Chapitre 9

Le langage SQL avancé

Ce document reprend les requêtes SQL du chapitre 9 de l'ouvrage *Bases de données* - *Concepts, utilisation et développement*.

9.1 INTRODUCTION

9.2 LE CONTRÔLE D'ACCÈS

```
grant select, update(QSTOCK,PRIX)
on    PRODUIT
to    P_MERCIER, S_FINANCIERS;

grant all privileges
on    CLIENT
to    P_MERCIER, S_FINANCIERS
with  grant option;

grant select
on    COM_COMPLETE
to    public;
grant run
on    SUP_DETAIL
to    public;

grant run
on    COMPTA01
to    S_FINANCIERS
```

```
with  grant option;

revoke update(PRIX)
on     PRODUIT
from     P_MERCIER;

revoke run
on     COMPTA01
from   P_MERCIER;

revoke grant option for update (COMPTE)
on CLIENT
from P_MERCIER;
```

a) Les rôles

```
create role CONSULTANT;
grant select on CLIENT to CONSULTANT;
grant update (ADRESSE, LOCALITE) on CLIENT to CONSULTANT;
grant CONSULTANT to P_MERCIER;

revoke select (LOCALITE) from CONSULTANT;
revoke CONSULTANT from P_MERCIER;
drop role CONSULTANT;
```

9.3 LES VUES SQL

9.3.1 Principes et objectifs des vues

9.3.2 Définition et utilisation d'une vue

```
create view COM_COMPLETE(NCOM,NCLI,NOMCLI,LOC,DATECOM)
as  select NCOM,COM.NCLI,NOM,LOCALITE,DATECOM
    from   CLIENT CLI,COMMANDE COM
    where  COM.NCLI = CLI.NCLI;

select NOMCLI,NCOM,DATECOM
from   COM_COMPLETE
where  LOC = 'Toulouse';

select NPRO
from   DETAIL
where  NCOM in ( select NCOM
                 from   COM_COMPLETE
                 where  LOC = 'Toulouse');

select LOC, count(*)
from   COM_COMPLETE CC, DETAIL D
where  CC.NCOM = D.NCOM
group by LOC;
```

```
select NOM,NCOM,DATECOM
from   CLIENT CLI,COMMANDE COM
where  COM.NCLI = CLI.NCLI
and    LOC = 'Toulouse';
```

```
drop view COM_COMPLETE;
```

9.3.3 Les vues comme interface pour des besoins particuliers

```
create view HABITUDE_ACHAT(LOCALITE,NPRO,VOLUME)
as  select LOCALITE,P.NPRO,sum(QCOM*PRIX)
    from   CLIENT CLI,COMMANDE COM,DETAIL D,PRODUIT P
    where  COM.NCLI = CLI.NCLI
    and    D.NCOM = COM.NCOM
    and    P.NPRO = D.NPRO
    group by LOCALITE, P.NPRO;
```

9.3.4 Les vues comme mécanisme de contrôle d'accès

```
create view ANALYSE(LOCALITE, CAT, DATE, NPRO, QCOM) as
select LOCALITE, CAT, DATECOM, NPRO, QCOM
from   CLIENT C, COMMANDE M, DETAIL D
where  C.NCLI = M.NCLI and M.NCOM = D.NCOM;
```

9.3.5 Les vues comme mécanisme d'évolution de la base de données

9.3.6 Les vues comme aide à l'expression de requêtes complexes

```
create view VAL_STOCK(STOCK,VALEUR) as
select P.NPRO, (QSTOCK - sum(D.QCOM))*PRIX
from   DETAIL D, PRODUIT P
where  D.NPRO = P.NPRO
group by P.NPRO, QSTOCK, PRIX;[1]
```

```
select sum(VALEUR) from VAL_STOCK;
```

9.3.7 Mise à jour des données via une vue

```
create view CLI(NCLI,NOM,ADRESSE,LOCALITE,CAT,COMPTE) as
select *
from   CLIENT
where  CAT is null or CAT in ('B1','B2','C1','C2')
with check option;
```

```
insert into CLI
values ('B313','DURAND','place Monge','Mons','D7',0);
```

1. Pourquoi QSTOCK et PRIX ?

9.4 EXTENSION DE LA STRUCTURE DES REQUÊTES SFW

9.4.1 Extension de la clause *select*

```
select NCOM, (select sum(QCOM*PRIX)
              from   DETAIL D, PRODUIT P
              where  D.NPRO = P.NPRO
              and    D.NCOM = M.NCOM) as MONTANT
from   COMMANDE M
where  MONTANT > 1000;
```

9.4.2 Extension de la clause *from*

```
select NCLI, NOM
from ((select NCLI, NOM, LOCALITE from CLIENT)
       except
      (select NCLI, NOM, LOCALITE from BON_CLIENT)
       union
      (select NCLI, NOM, LOCALITE from PROSPECT))
where LOCALITE = 'Poitiers';
```

```
select avg(MONTANT)
from  (select NCOM, sum(QCOM*PRIX) as MONTANT
       from   DETAIL D, PRODUIT P
       where  D.NPRO = P.NPRO
       group by NCOM);
```

```
select NPRO, TOTAL_QTE
from  ( (select NPRO, sum(QCOM)
         from   PRODUIT P, DETAIL D
         where  P.NPRO=D.NPRO
         group by NPRO)
         union
        (select NPRO, 0
         from   PRODUIT
         where  NPRO not in (select NPRO from DETAIL))
      )
         as DP(NPRO,TOTAL_QTE)
where TOTAL_QTE < 1000;
```

```
select *
from   CLIENT cross join COMMANDE
where  ...
```

```
select *
from   CLIENT, COMMANDE
where  ...
```

```
select *
from   CLIENT natural join COMMANDE
where  ...
```

```
select *
```

```
from   CLIENT C, COMMANDE M
where  C.NCLI = M.NCLI
and    ...
```

```
select *
from   CLIENT C join COMMANDE M
       on (C.NCLI = M.NCLI)
where  ...
```

```
select *
from   CLIENT C join COMMANDE M on (C.NCLI = M.NCLI)
       join DETAIL D on (M.NCOM = D.NCOM)
       join PRODUIT P on (D.NPRO = P.NPRO)
where  ...
```

```
select *
from   CLIENT C,COMMANDE M, DETAIL D,PRODUIT P
where  C.NCLI = M.NCLI
and    M.NCOM = D.NCOM
and    D.NPRO = P.NPRO;
```

```
select *
from   CLIENT join COMMANDE using (NCLI)
where  ...
```

```
from   CLIENT inner join COMMANDE using (NCLI)
```

```
select NCOM, C.NCLI, DATECOM, NOM, LOCALITE
from   COMMANDE M, CLIENT C
where  M.NCLI = C.NCLI
union
select null, NCLI, null, NOM, ADRESSE
from   CLIENT
where  NCLI not in (select NCLI from COMMANDE);
```

```
select NCOM, C.NCLI, DATECOM, NOM, LOCALITE
from   COMMANDE M right outer join CLIENT C
       on (M.NCLI = C.NCLI);
```

```
select NCOM, C.NCLI, DATECOM, NOM, LOCALITE
from   COMMANDE M, CLIENT C
where  M.NCLI = (+) C.NCLI;
```

```
select NCLI,NOM
from   CLIENT
where  NCLI not in (select NCLI from COMMANDE);
```

```
select NCLI,NOM
from   CLIENT C
where  not exists (select * from COMMANDE
                   where NCLI = C.NCLI);
```

```
select C.NCLI,NOM
from   CLIENT C left outer join COMMANDE M
       on (C.NCLI = M.NCLI)
where  M.NCOM is null;
```

9.5 LES REQUÊTES RÉCURSIVES

```
with ORGAN (NIVEAU, NPERS, NOM, RESP)
as (
-- Initialisation (E0)
    select 1, NPERS, NOM, RESPONSABLE
    from   PERSONNE
    where  NOM = 'p4'
    union all
-- Incrémentation (Ei)
    select O.NIVEAU + 1, P.NPERS, P.NOM, P.RESPONSABLE
    from   ORGAN O, PERSONNE P
    where  P.RESPONSABLE = O.NPERS
    )
-- Elaboration du résultat
select NIVEAU, NPERS, NOM, RESP from ORGAN;
```

9.6 LES EXTENSIONS OBJET DE SQL3

9.6.1 Types de données complexes (`row` et `array`)

```
create table CLIENT(
   NCLI    char(10) not null primary key,
   NOM     char(32) not null,
   ADRESSE row(RUE char(30), LOCALITE char(60)),
   CAT     char(2));

select NCLI, NOM, ADRESSE.RUE
from   CLIENT
where  ADRESSE.LOCALITE = 'Poitiers';

create table CLIENT2(
   NCLI     char(10) not null primary key,
   NOM      char(32) not null,
   PRENOM   char(15) array(4),
   ADRESSES row(RUE char(30), LOCALITE char(60)) array(2),
   CAT      char(2));

select NCLI, NOM, PRENOM[1], PRENOM[2], ADRESSES[1].RUE
from   CLIENT2
where  ADRESSES[1].LOCALITE = 'Poitiers';
```

9.6.2 Type défini par l'utilisateur (TDU)

```
create type Chaine as varchar(60) default '?';
create type Contact as (RUE Chaine, LOCALITE Chaine);

create table CLIENT(
   NCLI    char(10) not null primary key,
   NOM     Chaine not null,
   ADRESSE Contact,
   CAT     char(2));
```

9.6.3 Table typée

```
create type TPERSONNE as (
   NCLI     char(10),
   NOM      Chaine,
   ADRESSE  Contact,
   LOCALITE char(2));

create table CLIENT of TPERSONNE;

create table PROSPECT of TPERSONNE;
```

9.6.4 Hiérarchie de types

```
create type TPERSONNE as (
   NCLI     char(10),
   NOM      Chaine,
   ADRESSE  Contact,
   LOCALITE char(2))
   REF is system generated;

create type TCLIENT under TPERSONNE as (
   CAT    char(2),
   COMPTE decimal (9,2));
```

9.6.5 Hiérarchie de tables typées

```
create table PERSONNE of TPERSONNE
(REF is IdP system generated,
 NCLI     with options not null,
 NOM      with options not null,
 ADRESSE  with options not null,
 LOCALITE with options not null,
 primary key (NCLI));

create table CLIENT of TCLIENT under PERSONNE
(COMPTE   with options not null default 0
                       check(COMPTE >= 0));

select NCLI, NOM, ADRESSE
```

```
from   PERSONNE
where  LOCALITE = 'Poitiers';

from   only(PERSONNE)
```

9.6.6 Références entre tables

```
create table COMMANDE (
   NCOM char(10) not null primary key,
   DATECOM  date not null,
   REFCLI REF(TCLIENT) scope CLIENT not null);

select NCOM, DATECOM, REFCLI->NCLI, REFCLI->NOM
from   COMMANDE
where  REFCLI->LOCALITE = 'Poitiers';
```

9.7 LES PRÉDICATS (*check*)

```
create table CLIENT ( NCLI  ...,
                      ...,
                      CAT   char(2),
             primary key (NCLI),
             check (CAT is null or
                    CAT in ('B1','B2','C1','C2'));

alter table CLIENT
add check (CAT is null or CAT in ('B1','B2','C1','C2'));

alter table CLIENT
add constraint CHK_CAT
check (CAT is null or CAT in ('B1','B2','C1','C2'));

alter table COMMANDE
add check  ((DATECOM >= (select max(DATECOM)
                        from COMMANDE)
             and DATECOM <= CURRENT_DATE) is not false);

alter table COMMANDE
add check (NCLI in (select NCLI from CLIENT));

alter table CLIENT
drop constraint CHK_CAT;

CAT char(2) check(CAT is null or
                  CAT in ('B1','B2','C1','C2'))

create domain MONTANT integer check(value >= 0);
```

9.8 LES PROCÉDURES SQL (*stored procedures*)

```
create procedure SUP_DETAIL (in COM char(12),
                             in PRO char(15))
begin
    delete from DETAIL
    where NCOM = :COM and NPRO = :PRO;
    if (select count(*) from DETAIL
        where NCOM=:COM) = 0
    then delete from COMMANDE
         where NCOM = :COM
    end if;
end;

call SUP_DETAIL('30182','PA60');
```

9.9 LES DÉCLENCHEURS (T*riggers*)

```
create trigger SUP_COM
after update of CAT on CLIENT
for each row
when (new.COMPTE < 0)
begin
  if (old.CAT is not null and new.CAT is null)
     then delete from COMMANDE where NCLI = old.NCLI;
  end if;
end;
```

```
before insert on T
for each row
when not C
begin
   abort();
end;
```

9.10 LE CATALOGUE

```
select  CNAME, CTYPE, LEN1, NULLS
from    SYS_COLUMN
where   TNAME = 'DETAIL';

select TNAME
from   SYS_TABLE
where  TNAME  in (select TNAME
                  from   SYS_COLUMN
                  where  CNAME like 'NCOM%')
```

```
and    TTYPE = 'R';
```

```
select distinct CREATOR
from   SYS_TABLE
where  TNAME in
             (select TNAME
              from   SYS_KEY
              where  KEYID in
                           (select KTARG
                            from   SYS_KEY
                            where  TNAME = 'DETAIL'));
```

9.11 LES INTERFACES SQL DES PROGRAMMES D'APPLICATION

```
select NOM, LOCALITE
from   CLIENT
where  NCLI = 'C400';
```

9.11.1 L'interface SQL statique classique

```
select NOM, LOCALITE into :varNOM, :varLOC
from   CLIENT
where  NCLI = :varNCLI;
```

```
exec SQL  begin declare section;
```

```
exec SQL  end declare section;
```

```
exec SQL  include SQLCA;
```

```
exec SQL begin declare section;
  varNOM    char(32);
  varLOC    char(30);
  varNCLI   char(10);
exec SQL end declare section;
exec SQL include SQLCA;
read varNCLI;
exec SQL
  select NOM,LOCALITE into :varNOM,:varLOC
  from   CLIENT
  where  NCLI = :varNCLI;
write varNOM, varLOC;
```

```
exec SQL declare CURCLI cursor for
     select NCLI, NOM
     from   CLIENT
     where  LOCALITE = :varLOC;
```

```
varLOC := "Toulouse";
exec SQL open CURCLI;
exec SQL fetch CURCLI into :varNCLI,:varNOM;
while SQLCODE = 0 do
   <traiter les valeurs de varNCLI et varNOM>
   exec SQL fetch CURCLI into :varNCLI,:varNOM;
endwhile;
exec SQL close CURCLI;
```

```
exec SQL declare CURCLI cursor for
     select NCLI, NOM from CLIENT where LOCALITE = :varLOC;
varLOC := "Toulouse";
exec SQL open CURCLI;
exec SQL fetch CURCLI into :varNCLI,:varNOM;
while SQLCODE = 0 do
   if STAT > 0 then
      exec SQL
         update CLIENT
         set CAT = 'A1'
         where current of CURCLI;
   endif
   exec SQL fetch CURCLI into :varNCLI,:varNOM;
endwhile;
exec SQL close CURCLI;
```

9.11.2 SQLJ : une interface statique pour Java

```
String varNCLI, varNOM, varLOC;
varNCLI = "C400";
#sql{select NOM, LOCALITE into :varNOM, :varLOC
     from   CLIENT where NCLI = :varNCLI"};
<traiter les valeurs de varNOM et varLOC>
```

```
#sql iterator IterateurClient(String NCLI, String NOM);   [1]
IterateurClient CurCli;                                    [2]
String varLOC = "Poitiers";
#sql CurCli = {select NCLI, NOM from CLIENT
               where LOCALITE = :varLOC};                  [3]
while (CurCli.next()) {                                    [4]
      varNCLI = CurCli.NCLI();                             [5]
      varNOM  = CurCli.NOM();                              [5]
      <traiter les valeurs de varNCLI et varNOM>
}
```

```
CurCli.close();                                           [7]

#sql iterator IterateurClient(String, String);            [1]
IterateurClient CurCli;                                   [2]
varLOC = "Poitiers";
#sql CurCli = {select NCLI, NOM from CLIENT
               where LOCALITE = :varLOC};                 [3]
#sql {fetch :CurCli into :varNCLI, :varNOM};              [6]
while (!CurCli.endFetch()) {
      <traiter les valeurs de varNCLI et varNOM>
      #sql {fetch :CurCli into :varNCLI, :varNOM};        [6]
}
CurCli.close();                                           [7]

varCAT = "C2"; varNCLI = "C400";
#sql {update CLIENT set CAT = :varCAT
      where NCLI = :varNCLI};
```

9.11.3 L'interface SQL dynamique classique

```
varNCLI   char(10);
exec SQL include SQLCA;
exec SQL begin declare section;
   varNOM    char(32);
   varLOC    char(30);
   Requete   char(250);
exec SQL end declare section;
varNCLI := "C400";                                        [C]
Requete := "select NOM, LOCALITE from CLIENT "
        + "where  NCLI = '" + varNCLI + "'";              [C]
exec SQL prepare Q from :Requete;                         [P]
exec SQL execute Q into :varNOM, :varLOC;                 [E]
```

```
exec SQL execute immediate from :Requete
into :varNOM, varLOC;                                     [PE]
```

```
varNCLI := "C400";                                        [C]
Requete := "select NOM, LOCALITE from CLIENT "
        + "where  NCLI = ?";                              [C]
exec SQL prepare Q from :Requete;                         [P]
exec SQL execute Q using :varNCLI into :varNOM, :varLOC;  [E]
```

```
var1 := "select ";
var2 := "NOM, ADRESSE";
var3 := "CLIENT";
var4 := "LOCALITE = ?";
Requete := var1 + var2 + " from " + var3 + " where " + var4;
```

```
Requete := "select NCLI, NOM from CLIENT "
        + "where  LOCALITE = ?";
exec SQL declare CURCLI cursor for inst;
exec SQL prepare inst from :Requete;
varLOC := "Poitiers";
exec SQL open CURCLI using :varLOC;
exec SQL fetch CURCLI into :varNCLI, :varNOM;
while SQLCODE = 0 do
    <traiter les valeurs de varNCLI et varNOM>
    exec SQL fetch CURCLI into :varNCLI, :varNOM;
endwhile;
exec SQL close CURCLI;
```

```
Requete := "insert into DETAIL values (?, ?, ?)";
exec SQL prepare Q from :Requete;
exec SQL execute Q using :valNCOM, :valNPRO, :varQCOM;
```

```
Requete = "insert into DETAIL values ('"
         + valNCOM + "','" + valNPRO + "'," + valQCOM + ")";
exec SQL execute immediate from :Requete;
```

```
exec SQL execute immediate from
         "insert into DETAIL values ('"
        + valNCOM + "','" + valNPRO + "'," + valQCOM + ")";
```

9.11.4 JDBC : une interface CLI-SQL dynamique pour Java

```
String locBD, login, motDePasse;
String requete, varNCLI, varNOM, varLOC;
Connection conn;
PreparedStatement inst;
ResultSet res;
urlBD = <adresse de la BD>; login = "jlh"; motPasse = ..; [1]
conn = DriverManager.getConnection(urlBD,login,motPasse); [1]
```

```
requete = "select NOM, LOCALITE from CLIENT "
        + "where NCLI = ?";                                [2]
inst = conn.prepareStatement(requete);                     [3]
varNCLI = "C400";                                          [4]
inst.setString(1, varNCLI);                                [4]
res = inst.executeQuery();                                 [5]
if (res.next()){                                           [6]
    varNOM = res.getString(1)[2];                          [7]
    varLOC = res.getString(2);
    <traiter les valeurs de varNOM et varLOC>
};
res.close();                                               [8]
```

```
Statement inst;
inst = conn.createStatement();                             [3]
varNCLI = "C400";                                          [4]
requete = "select NOM, LOCALITE FROM CLIENT where NCLI = '"
         + varNCLI + "'";                                  [4]
res = inst.executeQuery(requete);                          [5]
if (res.next()){                                           [6]
    varNOM = res.getString(1);                             [7]
    varLOC = res.getString(2);
    <traiter les valeurs de varNOM et varLOC>
};
res.close();                                               [8]
```

```
requete = "select NCLI, NOM from CLIENT "
        + "where LOCALITE = ?";                            [2]
inst = conn.prepareStatement(requete);                     [3]
varLOC = "Poitiers";                                       [4]
inst.setString(1, varLOC);                                 [4]
res = inst.executeQuery();                                 [5]
while (res.next()){                                        [6]
    varNCLI = res.getString(1);                            [7]
    varNOM  = res.getString(2);
    <traiter les valeurs de varNCLI et varNOM>
};
res.close();                                               [8]
```

2. On peut aussi écrire `res.getString("NOM")`

```
requete = "update CLIENT set CAT = ? where NCLI = ?";
```

```
inst = con.prepareStatement(requete);
```

```
varCAT = "C2"; varNCLI = "C400";
inst.setInt(1, varCAT);
inst.setInt(2, varNCLI);
```

```
inst.executeUpdate();
```

9.11.5 Comparaison des modèles d'interaction

9.11.6 Un problème de sécurité : l'injection de code SQL

```
getForm(formID, varLogin, varPW);
requete = "select count(*) from  SYS_USERS "
        + " where ID_USER = '" + varLogin + "'"
        + " and PW = '" + varPW + "' ";
exec SQL execute  immediate  from  :requete  into :N;
if (N = 0) then accepte = False else accepte = True;
```

```
select count(*) into :N
from   SYS_USERS
where  ID_USER = 'Albert-Durant'
and    PW = 'A7cfg990';
```

```
select count(*) into :N
from   SYS_USERS
where  ID_USER = 'X'
or     'X' = 'X'
and    PW = ' '
or     'X' = 'X';
```

9.12 SQL ET L'INFORMATION INCOMPLÈTE

9.12.1 Introduction

9.12.2 La valeur `null` de SQL

9.12.3 La logique ternaire de SQL

- (CAT = 'B1') is true
- (CAT = 'B1') is not true
- (CAT = 'B1') is false
- (CAT = 'B1') is not false
- (CAT = 'B1') is unknown

- (CAT = ’B1’) is not unknown

9.12.4 La propagation de `null` en SQL

9.12.5 La propagation de unknown en SQL

9.12.6 Les problèmes de l’information incomplète en SQL

```
select NCLI
from CLIENT
where CAT = (select CAT
             from CLIENT
             where NCLI = 'K729');
```

```
select CAT, count(*)
from   CLIENT
group by CAT;
```

```
select distinct CAT from CLIENT order by CAT;
```

```
(CAT < 'C1') or (CAT = 'C1') or (CAT > 'C1')
```

```
select TITULAIRE, sum(H_COURS) + sum(H_TP) as CHARGE
from   ACTIVITE
group by TITULAIRE;
```

```
select TITULAIRE, sum(H_COURS + H_TP) as CHARGE
from   ACTIVITE
group by TITULAIRE;
```

```
(select sum(H_COURS) from ACTIVITE)
```

```
  (select sum(H_COURS) from ACTIVITE where P)
+ (select sum(H_COURS) from ACTIVITE where not P)
```

```
select NCLI,CAT
from   CLIENT C1
where not exists (select *
                  from   CLIENT C2
                  where  C2.CAT > C1.CAT);
```

9.12.7 Deux recommandations

```
create table CLIENT (NCLI .. not null primary key,
                     NOM .. not null,
                     ADRESSE .. not null,
                     LOCALITE .. not null,
                     COMPTE .. not null);
create table CLICAT( NCLI .. not null primary key,
                     CAT char(2) not null,
                     foreign key (NCLI) reference CLIENT);

create table CLIENT (NCLI .. not null primary key,
                     NOM .. not null,
                     ADRESSE .. not null,
                     LOCALITE .. not null,
                     CAT char(2) default 'A0' not null,
                     COMPTE .. not null);
```