# Corrigés des exercices
# *SQL pour Oracle*

## Chapitre 1

### Création des tables (creParc.sql)

```
CREATE TABLE Segment
      (indIP      VARCHAR2(11),
       nomSegment VARCHAR2(20) CONSTRAINT nn_nomSegment NOT NULL,
       etage NUMBER(2),
       CONSTRAINT pk_Segment PRIMARY KEY (indIP));

CREATE TABLE Salle
      (nSalle     VARCHAR2(7),
       nomSalle   VARCHAR2(20) CONSTRAINT nn_nomSalle NOT NULL,
       nbPoste    NUMBER(2), indIP      VARCHAR2(11),
       CONSTRAINT pk_salle PRIMARY KEY (nSalle));

CREATE TABLE Poste
      (nPoste     VARCHAR2(7),
       nomPoste   VARCHAR2(20) CONSTRAINT nn_nomPoste NOT NULL,
       indIP      VARCHAR2(11), ad         VARCHAR2(3),
       typePoste  VARCHAR2(9), nSalle     VARCHAR2(7),
       CONSTRAINT pk_Poste PRIMARY KEY (nPoste),
       CONSTRAINT ck_ad    CHECK (ad BETWEEN '0' AND '255'));

CREATE TABLE Logiciel
      (nLog       VARCHAR2(5),
       nomLog     VARCHAR2(20) CONSTRAINT nn_nomLog NOT NULL,
       dateAch    DATE, version    VARCHAR2(7),
       typeLog    VARCHAR2(9), prix       NUMBER(6,2),
       CONSTRAINT pk_Logiciel PRIMARY KEY (nLog),
       CONSTRAINT ck_prix     CHECK (prix >= 0));

CREATE TABLE Installer
      (nPoste     VARCHAR2(7), nLog       VARCHAR2(5),
       numIns     NUMBER(5), dateIns    DATE DEFAULT SYSDATE,
       delai      INTERVAL DAY(5) TO SECOND(2),
       CONSTRAINT pk_Installer PRIMARY KEY(nPoste,nLog));

CREATE TABLE Types
      (typeLP   VARCHAR2(9), nomType VARCHAR2(20),
       CONSTRAINT pk_types PRIMARY KEY(typeLP));
```





### Structure des tables (descParc.sql)

```
DESC Segment;
DESC Salle;
DESC Poste;
DESC Logiciel;
DESC Installer;
DESC Types;
```

### Destruction des tables (dropParc.sql)

```
DROP TABLE  Types;
DROP TABLE  Installer;
DROP TABLE  Logiciel;
DROP TABLE  Poste;
DROP TABLE  Salle;
DROP TABLE  Segment;
```

## Chapitre 2

### Insertion de données

```
INSERT INTO Segment VALUES ('130.120.80','Brin RDC',NULL);
INSERT INTO Segment VALUES ('130.120.81','Brin 1er  étage',NULL);
INSERT INTO Segment VALUES ('130.120.82','Brin 2ème étage',NULL);

INSERT INTO Salle VALUES ('s01','Salle 1',3,'130.120.80');
INSERT INTO Salle VALUES ('s02','Salle 2',2,'130.120.80');
INSERT INTO Salle VALUES ('s03','Salle 3',2,'130.120.80');
INSERT INTO Salle VALUES ('s11','Salle 11',2,'130.120.81');
INSERT INTO Salle VALUES ('s12','Salle 12',1,'130.120.81');
INSERT INTO Salle VALUES ('s21','Salle 21',2,'130.120.82');
INSERT INTO Salle VALUES ('s22','Salle 22',0,'130.120.83');
INSERT INTO Salle VALUES ('s23','Salle 23',0,'130.120.83');

INSERT INTO Poste VALUES ('p1','Poste 1','130.120.80','01','TX','s01');
INSERT INTO Poste VALUES ('p2','Poste 2','130.120.80','02','UNIX','s01');
INSERT INTO Poste VALUES ('p3','Poste 3','130.120.80','03','TX','s01');
INSERT INTO Poste VALUES ('p4','Poste 4','130.120.80','04','PCWS','s02');
INSERT INTO Poste VALUES ('p5','Poste 5','130.120.80','05','PCWS','s02');
INSERT INTO Poste VALUES ('p6','Poste 6','130.120.80','06','UNIX','s03');
INSERT INTO Poste VALUES ('p7','Poste 7','130.120.80','07','TX','s03');
INSERT INTO Poste VALUES ('p8','Poste 8','130.120.81','01','UNIX','s11');
INSERT INTO Poste VALUES ('p9','Poste 9','130.120.81','02','TX','s11');
INSERT INTO Poste VALUES ('p10','Poste 10','130.120.81','03','UNIX','s12');
INSERT INTO Poste VALUES ('p11','Poste 11','130.120.82','01','PCNT','s21');
INSERT INTO Poste VALUES ('p12','Poste 12','130.120.82','02','PCWS','s21');
```

```
INSERT INTO Logiciel VALUES ('log1','Oracle 6','13-05-1995','6.2','UNIX',3000);
INSERT INTO Logiciel VALUES ('log2','Oracle 8','15-09-1999','8i','UNIX',5600);
INSERT INTO Logiciel VALUES ('log3','SQL Server','12-04-1998','7','PCNT',3000);
INSERT INTO Logiciel VALUES ('log4','Front Page','03-06-1997','5','PCWS',500);
INSERT INTO Logiciel VALUES ('log5','WinDev','12-05-1997','5','PCWS',750);
INSERT INTO Logiciel VALUES ('log6','SQL*Net',NULL,'2.0','UNIX',500);
INSERT INTO Logiciel VALUES ('log7','I. I. S.','12-04-2002','2','PCNT',900);
INSERT INTO Logiciel  VALUES ('log8','DreamWeaver','21-09-
2003','2.0','BeOS',1400);

INSERT INTO Types VALUES ('TX',  'Terminal X-Window');
INSERT INTO Types VALUES ('UNIX','Système Unix');
INSERT INTO Types VALUES ('PCNT','PC Windows  NT');
INSERT INTO Types VALUES ('PCWS','PC Windows');
INSERT INTO Types VALUES ('NC',  'Network Computer');
```

## Gestion d'une séquence

```
CREATE SEQUENCE sequenceIns
  INCREMENT BY 1 START WITH 1
  MAXVALUE 10000 NOCYCLE;

INSERT INTO Installer VALUES
('p2', 'log1', sequenceIns.NEXTVAL,'15-05-2003',NULL);
INSERT INTO Installer VALUES
('p2', 'log2', sequenceIns.NEXTVAL,'17-09-2003',NULL);
INSERT INTO Installer VALUES
('p4', 'log5', sequenceIns.NEXTVAL,NULL,NULL);
INSERT INTO Installer VALUES
('p6', 'log6', sequenceIns.NEXTVAL,'20-05-2003',NULL);
INSERT INTO Installer VALUES
('p6', 'log1', sequenceIns.NEXTVAL,'20-05-2003',NULL);
INSERT INTO Installer VALUES
('p8', 'log2', sequenceIns.NEXTVAL,'19-05-2003',NULL);
INSERT INTO Installer VALUES
('p8', 'log6', sequenceIns.NEXTVAL,'20-05-2003',NULL);
INSERT INTO Installer VALUES
('p11','log3', sequenceIns.NEXTVAL,'20-04-2003',NULL);
INSERT INTO Installer VALUES
('p12','log4', sequenceIns.NEXTVAL,'20-04-2003',NULL);
INSERT INTO Installer VALUES
('p11','log7', sequenceIns.NEXTVAL,'20-04-2003',NULL);
INSERT INTO Installer VALUES
('p7', 'log7', sequenceIns.NEXTVAL,'01-04-2002',NULL);
```

## Modification de données

```
UPDATE Segment SET etage=0 WHERE indIP = '130.120.80';
UPDATE Segment SET etage=1 WHERE indIP = '130.120.81';
UPDATE Segment SET etage=2 WHERE indIP = '130.120.82';
```





```
UPDATE Logiciel SET prix = prix*0.9 WHERE typeLog = 'PCNT';
```

# Chapitre 3

## Ajout de colonnes

```
ALTER TABLE Segment  ADD (nbSalle NUMBER(2), nbPoste NUMBER(2));
ALTER TABLE Logiciel ADD nbInstall NUMBER(2);
ALTER TABLE Poste    ADD nbLog NUMBER(2);
```

## Modification de colonnes

```
ALTER TABLE Salle   MODIFY nomSalle VARCHAR2(30);

ALTER TABLE Segment MODIFY nomSegment VARCHAR(15);
-- +long 'Brin 2ème étage' : 15 caractères
-- tentative
ALTER TABLE Segment MODIFY nomSegment VARCHAR(14);
```

## Ajout de contraintes

```
ALTER TABLE Poste
  ADD CONSTRAINT fk_Poste_indIP_Segment FOREIGN KEY(indIP)
  REFERENCES Segment(indIP);

ALTER TABLE Poste
  ADD CONSTRAINT fk_Poste_nSalle_Salle FOREIGN KEY(nSalle)
  REFERENCES Salle(nSalle);

ALTER TABLE Poste
  ADD CONSTRAINT fk_Poste_typePoste_Types FOREIGN KEY(typePoste)
  REFERENCES Types(typeLP);

ALTER TABLE Installer
  ADD CONSTRAINT fk_Installer_nPoste_Poste FOREIGN KEY(nPoste)
  REFERENCES Poste(nPoste);

ALTER TABLE Installer
  ADD CONSTRAINT fk_Installer_nLog_Logiciel FOREIGN KEY(nLog)
  REFERENCES Logiciel(nLog);
```

Le script de destruction des tables devient :

```
DROP TABLE  Installer;
DROP TABLE  Logiciel;
DROP TABLE  Poste;
DROP TABLE  Types;
DROP TABLE  Salle;
DROP TABLE  Segment;
```

## Traitements des rejets

```
CREATE TABLE Rejets
  (ligne ROWID, propriétaire VARCHAR2(30),
  nomTable VARCHAR2(30), contrainte VARCHAR2(30));

ALTER TABLE Logiciel
  ADD CONSTRAINT fk_Logiciel_typeLog_Types FOREIGN KEY(typeLog)
  REFERENCES Types(typeLP) EXCEPTIONS INTO Rejets;

ALTER TABLE Salle
  ADD CONSTRAINT fk_Salle_indIP_Segment FOREIGN KEY(indIP)
  REFERENCES Segment(indIP) EXCEPTIONS INTO Rejets;
```

Résolution des rejets en :

- supprimant les enregistrements de la table `Rejets`.

```
DELETE FROM Rejets;
```

- supprimant les enregistrements de la table `Salle` qui ne respectent pas la contrainte.

```
DELETE FROM Salle WHERE indIP NOT IN (SELECT indIP FROM Segment);
```

- ajoutant le type de logiciel (‘BeOS’, ‘Système Be’)

```
INSERT INTO Types VALUES ('BeOS','Système Be');
```

L’ajout des deux contraintes de clé étrangère ne envoie plus d’erreur et la table `Rejets` reste vide.

# Chapitre 4

## Création dynamique de tables

```
CREATE TABLE Softs (nomSoft, Version)
       AS SELECT nomLog, Version FROM Logiciel;

CREATE TABLE PCSeuls (np, nomP, seg, ad, typeP,lieu)
       AS SELECT nPoste, nomPoste, IndIP, ad, typePoste, nSalle
       FROM Poste
       WHERE typePoste = 'PCNT' OR typePoste = 'PCWS';
```

## Requêtes mono-table

```
--1
SELECT nPoste, typePoste FROM Poste WHERE nPoste = 'p8';
--2
SELECT nomLog FROM Logiciel WHERE typeLog = 'UNIX';
--3
SELECT nomPoste, indIP, ad, nSalle
       FROM Poste WHERE typePoste = 'UNIX' OR typePoste = 'PCWS';
--4
```





```
SELECT nomPoste, indIP, ad, nSalle
  FROM Poste WHERE (typePoste = 'UNIX' OR typePoste = 'PCWS')
  AND  indIP = '130.120.80'
  ORDER BY nSalle DESC;
--5
SELECT nLog FROM Installer WHERE nPoste = 'p6';
--6
SELECT nPoste FROM Installer WHERE nLog = 'log1';
--7
SELECT nomPoste, indIP ||'.'|| ad FROM  Poste WHERE typePoste = 'TX';
```

## Fonctions et groupements

```
--8
SELECT nPoste, COUNT(nLog)
  FROM installer GROUP BY (nPoste);
--9
SELECT nSalle, COUNT(nPoste)
  FROM Poste GROUP BY (nSalle) ORDER BY 2;
--10
SELECT nLog, COUNT(nPoste)
  FROM Installer GROUP BY (nLog);
--11
SELECT AVG(prix)
  FROM Logiciel WHERE typeLog = 'UNIX';
--12
SELECT MAX(dateAch) FROM Logiciel;
--13
SELECT nPoste FROM Installer
  GROUP BY nPoste HAVING COUNT(nLog)=2;
--14
SELECT COUNT(*) FROM
      (SELECT nPoste FROM Installer GROUP BY nPoste HAVING COUNT(nLog)=2);
```

## Requêtes multi-tables

### Opérateurs ensemblistes

```
--15
SELECT typeLP FROM Types
  MINUS
SELECT DISTINCT typePoste FROM Poste;
--16
SELECT typeLog FROM Logiciel
  INTERSECT
SELECT typePoste FROM Poste;
--17
SELECT DISTINCT typePoste FROM Poste
  MINUS
```

```
SELECT DISTINCT typeLog FROM Logiciel;
```

### Jointures procédurales

```
--18
SELECT indIP || '.' || ad
  FROM Poste WHERE nPoste IN
      (SELECT nPoste
       FROM   Installer WHERE  nLog = 'log6');
--19
SELECT indIP || '.' || ad
  FROM Poste WHERE nPoste IN
      (SELECT nPoste
       FROM Installer WHERE nLog =
            (SELECT nLog
             FROM   Logiciel WHERE  nomLog = 'Oracle 8'));
--20
SELECT nomSegment
  FROM Segment WHERE indIP IN
      (SELECT   indIP
       FROM     Poste WHERE  typePoste = 'TX'
       GROUP BY indIP HAVING COUNT(*)=3);
--21
SELECT nomSalle
  FROM Salle WHERE nSalle IN
      (SELECT nSalle
       FROM   Poste WHERE nPoste IN
            (SELECT nPoste
             FROM Installer WHERE nLog =
                  (SELECT nLog
                   FROM   Logiciel WHERE  nomLog  = 'Oracle 6')));
--22
SELECT nomLog
  FROM Logiciel WHERE dateAch = (SELECT MAX(dateAch)  FROM Logiciel) ;
```

### Jointures relationnelles

```
--23
SELECT  p.indIP || '.' || p.ad
  FROM  Poste p, Installer i
  WHERE p.nPoste = i.nPoste
  AND   i.nLog   = 'log6';
--24
SELECT  p.indIP || '.' || p.ad
  FROM  Poste p, Installer i, Logiciel l
  WHERE p.nPoste = i.nPoste
  AND   l.nLog   = i.nLog
  AND   l.nomLog = 'Oracle 8';
--25
SELECT   s.nomSegment
  FROM   Segment s, Poste p
```





```
  WHERE  s.indIP     = p.indIP
  AND    p.typePoste = 'TX'
  GROUP  BY s.nomSegment
  HAVING COUNT(*)=3;
--26
SELECT  s.nomSalle
  FROM  Salle s, Poste p, Installer i, Logiciel l
  WHERE s.nSalle  = p.nSalle
  AND   p.nPoste  = i.nPoste
  AND   i.nLog    = l.nLog
  AND   l.nomLog  = 'Oracle 6';
--27
SELECT sg.nomSegment, s.nSalle, p.indIP||'.' || p.ad, l.nomLog, i.dateIns
  FROM  segment sg, Salle s, Poste p, Logiciel l, Installer i
  WHERE s.nSalle = p.nSalle
  AND   s.indIP  = sg.indIP
  AND   p.nPoste = i.nPoste
  AND   i.nLog   = l.nLog
  ORDER BY 1,2,3;
```

### Jointures SQL2

```
--28
SELECT  indIP || '.' || ad
  FROM  Poste NATURAL JOIN Installer
              WHERE nLog   = 'log6';
--29
SELECT indIP || '.' || ad
  FROM Poste NATURAL JOIN Installer
                NATURAL JOIN Logiciel
                WHERE nomLog = 'Oracle 8';
--30
SELECT   nomSegment
  FROM   Segment JOIN Poste USING(indIP)
           WHERE  typePoste = 'TX'
           GROUP  BY nomSegment
           HAVING COUNT(*)=3;
--31
SELECT  nomSalle
  FROM  Salle NATURAL JOIN  Poste
           NATURAL JOIN  Installer
                      NATURAL JOIN  Logiciel
                 WHERE nomLog  = 'Oracle 6';
```

## Modifications synchronisées

```
INSERT INTO installer VALUES ('p2','log6',  sequenceIns.NEXTVAL,NULL,NULL);
INSERT INTO installer VALUES ('p8','log1',  sequenceIns.NEXTVAL,NULL,NULL);
INSERT INTO installer VALUES ('p10','log1', sequenceIns.NEXTVAL,NULL,NULL);
```

```
UPDATE Segment seg
 SET seg.nbSalle = (SELECT COUNT(*) FROM Salle sal WHERE seg.indIP=sal.indIP);

UPDATE Segment seg
 SET seg.nbPoste = (SELECT COUNT(*) FROM Poste pos WHERE seg.indIP=pos.indIP);

UPDATE Logiciel l
  SET l.nbInstall = (SELECT COUNT(*) FROM Installer i WHERE l.nLog = i.nLog);

UPDATE Poste p
  SET p.nbLog = (SELECT COUNT(*) FROM Installer i WHERE p.nPoste = i.nPoste);
```

## Opérateurs existentiels

### Sous-interrogation synchronisé

```
--32
SELECT nomPoste
  FROM Poste p WHERE EXISTS
      (SELECT i1.nLog  FROM Installer i1 WHERE i1.nPoste = p.nPoste
      INTERSECT
       SELECT i2.nLog FROM Installer i2 WHERE i2.nPoste = 'p6')
  AND NOT (nPoste ='p6');
```

### Divisions

L'ensemble de référence est noté *A* dans les requêtes suivantes. Il est constitué des logiciels d'un poste donné.

```
--33
SELECT  nomPoste
  FROM Poste p WHERE NOT EXISTS
      (SELECT i2.nLog                                          A
       FROM Installer i2 WHERE i2.nPoste = 'p6'
      MINUS
       SELECT i1.nLog  FROM Installer i1 WHERE i1.nPoste = p.nPoste)
  AND NOT (nPoste ='p6');
--34
SELECT  nomPoste
  FROM Poste p WHERE NOT EXISTS
    (SELECT i2.nLog FROM Installer i2 WHERE i2.nPoste = 'p2'
    MINUS                                  A-B
     SELECT i1.nLog FROM Installer i1 WHERE i1.nPoste = p.nPoste)
  AND NOT EXISTS
    (SELECT i1.nLog FROM Installer i1 WHERE i1.nPoste = p.nPoste
    MINUS                                      B-A
     SELECT i2.nLog FROM Installer i2 WHERE i2.nPoste = 'p2')
  AND NOT (nPoste ='p2');
```





# Chapitre 5

## Vues mono-table

### Vues sans contraintes

```
CREATE VIEW LogicielsUnix
  AS SELECT *
     FROM Logiciel WHERE  typeLog = 'UNIX';

CREATE VIEW Poste0 (nPos0, nomPoste0, nSalle0, TypePoste0, indIP, ad0)
  AS SELECT nPoste, nomPoste, nSalle, typePoste, indIP, ad
     FROM Poste WHERE indIP IN
          (SELECT indIP FROM Segment WHERE etage = 0);

INSERT INTO Poste0
       VALUES ('p15','Bidon15', 's01','UNIX','130.120.80','20');
INSERT INTO Poste0
       VALUES ('p16', 'Bidon16','s21','UNIX','130.120.82','20');
```

Les deux postes sont présents dans la table Poste, mais seul le poste 'p15' est extrait de la vue `Poste0` !

```
DELETE FROM Poste WHERE nPoste IN ('p15','p16');
```

### Résoudre une requête complexe

L'expression `nbPoste*100` sert à la définition de la colonne `prixLocation`.

```
CREATE VIEW SallePrix (nSalle, nomSalle, nbPoste, prixLocation)
       AS SELECT nSalle, nomSalle, nbPoste, nbPoste*100
       FROM Salle;

SELECT * FROM SallePrix
       WHERE prixLocation > 150;

ALTER TABLE Types ADD tarif NUMBER(3);

UPDATE Types SET tarif=50  WHERE typeLP ='TX';
UPDATE Types SET tarif=100 WHERE typeLP ='PCWS';
UPDATE Types SET tarif=120 WHERE typeLP ='PCNT';
UPDATE Types SET tarif=200 WHERE typeLP ='UNIX';
UPDATE Types SET tarif=80  WHERE typeLP ='NC';
UPDATE Types SET tarif=400 WHERE typeLP ='BeOS';
```

La particularité de la vue `SalleIntermédiaire` réside dans la condition de regroupement (3 colonnes sont nécessaires car 3 colonnes doivent être extraites par `SELECT`).

```
CREATE VIEW SalleIntermédiaire(nSalle, typePoste, nombre, tarif)
  AS SELECT p.nSalle, p.typePoste, COUNT(p.nPoste), t.tarif
     FROM   Poste p, Types t
```

```
WHERE  p.typePoste = t.typeLP
GROUP  BY p.nSalle, p.typePoste, t.tarif;
```

La vue `SallePrixTotal` est définie à partir de la vue `SalleIntermédiaire` en groupant sur le numéro de salle et en faisant la somme du produit des tarifs par le nombre de poste de chaque type.

```
CREATE VIEW SallePrixTotal(nSalle, PrixRéel)
  AS SELECT   nSalle, SUM(nombre*tarif)
     FROM     SalleIntermédiaire
     GROUP BY nSalle;

SELECT * FROM SallePrixTotal
      WHERE PrixRéel = (SELECT MIN(PrixRéel) FROM SallePrixTotal);
```

### Vues avec contraintes

```
CREATE OR REPLACE VIEW Poste0
      (nPos0, nomPoste0, nSalle0, TypePoste0, indIP, ad0)
  AS SELECT nPoste, nomPoste, nSalle, typePoste, indIP, ad
     FROM Poste WHERE indIP IN
          (SELECT indIP FROM Segment WHERE etage = 0)
  WITH CHECK OPTION CONSTRAINT wco_Poste0;
```

Insertion désormais impossible du fait de la contrainte de vérification.

```
INSERT INTO Poste0 VALUES
      ('p16','Bidon15', 's21','UNIX','130.120.82','20');

CREATE OR REPLACE VIEW Installer0 (nPoste, nLog, num, dateIns)
  AS SELECT nPoste, nLog, numIns, dateIns FROM Installer
     WHERE nLog NOT IN
     (SELECT nLog FROM Logiciel WHERE typeLog = 'PCNT')
     AND nPoste IN
     (SELECT nPoste FROM Poste WHERE  indIP IN
           (SELECT indIP FROM Segment WHERE etage=0 ))
  WITH CHECK OPTION CONSTRAINT wco_Installer0;
```

Insertions impossibles du fait de la contrainte de vérification.

```
INSERT INTO Installer0 VALUES('p11','log7',sequenceIns.NEXTVAL,SYSDATE);
INSERT INTO Installer0 VALUES('p1','log7' ,sequenceIns.NEXTVAL, SYSDATE);
```

Insertion valable.

```
INSERT INTO Installer0 VALUES('p6','log2' ,sequenceIns.NEXTVAL, SYSDATE);
```

## Vue multi-tables

```
CREATE VIEW SallePoste (nomSalle, nomPoste, adrIP, nomTypePoste)
  AS SELECT s.nomSalle, p.nomPoste, p.indIP ||'.'|| p.ad, t.nomType
     FROM   Salle s, Poste p, Types t
     WHERE  s.nSalle    = p.nSalle
     AND    p.typePoste = t.typeLP;
```





## Mises à jour conditionnées

Les tables initiales sont les suivantes.

```
CREATE TABLE Primes
      (brevet CHAR(6), nom CHAR(20), paye NUMBER(7,2), compa CHAR(4));
CREATE TABLE Vols
      (brevet CHAR(6), dateVol DATE, bonus NUMBER(3));
INSERT INTO Primes VALUES ('PL-1', 'Aurélia Ente', 100, 'AF');
INSERT INTO Primes VALUES ('PL-2', 'Agnès Bidal', 100, 'AF');
INSERT INTO Primes VALUES ('PL-3', 'Sylvie Payrissat', 0, 'SING');
INSERT INTO Vols VALUES ('PL-1', SYSDATE-15, 50);
INSERT INTO Vols VALUES ('PL-1', SYSDATE-10, 50);
INSERT INTO Vols VALUES ('PL-1', SYSDATE,    50);
INSERT INTO Vols VALUES ('PL-3', SYSDATE-5,  40);
INSERT INTO Vols VALUES ('PL-3', SYSDATE,    40);
INSERT INTO Vols VALUES ('PL-4', SYSDATE,    20);
```

La vue mono-table à définir doit contenir la somme des bonus pour chaque pilote car `MERGE` ne peut pas mettre à jour plusieurs fois le même enregistrement de la table cible en une seule instruction.

```
CREATE VIEW v_Vols(brevet, sommeBonus)
       AS SELECT brevet, SUM(bonus) FROM Vols GROUP BY brevet;
```

La fusion utilise la vue qui ne contient aucune redondance au niveau des codes brevet. L'ajout du bonus à la table prime prend donc en compte tous les bonus que le pilote a eu.

```
MERGE INTO Primes p
   USING (SELECT brevet, sommeBonus FROM v_Vols) v
   ON    (p.brevet = v.brevet)
   WHEN MATCHED     THEN UPDATE SET p.paye = p.paye + v.sommeBonus
   WHEN NOT MATCHED THEN INSERT (brevet, paye) VALUES (v.brevet, v.sommeBonus);
```

# Chapitre 6

## Tableaux et structures de contrôle

L'algorithme de la fusion est un classique du genre : 3 *tant que* qui s'enchaînent. Le premier traite simultanément les 2 tableaux en remplissant le dernier jusqu'à ce qu'un des tableau soit traité en totalités. Il reste ensuite à remplir le tableau résultat avec la fin du tableau restant à parcourir (par les deux derniers *tant que*).

```
SET SERVEROUTPUT ON
DECLARE
  TYPE nomComp_tytab IS TABLE OF VARCHAR2(15) INDEX BY BINARY_INTEGER;
-- tableaux
  tab_compFrance nomComp_tytab;
```

```
  tab_compMonde  nomComp_tytab;
  tab_résultat   nomComp_tytab;
  v_indiceFrance NUMBER(1) := 1;
  v_indiceMonde  NUMBER(1) := 1;
  v_indiceRésultat  NUMBER(2) := 1;
BEGIN
  tab_compFrance(1) := 'AERIS';
  tab_compFrance(2) := 'Air France';
  tab_compFrance(3) := 'Air Littoral';
  tab_compFrance(4) := 'Regional';
  tab_compMonde(1)  := 'ALITALIA';
  tab_compMonde(2)  := 'Quantas';
  tab_compMonde(3)  := 'SABENA';
-- parcours des deux tableaux en parallèle (stop à la fin d'un des 2)
  WHILE (tab_compFrance.EXISTS(v_indiceFrance) AND
         tab_compMonde.EXISTS(v_indiceMonde)) LOOP
   IF tab_compFrance(v_indiceFrance) > tab_compMonde(v_indiceMonde)THEN
     tab_résultat(v_indiceRésultat) :=  tab_compMonde(v_indiceMonde);
     v_indiceMonde    := v_indiceMonde    + 1;
   ELSE
     tab_résultat(v_indiceRésultat) :=  tab_compFrance(v_indiceFrance);
     v_indiceFrance   := v_indiceFrance   + 1;
   END IF;
   v_indiceRésultat := v_indiceRésultat + 1;
  END LOOP;
-- Traitement de la fin du tableau qui reste à parcourir
  WHILE (tab_compFrance.EXISTS(v_indiceFrance)) LOOP
   tab_résultat(v_indiceRésultat) :=  tab_compFrance(v_indiceFrance);
   v_indiceRésultat := v_indiceRésultat + 1;
   v_indiceFrance   := v_indiceFrance   + 1;
  END LOOP;
  WHILE (tab_compMonde.EXISTS(v_indiceMonde)) LOOP
   tab_résultat(v_indiceRésultat) :=  tab_compMonde(v_indiceMonde);
   v_indiceRésultat := v_indiceRésultat + 1;
   v_indiceMonde    := v_indiceMonde    + 1;
  END LOOP;
-- Affichage des résultats
  DBMS_OUTPUT.PUT_LINE
      ('Nombre éléments de tab_résultat ' || tab_résultat.COUNT);
  FOR v_entier IN 1 ..tab_résultat.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE
       ('tab_résultat(' || v_entier || ') : ' || tab_résultat(v_entier));
  END LOOP;
END;
/
```





## Bloc PL/SQL et variables %TYPE

```
SET SERVEROUTPUT ON
DECLARE
  v_sequenceInsMax Installer.numIns%TYPE;
  v_nPoste         Installer.nPoste%TYPE;
  v_nLog           Installer.nLog%TYPE;
  v_dateIns        Installer.dateIns%TYPE;
  v_nSalle         Poste.nSalle%TYPE;
  v_nomLog         Logiciel.nomLog%TYPE;
BEGIN
-- Extraction de la dernière extraction
 SELECT   numIns, nPoste, nLog, dateIns
    INTO v_sequenceInsMax, v_nPoste, v_nLog, v_dateIns
    FROM Installer WHERE numIns = (SELECT MAX(numIns) FROM Installer);
-- Extraction du numéro de la salle du poste saisi
  SELECT nSalle INTO v_nSalle FROM  Poste WHERE nPoste = v_nPoste;
-- Extraction du nom du logiciel de numéro saisi
  SELECT nomLog INTO v_nomLog FROM  Logiciel WHERE nLog = v_nLog;
-- état de sortie
  DBMS_OUTPUT.PUT_LINE('Dernière installation en salle : '|| v_nSalle);
  DBMS_OUTPUT.PUT_LINE('-------------------------------------------');
  DBMS_OUTPUT.PUT_LINE('Poste : '|| v_nPoste || ' Logiciel : ' || v_nomLog ||
                       ' en date du ' || v_dateIns);
END;
/
```

## Variables de substitution et globales

```
-- Saisies au clavier
ACCEPT s_nSalle        PROMPT 'Numéro de Salle : '
ACCEPT s_typePoste     PROMPT 'Type de poste : '
VARIABLE g_nbPoste     NUMBER;
VARIABLE g_nbInstall   NUMBER;
BEGIN
-- Extraction du nombre de postes
  SELECT  COUNT(*) INTO :g_nbPoste
    FROM  Poste WHERE nSalle = '&s_nSalle' AND typePoste = '&s_typePoste';
-- Extraction du nombre d'installations
  SELECT COUNT(*)INTO :g_nbInstall FROM Installer
    WHERE nPoste IN (SELECT nPoste FROM Poste
                     WHERE nSalle = '&s_nSalle' AND typePoste = '&s_typePoste');
END;
/
-- Affichage des résultats
PRINT :g_nbPoste;
PRINT :g_nbInstall;
```

## Transaction

```
SET ECHO OFF
SET SERVEROUTPUT ON
SET VERIFY OFF
-- Saisies au clavier
ACCEPT s_nLog        PROMPT 'Numéro de logiciel : '
ACCEPT s_nomLog      PROMPT 'Nom du logiciel    : '
ACCEPT s_version     PROMPT 'Version du logiciel : '
ACCEPT s_typeLog     PROMPT 'Type du logiciel    : '
ACCEPT s_prix        PROMPT 'Prix du logiciel (en euros) : '
DECLARE
  v_nPoste Poste.nPoste%TYPE := 'p7';
  v_dateAchat DATE;
BEGIN
-- Insère dans Logiciel
  INSERT INTO Logiciel  VALUES
        ('&s_nLog','&s_nomLog',SYSDATE,'&s_version','&s_typeLog','&s_prix',0);
  DBMS_OUTPUT.PUT_LINE('Logiciel inséré dans la base');
-- Extraction de la date de l'achat
  SELECT dateach INTO v_dateAchat FROM Logiciel WHERE nLog = '&s_nLog';
  DBMS_OUTPUT.PUT_LINE
      ('Date achat : ' || TO_CHAR(v_dateAchat,'DD-MM-YYYY HH24:MI:SS'));
-- On attend 5 petites secondes
  DBMS_LOCK.SLEEP(5);
  DBMS_OUTPUT.PUT_LINE
      ('Date installation : ' || TO_CHAR(SYSDATE,'DD-MM-YYYY HH24:MI:SS'));
-- Insertion dans Installer
  INSERT INTO Installer
     VALUES (v_nPoste, '&s_nLog', sequenceIns.NEXTVAL, SYSDATE,
              NUMTODSINTERVAL(SYSDATE-v_dateAchat,'SECOND'));
 DBMS_OUTPUT.PUT_LINE('Logiciel installé sur le poste');
 COMMIT;
END;
/
-- Affichage des tables modifiées
SELECT * FROM Logiciel;
SELECT * FROM Installer;
```

# Chapitre 7

## Curseur

```
CREATE OR REPLACE PROCEDURE calculTemps IS
   CURSOR curseur IS
     SELECT l.nomLog,p.nomPoste,l.dateAch,i.dateIns,i.nLog, i.nPoste
     FROM Installer i, Logiciel l, Poste p
```





```
     WHERE i.nPoste = p.nPoste AND i.nLog = l.nLog;
   atte NUMBER(4);
BEGIN
   FOR enreg IN curseur LOOP
      IF enreg.dateIns IS NULL THEN
        DBMS_OUTPUT.PUT_LINE('Pas de date d''installation pour le logiciel '
         || enreg.nomLog  || ' sur ' || enreg.nomPoste);
      ELSE
           IF enreg.dateAch IS NULL THEN
              DBMS_OUTPUT.PUT_LINE('Date d''achat inconnue pour le logiciel '
                   || enreg.nomLog  || ' sur ' || enreg.nomPoste);
           ELSE
               atte :=  enreg.dateIns - enreg.dateAch;
               IF atte < 0 THEN
                DBMS_OUTPUT.PUT_LINE('Logiciel ' ||  enreg.nomLog ||
                  ' installé sur ' ||  enreg.nomPoste || ' ' ||  -atte ||
                  ' jour(s) avant d''être acheté!');
               ELSE
         IF atte = 0 THEN
            DBMS_OUTPUT.PUT_LINE('Logiciel ' || enreg.nomLog  || ' sur ' ||
             enreg.nomPoste || ' acheté et installé le même jour!');
         ELSE
            DBMS_OUTPUT.PUT_LINE('Logiciel ' || enreg.nomLog  || ' sur ' ||
             enreg.nomPoste || ', attente ' || atte || ' jour(s).');
            UPDATE Installer SET delai =
                NUMTODSINTERVAL(enreg.dateIns - enreg.dateAch,'DAY')
            WHERE  nPoste = enreg.nPoste AND nLog = enreg.nLog;
         END IF;
        END IF;
       END IF;
      END IF;
   END LOOP;
   COMMIT;
END calculTemps;
```

## Transaction

```
CREATE OR REPLACE PROCEDURE installLogSeg (param1 IN VARCHAR2, param2 IN
VARCHAR2, param3 IN VARCHAR2, param4 IN DATE, param5 IN VARCHAR2, param6 IN
VARCHAR2, param7 IN NUMBER) IS
 CURSOR  curseur IS SELECT p.nomPoste, p.nPoste, s.nomSalle
   FROM  Poste p, Salle s
   WHERE p.indIP  = param1 AND p.typePoste = param6
   AND   p.nSalle = s.nSalle;
BEGIN
  INSERT INTO Logiciel VALUES (param2,param3,param4,param5,param6,param7,0);
  DBMS_OUTPUT.PUT_LINE(param3 || ' stocké dans la table Logiciel');
  FOR enreg IN curseur LOOP
   INSERT INTO Installer
     VALUES(enreg .nPoste, param2, sequenceIns.NEXTVAL ,SYSDATE,
```

```
            NUMTODSINTERVAL(SYSDATE-param4,'DAY'));
    DBMS_OUTPUT.PUT_LINE('Installation sur '|| enreg.nomPoste ||' dans '
                         || enreg.nomSalle);
  END LOOP;
  COMMIT;
END installLogSeg ;
```

## Exceptions

```
CREATE OR REPLACE PROCEDURE installLogSeg (param1 IN VARCHAR2, param2 IN
VARCHAR2, param3 IN VARCHAR2,
 param4 IN DATE, param5 IN VARCHAR2, param6 IN VARCHAR2, param7 IN NUMBER) IS
 CURSOR  curseur IS SELECT p.nomPoste, p.nPoste, s.nomSalle
   FROM  Poste p, Salle s
   WHERE p.indIP  = param1 AND p.typePoste = param6
   AND   p.nSalle = s.nSalle;
 p_nomSegment Segment.nomSegment%TYPE;
 toutsePasseBien BOOLEAN := TRUE;
 date_fausse          EXCEPTION;
 pas_install_possible EXCEPTION;
 enfant_sans_parent   EXCEPTION;
 PRAGMA EXCEPTION_INIT(enfant_sans_parent,-2291);
 nbrInstall  NUMBER := 0;
BEGIN
 BEGIN
  SELECT nomSegment INTO p_nomSegment FROM Segment WHERE indIP  = param1 ;
  EXCEPTION
     WHEN NO_DATA_FOUND THEN
        toutsePasseBien := FALSE;
        DBMS_OUTPUT.PUT_LINE('Segment inconnu!');
      WHEN OTHERS THEN
        toutsePasseBien := FALSE;
        DBMS_OUTPUT.PUT_LINE('Erreur d''Oracle ' || SQLERRM  ||
                     '(' || SQLCODE || ')');
 END;
 IF param4 > SYSDATE THEN RAISE date_fausse;
 END IF;
 IF toutsePasseBien THEN
  INSERT INTO Logiciel VALUES (param2,param3,param4,param5,param6,param7,0);
  DBMS_OUTPUT.PUT_LINE(param3 || ' stocké dans la table Logiciel');
  FOR enreg IN curseur LOOP
   nbrInstall  := nbrInstall + 1;
   INSERT INTO Installer VALUES(enreg .nPoste, param2, sequenceIns.NEXTVAL
                          ,SYSDATE, NUMTODSINTERVAL(SYSDATE-param4,'DAY'));
   DBMS_OUTPUT.PUT_LINE('Installation sur '|| enreg.nomPoste ||' dans '
                         || enreg.nomSalle);
  END LOOP;
  COMMIT;
```





```
  IF nbrInstall = 0 THEN
    RAISE pas_install_possible;
  ELSE
    DBMS_OUTPUT.PUT_LINE(nbrInstall  || ' installations ont été faites.');
  END IF;
 END IF;
EXCEPTION
  WHEN enfant_sans_parent THEN
    DBMS_OUTPUT.PUT_LINE('Type de logiciel inconnu!');
  WHEN DUP_VAL_ON_INDEX THEN
    DBMS_OUTPUT.PUT_LINE('Logiciel déja présent!');
  WHEN date_fausse THEN
    DBMS_OUTPUT.PUT_LINE('Date d''achat supérieure à celle du jour!');
  WHEN pas_install_possible THEN
    DBMS_OUTPUT.PUT_LINE('Pas de poste de ce type sur le segment');
  WHEN OTHERS THEN
       DBMS_OUTPUT.PUT_LINE('Erreur d''Oracle ' ||
                  SQLERRM  || '(' || SQLCODE || ')');
END installLogSeg ;
/
```

## Paquetage

```
CREATE OR REPLACE PACKAGE paquetageParcInfo AS
      PROCEDURE installLogSeg (param1 IN VARCHAR2, param2 IN VARCHAR2, param3
      IN VARCHAR2, param4 IN DATE, param5 IN VARCHAR2, param6 IN VARCHAR2,
      param7 IN NUMBER);
      PROCEDURE calculTemps;
END paquetageParcInfo;
/

CREATE OR REPLACE PACKAGE BODY paquetageParcInfo AS
      PROCEDURE calculTemps IS
      …
      END calculTemps;
      PROCEDURE installLogSeg (param1 IN VARCHAR2, param2 IN VARCHAR2, param3
      IN VARCHAR2, param4 IN DATE, param5 IN VARCHAR2, param6 IN VARCHAR2,
      param7 IN NUMBER) IS
      …
      END installLogSeg ;
END paquetageParcInfo;
/
```

## Déclencheurs

### Mises à jour de colonnes

```
CREATE OR REPLACE TRIGGER Trig_Après_DI_Installer
 AFTER INSERT OR DELETE ON Installer
```

```
 FOR EACH ROW
BEGIN
 IF DELETING THEN
   UPDATE Poste SET nbLog=nbLog - 1 WHERE nPoste = :OLD.nPoste;
   UPDATE Logiciel SET nbInstall = nbInstall - 1 WHERE nLog = :OLD.nLog;
  ELSE
   IF INSERTING THEN
      UPDATE Poste SET nbLog = nbLog + 1 WHERE nPoste = :NEW.nPoste;
      UPDATE Logiciel SET nbInstall = nbInstall + 1 WHERE nLog = :NEW.nLog;
   END IF;
 END IF;
END;
/

CREATE OR REPLACE TRIGGER Trig_Après_DI_Poste
 AFTER INSERT OR DELETE ON Poste
 FOR EACH ROW
BEGIN
 IF DELETING THEN
    UPDATE Salle SET nbPoste = nbPoste - 1 WHERE nSalle = :OLD.nSalle;
 ELSE
    UPDATE Salle SET nbPoste = nbPoste + 1 WHERE nSalle = :NEW.nSalle;
 END IF;
END;
/

CREATE OR REPLACE TRIGGER Trig_Après_U_Salle
 AFTER UPDATE OF nbPoste ON Salle
 FOR EACH ROW
DECLARE
 differ NUMBER;
BEGIN
 differ := :NEW.nbPoste - :OLD.nbPoste;
 UPDATE Segment SET nbPoste = nbPoste + differ WHERE indIP = :NEW.indIP;
END;
/
```

### Programmation de contraintes

```
CREATE OR REPLACE TRIGGER Trig_Avant_UI_Installer
 BEFORE INSERT OR UPDATE OF nPoste, nLog ON installer
 FOR EACH ROW
DECLARE
 type_log   Types.typeLP%TYPE;
 type_pos   Types.typeLP%TYPE;
 date_achat DATE;
BEGIN
 SELECT typeLog, dateAch INTO type_log,date_achat
        FROM logiciel WHERE :NEW.nLog = nLog;
 SELECT typePoste INTO type_pos
        FROM Poste WHERE :NEW.nPoste = nPoste;
```





```
 IF type_log != type_pos THEN
    RAISE_APPLICATION_ERROR(-20111,'Poste : '||type_pos||'
                  différent du logiciel : '||type_log);
 END IF;
 IF :NEW.dateIns IS NOT NULL THEN
    IF :NEW.dateIns < date_achat THEN
     RAISE_APPLICATION_ERROR(-20112,'Installation ('|| :NEW.dateIns
           ||') inférieure a la date d''achat ('||date_achat||')');
    END IF;
 END IF;
END;
/
```

# Chapitre 9 (JDBC)

## Curseur statique

```
import java.util.ArrayList;
public class ExoJDBC {
 public static Connection cx;
 public static ResultSet rs, rs2;
 public static Statement etat;
 public static Statement etatModifiable;
 public static CallableStatement cetat;
 public static ArrayList getSalles() {
      ArrayList tableauRésultat = new ArrayList();
      try { etat = cx.createStatement();
             rs = etat.executeQuery("SELECT * FROM Salle");
             String [] ligne = null;
            while (rs.next()) { ligne = new String[4];
                  ligne[0] = rs.getString(1);
                  ligne[1] = rs.getString(2);
                  ligne[2] = (new Integer(rs.getInt(3))).toString();
                  ligne[3] = rs.getString(4);
                  tableauRésultat.add(ligne); }
                  rs.close(); etat.close(); }
      catch (SQLException ex) { while (ex != null) {
            System.out.println ("Statut SQL  : "+ex.getSQLState());
            System.out.println ("Message     : "+ex.getMessage());
            System.out.println ("Code erreur : "+ex.getErrorCode());
            ex = ex.getNextException(); } }
            return tableauRésultat; }

public static void main(String args[])
      {try {
      DriverManager.registerDriver (new oracle.jdbc.driver.OracleDriver());
      cx = DriverManager.getConnection
      ("jdbc:oracle:thin:@CAMPAROLS:1521:BDSoutou","soutou","ingres");
```

```
ArrayList lignes = getSalles();
System.out.println("Liste des salles :\n");
System.out.println("nSalle\tnomSalle  \tnbPoste\tindIP");
System.out.println("----------------------------------------");
String[] lig;
for (int i=0;i<lignes.size();i++) {
  lig = (String []) lignes.get(i);
  System.out.println(lig[0]+" \t"+lig[1]+"  \t"+lig[2]+" \t"+lig[3]);}
deleteSalle(6);
int excep = deleteSallePL("s9");
if (excep==0)
    System.out.println("Salle 9 supprimée");
      else
      System.out.println("Suppression impossible, code retour : "+excep);
// Fils présents
excep = deleteSallePL("s01");
if (excep==0) System.out.println("Salle ?? supprimée");
 else System.out.println("Suppression impossible, code: "+excep);}
catch (SQLException ex) {while (ex != null) {
System.out.println ("Statut SQL  : "+ex.getSQLState());
System.out.println ("Message     : "+ex.getMessage());
System.out.println ("Code erreur : "+ex.getErrorCode());
ex = ex.getNextException(); } } } }
```

## Curseur modifiable

```
public static void deleteSalle(int nl)
{try {
 etatModifiable = cx.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                  ResultSet.CONCUR_UPDATABLE);
cx.setAutoCommit(false);
rs2 = etatModifiable.executeQuery("SELECT s.* FROM Salle s");
if (rs2.absolute(nl))
  {rs2.deleteRow();
   cx.commit();
   System.out.println("Salle supprimée");}
else System.out.println("Désolé, pas de "+ nl +" ème salle !");
rs2.close(); etatModifiable.close(); }
catch (SQLException ex) { while (ex != null) {
      System.out.println ("Statut SQL  : "+ex.getSQLState());
      System.out.println ("Message     : "+ex.getMessage());
      System.out.println ("Code erreur : "+ex.getErrorCode());
      ex = ex.getNextException(); } } }
```

## Appel d'un sous-programme

```
public static int deleteSallePL(String ns) {
 int result = 0;
  try {cetat = cx.prepareCall("{? = call supprimeSalle(?)}");
```





```
cetat.registerOutParameter(1,java.sql.Types.INTEGER);
cetat.setString(2,ns);
cetat.execute();
result = cetat.getInt(1);
cetat.close(); }
catch (SQLException ex) { while (ex != null) {
     System.out.println ("Statut SQL  : "+ex.getSQLState());
     System.out.println ("Message     : "+ex.getMessage());
     System.out.println ("Code erreur : "+ex.getErrorCode());
     ex = ex.getNextException(); } }
return result;}
```

# Chapitre 10 (SQLJ)

## Itérateur nommé

```
import java.io.*;
import java.sql.*;
import oracle.sqlj.runtime.Oracle;
public class Consulte {
   public static void afficheSalle(String ns) {
      try {Oracle.connect(Consulte.class,"connect.properties");
         String noms, ip, nbp;
         #sql { SELECT nomSalle, nbPoste, indIP INTO :noms, :nbp, :ip
                FROM Salle WHERE nSalle = :ns };
         System.out.println(noms+" ("+nbp+" postes, segment IP : "+ip+")");}
      catch(SQLException ex) { ex.printStackTrace(); } }
   public static void affichePostes(String ns) {
      try {Oracle.connect(Consulte.class,"connect.properties");
         #sql iterator IterateurPoste (String nPoste, String nomPoste,
           String indIP, String ad,String typePoste);
         IterateurPoste rs_poste;
         #sql rs_poste = { SELECT nPoste, nomPoste, indIP, ad, typePoste
                          FROM Poste WHERE nSalle = :ns };
         System.out.println("Liste des postes : ");
         while(rs_poste.next()) {
            System.out.print("Numero: "+rs_poste.nPoste());
            System.out.print(" Nom Poste: "+rs_poste.nomPoste());
            System.out.print(" IP: "+rs_poste.indIP());
            System.out.print(" Adr: "+rs_poste.ad());
            System.out.println("Type Poste: "+rs_poste.typePoste()); }
         rs_poste.close(); }
      catch(SQLException ex) { ex.printStackTrace(); } }
   public static void main ( String[] args )
      {String ns;
      System.out.print("Saisir le numero d'une salle : ");
      ns = lire();
```

```
            afficheSalle(ns);
            affichePostes(ns);}
         private static String lire()
            {String c="";
            try {BufferedReader entrée =
                  new BufferedReader(new InputStreamReader(System.in));
                  c = entree.readLine();}
            catch (java.io.IOException e)
                  {System.out.println("Une erreur d'entree/sortie est survenue!!!");
                   System.exit(0);}
            return c;} }
```

## Mise à jour de la base

```
import java.io.*;
import java.sql.*;
import oracle.sqlj.runtime.Oracle;
public class Insere {
   public static void insere(String[] valeurs) {
    try {Oracle.connect(Insere.class,"connect.properties");
       String v0 = valeurs[0], v1=valeurs[1], v3=valeurs[3], v4=valeurs[4];
       int v5 = Integer.parseInt(valeurs[5]), v6=0;
       java.sql.Date v2 = null;
       if (!valeurs[2].equals("NULL")) v2 = java.sql.Date.valueOf(valeurs[2]);
       #sql { INSERT INTO logiciel VALUES (:v0,:v1,:v2,:v3,:v4,:v5,:v6) };
       #sql { COMMIT };
       Oracle.close(); }
     catch(SQLException ex) {
   if (ex.getErrorCode() == 1) System.out.println("Logiciel déjà existant!");
   else if (ex.getErrorCode() == 913) System.out.println("Trop de valeurs!");
   else if (ex.getErrorCode() == 942) System.out.println("Table inconnue!");
   else if (ex.getErrorCode() == 947) System.out.println("Manque de valeurs!");
   else if (ex.getErrorCode()==1401) System.out.println("Valeur trop longue!");
   else if (ex.getErrorCode()==1438) System.out.println("Valeur trop grande!");
   else if (ex.getErrorCode() == 2291) System.out.println("Type inconnu!"); } }
   public static void main ( String[] args )
      {String[][] valeurs = new String[3][8];
      valeurs[0][0] = "v1"; valeurs[0][1] = "Forms";
      valeurs[0][2] ="13-05-1995"; valeurs[0][3]="4";
      valeurs[0][4] = "UNIX"; valeurs[0][5] = "300";
      valeurs[1][0] = "v2"; valeurs[1][1] ="SQLJ";
      valeurs[1][2] ="15-09-1999"; valeurs[1][3] = "1";
      valeurs[1][4] = "UNIX"; valeurs[1][5] ="560";
      valeurs[2][0] = "v3"; valeurs[2][1] = "MySQL";
      valeurs[2][2] ="12-04-1998"; valeurs[2][3] = "7";
      valeurs[2][4] = "PCNT"; valeurs[2][5] = "0";
      for (int i=0; i<valeurs.length; i++) insere(valeurs[i]);}  }
```