

# Utilisation des bases de données

CM5 : SQL avancé et OLAP

Mickaël Martin-Nevot

V1.5.0



Cette œuvre est mise à disposition selon les termes de la [licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Partage à l'Identique 3.0 non transposé](https://creativecommons.org/licenses/by-nc-sa/3.0/).

# Utilisation des bases de données

- I. Prés.
- II. BD et SGBD
- III. Merise
- IV. LDD
- V. LMD
- VI. LCT
- VII. Droits
- VIII. LDSP
- IX. SQL avancé

# Jointures externes

- Permet de récupérer les tuples de la jointure interne **plus** certains tuples sans correspondance dans au moins l'une des relations jointes :
  - Jointure interne
  - Jointure externe gauche
  - Jointure externe droite
  - Jointure externe complète

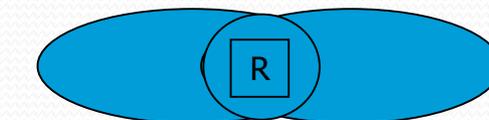
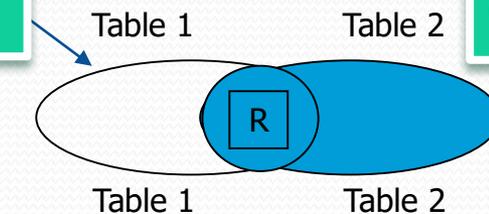
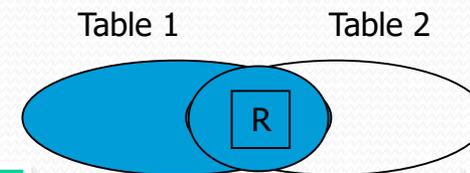
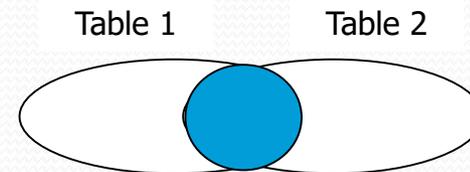


Table subordinée

Table dominante

Permet d'extraire des tuples ne répondant pas aux critères de jointure (interne)

# Jointures externes

- Jointure interne

-- Donner les noms des étudiants nés en 2001 et ayant réalisé (au moins) un stage en  
-- entreprise ainsi que sa durée.

```
SELECT nom, duree
FROM Etudiant E INNER JOIN Convention C
     ON E.ide = C.ide
WHERE DATE_PART('year', daten) = 2001;
```

- Jointure externe gauche

-- Quels sont les noms des étudiants nés en 2001, qu'ils aient ou non réalisé un  
-- stage en entreprise, ainsi que sa durée (s'il est réalisé).

```
SELECT nom, duree
FROM Etudiant E LEFT OUTER JOIN Convention C
     ON E.ide = C.ide
WHERE DATE_PART('year', daten) = 2001;
```

- Jointure externe droite (symétrique)

# Jointures externes

- Jointure externe complète (bilatérale)

-- Jointure interne : donner les noms des étudiants et des sociétés dont l'adresse  
-- est la même (tout en étant différente d'Aix\_en\_Pce).

```
SELECT E.nom AS nome, S.nom AS noms, E.adresse AS adr
FROM Etudiant E INNER JOIN Societe S
  ON E.adresse = S.adresse
WHERE E.adresse <> 'Aix_en_Pce';
```

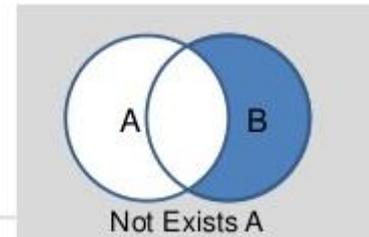
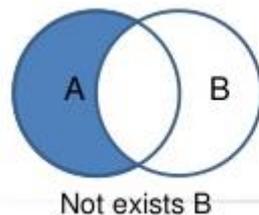
-- Jointure externe complète : donner les noms des étudiants et des sociétés ainsi  
-- que leurs adresses, quelles que soient ces adresses (mais différentes d'Aix\_en\_Pce  
-- et associées à tous les étudiants référencés).

```
SELECT E.nom AS nome, S.nom AS noms, E.adresse AS adr
FROM Etudiant E FULL OUTER JOIN Societe S
  ON E.adresse = S.adresse
WHERE E.adresse <> 'Aix_en_Pce' OR E.adresse IS NULL;
```

# Existence

- EXISTS : vrai si ensemble non nul

```
-- Quels sont les étudiants n'ayant réalisé aucun stage en entreprise ?  
SELECT ide  
FROM Etudiant  
WHERE NOT EXISTS (  
    SELECT * FROM Convention  
    -- ou SELECT ide FROM Convention  
    WHERE Convention.ide = Etudiant.ide);
```



# Division

- Permet (généralement) d'obtenir les tuples d'une relation qui sont associés à **tous les** tuples d'une autre relation :
  - En SQL, le quantificateur  $\forall$  n'existe pas
  - Il est remplacé par une double négation :
$$\forall x, P(x) \Leftrightarrow \neg(\exists x, \neg P(x))$$
    - « Un tuple A est en relation avec tous les enregistrements »
    - « Il n'existe pas de tuple qui n'est pas en relation avec le tuple A »

```
-- Trouver une société...
-- ...qui a une convention avec tous les étudiants.
-- = ...telle qu'il n'existe pas d'étudiant qui n'a pas de convention avec cette société.
SELECT ids FROM Societe S
WHERE NOT EXISTS (
  SELECT ide FROM Etudiant E
  WHERE NOT EXISTS (
    SELECT * FROM Convention C
    WHERE C.ids = S.ids AND C.ide = E.ide));
```

# Division (par cardinalités)

- Après calcul du nombre d'éléments dans chaque ensemble, est extrait les éléments de même cardinalité ↗

```
SELECT S.ids
FROM Societe S INNER JOIN Convention C
  ON S.ids = C.ids
GROUP BY S.ids
HAVING COUNT(DISTINCT ide) = (SELECT COUNT(ide) FROM Etudiant E)
```

Cette approche n'est pas toujours possible

R1	IdE
	8
	17

R2	IdS	IdE
	8	8
	8	17
	13	15
	34	8
	34	17
	13	12
	21	14
	8	15
	34	15
	2	17

ResDiv	IdS
	8
	34

# Vue

- Relation virtuelle
- Regroupement **logique** de données ← D'une ou plusieurs relations
  - Pas de stockage distinct de l'existant
- Manipulable **comme** une relation ordinaire
- Spécification d'une vue avec une expression de sélection

## Syntaxe :

```
CREATE [OR REPLACE] [...] VIEW name [(column_name [, ...])] ... AS query
```

```
CREATE OR REPLACE VIEW Vue1 (nome, nomt) AS
SELECT E.nom, P.nom
FROM Etudiant E
     INNER JOIN Convention C
           ON E.ide = C.ide
     INNER JOIN Personnel P
           ON C.ids = P.ids;
```

# Vue

- Nouveaux champs

```
CREATE OR REPLACE VIEW A3 (nom, age) AS  
SELECT nom, DATE_PART('year', AGE(daten))  
FROM Etudiant  
WHERE annee = 3;
```

Un champ peut être « créé »  
par calcul ou renommage

- L'écriture dans une vue est interdite lorsque :

- Elle est définie par plus d'une relation
- Elle comporte le résultat d'un calcul
- Elle ne respecte pas une contrainte d'intégrité d'une relation

Par exemple, de non nullité  
sur un champ non projeté

- Modification/suppression (semblable à une relation)

```
DROP VIEW Vue1;
```

Lorsqu'on supprime une relation associée, la vue n'est plus valide

Permet de simplifier (boîte noire), limiter ou sécuriser l'accès à des données

Ne devrait pas être utilisé comme table « temporaire »

# Table commune

- **Relation précalculée** avant la requête principale

WITH

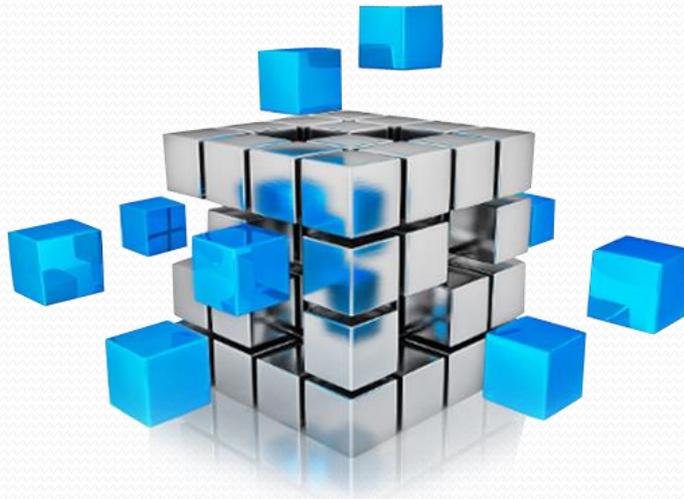
```
T1 (nom, nb_ade) AS (  
    SELECT nom, COUNT(DISTINCT adresse) AS nb_ade  
    FROM Etudiant  
    GROUP BY nom  
) ,  
T2 (nom, nb_ads) AS (  
    SELECT P.nom, COUNT(DISTINCT adresse) AS nb_ads  
    FROM Personnel P INNER JOIN Societe S  
        ON P.ids = S.ids  
    GROUP BY P.nom  
)  
SELECT T1.nom, nb_ade, nb_ads  
FROM T1, T2  
WHERE T1.nom = T2.nom;
```



Plusieurs tables communes peuvent être imbriquées

# Agrégation étendue (OLAP)

- GROUP BY ne construit **qu'une seule partition** des résultats
- Possibilité d'y adjoindre des opérateurs afin de visualiser **plusieurs partitions** en même temps



**OLAP** (*online analytical processing*) : traitement analytique en ligne (couramment utilisé en informatique décisionnelle) permet l'analyse sur-le-champ d'informations selon plusieurs axes, dans le but d'obtenir des rapports de synthèse

# OLAP : sélection de partitions

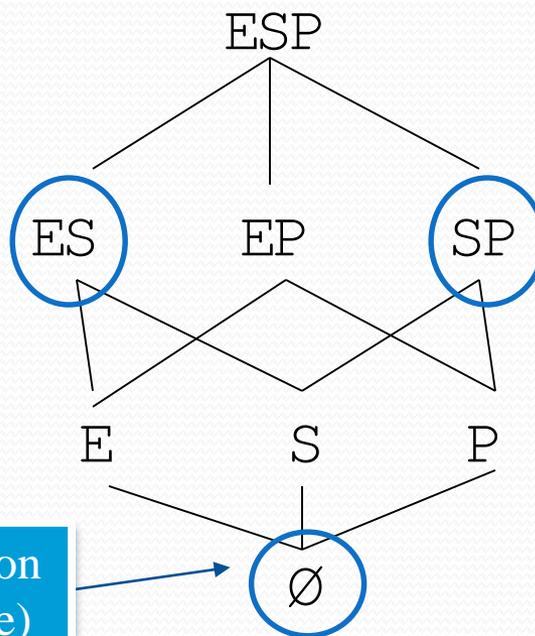
-- Donner les durées totales des stages en entreprise effectués d'une façon générale ; par  
-- étudiant et par société ; et enfin par société et par tuteur.

```
SELECT ide, ids, idp, SUM(duree) AS dureet
```

```
FROM Convention
```

```
GROUP BY GROUPING SETS((), (ide, ids), (ids, idp));
```

ide	ids	idp	dureet
8	8	NULL	3
8	21	NULL	5
8	34	NULL	6
12	13	NULL	6
15	8	NULL	6
15	13	NULL	5
17	8	NULL	4
17	34	NULL	6
NULL	NULL	NULL	41
NULL	8	2	3
NULL	8	4	6
NULL	8	7	4
NULL	13	12	11
NULL	21	19	5
NULL	34	53	12



Pas de partition  
(partition vide)

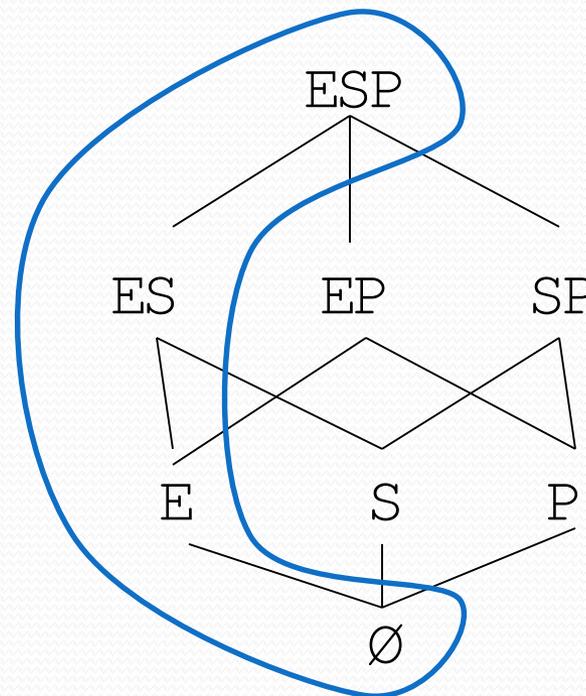
E : ide  
S : ids  
P : idp  
∅ : absence de partition

# OLAP : chemin de partitions

-- Donner les durées totales des stages en entreprise effectués d'une façon générale ; par  
-- étudiant ; par étudiant et par société ; et enfin par étudiant, par société et par tuteur.

```
SELECT ide, ids, idp, SUM(duree) AS dureet
FROM Convention
GROUP BY ROLLUP(ide, ids, idp);
```

ide	ids	idp	dureet
8	8	2	3
8	8	NULL	3
8	21	19	5
8	21	NULL	5
8	34	53	6
8	34	NULL	6
8	NULL	NULL	14
12	13	12	6
12	13	NULL	6
12	NULL	NULL	6
15	8	4	6
15	8	NULL	6
15	13	12	5
15	13	NULL	5
15	NULL	NULL	11
17	8	7	4
17	8	NULL	4
17	34	53	6
17	34	NULL	6
17	NULL	NULL	10
NULL	NULL	NULL	41

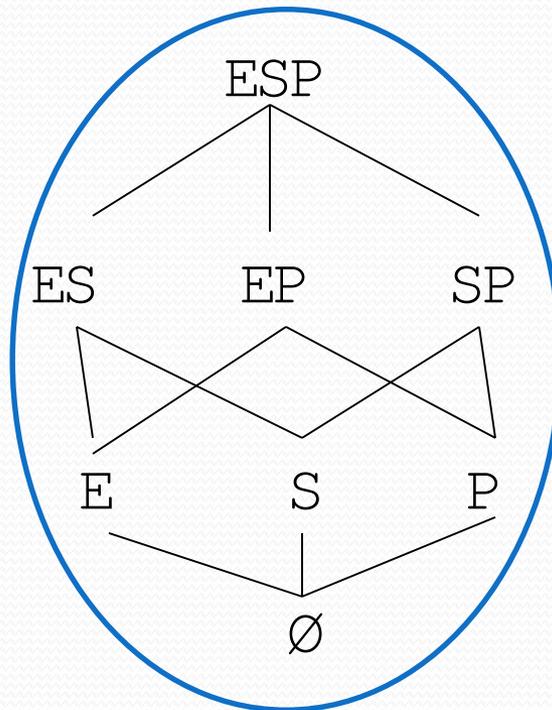


E : ide  
S : ids  
P : idp  
Ø : absence de partition

# OLAP : toutes les partitions

-- Donner les durées totales des stages en entreprise effectués d'une façon générale ; par  
-- étudiant ; par société ; par tuteur ; par étudiant et par société ; par étudiant et par  
-- tuteur ; par société et par tuteur ; et enfin par étudiant, par société et par tuteur.

```
SELECT ide, ids, idp, SUM(duree) AS dureet  
FROM Convention  
GROUP BY CUBE(ide, ids, idp);
```



E : ide  
S : ids  
P : idp  
∅ : absence de partition

# Ordonnancement

- Connaître le rang d'une donnée

-- Donner par année le classement des étudiants par rapport aux notes de stages en  
-- entreprise obtenues.

```
SELECT DATE_PART('year', datec) AS annee,  
       ide,  
       note,  
       RANK() OVER (PARTITION BY DATE_PART('year', datec) ORDER BY note DESC) AS rank  
FROM Convention  
ORDER BY annee, rank;
```



annee	ide	note	rank
2020	8	17	1
2020	8	16	2
2020	17	14	3
2020	12	13	4
2020	15	11	5
2021	8	18	1
2021	17	14	2
2021	15	10	3

# Aller plus loin

- Fusion (MERGE)
- Vues matérialisées
- Séquences (approfondissement)
- Tables sommaires
- Récursivité

# Crédits

## Auteur

Mickaël Martin-Nevot

[mmartin.nevot@gmail.com](mailto:mmartin.nevot@gmail.com)

- Laurent Carmignac



Carte de visite électronique

## Relecteurs

Cours en ligne sur : [www.mickaël-martin-nevot.com](http://www.mickaël-martin-nevot.com)

