
               conversion_repere_y(canvas, borne_inf_y, 6, zoom),
               conversion_repere_x(canvas, borne_inf_x, 13, zoom),
               conversion_repere_y(canvas, borne_inf_y, 1, zoom), couleur_ext, epaiss_segment);
affiche_point(ctx, conversion_repere_x(canvas, borne_inf_x, 13, zoom),
              conversion_repere_y(canvas, borne_inf_y, 1, zoom), couleur_ext, epaiss_point);
affiche_droite(ctx, conversion_repere_x(canvas, borne_inf_x, 13, zoom),
               conversion_repere_y(canvas, borne_inf_y, 1, zoom),
               conversion_repere_x(canvas, borne_inf_x, 19, zoom),
               conversion_repere_y(canvas, borne_inf_y, 1, zoom), couleur_ext, epaiss_segment);

```



```

conversion_repere_y(canvas,borne_inf_y,5,zoom),
conversion_repere_x(canvas,borne_inf_x,-1,zoom),
conversion_repere_y(canvas,borne_inf_y,5,zoom),
conversion_repere_x(canvas,borne_inf_x,-1,zoom),
conversion_repere_y(canvas,borne_inf_y,3,zoom),couleur_ext,epaiss_segment);
affiche_point(ctx,conversion_repere_x(canvas,borne_inf_x,-1,zoom),
conversion_repere_y(canvas,borne_inf_y,5,zoom),couleur_ext,epaiss_point);
affiche_point(ctx,conversion_repere_x(canvas,borne_inf_x,-1,zoom),
conversion_repere_y(canvas,borne_inf_y,3,zoom),couleur_ext,epaiss_point);
affiche_arc_cercle(ctx,conversion_repere_x(canvas,borne_inf_x,-1,zoom),
conversion_repere_y(canvas,borne_inf_y,3,zoom),
conversion_repere_x(canvas,borne_inf_x,-1,zoom),
conversion_repere_y(canvas,borne_inf_y,4,zoom),
conversion_repere_x(canvas,borne_inf_x,1,zoom),
conversion_repere_y(canvas,borne_inf_y,4.5,zoom),
couleur_ext,epaiss_segment);
affiche_point(ctx,conversion_repere_x(canvas,borne_inf_x,-1,zoom),
conversion_repere_y(canvas,borne_inf_y,4,zoom),couleur_ext,epaiss_point);
affiche_point(ctx,conversion_repere_x(canvas,borne_inf_x,1,zoom),
conversion_repere_y(canvas,borne_inf_y,4.5,zoom),couleur_ext,epaiss_point);
affiche_arc_cercle(ctx,conversion_repere_x(canvas,borne_inf_x,1,zoom),
conversion_repere_y(canvas,borne_inf_y,4.5,zoom),
conversion_repere_x(canvas,borne_inf_x,0,zoom),
conversion_repere_y(canvas,borne_inf_y,4,zoom),
conversion_repere_x(canvas,borne_inf_x,0,zoom),
conversion_repere_y(canvas,borne_inf_y,2,zoom),couleur_ext,epaiss_segment);
affiche_point(ctx,conversion_repere_x(canvas,borne_inf_x,0,zoom),
conversion_repere_y(canvas,borne_inf_y,4,zoom),couleur_ext,epaiss_point);
affiche_point(ctx,conversion_repere_x(canvas,borne_inf_x,0,zoom),
conversion_repere_y(canvas,borne_inf_y,2,zoom),couleur_ext,epaiss_point);
affiche_arc_cercle(ctx,conversion_repere_x(canvas,borne_inf_x,0,zoom),
conversion_repere_y(canvas,borne_inf_y,2,zoom),
conversion_repere_x(canvas,borne_inf_x,0,zoom),
conversion_repere_y(canvas,borne_inf_y,3,zoom),
conversion_repere_x(canvas,borne_inf_x,1,zoom),
conversion_repere_y(canvas,borne_inf_y,4,zoom),couleur_ext,epaiss_segment);
affiche_point(ctx,conversion_repere_x(canvas,borne_inf_x,0,zoom),
conversion_repere_y(canvas,borne_inf_y,3,zoom),couleur_ext,epaiss_point);
affiche_point(ctx,conversion_repere_x(canvas,borne_inf_x,1,zoom),
conversion_repere_y(canvas,borne_inf_y,4,zoom),couleur_ext,epaiss_point);
affiche_rectangle(ctx,conversion_repere_x(canvas,borne_inf_x,1,zoom),
conversion_repere_y(canvas,borne_inf_y,1,zoom),
conversion_repere_x(canvas,borne_inf_x,2,zoom),
conversion_repere_y(canvas,borne_inf_y,2,zoom),
couleur_int,epaiss_segment);
affiche_point(ctx,conversion_repere_x(canvas,borne_inf_x,1,zoom),
conversion_repere_y(canvas,borne_inf_y,1,zoom),couleur_int,epaiss_point);
affiche_point(ctx,conversion_repere_x(canvas,borne_inf_x,2,zoom),
conversion_repere_y(canvas,borne_inf_y,2,zoom),couleur_int,epaiss_point);
*/
}
// *****
// affiche un quadrillage
// *****
function affiche_quadrillage(canvas,ctx,borne_inf_x,borne_inf_y,borne_sup_x,borne_sup_y,
zoom,couleur_quadrill,couleur_quadrill_zero,
epaiss_quadrill,espacemt_quadrill) {
// fenêtre, contexte, 4 bornes issues des métadonnées de l'attribut spatial,
// 2 couleurs, épaisseur, espacement
var i;
for (i = borne_inf_x; i <= borne_sup_x; i += espacemt_quadrill)
affiche_droite(ctx,conversion_repere_x(canvas,borne_inf_x,i,zoom),
conversion_repere_y(canvas,borne_inf_y,borne_inf_y,zoom),
conversion_repere_x(canvas,borne_inf_x,i,zoom),
conversion_repere_y(canvas,borne_inf_y,borne_sup_y,zoom),
i==0?couleur_quadrill_zero:couleur_quadrill,epaiss_quadrill);
for (i = borne_inf_y; i <= borne_sup_y; i += espacemt_quadrill)
affiche_droite(ctx,conversion_repere_x(canvas,borne_inf_x,borne_inf_x,zoom),
conversion_repere_y(canvas,borne_inf_y,i,zoom),
conversion_repere_x(canvas,borne_inf_x,borne_sup_x,zoom),
conversion_repere_y(canvas,borne_inf_y,i,zoom),
i==0?couleur_quadrill_zero:couleur_quadrill,epaiss_quadrill);
}

```

```

}
// *****
// affiche un texte
// *****
function affiche_texte(canvas,ctx,borne_inf_x,borne_inf_y,zoom,pt_x,pt_y,texte,couleur,
                    police) {
    // fenêtre, contexte, 2 bornes inférieures issues des métadonnées de l'attribut
    // spatial, zoom, coordonnées, texte, couleur, police
    ctx.fillStyle = couleur;
    ctx.font = police;
    ctx.fillText(texte,conversion_repere_x(canvas,borne_inf_x,pt_x,zoom),
                conversion_repere_y(canvas,borne_inf_y,pt_y,zoom));
}
// *****
// affiche un point
// *****
function affiche_point(ctx,pt_x,pt_y,couleur,epaisseur) {
    // contexte, coordonnées du point, couleur, épaisseur
    ctx.beginPath();
    ctx.moveTo(pt_x,pt_y);
    ctx.arc(pt_x,pt_y,epaisseur,0,2*Math.PI,true); // cercle
    ctx.lineWidth = epaisseur;
    ctx.strokeStyle = couleur;
    ctx.fillStyle = couleur;
    ctx.fill(); // remplissage
    ctx.stroke();
    ctx.closePath();
}
// *****
// affiche une droite (entre 2 points)
// *****
function affiche_droite(ctx,pt1_x,pt1_y,pt2_x,pt2_y,couleur,epaisseur) {
    // contexte, coordonnées des 2 points, couleur, épaisseur
    ctx.beginPath();
    ctx.moveTo(pt1_x,pt1_y); // coordonnées de début
    ctx.lineTo(pt2_x,pt2_y); // coordonnées de fin
    ctx.lineWidth = epaisseur;
    ctx.strokeStyle = couleur;
    ctx.lineCap = "butt"; // extrémités : "butt" (carré sans dépasser)
    ctx.stroke();
    ctx.closePath();
}
// *****
// affiche un rectangle (entre 2 points)
// *****
function affiche_rectangle(ctx,pt1_x,pt1_y,pt2_x,pt2_y,couleur,epaisseur) {
    // contexte, coordonnées des 2 points, couleur, épaisseur
    affiche_droite(ctx,pt1_x,pt1_y,pt1_x,pt2_y,couleur,epaisseur);
    affiche_droite(ctx,pt1_x,pt2_y,pt2_x,pt2_y,couleur,epaisseur);
    affiche_droite(ctx,pt2_x,pt2_y,pt2_x,pt1_y,couleur,epaisseur);
    affiche_droite(ctx,pt2_x,pt1_y,pt1_x,pt1_y,couleur,epaisseur);
}
// *****
// affiche un cercle (passant par 3 points)
// *****
function affiche_cercle(ctx,pt1_x,pt1_y,pt2_x,pt2_y,pt3_x,pt3_y,couleur,epaisseur) {
    // contexte, coordonnées des 3 points sur le cercle, couleur, épaisseur
    ctx.beginPath();
    var centre_x = centre_x_cercle_3points(pt1_x,pt1_y,pt2_x,pt2_y,pt3_x,pt3_y);
    var centre_y = centre_y_cercle_3points(pt1_x,pt1_y,pt2_x,pt2_y,pt3_x,pt3_y);
    ctx.arc(
        centre_x,centre_y, // coordonnées du centre
        rayon_cercle_point_centre(pt1_x,pt1_y,centre_x,centre_y), // rayon
        0,2*Math.PI, // un tour complet
        true); // ou false !
    ctx.lineWidth = epaisseur;
    ctx.strokeStyle = couleur;
    ctx.stroke();
    ctx.closePath();
}
// *****
// affiche un arc de cercle (passant par 3 points)

```

```

// *****
function affiche_arc_cercle(ctx,pt1_x,pt1_y,pt2_x,pt2_y,pt3_x,pt3_y,couleur,epaisseur) {
    // contexte, coordonnées des 3 points sur le cercle, couleur, épaisseur
    ctx.beginPath();
    var centre_x = centre_x_cercle_3points(pt1_x,pt1_y,pt2_x,pt2_y,pt3_x,pt3_y);
    var centre_y = centre_y_cercle_3points(pt1_x,pt1_y,pt2_x,pt2_y,pt3_x,pt3_y);
    var angle1 = angle_point_cercle(pt1_x-centre_x,pt1_y-centre_y); // angle 1er point
    var angle2 = angle_point_cercle(pt2_x-centre_x,pt2_y-centre_y); // angle 2e point
    var angle3 = angle_point_cercle(pt3_x-centre_x,pt3_y-centre_y); // angle 3e point
    ctx.arc(
        centre_x,centre_y, // coordonnées du centre
        rayon_cercle_point_centre(pt1_x,pt1_y,centre_x,centre_y), // rayon du cercle
        angle1,angle3, // angles de début et de fin
        ((angle1 < angle2 && angle2 < angle3) || (angle2 < angle3 && angle3 < angle1) ||
        (angle3 < angle1 && angle1 < angle2)) ? false : true);
    // sens rotation : sens trigonométrique = anti-horaire (true) sens horaire (false)
    ctx.lineWidth = epaisseur;
    ctx.strokeStyle = couleur;
    ctx.stroke();
    ctx.closePath();
}
// *****
// première et seconde coordonnée du centre d'un cercle donné par 3 points
// *****
function centre_x_cercle_3points(pt1_x,pt1_y,pt2_x,pt2_y,pt3_x,pt3_y) {
    // coordonnées des 3 points sur le cercle
    return (1/2) * ( pt3_y*Math.pow(pt2_x,2) - pt1_y*Math.pow(pt2_x,2) +
        Math.pow(pt3_y,2)*pt1_y - pt2_y*Math.pow(pt3_y,2) +
        Math.pow(pt1_x,2)*pt2_y + pt2_y*Math.pow(pt1_y,2) -
        Math.pow(pt1_x,2)*pt3_y + pt1_y*Math.pow(pt3_x,2) +
        Math.pow(pt2_y,2)*pt3_y - Math.pow(pt2_y,2)*pt1_y -
        pt3_y*Math.pow(pt1_y,2) - Math.pow(pt3_x,2)*pt2_y )
        / ( -pt3_x*pt2_y + pt1_x*pt2_y - pt1_x*pt3_y + pt3_y*pt2_x -
        pt1_y*pt2_x + pt1_y*pt3_x );
}
function centre_y_cercle_3points(pt1_x,pt1_y,pt2_x,pt2_y,pt3_x,pt3_y) {
    // coordonnées des 3 points sur le cercle
    return -(1/2) * ( pt3_x*Math.pow(pt2_x,2) + pt3_x*Math.pow(pt2_y,2) -
        pt1_x*Math.pow(pt2_x,2) - pt1_x*Math.pow(pt2_y,2) +
        pt1_x*Math.pow(pt3_x,2) + pt1_x*Math.pow(pt3_y,2) -
        Math.pow(pt3_x,2)*pt2_x - Math.pow(pt3_y,2)*pt2_x +
        Math.pow(pt1_x,2)*pt2_x - Math.pow(pt1_x,2)*pt3_x +
        Math.pow(pt1_y,2)*pt2_x - Math.pow(pt1_y,2)*pt3_x )
        / ( -pt3_x*pt2_y + pt1_x*pt2_y - pt1_x*pt3_y + pt3_y*pt2_x -
        pt1_y*pt2_x + pt1_y*pt3_x );
}
// *****
// rayon d'un cercle donné par 1 point sur le cercle et son centre
// *****
function rayon_cercle_point_centre(pt_x,pt_y,centre_x,centre_y) {
    // coordonnées du point sur le cercle et du centre du cercle
    return Math.pow(Math.pow(pt_x-centre_x,2)+Math.pow(pt_y-centre_y,2),0.5);
}
// *****
// angle d'un point d'un cercle
// cf. : http://fr.wikipedia.org/wiki/Coordonnées\_polaires
// *****
function angle_point_cercle(x,y) {
    // coordonnées relativement à (0,0)
    if (x == 0 && y == 0)
        return 0; // en fait, cas d'erreur
    else if (x == 0 && y > 0)
        return Math.PI/2;
    else if (x == 0 && y < 0)
        return Math.PI*3/2;
    else if (x > 0 && y >= 0)
        return Math.atan(y/x);
    else if (x > 0 && y < 0)
        return Math.atan(y/x)+2*Math.PI;
    else // (x < 0)
        return Math.atan(y/x)+Math.PI;
}

```

```

// *****
// conversion de repère pour la première coordonnée
// *****
function conversion_repere_x(canvas,borne_inf_x,x,zoom) {
    // fenêtre, borne inférieure de la première coordonnée issue des métadonnées de
    // l'attribut spatial, première coordonnée, zoom
    return (x-borne_inf_x)*zoom;
}
// *****
// conversion de repère pour la seconde coordonnée
// *****
function conversion_repere_y(canvas,borne_inf_y,y,zoom) {
    // fenêtre, borne inférieure de la seconde coordonnée issue des métadonnées de
    // l'attribut spatial, seconde coordonnée, zoom
    return canvas.height-(y-borne_inf_y)*zoom;
}
// *****
// calcul du zoom permettant de dessiner la figure géométrique la plus grande
// possible dans la fenêtre graphique
// *****
function calcul_zoom(canvas,borne_inf_x,borne_inf_y,borne_sup_x,borne_sup_y) {
    // fenêtre, bornes inférieures et supérieures des deux coordonnées issues des
    // métadonnées de l'attribut spatial
    return Math.min(canvas.width/(borne_sup_x-borne_inf_x),
        canvas.height/(borne_sup_y-borne_inf_y));
}
}
</script>
</body>
</html>

```

Visualisation des figures géométriques avec un navigateur Web (supportant la balise Canvas)

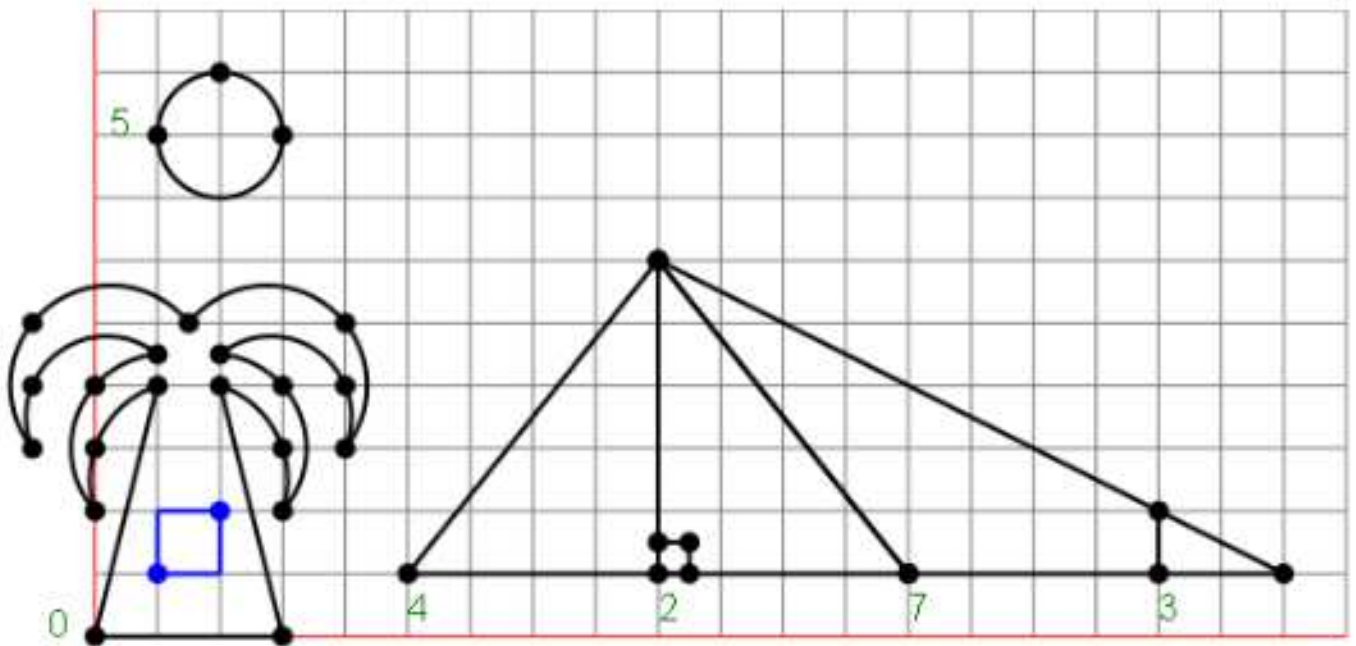
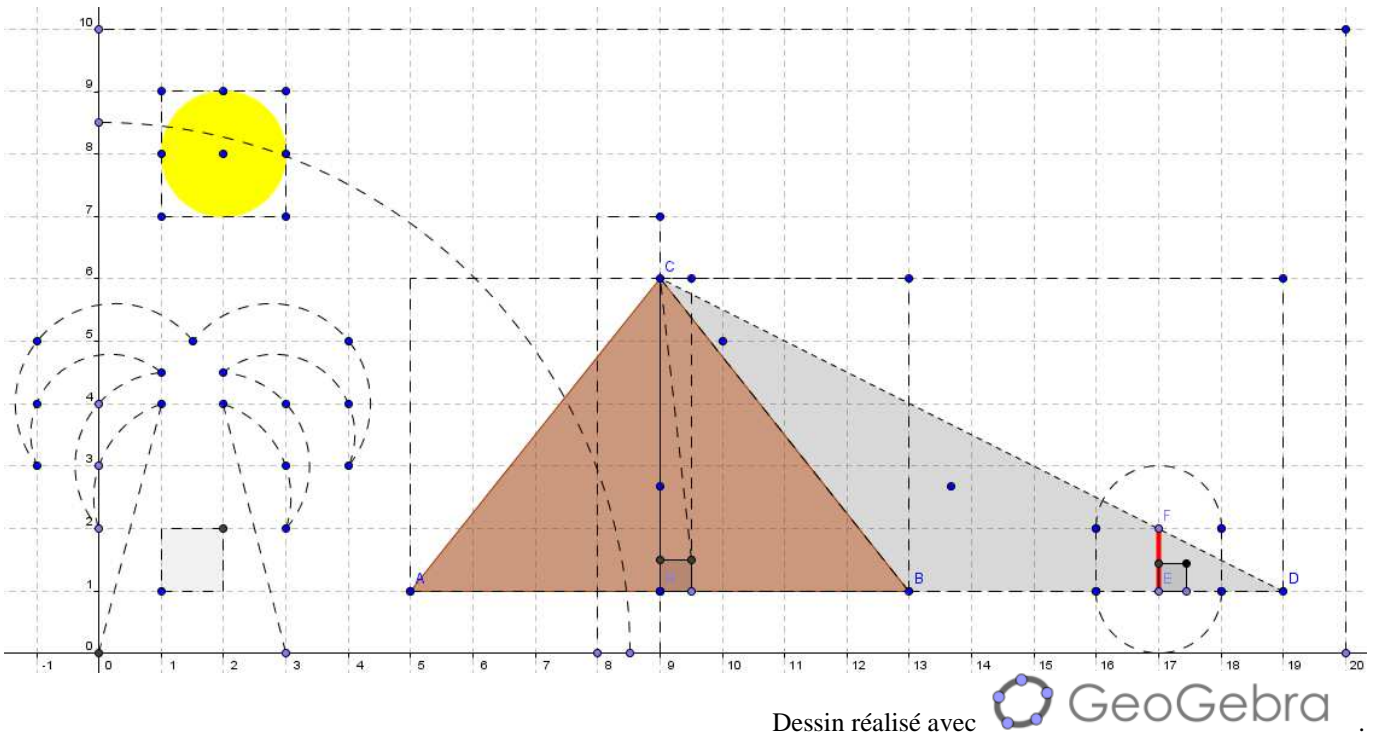


Figure affichée par un navigateur Web pour la balise Canvas d'

codée en JavaScript et où ont été insérées les instructions créées par le code PL/SQL.



N. B. (cf. <http://fr.wikipedia.org/wiki/Thalès>) : Plutarque, dans *Le Banquet des Sept Sages*, §2, écrit « Ainsi, vous, Thalès [de Milet], le roi d'Egypte [le pharaon Amasis] vous admire beaucoup, et, entre autres choses, il a été, au-delà de ce qu'on peut dire, ravi de la manière dont vous avez mesuré [la hauteur de] la pyramide [la grande pyramide de Gizeh (c.-à-d. la pyramide de Khéops)] sans le moindre embarras et sans avoir eu besoin d'aucun instrument. Après avoir dressé votre bâton à l'extrémité de l'ombre que projetait la pyramide, vous construisîtes deux triangles par la tangence d'un rayon, et vous démontrâtes qu'il y avait la même proportion entre la hauteur du bâton et la hauteur de la pyramide qu'entre la longueur des deux ombres. ». On a $AH = HB = 116$ m, $BD = 73$ m, $ED = 1,3$ m et $EF = 1$ m ; le théorème de Thalès nous dit que $ED/HD = EF/HC$ et on obtient $HC = EF \cdot HD / ED = 1 \cdot (116 + 73) / 1,3 \approx 145$ m. En fait, sa hauteur initiale était de $146,58$ m et est aujourd'hui de 137 m (cf. http://fr.wikipedia.org/wiki/Pyramide_de_Khéops).