Département **Informatique**
BUT 1

Ressource R2.06 :

Exploitation d'une Base de Données

Chapitre 1 : Requêtes SQL Avancées

LE LANGAGE

- **1 INTERROGATION.**.

  1.1 IMBRICATION DE SOUS QUESTION.
  1.2 SOUS QUESTION AVEC REFERENCE EXTERNE.
  1.3 EXPRESSION DE LA DIVISION.
  1.4 IMBRICATION AU NIVEAU RESULTAT.
  1.5 IMBRICATION AU NIVEAU TABLE.
  1.6 OPERATIONS SUR DES ENSEMBLES CALCULES.

- **2 MANIPULATION.**

  2.1 INSERTION.
  2.2 SUPPRESSION.
  2.3 MODIFICATION.

- **3 DESCRIPTION.**.
  3.1 DESCRIPTION DES VUES.

LE LANGAGE

- **S**tructured **Q**uery **L**anguage.
- Proche du langage des ensembles.
- Support ORACLE (version 9), il y a différents "dialectes".
- Permet l'Interrogation, la Manipulation, la Description

1 INTERROGATION

- **1.1 IMBRICATION DE SOUS QUESTION.**

  - Un SELECT (une question) définit un ensemble. On pourra dans le prédicat de qualification exprimer l'appartenance (la non appartenance) d'une valeur à un tel ensemble par **IN (NOT IN).** C'est l'imbrication de sous question.

1.1 IMBRICATION

- **1.1.1 INTERPRETATION DANS LE LANGAGE DES ENSEMBLES.**
  - Avec la non appartenance (x_k et y_k sont la même colonne).

$\{r[x_i,x_j] \mid x_k \notin \{s[y_k] \mid P(y_1,\ldots,y_n)\}\}$

SELECT x_i,x_j FROM R WHERE x_k NOT IN (SELECT y_k FROM S WHERE $P(y_1,\ldots,y_n)$);

On exprime ainsi la différence.

1.1 IMBRICATION

– Ou avec l'appartenance (x_k et y_k sont la même colonne).

$\{r[x_i,x_j] \mid r[x_k] \in \{s[y_k] \mid \wp(y_1,\ldots,y_n)\}\}$

SELECT x_i,x_j FROM R WHERE x_k IN (SELECT y_k FROM S WHERE $\wp(y_1,\ldots,y_n)$);

On exprime ainsi une jointure.

Notez les parenthèses qui permettent plusieurs niveaux d'imbrication.

1.1 IMBRICATION

- **1.1.2 EXEMPLES.**

  CLIENT(NUCLI, NOM, ADR)

  COM(NUCO, #NUCLI, DAT)

  ARTICLE(NUART, PU, DESIGN)

  LIGNE(NUCO, NUART, QTE)

  – **Liste des clients qui n'ont pas commandés.**

```
SELECT NUCLI, NOM FROM CLIENT
    WHERE NUCLI NOT IN
        (SELECT NUCLI FROM COM);
```

  – **Liste des clients qui ont passé au moins une commande.**

```
SELECT NUCLI, NOM FROM CLIENT
WHERE NUCLI  IN
   (SELECT NUCLI FROM COM);
```

– On peut écrire

```
SELECT NOM, DAT FROM CLIENT, COM
      WHERE (NUCLI,NUCO) IN  (SELECT NUCLI,NUCO FROM COM);
```

Notez les parenthèses qui indiquent un couple de valeur, attention à l'ordre.

```
SELECT NOM, DESIGN FROM CLIENT, ART
 WHERE (NUCLI,NUART) IN (SELECT NUCLI,NUART
                                FROM COM,LIGNE WHERE
                                      COM.NUCO = LIGNE.NUCO);
```

1 INTERROGATION

- **1.2 SOUS QUESTION AVEC REFERENCE EXTERNE.**

  – Dans une sous question (un SELECT imbriqué), on peut faire référence à une table citée dans la question (le FROM du SELECT qui le contient). C'est ce que l'on appelle une référence externe (on parle parfois de sous question corrélative).

1.2 INTERROGATION - REFERENCE EXTERNE

- Cela permet de qualifier une ligne en utilisant une fonction d'évaluation d'un ensemble calculé en utilisant cette ligne.

- Cela permet de qualifier une ligne en utilisant le fait qu'un ensemble calculé en utilisant cette ligne est vide (ou non) par [NOT] EXISTS.

1.2 INTERROGATION - REFERENCE EXTERNE

- **1.2.1 EXEMPLES AVEC UNE FONCTION D'EVALUATION D'ENSEMBLE.**

  – Liste des commandes qui comportent plus de cinq articles.

SELECT NUCO FROM **COM** WHERE
(SELECT COUNT(*) FROM LIGNE
WHERE LIGNE.NUCO = **COM.NUCO**) > 5;

COM.NUCO est une référence externe à la table **COM** du premier select

– Liste des clients qui ont un nombre de commandes supérieur à celui de DUPOND.

```
SELECT NOM FROM CLIENT WHERE
    (SELECT COUNT(*) FROM COM
    WHERE NUCLI = CLIENT.NUCLI)
>
    (SELECT COUNT(*) FROM COM
  WHERE NUCLI  = (SELECT NUCLI FROM CLIENT
                        WHERE NOM = 'DUPOND'));
```

1.2 INTERROGATION - REFERENCE EXTERNE

- **1.2.2 INTERPRETATION DANS LE LANGAGE DES ENSEMBLES.**

  – Comparaison à l'ensemble vide :

$\{r[x_i,x_j] \mid \{s \mid P(y_1,\ldots,y_m, x_1,\ldots,x_n)\}\ [!] = \not\in\}$

SELECT x_i,x_j FROM R WHERE [NOT] EXISTS (SELECT * FROM S WHERE $P(y_1,..,y_m, x_1,\ldots,x_n)$);

1.2 INTERROGATION - REFERENCE EXTERNE

- Notez que

$\{r[x_i,x_j] \mid \{s \mid y_1 = x_1\} != \not\in\}$

SELECT x_i,x_j FROM R WHERE EXISTS (SELECT * FROM S WHERE $y_1 = x_1$);

ou y_1 et x_1 sont la même colonne (domaine) est une expression de la jointure.

1.2 INTERROGATION - REFERENCE EXTERNE

- **1.2.3 EXEMPLES AVEC [NOT] EXISTS.**
  - Les clients qui n'ont pas commandé.

```
SELECT NOM FROM CLIENT
WHERE NOT EXISTS
   (SELECT * FROM COM
   WHERE COM.NUCLI = CLIENT.NUCLI);
```

C'est une différence.

Les noms des clients sont affichés si le deuxième select est vide

- **Les clients qui ont commandé.**

```
SELECT NOM FROM CLIENT
    WHERE EXISTS (SELECT * FROM COM
                  WHERE COM.NUCLI = CLIENT.NUCLI);
```

C'est une jointure. Dans ce cas le deuxième *select* ne doit pas être vide

1 INTERROGATION

- **1.3 EXPRESSION DE LA DIVISION.**
  - Les noms des clients qui ont commandé tous les articles.
  - Dans le langage des ensembles l'expression "naturelle" serait d'utiliser l'inclusion des ensembles.

$\{r[x_i,x_j] \mid \{s[y_k]\} \not\subset \{s[y_k] \mid P(x_1,\ldots,x_n,y_1,\ldots,y_m)\}\}$

$\{client[nom] \mid \{art[nuart]\} \not\subset$

$\{art[nuart] \mid com[nucli] = client[nucli]$

and $ligne[nuart]=art[nuart] \}$

1.3 INTERROGATION - DIVISION

- Malheureusement SQL ne permet pas l'expression de l'inclusion (ou de l'égalité) des ensembles. Il nous faut une expression équivalente.
- On a : A ⊄ B ⇔ A - B = ∅

Un client a commandé tous les articles ssi la différence de l'ensemble de nuart avec l'ensemble des nuart commandés par le client est vide

{client[nom] | {art[nuart]} -

{art[nuart] |com[nucli] = client[nucli] and ligne[nuart]=art[nuart]= ∅

1.3 INTERROGATION - DIVISION

– On a aussi l'expression avec les NOT IN:

```
SELECT NOM FROM CLIENT
WHERE NUCLI NOT IN
    (SELECT NUCLI FROM CLIENT, ART
     WHERE (NUCLI, NUART) NOT IN
     (SELECT NUCLI, NUART FROM
    COM C1,LIGNE L1
                    WHERE     C1.NUCO=L1.NUCO));
```

qui est la traduction de la formule

R/S(X) =R[X] - ((R[X] ⋇ S[Y]) – S'(X,Y))[X]

```
client[nucli] - ((client[nucli] ⋇ art[nuart]) -
                    com*ligne[nucli,nuart])[nucli]
```

L'ensemble des clients moins l'ensemble des clients qui n'ont pas commandé tous les articles

1.3 INTERROGATION - DIVISION

- Cela nous donne l'expression suivante :

```
SELECT NOM FROM CLIENT
WHERE NOT EXISTS
   (SELECT * FROM ART
    WHERE NUART
     NOT IN (SELECT NUART FROM COM C1, LIGNE L1
        WHERE C1.NUCLI = CLIENT.NUCLI
                And L1.NUCO=C1.NUCO));
```

Il n'existe pas d'articles telque cet article n'est pas commandé par le client

1.3 INTERROGATION - DIVISION

- Ou l'expression équivalente :

```
SELECT NOM FROM CLIENT
WHERE NOT EXISTS
   (SELECT * FROM ART A1
   WHERE NOT EXISTS
         (SELECT * FROM LIGNE L1, COM C1
         WHERE C1.NUCLI = CLIENT.NUCLI
         AND L1.NUART = A1.NUART AND
               L1.NUCO=C1.NUCO));
```

1.3 INTERROGATION - DIVISION

- On peut aussi utiliser l'artifice suivant :

  Le nombre d'articles commandés par le client est égale aux nombre d'articles

```
SELECT NOM FROM CLIENT WHERE
   (SELECT COUNT(DISTINCT NUART)
   FROM COM C1,LIGNE L1WHERE
   NUCLI = CLIENT.NUCLI and C1.NUCO=L1.NUCO)
=
   (SELECT COUNT(NUART)  FROM ART);
```

ATTENTION l'égalité du nombre des éléments n'implique pas en général l'égalité des ensembles.

1 INTERROGATION

- **1.4 IMBRICATION AU NIVEAU RESULTAT.**
  - Le résultat du SELECT imbriqué au niveau résultat permet de "prolonger" chaque ligne du résultat.
  - Il doit par une référence externe faire référence au résultat qu'il prolonge.
  - Il ne peut avoir comme résultat qu'une seule ligne.

1.4 INTERROGATION - RESULTAT

```
SELECT NOM, (SELECT COUNT(*)
                    FROM COM WHERE
                    NUCLI = CLIENT.NUCLI)
FROM CLIENT WHERE ADR = 'NANTES';

SELECT DESIGN, (SELECT SUM(QTE)
                      FROM LIGNE WHERE
                    NUART = ART.NUART)
FROM ARTICLE;
```

1 INTERROGATION

- **1.5 IMBRICATION AU NIVEAU TABLE.**
  - Il est nécessaire d'utiliser des ALIAS.

```
SELECT NOM, DESIGN, SUM(MCOM.QTE)
FROM CLIENT, ARTICLE, (SELECT NUCLI,NUART,QTE
                       FROM LIGNE, COM
                       WHERE LIGNE.NUCO = COM.NUCO
                       AND DAT = 'MAI') MCOM
WHERE MCOM.NUCLI = CLIENT.NUCLI
AND MCOM.NUART = ART.NUART
GROUP BY NOM, DESIGN;
```

La table MCOM n'exsite pas, elle est crée temporairement dans le premier ***from***

1 INTERROGATION

- **1.6 OPERATIONS SUR DES ENSEMBLES CALCULES.**

  – On peut exprimer les opérations ensemblistes sur le résultat de SELECT.

  – Il est TOUJOURS préférable d'avoir un seul SELECT.

1.6 INTERROGATION - OPERATIONS

- SELECT … UNION SELECT …;
- SELECT … UNION ALL SELECT …;
  conserve les doublons
- SELECT … INTERSECT SELECT …;
- SELECT … MINUS SELECT …;

1.6 INTERROGATION - OPERATIONS

```
SELECT NOM, COUNT(*) FROM CLIENT, COM
WHERE CLIENT.NUCLI = COM.NUCLI
GROUP BY NOM;
    UNION
SELECT NOM, 0 FROM CLIENT
WHERE NUCLI NOT IN
    (SELECT NUCLI FROM COM);
```

Pour les clients qui n'ont pas commandé, un 0 est affiché devant leur nom

Réécriture des requêtes

Select a,b from **T1,T2** where **T1.c=T2.c and Cond1**	Select a,b from **T1** where **T1.c in (select c from T2 and Cond1)** ou Select a, b from T1 where **exists** (select * from T2 where **T1.c=T2.c and Cond1**)

```
SELECT NUCLI, NOM FROM CLIENT CL, COM  C
        WHERE CL.NUMCLI  = C. NUMCLI ;

SELECT NUCLI, NOM FROM CLIENT CL
    WHERE CL.NUCLI IN   (SELECT C.NUCLI FROM COM C);

SELECT NUCLI, NOM FROM CLIENT CL
    WHERE Exists   (SELECT *  FROM COM C
                        where CL.NUCLI  = C. NUCLI);
```

2 MANIPULATION

- **2.1 INSERTION.**

INSERT INTO *nom_table* [({*nom_col,*}.)]
VALUES ({*valeur,*}.)
SELECT … ;

  - Par défaut, insertion de la valeur NULL dans les colonnes non désignées.

2.1 MANIPULATION - INSERTION

```
INSERT INTO COM (NUCO, NUCLI, DAT)
                VALUES (237,9,'MAI');

INSERT INTO LIGCALC
    (NUCO, NUART, MONTANT)
SELECT NUCO, NUART, QTE*PU
    FROM LIGNE, ART
    WHERE LIGNE.NUART = ART.NUART;
```

2.1 MANIPULATION - INSERTION

```
INSERT INTO LIGNE (NUCO,NUART,QTE)
SELECT NUCO,99,1
FROM LIGNE GROUP BY NUCO
HAVING COUNT(*) >5;
```

- On peut utiliser la table ou l'on insère pour calculer une insertion dans cette même table. Le calcul est fait sur la table dans l'état "avant" l'insertion.

2 MANIPULATION

- **2.2 SUPPRESSION.**

DELETE FROM *nom_table* [WHERE *prédicat*];

DELETE FROM COM;

DELETE FROM COM WHERE
NUCO IN (129,238,452);

2.2 MANIPULATION - SUPPRESSION

- On peut désigner la table ou l'on supprime par une référence externe dans le prédicat.

```
DELETE FROM COM WHERE NOT EXISTS
   (SELECT * FROM LIGNE
   WHERE  NUCO = COM.NUCO);
```

- Ou sans la référence externe.

```
DELETE FROM COM  WHERE NUCO NOT IN
   (SELECT NUCO FROM LIGNE);
```

2.2 MANIPULATION - SUPPRESSION

```
DELETE FROM LIGNE WHERE NUART IN
   (SELECT L.NUART FROM LIGNE L
   GROUPE BY L.NUART HAVING SUM(QTE)
   >     (SELECT STOCK FROM ART
         WHERE NUART = L.NUART));
```

- On peut utiliser la table ou l'on veut supprimer pour qualifier les lignes à supprimer dans cette table. La qualification est faite sur la table dans l'état "avant" la suppression.

2 MANIPULATION

- **2.3 MODIFICATION.**

UPDATE *nom_table* SET {*nom_colonne* = *valeur*,}.
({*nom_colonne*,}.) = (SELECT …) [WHERE *prédicat*];

UPDATE LIGNE SET QTE = 2 WHERE NUCO = 127 AND
NUART = 36;

2.3 MANIPULATION - MODIFICATION

- On peut utiliser une référence externe vers la table modifiée, pour qualifier les lignes à modifier ou pour calculer la modification.
- **ATTENTION**, en cas de calcul a NULL.

```
UPDATE CLIENT SET NBCOM =
    (SELECT COUNT(*) FROM COM
    WHERE NUCLI = CLIENT.NUCLI);
```

2.3 MANIPULATION - MODIFICATION

- Mais

```
UPDATE S_ART SET S_QTE =
    (SELECT SUM(QTE) FROM LIGNE
    WHERE NUART = S_ART.NUART)
WHERE NUART IN
    (SELECT NUART FROM LIGNE);
UPDATE S_ART SET S_QTE = 0
WHERE NUART NOT IN
    (SELECT NUART FROM LIGNE);
```

2.3 MANIPULATION - MODIFICATION

- On peut utiliser la table que l'on veut modifier pour qualifier les lignes à modifier dans cette table.

```
UPDATE COM SET REM = 10
WHERE NUCLI IN
    (SELECT NUCLI FROM COM
    GROUP BY NUCLI HAVING COUNT(*) > 5);
```

Un remboursement de 10% pour les clients qui ont commandé plus de 5 articles

2.3 MANIPULATION - MODIFICATION

- On peut utiliser la table que l'on veut modifier pour calculer les modification à faire dans cette table.

```
UPDATE LIGNE  SET REM =
    (SELECT 1+SUM(L.QTE)/100 FROM LIGNE L
     WHERE LIGNE.NUART =  L.NUART)
WHERE NUART IN
    (SELECT NUART FROM PROMO);
```

2 MANIPULATION

- C'est bien la totalité du "traitement des données":
  - "qualification" des lignes à supprimer ou à modifier.
  - calcul des lignes à insérer ou des nouvelles valeurs à affecter.

qui est exprimé dans le langage des ensembles (sans recours à l'algorithmique).

3 DESCRIPTION

- **3.1. DESCRIPTION DES VUES.**

CREATE [OR REPLACE] VIEW *nom_vue* ({*alias*,}...) AS SELECT .....;

DROP VIEW *nom_vue*;

Les vues sont enregistrées dans le dictionnaire de données.

3.1. DESCRIPTION - VUES.

- Une vue peut être utilisée comme une relation de base. Pour une interrogation, il n'y a pas de problèmes. Par contre une mise à jour, pose le problème de pouvoir inférer ce qu'il y a à faire sur les relations de bases à partir d'une mise à jour de la vue. Ceci est en général impossible.

- La mise à jour à travers une vue est interdite sauf si la vue est une sélection/projection sur une seule table.

3.1. DESCRIPTION - VUES.

```
CREATE VIEW MARKET (NOM,ADR,ART,PU,QTE) AS
    SELECT NOMCLI, ADDR, DESIGN, PU,SUM(QTE)
        FROM CLIENT, COM, LIGNE, ART
        WHERE CLIENT.NUCLI = COM.NUCLI
        AND COM.NUCO = LIGNE.NUCO
        AND LIGNE.NUART = ART.NUART
GROUP BY NOMCLI, ADDR, DESIGN, PU;

SELECT NOM FROM MARKET  WHERE ADR = 'NANTES'  AND
ART = 'TELE';
```