Corrigés des exercices SQL pour MySQL

Christian Soutou – Eyrolles 2006

Chapitre 1

Création des tables

```
CREATE TABLE Segment
      (indIP      varchar(11),
       nomSegment varchar(20) NOT NULL,
       etage TINYINT(1),
       CONSTRAINT pk_Segment PRIMARY KEY (indIP));
CREATE TABLE Salle
      (nSalle     varchar(7),
       nomSalle   varchar(20) NOT NULL,
       nbPoste    TINYINT(2),
       indIP      varchar(11),
       CONSTRAINT pk_salle PRIMARY KEY (nSalle));
CREATE TABLE Poste
      (nPoste     varchar(7),
       nomPoste   varchar(20) NOT NULL,
       indIP      varchar(11),
       ad         varchar(3),
       typePoste  varchar(9),
       nSalle     varchar(7),
       CONSTRAINT pk_Poste PRIMARY KEY (nPoste),
       CONSTRAINT ck_ad    CHECK (ad BETWEEN '000' AND '255'));
CREATE TABLE Logiciel
      (nLog       varchar(5),
       nomLog     varchar(20) NOT NULL,
       dateAch    DATETIME,
       version    varchar(7),
       typeLog    varchar(9),
       prix       DECIMAL(6,2),
       CONSTRAINT pk_Logiciel PRIMARY KEY (nLog),
       CONSTRAINT ck_prix     CHECK (prix >= 0));
CREATE TABLE Installer
      (nPoste     varchar(7),
       nLog       varchar(5),
       numIns     INTEGER(5) AUTO_INCREMENT,
       dateIns    TIMESTAMP DEFAULT NOW(),
       delai      DECIMAL(8,2),
       CONSTRAINT pk_Installer PRIMARY KEY(numIns));
CREATE TABLE Types
      (typeLP   varchar(9), nomType varchar(20),
       CONSTRAINT pk_types PRIMARY KEY(typeLP));
```

Destruction des tables

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

Chapitre 2

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

INSERT INTO installer (nPoste,nLog,dateIns,delai) VALUES ('p2', 'log1', '2003-05-15',NULL);
INSERT INTO installer (nPoste,nLog,dateIns,delai) VALUES ('p2', 'log2', '2003-09-17',NULL);
INSERT INTO installer (nPoste,nLog,dateIns,delai) VALUES ('p4', 'log5',  NULL,NULL);
INSERT INTO installer (nPoste,nLog,dateIns,delai) VALUES ('p6', 'log6', '2003-05-20',NULL);
INSERT INTO installer (nPoste,nLog,dateIns,delai) VALUES ('p6', 'log1', '2003-05-20',NULL);
INSERT INTO installer (nPoste,nLog,dateIns,delai) VALUES ('p8', 'log2', '2003-05-19',NULL);
INSERT INTO installer (nPoste,nLog,dateIns,delai) VALUES ('p8', 'log6', '2003-05-20',NULL);
INSERT INTO installer (nPoste,nLog,dateIns,delai) VALUES ('p11','log3', '2003-04-20',NULL);
INSERT INTO installer (nPoste,nLog,dateIns,delai) VALUES ('p12','log4', '2003-04-20',NULL);
INSERT INTO installer (nPoste,nLog,dateIns,delai) VALUES ('p11','log7', '2003-04-20',NULL);
INSERT INTO installer (nPoste,nLog,dateIns,delai) VALUES ('p7', 'log7', '2002-04-01',NULL);
```

Modification des 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';
SELECT * FROM Segment;

UPDATE Logiciel
      SET prix = prix*0.9 WHERE typeLog = 'PCNT';
SELECT nLog, typeLog, prix FROM Logiciel;
```

Chapitre 3

Ajout de colonnes

```
ALTER TABLE Segment
      ADD (nbSalle TINYINT(2) DEFAULT 0, nbPoste TINYINT(2) DEFAULT 0);
ALTER TABLE Logiciel ADD nbInstall TINYINT(2) DEFAULT 0;
ALTER TABLE Poste ADD nbLog TINYINT(2) DEFAULT 0;
```

Modification de colonnes

```
ALTER TABLE Salle MODIFY nomSalle VARCHAR(30);
DESC Salle;
ALTER TABLE Segment MODIFY nomSegment VARCHAR(15);
DESC Segment;
```

Ajout de contraintes

```
ALTER TABLE Installer ADD CONSTRAINT un_installation UNIQUE(nPoste,nLog);

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);

-- commande refusées
ALTER TABLE Logiciel ADD CONSTRAINT fk_Logiciel_typeLog_Types
  FOREIGN KEY(typeLog) REFERENCES Types(typeLP);

ALTER TABLE Salle ADD CONSTRAINT fk_Salle_indIP_Segment
  FOREIGN KEY(indIP) REFERENCES Segment(indIP);

--erreurs :
SELECT nLog FROM Logiciel WHERE typeLog NOT IN (SELECT typeLP FROM Types);

SELECT nSalle FROM Salle WHERE indIP NOT IN (SELECT indIP FROM Segment);

--résolution des rejets
--Supprimer les enregistrements de la table Salle qui posent problème.
DELETE FROM Salle WHERE indIP NOT IN (SELECT indIP FROM Segment);

--Ajouter le type de logiciel (‘BeOS’, ‘Système Be’)
INSERT INTO Types VALUES ('BeOS','Système Be');

-- commandes OK
ALTER TABLE Logiciel ADD CONSTRAINT fk_Logiciel_typeLog_Types
  FOREIGN KEY(typeLog) REFERENCES Types(typeLP);

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

Chapitre 4

Création dynamique de tables

```
CREATE TABLE Softs AS SELECT nomLog, Version, prix FROM Logiciel;
ALTER TABLE Softs CHANGE nomLog nomSoft VARCHAR(20);
```

```
      CREATE TABLE PCSeuls AS SELECT nPoste, nomPoste, IndIP, ad, typePoste, nSalle
            FROM Poste WHERE typePoste = 'PCNT' OR typePoste = 'PCWS';
      ALTER TABLE PCSeuls CHANGE nPoste np VARCHAR(7);
      ALTER TABLE PCSeuls CHANGE nomPoste nomP VARCHAR(20);
      ALTER TABLE PCSeuls CHANGE IndIP seg VARCHAR(11);
      ALTER TABLE PCSeuls CHANGE typePoste typeP VARCHAR(9);
      ALTER TABLE PCSeuls CHANGE nSalle lieu VARCHAR(9);
```

Requêtes monotables

```
--1   Type du poste p8
SELECT nPoste, typePoste FROM Poste WHERE nPoste = 'p8';

--2   Noms des logiciels UNIX
SELECT nomLog FROM Logiciel WHERE typeLog = 'UNIX';

--3   Nom, adresse IP, numéro de salle des postes de type UNIX ou PCWS.
SELECT nomPoste, indIP, ad, nSalle FROM poste
  WHERE typePoste = 'UNIX' OR typePoste = 'PCWS';

--4   Même requête pour les postes du segment 130.120.80 triés
--par numéro de salle décroissant
SELECT nomPoste, indIP, ad, nSalle FROM poste
  WHERE (typePoste = 'UNIX' OR typePoste = 'PCWS')
  AND   indIP = '130.120.80' ORDER BY nSalle DESC;

--5  Numéros des logiciels installés sur le poste p6.
SELECT  nLog FROM Installer WHERE nPoste = 'p6';

--6   Numéros des postes qui hébergent le logiciel log1.
SELECT  nPoste FROM  Installer WHERE nLog = 'log1';

--7   Nom et adresse IP complète (ex : 130.120.80.01) des postes de type TX
SELECT nomPoste, CONCAT(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) T;
```

Requêtes multitables

Opérateurs ensemblistes

```
--15
SELECT DISTINCT typeLP FROM Types
 WHERE typeLP NOT IN (SELECT DISTINCT typePoste FROM Poste);

--16
```

```
SELECT DISTINCT typeLog FROM Logiciel
  WHERE typeLog IN (SELECT typePoste FROM Poste);

--17
SELECT DISTINCT typePoste FROM Poste
  WHERE typePoste NOT IN (SELECT typeLog FROM Logiciel);
```

Jointures procédurales

```
--18
SELECT CONCAT(indIP,'.',ad) FROM Poste
  WHERE nPoste IN
      (SELECT nPoste FROM   Installer WHERE  nLog = 'log6');

--19
SELECT CONCAT(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  Nom du logiciel ayant la date d'achat la plus récente (utiliser la requête 12).
SELECT  nomLog FROM Logiciel WHERE dateAch =
      (SELECT MAX(dateAch) FROM Logiciel);
```

Jointures relationnelles

```
SELECT CONCAT(indIP,'.',ad) FROM Poste p, Installer i
  WHERE p.nPoste = i.nPoste AND i.nLog = 'log6';

--24  Adresse IP des postes qui hébergent le logiciel de nom 'Oracle 8'
SELECT  CONCAT(indIP,'.',ad) FROM Poste p, Installer i, Logiciel l
  WHERE p.nPoste = i.nPoste AND l.nLog = i.nLog AND l.nomLog = 'Oracle 8';

--25  Noms des segments possédant exactement trois postes de travail de type 'TX'
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  Noms des salles ou l'on peut trouver au moins un poste hébergeant 'Oracle 6'
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  Adresse IP des postes qui hébergent le logiciel 'log6'.
SELECT CONCAT(indIP,'.',ad) FROM  Poste NATURAL JOIN Installer
  WHERE nLog   = 'log6';

--29  Adresse IP des postes qui hébergent le logiciel de nom 'Oracle 8'
SELECT CONCAT(indIP,'.',ad) FROM Poste NATURAL JOIN Installer
      NATURAL JOIN Logiciel WHERE nomLog = 'Oracle 8';
```

```
--30  Noms des segments possédant exactement trois postes de travail de type 'TX'
SELECT nomSegment FROM Segment JOIN Poste USING(indIP)
  WHERE  typePoste = 'TX' GROUP  BY nomSegment HAVING COUNT(*)=3;

--31  Noms des salles ou l'on peut trouver au moins un poste hébergeant 'Oracle 6'
SELECT  nomSalle FROM  Salle NATURAL JOIN  Poste
      NATURAL JOIN  Installer
            NATURAL JOIN  Logiciel WHERE nomLog  = 'Oracle 6';
```

Modifications synchronisées

```
INSERT INTO installer (nPoste,nLog,dateIns,delai)
      VALUES ('p2','log6',SYSDATE(),NULL);
INSERT INTO installer (nPoste,nLog,dateIns,delai)
      VALUES ('p8','log1',SYSDATE(),NULL);
INSERT INTO installer (nPoste,nLog,dateIns,delai)
      VALUES ('p10','log1',SYSDATE(),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 synchronisée

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

Divisions

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

Chapitre 5

Vues monotables

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

```
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);
DESCRIBE Poste0;
SELECT * FROM Poste0;

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 sont présents ....
SELECT * FROM Poste;
-- seul le poste p15 est présent ....
SELECT * FROM Poste0;
DELETE FROM Poste WHERE nPoste IN ('p15','p16');
```

Résoudre une requête complexe

```
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 DROP COLUMN tarif;
ALTER TABLE Types ADD tarif SMALLINT(4);

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';

CREATE VIEW SalleIntermediaire(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;

CREATE VIEW SallePrixTotal(nSalle, PrixReel)
  AS SELECT nSalle, SUM(nombre*tarif) FROM SalleIntermediaire
     GROUP BY nSalle;
SELECT * FROM SallePrixTotal
       WHERE PrixReel = (SELECT MIN(PrixReel) FROM SallePrixTotal);
```

Vues avec contraintes

```
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)
  WITH CHECK OPTION;
--ck option failed
INSERT INTO Poste0 VALUES('p16','Bidon15', 's21','UNIX','130.120.82','20');

CREATE 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 ;
--ck option failed
INSERT INTO Installer0 (nPoste,nLog,dateIns) VALUES ('p11','log7',SYSDATE());
--ck option failed
INSERT INTO Installer0 (nPoste,nLog,dateIns) VALUES ('p1','log7',SYSDATE());
--bonne installation
INSERT INTO Installer0 (nPoste,nLog,dateIns) VALUES ('p6','log2',SYSDATE());
```

Vue multitable

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

Chapitre 6

Extraction de données

```
delimiter $
DROP PROCEDURE sp1$

CREATE PROCEDURE sp1()
BEGIN
DECLARE v_sequenceInsMax INTEGER(5);
DECLARE v_nPoste         VARCHAR(7);
DECLARE v_nLog           VARCHAR(5);
DECLARE v_dateIns      TIMESTAMP;
DECLARE v_nSalle         VARCHAR(7);
DECLARE v_nomLog         VARCHAR(20);
 SELECT numIns, nPoste, nLog, dateIns
  INTO v_sequenceInsMax, v_nPoste, v_nLog, v_dateIns
    FROM  Installer WHERE numIns = (SELECT MAX(numIns) FROM Installer);
  SELECT  nSalle INTO v_nSalle FROM Poste WHERE nPoste = v_nPoste;
  SELECT  nomLog INTO v_nomLog FROM Logiciel WHERE nLog = v_nLog;
  SELECT CONCAT('Derniere installation en salle : ',v_nSalle) "Resultat 1 exo 1";
  SELECT  CONCAT('Poste  :  ',v_nPoste,'  Logiciel  :  ',  v_nomLog  ,'  en  date  du  ',v_dateIns)
"Resultat 2 exo 1";
END;
$
--trace :
CALL sp1()$
```

Variables de session

```
delimiter $
SET @vs_nSalle    = 's01'$
SET @vs_typePoste = 'UNIX'$
SET @vs_nbPoste   = ''$
SET @vs_nbInstall = ''$

DROP PROCEDURE sp1$

CREATE PROCEDURE sp1()
BEGIN
  SELECT COUNT(*) INTO @vs_nbPoste FROM Poste  WHERE nSalle=@vs_nSalle
      AND typePoste=@vs_typePoste ;
  SELECT COUNT(*) INTO @vs_nbInstall
      FROM Installer WHERE nPoste IN
            (SELECT nPoste FROM Poste
            WHERE nSalle=@vs_nSalle AND typePoste=@vs_typePoste);
END;
$
--trace :
CALL sp1()$
SELECT CONCAT(@vs_nbPoste, ' poste(s) installe(s) en salle ',@vs_nSalle,', ',
@vs_nbInstall, ' installation(s) de type ',@vs_typePoste) "Resultat exo2"$
```

Transaction

```
delimiter $
SET @vs_nLog   = 'log15'$
SET @vs_nomLog = 'MySQL Query'$
SET @vs_version= '1.4'$
SET @vs_typeLog= 'PCWS'$
SET @vs_prix   = '95'$

DROP PROCEDURE sp1$

CREATE PROCEDURE sp1()
BEGIN
DECLARE  v_nPoste  VARCHAR(7) DEFAULT 'p7';
DECLARE  v_dateAchat DATETIME;
SET AUTOCOMMIT = 0;
--Insère dans Logiciel
  INSERT INTO Logiciel
    VALUES (@vs_nLog,@vs_nomLog,NOW(),@vs_version,@vs_typeLog,@vs_prix,0) ;
  SELECT('Logiciel insere dans la base') "message1";
--récupère la date de l'achat
  SELECT dateach INTO v_dateAchat FROM Logiciel WHERE nLog = @vs_nLog;
  SELECT CONCAT('Date achat : ',v_dateAchat) "message2";
--On attend 5 petites secondes
  SELECT SLEEP(5);
--Insère dans Installer
  SELECT CONCAT('Date installation : ',SYSDATE()) "message3";
  INSERT INTO Installer (nPoste,nLog,dateIns,delai) VALUES
      (v_nPoste, @vs_nLog, SYSDATE(),TIMEDIFF(SYSDATE(),v_dateAchat));
 SELECT('Logiciel installe sur le poste') "message4";
 COMMIT;
END;
$
```

Chapitre 7

Curseur

```
delimiter $
DROP TABLE IF EXISTS test.Trace$
CREATE TABLE test.Trace(message VARCHAR(80))$
DROP PROCEDURE IF EXISTS calculTemps$
CREATE PROCEDURE calculTemps()
BEGIN
 DECLARE fincurs BOOLEAN DEFAULT 0;
 DECLARE v_nomLog   VARCHAR(20);
 DECLARE v_nomPoste VARCHAR(20);
 DECLARE v_dateAch  DATETIME;
 DECLARE v_dateIns  TIMESTAMP;
 DECLARE v_nLog     VARCHAR(5);
 DECLARE v_nPoste   VARCHAR(7);
--nb jours entier
 DECLARE v_attente  SMALLINT;
--nb jour décimal
 DECLARE v_jourdecimal DECIMAL(8,2);
--écriture en format "TIME étendu"
 DECLARE v_chainejour VARCHAR(30);

 DECLARE curseur CURSOR FOR
         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;
 DECLARE CONTINUE HANDLER FOR NOT FOUND SET fincurs := 1;
 OPEN curseur;
 FETCH curseur INTO v_nomLog,v_nomPoste,v_dateAch,v_dateIns,v_nLog,v_nPoste;
 WHILE (NOT fincurs) DO
  IF v_dateAch IS NULL THEN
     INSERT INTO test.Trace VALUES
      (CONCAT('Date d''achat inconnue pour le logiciel ',
               v_nomLog,' sur ',v_nomPoste));
  ELSE
     SET v_attente :=  DATEDIFF(v_dateIns,v_dateAch);
     IF v_attente < 0 THEN
        INSERT INTO test.Trace VALUES
```

```
                  (CONCAT('Logiciel ',v_nomLog,' installé sur ',
                   v_nomPoste,' ', -v_attente,' jour(s) avant l''achat!'));
      ELSE
         IF v_attente = 0 THEN
            INSERT INTO test.Trace VALUES (CONCAT(v_nomLog,' sur ',v_nomPoste,
                                     ' acheté et installé le même jour!')) ;
       ELSE
         INSERT INTO test.Trace VALUES
               (CONCAT('Logiciel ',v_nomLog,' sur ',v_nomPoste,
                 ' attente ',v_attente,' jour(s).'));
           SET  v_jourdecimal :=
                      TIMESTAMPDIFF(SECOND,v_dateAch,v_dateIns)/(24*3600);
           SET v_chainejour :=
            CONCAT(SIGN(v_jourdecimal) * FLOOR(ABS(v_jourdecimal))," j ",
            SEC_TO_TIME((ABS(v_jourdecimal)-FLOOR(ABS(v_jourdecimal)))* 86400));
         INSERT INTO test.Trace VALUES
                  (CONCAT('En format TIME étendu ', v_chainejour));
         UPDATE Installer SET delai = v_jourdecimal
               WHERE  nPoste = v_nPoste AND nLog = v_nLog;
         END IF;
      END IF;
 END IF;
 FETCH curseur INTO v_nomLog,v_nomPoste,v_dateAch,v_dateIns,v_nLog,v_nPoste;
 END WHILE;
 CLOSE curseur;
 SELECT * FROM test.Trace;
END;
$
--Test et appel
UPDATE Installer SET delai = NULL$
SELECT * FROM Installer$
DELETE FROM test.Trace$
CALL calculTemps()$
--Vérification
SELECT * FROM Installer$
```

Transaction

```
delimiter $
DROP TABLE IF EXISTS test.Trace$
CREATE TABLE test.Trace(message VARCHAR(80))$
DROP PROCEDURE IF EXISTS installLogSeg$
CREATE  PROCEDURE  installLogSeg  (IN  param1  VARCHAR(11),  IN  param2  VARCHAR(5),IN  param3
VARCHAR(20),  IN  param4  TIMESTAMP,  IN  param5  VARCHAR(7),  IN  param6  VARCHAR(9),  IN  param7
DECIMAL(6,2))
BEGIN
 DECLARE fincurs    BOOLEAN DEFAULT 0;
 DECLARE v_nomPoste VARCHAR(20);
 DECLARE v_nomSalle VARCHAR(20);
 DECLARE v_nPoste   VARCHAR(7);
 DECLARE curseur    CURSOR FOR
         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;
 DECLARE CONTINUE HANDLER FOR NOT FOUND SET fincurs := 1;

 SET AUTOCOMMIT = 0;
 INSERT INTO Logiciel VALUES (param2,param3,param4,param5,param6,param7,0);
 INSERT INTO test.Trace VALUES
             (CONCAT(param3,' stocké dans la table Logiciel'));
 OPEN curseur;
 FETCH curseur INTO v_nomPoste,v_nPoste,v_nomSalle;
 WHILE (NOT fincurs) DO
   INSERT INTO Installer (nPoste,nLog,delai)
         VALUES(v_nPoste,
                param2, TIMESTAMPDIFF(SECOND,param4,SYSDATE())/(24*3600) );
   INSERT INTO test.Trace VALUES
                (CONCAT('Installation sur ',v_nomPoste,' dans ',v_nomSalle));
   FETCH curseur INTO v_nomPoste,v_nPoste,v_nomSalle;
 END WHILE;
 CLOSE curseur;
 COMMIT;
 SELECT * FROM test.Trace;
END;
$
CALL installLogSeg('130.120.80', 'log99','Blaster', '2005-09-05', '9.9', 'PCWS', 999.9 )$
SELECT * FROM Logiciel$
SELECT * FROM Installer WHERE nLog='log99'$
--
DELETE FROM Installer WHERE nLog='log99'$
DELETE FROM Logiciel WHERE nLog='log99'$
```

Exceptions

```
delimiter $
DROP TABLE IF EXISTS test.Trace$
CREATE TABLE test.Trace(message VARCHAR(80))$
DROP PROCEDURE IF EXISTS installLogSeg$
CREATE  PROCEDURE  installLogSeg  (IN  param1  VARCHAR(11),  IN  param2  VARCHAR(5),IN  param3
VARCHAR(20),  IN  param4  TIMESTAMP,  IN  param5  VARCHAR(7),  IN  param6  VARCHAR(9),  IN  param7
DECIMAL(6,2))
BEGIN
 DECLARE fincurs    BOOLEAN DEFAULT 0;
 DECLARE doublonTrouve  BOOLEAN DEFAULT 0;
 DECLARE pereInexistant BOOLEAN DEFAULT 0;
 DECLARE nbrInstall TINYINT DEFAULT 0;
 DECLARE v_nomSeg   VARCHAR(20);
 DECLARE v_nomPoste VARCHAR(20);
 DECLARE v_nomSalle VARCHAR(20);
 DECLARE v_nPoste   VARCHAR(7);
 DECLARE curseur    CURSOR FOR
         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;
 DECLARE CONTINUE HANDLER FOR NOT FOUND SET fincurs  := 1;
 DECLARE CONTINUE HANDLER FOR 1062 SET doublonTrouve := 1;
 DECLARE CONTINUE HANDLER FOR 1452 SET pereInexistant := 1;
 --  numéro de segment inconnu?
 SELECT nomSegment INTO v_nomSeg FROM Segment WHERE indIP=param1;
 IF (fincurs) THEN
    INSERT INTO test.Trace VALUES (CONCAT('Mauvais code segment : ',param1));
 ELSE
 --  numéro de logiciel déjà présent ?
  SET AUTOCOMMIT = 0;
  INSERT INTO Logiciel VALUES (param2,param3,param4,param5,param6,param7,0);
  IF (doublonTrouve) THEN
     INSERT INTO test.Trace VALUES
               (CONCAT('Logiciel : ',param2,' déjà présent!'));
  ELSE
    IF (pereInexistant) THEN
     INSERT INTO test.Trace VALUES
                (CONCAT('Type du logiciel : ',param6,' non référencé!'));
    ELSE
     IF (DATEDIFF(SYSDATE(),param4)<0) THEN
      INSERT INTO test.Trace VALUES (CONCAT('Date achat plus grande que celle du jour!'));
     ELSE
      INSERT INTO test.Trace VALUES
                    (CONCAT(param3,' stocké dans la table Logiciel'));
      OPEN curseur;
      FETCH curseur INTO v_nomPoste,v_nPoste,v_nomSalle;
      WHILE (NOT fincurs) DO
      SET nbrInstall := nbrInstall +1;
        INSERT INTO Installer (nPoste,nLog,delai)
              VALUES(v_nPoste, param2,
                     TIMESTAMPDIFF(SECOND,param4,SYSDATE())/(24*3600) );
        INSERT INTO test.Trace VALUES
                 (CONCAT('Installation sur ',v_nomPoste,' dans ',v_nomSalle));
        FETCH curseur INTO v_nomPoste,v_nPoste,v_nomSalle;
      END WHILE;
      IF (nbrInstall=0) THEN
       INSERT INTO test.Trace VALUES
         (CONCAT('Aucune installation sur le segment',param1,' de ',param2));
      END IF;
      CLOSE curseur;
     END IF;
    END IF;
  END IF;
 END IF;
 SELECT * FROM test.Trace;
END;
$
--test segment
DELETE FROM test.Trace$
CALL installLogSeg('toto', 'log99','Blaster', '2005-09-05', '9.9', 'PCWS', 999.9)$
SELECT * FROM test.Trace$
--test logiciel déjà présent
--ERROR 1062 (23000): Duplicate entry 'log1' for key 1
DELETE FROM test.Trace$
CALL installLogSeg('130.120.80', 'log1','Blaster', '2005-09-05', '9.9', 'PCWS', 999.9)$
--test type du logiciel
DELETE FROM test.Trace$
CALL installLogSeg('130.120.80', 'log98','Mozilla', '2005-11-04', '1', 'toto', 100.0)$
--date d'achat plus grande que celle du jour ?
-- DATEDIFF(v_dateIns,v_dateAch);
DELETE FROM test.Trace$
CALL installLogSeg('130.120.80', 'log98','Mozilla', '2010-11-04', '1', 'PCWS', 100.0)$
--aucune install
DELETE FROM test.Trace$
CALL installLogSeg('130.120.81', 'log55','Eudora', '2005-12-06', '5', 'PCWS', 540)$
--bonne installation
DELETE FROM test.Trace$
CALL installLogSeg('130.120.80', 'log77','Blog Up', '2005-12-05', '1.3', 'PCWS', 90)$
SELECT * FROM Logiciel$
SELECT * FROM Installer WHERE nLog='log77'$
```

Déclencheurs

Mises à jour de colonnes

```
CREATE TRIGGER Trig_AD_Installer AFTER DELETE ON Installer FOR EACH ROW
BEGIN
  UPDATE Poste SET nbLog=nbLog - 1 WHERE nPoste = OLD.nPoste;
  UPDATE Logiciel SET nbInstall = nbInstall - 1 WHERE nLog = OLD.nLog;
END;
$



CREATE TRIGGER Trig_AI_Installer AFTER INSERT ON Installer FOR EACH ROW
BEGIN
 UPDATE Poste SET nbLog = nbLog + 1 WHERE nPoste = NEW.nPoste;
 UPDATE Logiciel SET nbInstall = nbInstall + 1 WHERE nLog = NEW.nLog;
END;
$

CREATE TRIGGER Trig_AI_Poste AFTER INSERT ON Poste FOR EACH ROW
BEGIN
 UPDATE Salle SET nbPoste=nbPoste+1 WHERE nSalle = NEW.nSalle;
END;
$

CREATE TRIGGER Trig_AD_Poste AFTER DELETE ON Poste FOR EACH ROW
BEGIN
  UPDATE Salle SET nbPoste = nbPoste - 1 WHERE nSalle = OLD.nSalle;
END;
$

CREATE TRIGGER Trig_AU_Salle AFTER UPDATE ON Salle FOR EACH ROW
```

```
BEGIN
 DECLARE differ TINYINT(2);
 SET differ := NEW.nbPoste - OLD.nbPoste;
 UPDATE Segment SET nbPoste = nbPoste + differ WHERE indIP = NEW.indIP;
END;
$
```

Programmation de contraintes

```
DROP TABLE IF EXISTS test.Trace$
CREATE TABLE test.Trace(col VARCHAR(80) PRIMARY KEY)$

CREATE TRIGGER Trig_BI_Installer BEFORE INSERT ON Installer FOR EACH ROW
BEGIN
DECLARE v_type_log   VARCHAR(9);
DECLARE v_type_pos   VARCHAR(9);
DECLARE v_date_achat DATETIME;
 SELECT typeLog, dateAch INTO v_type_log,v_date_achat
        FROM Logiciel WHERE NEW.nLog = nLog;
 SELECT typePoste INTO v_type_pos
        FROM Poste WHERE NEW.nPoste = nPoste;
 IF NOT (v_type_log = v_type_pos) THEN
 -- Les types ne correspondent pas : on fait planter...
    INSERT INTO test.Trace VALUES (NULL);
 END IF;
 IF NEW.dateIns IS NOT NULL THEN
     IF DATEDIFF(NEW.dateIns,v_date_achat) < 0 THEN
    -- Installation antérieure a la date achat
       INSERT INTO test.Trace VALUES (NULL);
     END IF;
 END IF;
END;
$
```

Chapitre 8

Curseur statique

```
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;
            }
--main()
…
```

```
        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]);}
        …
```

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 deleteSalleSP(String ns) {
              int result = 0;
              try {cetat = cx.prepareCall("{call supprimeSalle(?,?)}");
                     cetat.registerOutParameter(2,java.sql.Types.INTEGER);
                     cetat.setString(1,ns);
                     cetat.execute();
                     result = cetat.getInt(2);
                     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 9

Extraction préparée (`exo1suite.php`)

```
<html> <head> <title>Installations d'une salle</title> </head>
<body>
<?php
if ( ($service = mysqli_connect('localhost','soutou','iut','bdsoutou')) > 0)
 {
 $numsalle = $_POST['ns'];
 $requete = "SELECT l.nomLog,i.nPoste,i.dateins,l.dateAch FROM bdsoutou.Installer i, bdsoutou.Logiciel l, bdsoutou.Poste p WHERE l.nLog=i.nLog AND p.nPoste=i.nPoste AND p.nSalle='$numsalle' ORDER BY 1,2,3";
 $ordre = mysqli_prepare($service,$requete);
 if ( ($res = mysqli_stmt_execute($ordre)) > 0)
  {
   if ( ($resbind = mysqli_stmt_bind_result($ordre,$v1,$v2,$v3,$v4)) > 0)
     {
    print "<H4>Liste des Installation de la salle $numsalle</H4>";
    $trouve=0;
    print "<TABLE BORDER=1> ";
    print "<tr><th>Nom Logiciel</th><th>Poste</th><th>Installation
               </th><th>Achat</th></tr>";
    while  (mysqli_stmt_fetch($ordre))
          {
          $trouve=1;
          print "<TR> <TD> $v1</TD>" ;
          print "     <TD> $v2</TD>";
          print "     <TD> $v3</TD>";
          print "     <TD> $v4</TD> </TR> ";
          }
    print "</TABLE> ";
    if ($trouve==0)
         print "<BR>Aucune installation dans la salle";
  }
  else print "<BR>La liaison est un échec!";
 }
 else print "<BR>La requete est un échec!";
 mysqli_stmt_close($ordre);
 mysqli_close($service);
 }
 else print "<BR> La connexion est un échec!";
?>
</body> </html>
```

Appel d’un sous-programme (exo2suite.php)

```
<html> <head> <title>Suppression d'une salle</title> </head>
<body>
<?php
if ( ($service = mysqli_connect('localhost','soutou','iut','bdsoutou')) > 0)
{ mysqli_autocommit($service,FALSE);
 $numsalle = $_POST['ns'];
 if ($result = mysqli_multi_query($service,
            "call bdsoutou.supprimeSalle('$numsalle',@v_retour)") > 0)
 {print "<BR>Procédure réalisée correctement.";
  if ($result2 = mysqli_query($service,"SELECT @v_retour"))
       {$ligne = mysqli_fetch_array($result2, MYSQLI_NUM);
        if  ($ligne[0] == -1)
            print "<BR>Désolé, la salle $numsalle n'existe pas!";
        if  ($ligne[0] == 0)
            print "<BR>La salle $numsalle est supprimée";
        if  ($ligne[0] == -2)
            print "<BR>Désolé, la salle $numsalle est référencée par un poste de travail!";
        mysqli_free_result($result2);
       }
     else
       { print "<BR>Problème au retour du paramètre ".mysqli_error($service); }
  }
 else
 { print "<BR>La procédure est un échec! ".mysqli_error($service);  }
  mysqli_close($service);
 }
 else print "<BR> La connexion est un échec!";
?>
</body> </html>
```

Insertion préparée (exo3suite.php)

```
<html> <head> <title>Ajout d'une installation</title> </head>
<body>
<?php
if ( ($service = mysqli_connect('localhost','soutou','iut','bdsoutou')) > 0)
{mysqli_autocommit($service,FALSE);
 $numposte = $_POST['np'];
 $numlogi  = $_POST['nl'];
 $insert3    =    "INSERT    INTO    bdsoutou.Installer    (nPoste,nLog,dateIns,delai)    VALUES
(?,?,SYSDATE(),NULL)";
 $ordre = mysqli_prepare($service, $insert3);
 if ( (mysqli_stmt_bind_param($ordre,'ss',$numposte,$numlogi)) > 0)
 {if ( ($res = mysqli_stmt_execute($ordre)) > 0)
     {print    "<BR>Enregistrement    ($numlogi,    $numposte)    inséré,    (en    sequence    :
".mysqli_insert_id($service).")";
     mysqli_commit($service);
     mysqli_stmt_free_result($ordre);
     }
    else
     {print "<BR>L'insertion de $numlogi sur $numposte est un échec!";
      print "<BR><B>Message : </B>".mysqli_stmt_error($ordre);
      print "<BR><B>Code : </B>".mysqli_stmt_errno($ordre);
     }
 }
 else print "<BR>Problème au bind!";
 mysqli_close($service);
}
 else print "<BR> La connexion est un échec!";
?>
</body> </html>
```