Introduction aux Bases de données

Cours 5 : Requêtes SQL sur plusieurs tables

UFR 919 – Licence

2e année

Rappel : Requêtes simples

SELECT liste-colonnes
FROM Table
WHERE Condition;

Sélectionne

- les colonnes précisées dans le SELECT
- provenant de la table précisée dans le FROM
- dont les lignes vérifient les conditions précisées dans le WHERE.

Requêtes complexes : produit cartésien

SELECT Atts
FROM T1, T2 …, Tn
WHERE Condition ;

Atts *liste d'attributs* ∈Att(T1)*Att(T2)*…*Att(Tn)
Condition *contient des sous-conditions de la forme*
- Ti.a θTi.b ou
- Ti.a θ cste

où a *et* b *sont des attributs de* Ti, cste *une constante et* i=1..n

- Schéma = concaténation des attributs des Ti restreints à Atts
- Résultat = TOUTES les combinaisons des n-uplets de T1, …,Tn vérifiant Condition

Exemple

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	**M. Smith**	**Analyst**
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	**B. Casey**	**Analyst**
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	**J. Jones**	**Analyst**

PAY

TITLE	SALARY
Elect. Eng.	55000
Analyst	70000
Mech. Eng.	45000
Programmer	60000

SELECT **ENO, ENAME, EMP.TITLE, SALARY**
FROM EMP, PAY
WHERE **EMP.TITLE='Analyst'**

ENO	ENAME	EMP.TITLE	SALARY
E2	M. Smith	Analyst	55000
E2	M. Smith	Analyst	70000
E2	M. Smith	Analyst	45000
E2	M. Smith	Analyst	60000
E5	B. Casey	Analyst	55000
E5	B. Casey	Analyst	70000
E5	B. Casey	Analyst	45000
E5	B. Casey	Analyst	60000
E8	J. Jones	Analyst	55000
E8	J. Jones	Analyst	70000
E8	J. Jones	Analyst	45000
E8	J. Jones	Analyst	60000

Requêtes complexes : jointure

SELECT liste-colonnes
FROM T1, …, Tn
WHERE Condition;

Condition *contient des sous-conditions de la forme*
Ti.a θ Tj.b (condition de jointure) *ou* Ti.a θ cste

- Schéma avant la projection = concaténation des attributs des différentes tables Ti
- Résultat avant la projection = TOUTES les combinaisons des lignes de T1, …,Ti dont on ne garde que les lignes vérifiant Condition

Exemple

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	**M. Smith**	**Analyst**
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	**B. Casey**	**Analyst**
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	**J. Jones**	**Analyst**

PAY

TITLE	SALARY
Elect. Eng.	55000
Analyst	70000
Mech. Eng.	45000
Programmer	60000

```
SELECT ENO, ENAME, EMP.TITLE, SALARY
FROM EMP, PAY
WHERE EMP.TITLE=PAY.TITLE
      And PAY.TITLE='Analyst';
```

EXERCICE : TROUVER LE RESULTAT

Jointure : remarques

- Un tuple de la table *nomtable1* apparait dans le résultat de le jointure s'il joint **avec au moins 1** tuple de *nomtable2*
- Souvent un tuple joint avec plusieurs, d'où l'utilisation fréquente d'un DISTINCT dans le SELECT
- Si un attribut de même nom dans les 2 tables alors nécessité de préfixer l'attribut par le nom de la table (ou par son renommage si la table est renommée)
- Si oubli de la condition de jointure... produit cartésien ! Donc si k tables dans le from, en général au minimum k-1 conditions de jointure dans le WHERE

Exemples

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Noms et salaire des employés ?

```
SELECT Ename, Salary
FROM   Emp, Pay
WHERE  Emp.Title = Pay.Title
```

Noms et titres des employés qui travaillent dans un projet pendant plus de 17 mois?

```
SELECT Ename, Title
FROM   Emp, Works
WHERE  Dur > 17
       AND Emp.Eno = Works.Eno
```

Exemples

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Numéros et noms des projets dans lesquels a travaillé l'employé 10?

```
SELECT Project.Pno, Pname
FROM   Project, Works
WHERE  Eno=10
       AND Project.Pno = Works.Pno
```

Noms et titres des employés qui travaillent dans un projet à Paris ?

```
SELECT Ename, Title
FROM   Emp E, Works W, Project P
WHERE  P.City = 'Paris'
       AND    E.Eno = W.Eno
       AND W.Pno = P.Pno
```


Exemples

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Analyst
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Analyst
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Analyst

PAY

TITLE	SALARY
Elect. Eng.	55000
Analyst	70000
Mech. Eng.	45000
Programmer	60000

```
SELECT ENO, ENAME, EMP.TITE, SALARY
FROM EMP, PAY
WHERE EMP.TITLE=PAY.TITLE
```

ENO	ENAME	EMP.TITLE	SALARY
E1	J. Doe	Elect. Eng.	55000
E2	M. Smith	Analyst	70000
E3	A. Lee	Mech. Eng.	45000
E4	J. Miller	Programmer	60000
E5	B. Casey	Analyst	70000
E6	L. Chu	Elect. Eng.	55000
E7	R. Davis	Mech. Eng.	45000
E8	J. Jones	Analyst	70000

Jointure : cas particulier

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Nom des employés originaires de la même ville?

```
SELECT E1.Ename, E2.Ename
FROM   Emp E1, Emp E2
WHERE  E1.City = E2.City
```

EMP :

Eno	Ename	Title	City
10	Pierre	Analyst	London
20	Lucy	Engineer	Paris
30	Yifan	RH	Munich
40	Anne	RH	London
50	Ryadh	Analyst	Paris

Résultat :

E1.Ename	E2.Ename
Pierre	Pierre
Pierre	Anne
Lucy	Lucy
Lucy	Ryadh
Yifan	Yifan
Anne	Anne
Anne	Pierre
Ryadh	Ryadh
Ryadh	Lucy

Cas particulier : l'auto-jointure

Auto-jointure : jointure impliquant 2 fois la même table

```
SELECT liste-colonnes
FROM  T1 var1, T1 var2
WHERE conditions;
```

- Renommage des tables dans le FROM obligatoire
- Tous les attributs préfixés par var1 et var2 car présents dans les 2 tables T1 et T2 respectivement
- Attention car un enregistrement dans T1 apparait aussi dans T2 (et donc peut joindre avec lui-même suivant le prédicat de jointure)

Auto-jointure : exemples

Comment améliorer ? 1ère idée

Nom des employés originaires de la même ville?

```
SELECT E1.Ename, E2.Ename
FROM   Emp E1, Emp E2
WHERE  E1.City = E2.City
AND    E1.Ename <> E2.Name
```

EMP :

Eno	Ename	Title	City
10	Pierre	Analyst	London
20	Lucy	Engineer	Paris
30	Yifan	RH	Munich
40	Anne	RH	London
50	Ryadh	Analyst	Paris

Résultat :

E1.Ename	E2.Ename
Pierre	Anne
Lucy	Ryadh
Anne	Pierre
Ryadh	Lucy

Auto-jointure : exemples

Comment améliorer ? 2ème idée

Nom des employés originaires de la même ville?

```
SELECT E1.Ename, E2.Ename
FROM   Emp E1, Emp E2
WHERE  E1.City = E2.City
AND    E1.Ename < E2.Name
```

EMP :

Eno	Ename	Title	City
10	Pierre	Analyst	London
20	Lucy	Engineer	Paris
30	Yifan	RH	Munich
40	Anne	RH	London
50	Ryadh	Analyst	Paris

Résultat :

E1.Ename	E2.Ename
Lucy	Ryadh
Anne	Pierre

SQL : Requêtes imbriquées

Requête imbriquée dans la clause WHERE d'une requête externe:

```
SELECT  ...
FROM    ...
WHERE   [Opérande] Opérateur (SELECT ...
                              FROM ...
                              WHERE ...)
```

3 imbrications possibles :

- **(A1,…An) IN <sous-req>** exprime inclusion ensembliste
- **EXISTS <sous-req>** exprime condition d'existence
- **(A1,…An) <comp> [ALL|ANY] <sous-req>** **exprime** comparaison avec quantificateur (ANY par défaut)

Opérateur IN

```
SELECT ...
FROM ...
WHERE (A1,…,An) [NOT] IN (SELECT B1,…,Bn
                          FROM   ...
                          WHERE  ...)
```

Sémantique : la condition est vraie si le n-uplet désigné par ($A_1, \ldots, A_n$) de la requête *externe* appartient (n'appartient pas) au résultat de la requête *interne*.

Exemple avec IN

Emp (Eno, Ename, Title, City)
Pay(Title, Salary)
Project(Pno, Pname, Budget, City)
Works(Eno, Pno, Resp, Dur)

Noms des employés qui habitent dans des villes où il y a des [n'y a pas de] projets de budget inférieur à 50?

```
SELECT  Ename
FROM Emp
WHERE City [NOT] IN (SELECT City
                     FROM   Project
                     WHERE Budget < 50)
```


Imbrications avec IN : remarques

- La requête interne est effectuée dans un premier temps et son résultat est stocké (temporairement) en RAM (ou sur disque si trop gros!)
- La requête interne n'utilise pas la (les) table(s) de la requête externe
- Le IN étant ensembliste, les nuplets retournés par la requête interne doivent être du même format que le nuplet avant le IN (même nombre d'attributs et de même domaine, par forcément de même nom)
- Le SELECT de la requête externe ne peut retourner que des attributs appartenant aux tables placées dans son FROM

opérateur EXISTS

Q

```
SELECT ...
FROM ...
WHERE
```

Q'

```
[NOT] EXISTS (SELECT *
              FROM ...
              WHERE P)
```

- *Sémantique procédurale :*
  - pour chaque n-uplet *x* de la requête externe Q, exécuter la requête interne Q'; s'il existe au moins un [s'il n'existe aucun] n-uplet y dans le résultat de la requête interne, alors sélectionner *x*.
- *Sémantique logique :*
  - $\{ x... \mid Q(x) \wedge [\neg]\exists y\ (Q'(y)) \}$

Imbrications avec EXISTS

- Les deux requêtes sont *corrélées* : la condition *P* dans la requête interne Q' exprime une jointure entre les tables de Q' et les tables de la requête externe Q.
  => requête interne exécutée pour chaque nuplet de la requête externe
- Seule l'existence d'un résultat importe d'où le * dans le SELECT de la requête interne
- Le SELECT de la requête externe ne peut retourner que des attributs appartenant aux tables placées dans son FROM

Exemples avec EXISTS 1/3

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Noms des employés qui habitent dans une ville où il y a un projet?

```
SELECT Ename FROM Emp
WHERE EXISTS   (SELECT  *
                FROM    Project
                WHERE   Emp.City=Project.City)
```

Pour chaque employé on va vérifier si un projet est localisé dans sa ville.

Si oui, EXISTS vaut vrai et on retourne le nom de l'employé

Exemples avec EXISTS 2/3

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Noms des employés qui habitent dans une ville sans projet?

```
SELECT Ename FROM Emp
WHERE NOT EXISTS  (SELECT  *
                   FROM    Project
                   WHERE   Emp.City=Project.City)
```

Impossible à faire avec condition dans le WHERE

Exercice : exprimer cette requête à l'aide d'un *NOT IN*

Exemples avec EXISTS 3/3

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Noms des projets ayant le plus grand budget?

```
SELECT Pname FROM Project P1
WHERE NOT EXISTS  (SELECT  *
                   FROM    Project P2
                   WHERE   P1.Budget < P2.Budget)
```