Corrigé du R2.06-Exploitation BD (Séance n° 3)
Interrogations en sqlplus

Première étape : Expression des jointures

Q1 Donnez, pour l'étudiant Stéphane Rocchi, la liste par ordre décroissant des moyennes de test obtenues avec le code de la matière associé.

```
SELECT CODE, MOY_TEST
FROM ETUDIANT, NOTATION
WHERE NOM_ET = 'ROCCHI' AND PRENOM_ET = 'STEPHANE' AND
       NOTATION.NUM_ET = ETUDIANT.NUM_ET
ORDER BY MOY_TEST DESC ;

SELECT CODE, MOY_TEST
FROM NOTATION
WHERE NUM_ET IN (
            SELECT NUM_ET
            FROM ETUDIANT
            WHERE NOM_ET = 'ROCCHI' AND PRENOM_ET = 'STEPHANE')
ORDER BY MOY_TEST DESC ;

SELECT CODE, MOY_TEST
FROM ETUDIANT
JOIN NOTATION
ON NOTATION.NUM_ET = ETUDIANT.NUM_ET
WHERE NOM_ET = 'ROCCHI' AND RENOM_ET = 'STEPHANE'
ORDER BY MOY_TEST DESC  ;
```

Q2 Donnez la liste des matières (leur code et libellé) enseignées par Didier Boitard.

```
SELECT DISTINCT MODULE.CODE, LIBELLE
FROM MODULE, PROF, ENSEIGNT
WHERE MODULE.CODE = ENSEIGNT.CODE AND
       ENSEIGNT.NUM_PROF = PROF.NUM_PROF AND NOM_PROF = 'BOITARD' AND
       PRENOM_PROF = 'DIDIER' ;

SELECT CODE, LIBELLE
FROM MODULE
WHERE CODE IN (
            SELECT CODE
            FROM ENSEIGNT
            WHERE NUM_PROF IN (
                        SELECT NUM_PROF
                        FROM PROF
                        WHERE NOM_PROF = 'BOITARD' AND
                               PRENOM_PROF = 'DIDIER')) ;

SELECT DISTINCT MODULE.CODE, LIBELLE
FROM MODULE
JOIN ENSEIGNT
```

```
ON MODULE.CODE = ENSEIGNT.CODE
JOIN PROF
ON ENSEIGNT.NUM_PROF = PROF.NUM_PROF
WHERE NOM_PROF = 'BOITARD' AND PRENOM_PROF = 'DIDIER' ;
```

Remarque : dans la première et la dernière version de la requête, il y a autant de tuples résultats que d'étudiants ayant un enseignement de Didier Boitard. Il faut donc utiliser **DISTINCT** pour éliminer les modules dupliqués. Dans la deuxième version de la requête, le premier bloc n'opère que sur la relation MODULE dont CODE est la clef primaire. Il ne peut donc pas y avoir de duplicats et **DISTINCT** ***ne doit pas être utilisé***.

Q3 Quels sont les groupes de deuxième année pour lesquels Marc Laporte a effectué un enseignement ?

```
SELECT DISTINCT GROUPE
FROM ETUDIANT, ENSEIGNT, PROF
WHERE ANNEE = 2 AND NOM_PROF = 'LAPORTE' AND
       PRENOM_PROF = 'MARC' AND ETUDIANT.NUM_ET = ENSEIGNT.NUM_ET
       AND ENSEIGNT.NUM_PROF = PROF.NUM_PROF ;
```

```
SELECT DISTINCT GROUPE
FROM ETUDIANT
WHERE ANNEE = 2 AND NUM_ET IN
            (SELECT NUM_ET
            FROM ENSEIGNT, PROF
            WHERE ENSEIGNT.NUM_PROF = PROF.NUM_PROF AND
                  NOM_PROF = 'LAPORTE' AND PRENOM_PROF = 'MARC') ;
```

```
SELECT DISTINCT GROUPE
FROM ETUDIANT
WHERE ANNEE = 2 AND NUM_ET IN
            (SELECT NUM_ET
            FROM ENSEIGNT
            WHERE NUM_PROF IN
                    (SELECT NUM_PROF
                    FROM PROF
                    WHERE NOM_PROF = 'LAPORTE'
                    AND PRENOM_PROF = 'MARC')) ;
```

```
SELECT DISTINCT GROUPE
FROM ETUDIANT
JOIN ENSEIGNT
ON ETUDIANT.NUM_ET = ENSEIGNT.NUM_ET
JOIN PROF
ON ENSEIGNT.NUM_PROF = PROF.NUM_PROF
WHERE ANNEE = 2 AND NOM_PROF = 'LAPORTE' AND
       PRENOM_PROF = 'MARC' ;
```

Remarque : quelle que soit la version de la requête, il y a autant de valeurs résultats que d'étudiants ayant un enseignement de Marc Laporte. Il faut donc toujours utiliser **DISTINCT** pour éliminer les numéros de groupe dupliqués.

Q4 Donnez la liste, par ordre alphabétique, des noms de tous les étudiants ayant suivi un

enseignement effectué par un professeur, par ailleurs responsable d'un module.

```
SELECT DISTINCT NOM_ET, ETUDIANT.NUM_ET
FROM ETUDIANT, ENSEIGNT, MODULE
WHERE ETUDIANT.NUM_ET = ENSEIGNT.NUM_ET AND ENSEIGNT.NUM_PROF = RESP
ORDER BY NOM_ET ;

SELECT DISTINCT NOM_ET, ETUDIANT.NUM_ET
FROM ETUDIANT
JOIN ENSEIGNT
ON ETUDIANT.NUM_ET = ENSEIGNT.NUM_ET
JOIN MODULE
ON ENSEIGNT.NUM_PROF = RESP
ORDER BY NOM_ET ;
```

Remarque : un étudiant pouvant avoir plusieurs fois le même professeur pour des enseignements différents, le mot-clef **DISTINCT** est nécessaire pour éliminer les duplicats, cependant pour conserver les homonymes dans le résultat, il convient d'effectuer la projection de l'attribut clef primaire NUM_ET.

```
SELECT NOM_ET, NUM_ET
FROM ETUDIANT
WHERE NUM_ET IN
            (SELECT NUM_ET
            FROM ENSEIGNT, MODULE
            WHERE NUM_PROF = RESP)
ORDER BY NOM_ET ;
```

Q5 Donnez la liste, par ordre alphabétique, des noms de tous les étudiants ayant suivi l'enseignement d'une matière effectué par le professeur responsable de cette matière.

```
SELECT DISTINCT NOM_ET, ETUDIANT.NUM_ET
FROM ETUDIANT, ENSEIGNT, MODULE
WHERE ETUDIANT.NUM_ET = ENSEIGNT.NUM_ET AND
       ENSEIGNT.CODE = MODULE.CODE AND ENSEIGNT.NUM_PROF = RESP
ORDER BY NOM_ET ;

SELECT NOM_ET, NUM_ET
FROM ETUDIANT
WHERE NUM_ET IN
            (SELECT NUM_ET
            FROM ENSEIGNT, MODULE
            WHERE NUM_PROF = RESP AND ENSEIGNT.CODE = MODULE.CODE)
ORDER BY NOM_ET ;

SELECT DISTINCT NOM_ET, ETUDIANT.NUM_ET
FROM ETUDIANT
JOIN ENSEIGNT
ON ETUDIANT.NUM_ET = ENSEIGNT.NUM_ET
JOIN MODULE
ON ENSEIGNT.CODE = MODULE.CODE AND ENSEIGNT.NUM_PROF = RESP
ORDER BY NOM_ET ;
```

Deuxième étape : Formulation de calculs verticaux et horizontaux

Q6 Combien y a-t-il de professeurs actuellement saisis dans la base ?

```
SELECT COUNT(*) FROM PROF ;
```

Q7 Quelle est la moyenne des notes de contrôle continu pour la matière de code PRL ?

```
SELECT AVG(MOY_CC) FROM NOTATION WHERE CODE = 'PRL' ;
```

Q8 Combien de professeurs ont donné un enseignement à l'étudiant Philippe Lyon ?

```
SELECT COUNT (DISTINCT NUM_PROF)
FROM ETUDIANT, ENSEIGNT
WHERE ETUDIANT.NUM_ET = ENSEIGNT.NUM_ET
       AND NOM_ET = 'LYON' AND PRENOM_ET = 'PHILIPPE' ;

SELECT COUNT (DISTINCT NUM_PROF)
FROM ENSEIGNT
WHERE NUM_ET IN (
            SELECT NUM_ET FROM  ETUDIANT
            WHERE NOM_ET = 'LYON' AND PRENOM_ET = 'PHILIPPE') ;
```

Remarque : sans l'utilisation de **DISTINCT**, la requête ne retourne pas le résultat voulu mais le nombre d'enseignements reçus par l'étudiant Philippe Lyon. S'il se trouve que cet étudiant n'a pas plusieurs fois le même enseignant, la requête sans **DISTINCT** donnera un résultat juste mais ***elle est néanmoins logiquement fausse***.

Q9 Pour la matière de libellé Prolog, donnez la note moyenne obtenue par les étudiants en tenant compte des coefficients de contrôle continu et de test.

```
SELECT AVG(((NVL(MOY_CC, 0)*COEFF_CC) +
       (NVL(MOY_TEST, 0)*COEFF_TEST)) /(COEFF_CC+COEFF_TEST) ) moyenne
FROM NOTATION, MODULE
WHERE NOTATION.CODE = MODULE.CODE AND  LIBELLE = 'PROLOG' ;
```

Remarque : L'alias `moyenne` spécifié après l'expression de calcul horizontal est simplement introduit pour éviter l'affichage de l'expression.

Q10 Quel est le coefficient de test le plus faible ?

```
SELECT MIN(COEFF_TEST) FROM MODULE ;
```

Q11 Quelles sont les matières (libellé) dont le coefficient de test est le plus faible ?

```
SELECT LIBELLE
FROM MODULE
WHERE COEFF_TEST =
            (SELECT MIN(COEFF_TEST)
            FROM MODULE)
```

```
SELECT LIBELLE ;
FROM MODULE
WHERE COEFF_TEST <= ALL (
            SELECT COEFF_TEST
            FROM MODULE
            WHERE COEFF_TEST IS NOT NULL) ;
```

Remarque : dans la deuxième requête, la condition `COEFF_TEST IS NOT NULL` doit être spécifiée. En effet si l'attribut concerné comprend des valeurs nulles, le résultat de la sous-requête est évalué à 'inconnu' et le premier bloc ne rend aucun tuple.

Q12 Donnez la moyenne générale actuelle de l'étudiante Sandrine Levy (en supposant que toutes les matières sont équivalentes en terme de points).

```
SELECT AVG(((NVL(MOY_CC, 0)*COEFF_CC) +
       (NVL(MOY_TEST, 0)*COEFF_TEST)) / 100) moyenne
FROM MODULE, NOTATION, ETUDIANT
WHERE MODULE.CODE = NOTATION.CODE AND NOTATION.NUM_ET = ETUDIANT.NUM_ET
       AND NOM_ET = 'LEVY' AND PRENOM_ET = 'SANDRINE' ;
```

Q13 Quelles sont les matières dans lesquelles la meilleure note de test a été obtenue ?

```
SELECT DISTINCT LIBELLE FROM MODULE, NOTATION
WHERE MODULE.CODE = NOTATION.CODE AND MOY_TEST = (
            SELECT MAX(MOY_TEST)
            FROM NOTATION) ;
```

Remarque : Il faut procéder à l'élimination des duplicats (par **DISTINCT**) car plusieurs étudiants peuvent avoir obtenu la note maximale dans la même matière.

Q14 Quels sont les noms des étudiants qui ont obtenu, toutes matières confondues, la meilleure note de test ?

```
SELECT DISTINCT NOM_ET, ETUDIANT.NUM_ET
FROM ETUDIANT, NOTATION
WHERE ETUDIANT.NUM_ET = NOTATION.NUM_ET AND MOY_TEST = (
            SELECT MAX(MOY_TEST)
            FROM NOTATION) ;
```

Remarque : Il faut procéder à l'élimination des duplicats (par **DISTINCT**) car le même étudiant peut avoir obtenu la note maximale dans plusieurs matières.

```
SELECT DISTINCT NOM_ET, ETUDIANT.NUM_ET
FROM ETUDIANT, NOTATION
WHERE ETUDIANT.NUM_ET = NOTATION.NUM_ET AND MOY_TEST >= ALL (
            SELECT MOY_TEST
            FROM NOTATION
            WHERE MOY_TEST IS NOT NULL) ;
```

Remarque : la condition de sélection `MOY_TEST IS NOT NULL`, dans le bloc imbriqué, est nécessaire car il suffit d'une note de test inconnue pour que la requête ne rende aucun résultat.

Troisième étape : Utilisation des opérateurs ensemblistes

Q15 Donnez la liste des villes de résidence des étudiants et des enseignants.

```
SELECT VILLE_ET FROM ETUDIANT
UNION
SELECT VILLE_PROF FROM PROF ;
```

Q16 Quels sont les enseignants responsables d'une matière qu'ils enseignent. Donnez le numéro de l'enseignant et le code de la matière.

```
SELECT RESP, CODE FROM MODULE
INTERSECT
SELECT NUM_PROF, CODE FROM ENSEIGNT ;
```

Q17 Existe-t-il des matières, si oui, donnez leur libellé, ne correspondant à la spécialité d'aucun professeur ?

```
SELECT LIBELLE FROM MODULE WHERE  CODE  IN
            (SELECT CODE FROM MODULE
            MINUS
            SELECT MAT_SPEC FROM PROF) ;
```

Quatrième étape : Equivalent des opérateurs ensemblistes

Q18 Quels sont les enseignants responsables d'une matière qu'ils enseignent. Donnez le numéro de l'enseignant et le code de la matière.

```
SELECT RESP, CODE
FROM MODULE
WHERE (RESP, CODE) IN (
            SELECT NUM_PROF, CODE
            FROM ENSEIGNT) ;

SELECT DISTINCT RESP, MODULE.CODE
FROM MODULE, ENSEIGNT
WHERE RESP = NUM_PROF AND MODULE.CODE = ENSEIGNT.CODE ;
```

Remarques : (a) dans la première requête, il est inutile de contrôler que la sous-requête puisse retourner des valeurs nulles. En effet, ce cas est impossible : les deux attributs projetés font partie de la clef primaire.
(b) dans la deuxième requête, le mot clef **DISTINCT** est obligatoire car un même responsable peut enseigner la même matière à des étudiants différents.

Q19 Existe-t-il des matières, si oui, donnez leur libellé, ne correspondant à la spécialité d'aucun professeur ?

```
SELECT LIBELLE FROM MODULE
WHERE CODE NOT IN (
            SELECT MAT_SPEC
            FROM PROF
            WHERE MAT_SPEC IS NOT NULL) ;
```

```
SELECT LIBELLE FROM MODULE
WHERE CODE <> ALL (
            SELECT MAT_SPEC
            FROM PROF
            WHERE MAT_SPEC IS NOT NULL) ;
```

Remarque : dans les deux solutions précédentes, il faut spécifier la condition **NOT NULL**, car le prédicat « **NOT IN** » et la condition « CODE <> **ALL** » sont évalués à « inconnu » quand l'ensemble retourné par la sous-requête comporte une valeur nulle.

```
SELECT LIBELLE
FROM MODULE M
WHERE NOT EXISTS
            (SELECT MAT_SPEC
            FROM PROF
            WHERE MAT_SPEC = M.CODE) ;

SELECT LIBELLE
FROM MODULE, PROF
WHERE CODE = MAT_PEC(+) AND MAT_SPEC IS NULL ;
```

Cinquième étape : Test d'absence de données

Proposer trois formulations différentes des requêtes suivantes, en ne faisant pas appel aux opérateurs ensemblistes.

Q20 Donner les nom et prénom des étudiants n'ayant aucune note.

```
SELECT NOM_ET, PRENOM_ET, NUM_ET
FROM ETUDIANT
WHERE NUM_ET NOT IN
            (SELECT NUM_ET
            FROM NOTATION) ;

SELECT NOM_ET, PRENOM_ET, NUM_ET
FROM ETUDIANT E
WHERE NOT EXISTS (
            SELECT *
            FROM NOTATION
            WHERE E.NUM_ET = NOTATION.NUM_ET) ;

SELECT NOM_ET, PRENOM_ET, NUM_ET
FROM ETUDIANT
WHERE NUM_ET <> ALL (
            SELECT NUM_ET
            FROM NOTATION) ;

SELECT NOM_ET, PRENOM_ET, ETUDIANT.NUM_ET
FROM ETUDIANT, NOTATION
WHERE ETUDIANT.NUM_ET = NOTATION.NUM_ET(+) AND NOTATION.NUM_ET IS NULL ;
```

Q21 Quels sont les étudiants (nom, prénom) n'ayant eu aucun enseignement de Marc Laporte ?

```
SELECT NOM_ET, PRENOM_ET FROM ETUDIANT
WHERE NUM_ET NOT IN (
             SELECT NUM_ET FROM ENSEIGNT, PROF
             WHERE ENSEIGNT.NUM_PROF = PROF.NUM_PROF AND
                   NOM_PROF = 'LAPORTE' AND PRENOM_PROF = 'MARC') ;

SELECT NOM_ET, PRENOM_ET
FROM ETUDIANT E
WHERE NOT EXISTS (
             SELECT *
             FROM ENSEIGNT, PROF
             WHERE ENSEIGNT.NUM_PROF = PROF.NUM_PROF AND
                   NOM_PROF = 'LAPORTE' AND PRENOM_PROF = 'MARC'
                   AND E.NUM_ET = ENSEIGNT.NUM_ET) ;

SELECT NOM_ET, PRENOM_ET
FROM ETUDIANT
WHERE NUM_ET <> ALL
           (SELECT NUM_ET
            FROM ENSEIGNT, PROF
            WHERE ENSEIGNT.NUM_PROF = PROF.NUM_PROF AND
                  NOM_PROF = 'LAPORTE' AND PRENOM_PROF = 'MARC') ;

SELECT NOM_ET, PRENOM_ET
FROM ETUDIANT, (SELECT NUM_ET   -- Etudiants de Marc Laporte
                  FROM ENSEIGNT, PROF
                  WHERE  ENSEIGNT.NUM_PROF = PROF.NUM_PROF AND
                  NOM_PROF = 'LAPORTE' AND PRENOM_PROF = 'MARC') MARC
WHERE ETUDIANT.NUM_ET = MARC.NUM_ET (+)
AND MARC.NUM_ET IS NULL ;
```

Remarque : sont données ci-dessous deux autres versions de la ***requête mais qui ne rendent pas le résultat escompté***, car dans les deux cas, ***un étudiant qui a eu plusieurs enseignements, dont un est effectué par Marc Laporte et au moins un autre est effectué par un autre enseignant, fait partie du résultat ce qui est faux***.

```
SELECT DISTINCT NOM_ET, PRENOM_ET
FROM ETUDIANT, ENSEIGNT, PROF
WHERE ETUDIANT.NUM_ET = ENSEIGNT.NUM_ET(+)  AND
       ENSEIGNT.NUM_PROF = PROF.NUM_PROF(+)  AND
       NOM_PROF(+) = 'LAPORTE' AND PRENOM_PROF(+) = 'MARC'
       AND NOM_PROF IS NULL ;

SELECT DISTINCT NOM_ET, PRENOM_ET
FROM ETUDIANT, ENSEIGNT
WHERE ETUDIANT.NUM_ET = ENSEIGNT.NUM_ET AND NUM_PROF NOT IN
           (SELECT NUM_PROF
            FROM PROF
            WHERE NOM_PROF = 'LAPORTE' AND PRENOM_PROF = 'MARC') ;
```

Remarque : en plus du problème indiqué, avec la version précédente de la requête, un étudiant n'ayant eu aucun enseignement ne fait pas partie du résultat : il est éliminé lors de l'exécution de la jointure prédicative du premier bloc.

Q22 Quels sont les enseignants (nom, prénom) n'étant pas responsables de matière ?

```
SELECT NOM_PROF, PRENOM_PROF
FROM PROF
WHERE NUM_PROF NOT IN (
            SELECT RESP
            FROM MODULE
            WHERE RESP IS NOT NULL) ;

SELECT NOM_PROF, PRENOM_PROF
FROM PROF, MODULE
WHERE NUM_PROF = RESP(+) AND RESP IS NULL ;

SELECT NOM_PROF, PRENOM_PROF
FROM PROF P
WHERE NOT EXISTS (
             SELECT *
             FROM MODULE
             WHERE MODULE.RESP = P.NUM_PROF) ;

SELECT NOM_PROF, PRENOM_PROF
FROM PROF
WHERE NUM_PROF <> ALL (
             SELECT RESP
             FROM MODULE
             WHERE RESP IS NOT NULL) ;
```

Remarque : dans la première version de la requête et dans la dernière, la condition de sélection « `RESP IS NOT NULL` » est nécessaire. Sans elle, l'attribut `RESP` ayant des valeurs nulles, le résultat de la sous-requête est considéré comme inconnu et la requête ne rend aucun tuple.

Sixième étape : Expression des partitionnements

Formulez les requêtes suivantes en SQLplus.

Q23 Donnez, pour la deuxième année, le nombre d'étudiants par groupe.

```
SELECT GROUPE, COUNT(*)
FROM ETUDIANT
WHERE ANNEE = 2
GROUP BY GROUPE ;
```

Q24 Donnez la meilleure note de test de chaque étudiant (numéro).

```
SELECT NUM_ET, MAX(MOY_TEST)
FROM NOTATION
GROUP BY NUM_ET ;
```

Q25 Donnez le nombre de professeurs qu'a eu chacun des étudiants (numéro, nom) de deuxième année dans chacune des matières (en se contentant de leur code).

```
SELECT ETUDIANT.NUM_ET, NOM_ET, CODE, COUNT(NUM_PROF)
FROM ETUDIANT, ENSEIGNT
WHERE ETUDIANT.NUM_ET = ENSEIGNT.NUM_ET AND ANNEE = 2
```

```
GROUP BY ETUDIANT.NUM_ET, NOM_ET, CODE ;
```

Remarques :

- la projection de l'attribut clef primaire NUM_ET assure que des classes différentes sont créées par le partitionnement même en cas d'homonymie.
- il est inutile de préciser **DISTINCT** avant l'argument de la fonction **COUNT**. En effet, par définition de la relation ENSEIGNT, pour une matière et un étudiant donnés, un numéro de professeur n'apparaît qu'une seule fois.

Q26 Donnez, pour chaque ville de résidence de plus de cinq professeurs, le nombre de professeurs y habitant.

```
SELECT VILLE_PROF, COUNT(*)
FROM PROF
GROUP BY VILLE_PROF
HAVING COUNT(*) > 5 ;
```