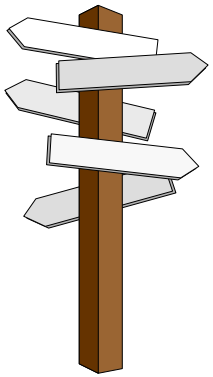
Cours n°5
SQL : Langage de manipulation des données (LMD) - Suite

Chantal Reynaud

Université Paris X - Nanterre UFR SEGMI - IUP MIAGE



Plan

I. SELECT multi-tables

II. L'insertion, la modification ou la suppression de valeurs de tuples

Partie I. SELECT multi-tables

I. Jointure

II. Requête imbriquée

III. Les opérateurs ensemblistes

I. Jointure (syntaxe SQL-89)

- Quand on précise plusieurs tables dans la clause FROM, on obtient le produit cartésien des tables. Ce qui est normalement souhaité, c 'est de joindre les informations de diverses tables, en précisant quelles relations les relient entre elles. C 'est la clause WHERE qui permet d 'obtenir ce résultat.

Ex :

```
SELECT Num-Stage, Libellé-Stage, Nom-Cat
FROM STAGE, CATEGORIE
WHERE STAGE.Num-Cat = CATEGORIE.Num-Cat;

SELECT Num-Stage, Libellé-Stage, Nom-Cat
FROM STAGE s, CATEGORIE c
WHERE s.Num-Cat = c.Num-Cat;
```

Alias

```
SELECT Num-Stage, Libellé-Stage, Nom-Cat
FROM STAGE s, CATEGORIE c
WHERE s.Num-Cat = c.Num-Cat and Num-Stage < 200;
```

Equijointures

Jointure : syntaxe SQL-2

- Cross Join = Produit cartésien

```
SELECT * FROM STAGE CROSS JOIN CATEGORIE;
```

- Jointure naturelle : SQL choisit automatiquement les attributs possédant le même nom dans les deux tables comme attribut de liaison entre les tables.

```
SELECT Num-Stage, Libellé-Stage, Nom-Cat
FROM STAGE NATURAL JOIN CATEGORIE
WHERE Num-Stage < 200;
```

- Jointure conditionnelle : séparation des conditions de jointure des autres conditions additionnelles de filtres

```
SELECT Num-Stage, Nom-Cat
FROM STAGE JOIN CATEGORIE
ON Num-Cat
WHERE Num-Stage < 200;
```

Jointure par non égalité

POIDS		
Nom-Poids	Poids-Min	Poids-Max
PLUME	0	100
LEGER	101	500
MOYEN	501	2500
LOURD	2501	9999

ARTICLES		
Num-Art	Nom-Art	Poids-Art
A10	CRAYON	20.00
A12	CRAYON LUXE	20.00
A01	AGRAFEUSE	150.0
A04	LAMPE	550.0

```
SELECT Num-Art, Nom-Art, Poids-Art, Nom-Poids
FROM ARTICLES, POIDS
WHERE Poids-Art BETWEEN Poids-Min AND Poids-Max;
```

En SQL-2 :

```
SELECT Num-Art, Nom-Art, Poids-Art, Nom-Poids
FROM ARTICLES JOIN POIDS
ON Poids-Art BETWEEN Poids-Min AND Poids-Max;
```

Auto-jointure

- Il peut être utile de rassembler des informations venant d 'une ligne d 'une table avec des informations venant d 'une autre ligne de la même table. Dans ce cas, il faut renommer au moins l 'une des deux tables en lui donnant un synonyme afin de pouvoir préfixer sans ambiguïté chaque nom de colonne.

```
SELECT a.Num-Stage, a.Libellé-Stage, a.Nb-jours, b.Num-Stage, B. Nb-jours
FROM STAGE a, STAGE b
WHERE a.Nb-jours > b.Nb-jours AND b.Num-Stage = ′471′;
```

En SQL-2 :

```
SELECT a.Num-Stage, a.Libellé-Stage, a.Nb-jours, b.Num-Stage, B. Nb-jours
FROM STAGE a  JOIN STAGE b
ON a.Nb-jours > b.Nb-jours
WHERE b.Num-Stage = ′471′;
```

Num-Stage	Libellé-Stage	Nb-jours	Num-Stage	Nb-jours
153	Windows 98	5	471	4
455	Windows NT4	5	471	4
323	Analyse objet	5	471	4

Jointure externe (syntaxe Oracle)

```
SELECT Num-Stage, Num-St
FROM STAGE, PARTICIPATION
WHERE STAGE.Num-Stage = PARTICIPATION.Num-Stage
```

Num-Stage	Num-St
350	106
350	345
471	457
215	987
467	342

Un stage qui n 'a pas de participant n 'apparaît pas

```
ELECT Num-Stage, Num-St
ROM STAGE, PARTICIPATION
/HERE STAGE.Num-Stage = PARTICIPATION.Num-Stage (+)
```

Num-Stage	Num-St
152	(NULL)
153	(NULL)
215	987
323	(NULL)
350	106
350	345
455	(NULL)
467	342
471	457
476	(NULL)

Liste des différents stages , avec les participants s 'ils en ont, sans omettre les stages sans participants.

Le (+) ajouté après un nom de colonne peut s 'interpréter comme l 'ajout, dans la table à laquelle la colonne appartient, d 'une ligne fictive qui réalise la correspondance avec les lignes de l 'autre table, qui n 'ont pas de correspondant réel.

Jointure externe - Syntaxe SQL-2

```
SELECT Num-Stage, Num-St
FROM STAGE RIGHT OUTER JOIN PARTICIPATION
ON Num-Stage;
```

De même, il existe **LEFT OUTER JOIN** qui correspond à l 'ajout d 'une ligne fictive dans la table de gauche et **FULL OUTER JOIN** pour l 'ajout de lignes fictives dans les deux tables.

Si on désire la liste des stages pour lesquels il n 'y a pas eu de participants :

```
SELECT Num-Stage, Num-St
FROM STAGE RIGHT OUTER JOIN PARTICIPATION
ON Num-Stage
WHERE Num-St IS NULL;
```

II. Requête imbriquée

- Une caractéristique puissante de SQL est la possibilité qu ’un critère de recherche employé dans une clause WHERE soit lui-même le résultat d ’un SELECT.

Exemple :
Sélection du numéro et du libellé du stage suivi par le stagiaire n°350.

```
SELECT Num-Stage, Libellé-Stage
FROM STAGE s, PARTICIPATION p
WHERE Num-St = ′ 350 ′ and s.num-Stage = p.num-Stage ;
```

Mais on peut aussi formuler cette requête par une sous-interrogation :

```
SELECT Num-Stage, Libellé-Stage
FROM STAGE
WHERE Num-Stage =       (SELECT distinct Num-Stage
                         FROM PARTICIPATION
                         WHERE Num-St =  = ′ 350 ′);
```

Le SELECT imbriquée équivaut à une valeur

Syntaxe : SELECT … WHERE exp op (SELECT …)
où op est un des opérateurs =, !=, <, >, <=, >=

Requête imbriquée renvoyant un ensemble de valeurs

- Les opérateurs utilisables, dans ce cas, sont : IN, ALL (tous), ANY (au moins un)

```
SELECT Num-ST, Nom-ST
FROM STAGIAIRE s
WHERE ′ 350 ′ IN
(SELECT Num-Stage
FROM STAGE JOIN PARTICIPATION
ON Num-Stage
WHERE Num-St = s.Num-St);
```

WHERE exp op ANY (SELECT …)
WHERE exp op ALL (SELECT …)
WHERE exp IN (SELECT …)
WHERE exp NOT IN (SELECT …)
où op est =, !=, <, >, <=, >=

Sous-interrogation synchronisée

```
SELECT Nom-Art
FROM ARTICLES
WHERE PV-Art > ANY      (SELECT PV-Art
                         FROM ARTICLES
                         WHERE Coul-Art = ′Blanc′);
```

Requête existentielle - Test d ’unicité

● La clause EXISTS est suivie d ’une sous-interrogation entre parenthèses, et prend la valeur vrai s ’il existe au moins une ligne satisfaisant les conditions de la sous-interrogation

```
SELECT Nom-Art
FROM ARTICLES a
WHERE EXISTS (
        SELECT *
        FROM ARTICLES b
        WHERE b.Coul-Art = ′ Blanc ′ AND PV-Art > b.PV-Art);
```

Donne les noms d ’articles plus chers qu’ un article de couleur blanche

● La clause UNIQUE est suivie d ’une sous-interrogation entre parenthèses, et prend la valeur vrai s ’il existe une seule ligne satisfaisant les conditions de la sous-interrogation

```
SELECT Num-St, Nom-St
FROM STAGIAIRE s
WHERE NOT UNIQUE (
        SELECT *
        FROM PARTICIPATION p
        WHERE s.Num-St = p.Num-St);
```

Donne les stagiaires qui ont participé à plus d ’un stage

Requête imbriquée dans un HAVING

- Un critère dans un HAVING peut dépendre du résultat d ’une requête imbriquée.

```
SELECT Num-Cat, SUM(Nb-jours) somme
FROM STAGE
GROUP BY Num-Cat
HAVING SUM(Nb-jours) >
        (SELECT AVG(Nb-jours)
        FROM STAGE);
```

Cette requête compare le total de jours de stage de chaque catégorie avec la moyenne du nombre de jours d ’un stage

III. Les opérateurs ensemblistes

- UNION

```
SELECT Num-Stage, Libellé-Stage
FROM STAGE
WHERE Num-Cat = ′1′ OR Num-Cat = ′2′
UNION
SELECT Num-Stage, Libellé-Stage
FROM STAGE
WHERE Nb-jours = 4;
```

Sélection des stages de catégorie 1 ou 2 dont la durée est de 4 jours

- INTERSECTION

```
SELECT Num-Stage, Libellé-Stage, ′1′ as indicateur
FROM STAGE
WHERE Num-Cat = ′1′
INTERSECT
SELECT Num-Stage, Libellé-Stage, ′non ouvert as indicateur′
FROM STAGE, PARTICIPATION
WHERE STAGE. Num-Stage = PARTICIPATION.Num-Stage (+) AND Num-St IS
NULL;
```

Sélection des stages de catégorie 1 sans participant

Partie II. L'insertion, la modification ou la suppression de valeurs de tuples

I. La commande INSERT

II. La commande UPDATE

III. La commande DELETE

La commande INSERT

- Ajout de lignes dans une table

```
INSERT INTO table (col1, …, coln)
VALUES (val1, …, valn);
```

ou

```
INSERT INTO table (col1, …, coln)
SELECT …;
```

La liste des noms de colonnes est optionnelle. Si elle est omise, Oracle prend par défaut l 'ensemble des colonnes de la table dans l 'ordre où elles ont été données lors de la création de la table. Si une liste de colonnes est spécifiée, les colonnes ne figurant pas dans la liste auront la valeur NULL.

```
INSERT INTO STAGIAIRE (Num-St, Nom-St, Prénom-St, Adr-St) VALUES (567, 'Dupont', 'Jean ', 'Paris');

INSERT INTO STAGIAIRE VALUES (567, 'Dupont', 'Jean ', 'Paris');

INSERT INTO PARIS_STAGIAIRE SELECT Num-ST, Nom-ST, Prénom-St
FROM STAGIAIRE WHERE Adr-St = 'Paris';
```

La commande UPDATE

● Cette commande permet de modifier les valeurs d 'un ou de plusieurs champs, dans une ou plusieurs lignes existantes d 'une table.

```
UPDATE table
SET col1 = exp1, col2 = exp2, …
WHERE prédicat
```

col1, col2, … sont les noms de colonnes qui seront modifiés.
La clause WHERE est facultative

```
UPDATE table
SET col1, col2 , …) = (SELECT …)
WHERE prédicat
```

```
UPDATE STAGE
SET Nb-jours = 4
WHERE Type-Stage = 'TP';
```

```
UPDATE STAGE
SET Nb-jours =
SELEC MAX(Nb-jours) FROM STAGE
GROUP BY NUM-Cat);
```

```
UPDATE STAGE
SET Nb-jours = 4
WHERE Type-Stage IN ( SELECT Num-St FROM PARTICIPATION);
```

La commande DELETE

- L ’ordre DELETE permet de supprimer des lignes d ’une table

```
DELETE FROM table
WHERE prédicat
```

La clause WHERE indique quelles lignes doivent être supprimées. Cette clause est facultative. Si elle n ’est pas précisée, TOUTES les lignes de la table sont supprimées.

```
DELETE FROM STAGE
WHERE Num-Cat = (
        SELECT Num-Cat
        FROM STAGE
        WHERE Libellé-St = 'Programmation java' );
```