Exploitation d'une BD

CM4 : SQL avancé et OLAP

Mickaël Martin Nevot

V1.4.0

Cette œuvre de Mickaël Martin Nevot est mise à disposition sous licence Creative Commons Attribution - Utilisation non commerciale - Partage dans les mêmes conditions.

(cc) BY-NC-SA

Rappel : opérateurs SQL

- Projection
- Sélection
- Union
- Différence
- Intersection
- Produit cartésien
- Jointure
- **Division**

Opérateurs ensemblistes

Opérateurs primitifs

Opérateurs dérivés

(cc) BY-NC-SA

Partitionnement (sous-tables)

- Partitionner les données afin d'effectuer des calculs par ensemble de données groupées :

```
SELECT prenom, COUNT(*)
FROM Etudiant WHERE sexe = 'M'
GROUP BY prenom;
```

On obtient autant de partitions que de valeurs distinctes dans l'ensemble d'attributs de la clause GROUP BY

- Partitionnement avec condition de sélection :

```
SELECT prenom, COUNT(ide)
FROM Etudiant WHERE sexe = 'M'
GROUP BY prenom
HAVING COUNT (ide) >= 2;
```

Les agrégations s'appliquent à chaque valeur de l'ensemble de la clause SELECT

Règle d'or : tous les attributs non agrégés projetés dans un SELECT **doivent** figurer dans le GROUP BY, et **inversement**

(cc) BY-NC-SA

Partitionnement

R	A	B	C
	a1	b1	c1
	a1	b1	c2
	a2	b1	c1
	a2	b2	c1
	a1	b2	c1
	a2	b1	c1

```
SELECT A, B, COUNT(*)
FROM R
GROUP BY A, B;
```

T1	A	B	(*)
	a1	b1	2
	a1	b2	1
	a2	b1	2
	a2	b2	1

```
SELECT C, A, COUNT(*)
FROM R
GROUP BY C, A;
```

T2	C	A	(*)
	c1	a1	2
	c1	a2	3
	c2	a1	1

(cc) BY-NC-SA

Sous-requêtes

- Sous-requête (sous-interrogation) dans une autre clause que WHERE :
  - SELECT :

```
SELECT DISTINCT ids, (SELECT AVG(duree) FROM Convention WHERE ids = 8) AS "Diff"
FROM Convention
WHERE ids = 34;
```

  - FROM : table imbriquée

```
SELECT MAX(avgd) AS summ
FROM (SELECT ids, AVG(duree) AS avgd FROM Convention GROUP BY ids) T;
```

Alias obligatoire dans ce cas

  - HAVING :

```
SELECT annee, COUNT(*)
FROM Etudiant WHERE sexe = 'M'
GROUP BY annee
HAVING COUNT (DISTINCT ide) >= (SELECT COUNT(*) FROM Convention WHERE ide = 8);
```

Il est important de distinguer les sous-requêtes qui retournent une seule valeur des sous requêtes qui retournent plusieurs valeurs

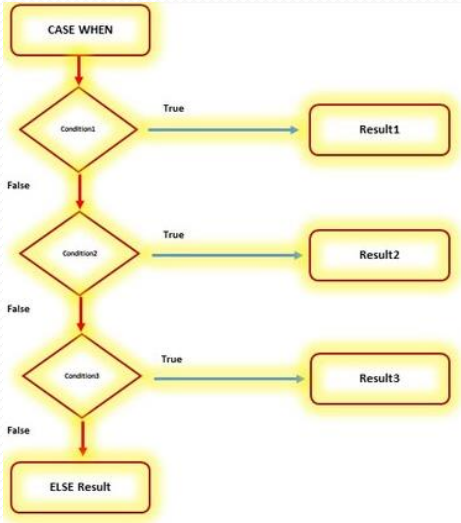
(cc) BY-NC-SA

Expressions conditionnelles

- CASE : recomposition contextuelle des données

```
SELECT CASE TO_CHAR(datec, 'MM')
         WHEN '03' THEN 'Mars'
         WHEN '04' THEN 'Avril'
         WHEN '05' THEN 'Mai'
       END AS mois,
       COUNT(DISTINCT ide) AS nombre
FROM Convention
WHERE TO_CHAR(datec, 'MM') IN ('03', '04', '05')
GROUP BY datec;
```

Branchement



(cc) BY-NC-SA

Gestion des valeurs nulles

- Renvoi le premier argument non nul (condition) :
  - NVL : évalue toujours les deux arguments

```
SELECT ide, ids, NVL(date_deb, TO_DATE('2021-06-01', 'YYYY-MM-DD'))
FROM Convention;
```

  - COALESCE : stop l'évaluation après le premier argument non nul

```
SELECT ide, ids, COALESCE(date_deb, TO_DATE('2021-06-01', 'YYYY-MM-DD'))
FROM Convention;
```

À la norme

- Avec les fonctions d'agrégation :

```
SELECT AVG(note) FROM Convention;
```

Valeurs nulles ignorées

- Avec les sous-requêtes :

```
SELECT * FROM Etudiant
WHERE prenom NOT IN (SELECT prenom FROM Personnel WHERE prenom IS NOT NULL);
```

Ici, sans le prédicat de la sous-requête, le prédicat de la requête principale est évalué à faux même s'il n'y a qu'une seule valeur nulle dans la sous-requête

(cc) BY-NC-SA

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

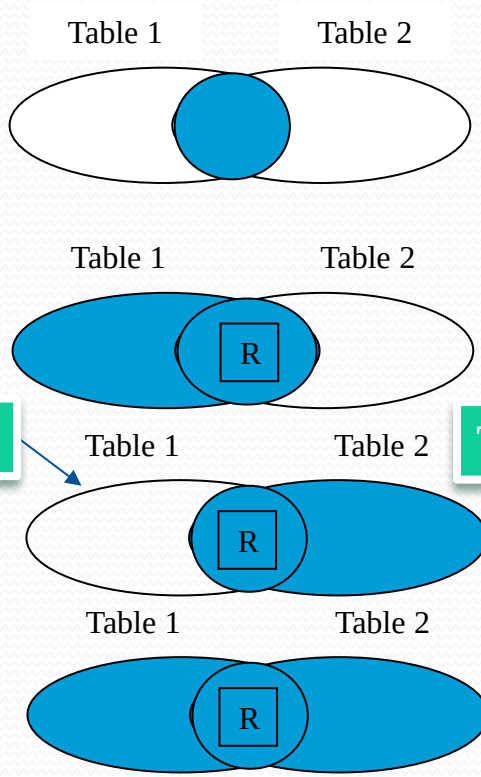


Table subordonnée

Table dominante

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

(cc) BY-NC-SA

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 TO_CHAR(daten, 'YYYY') = '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 TO_CHAR(daten, 'YYYY') = '2001';
```

Jointure externe droite (symétrique)

```
-- Quels sont les noms des étudiants, qu'ils aient ou non réalisé un
-- stage en entreprise, ainsi que sa durée (s'il est réalisé).
SELECT nom, duree
FROM Convention C RIGHT OUTER JOIN Etudiant E
  ON C.ide = E.ide;
```

(cc) BY-NC-SA

Jointures externes

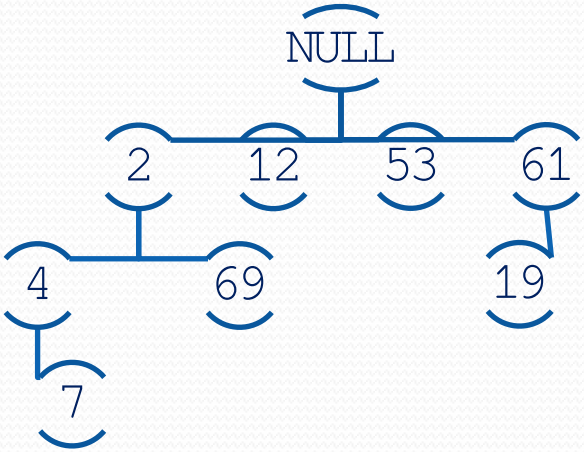
- Jointure externe complète (bilatérale)

```
-- 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;
```

(cc) BY-NC-SA

Recherche récursive

```
-- Racines de l'arbre.
SELECT employe
FROM Organigramme
WHERE responsable IS NULL;

-- Feuilles de l'arbre.
SELECT employe
FROM Organigramme
WHERE employe NOT IN (SELECT responsable FROM Organigramme
                      WHERE responsable IS NOT NULL);
```



- Parcours père-fils :

Syntaxe :

```
SELECT [...] CONNECT BY pere = PRIOR fils [AND cond] [START WITH start]
SELECT [...] CONNECT BY PRIOR fils = pere [AND cond] [START WITH start]
```

- Parcours fils-père :

Syntaxe :

```
SELECT [...] CONNECT BY fils = PRIOR pere [AND cond] [START WITH start]
SELECT [...] CONNECT BY PRIOR pere = fils [AND cond] [START WITH start]
```

(cc) BY-NC-SA

Recherche récursive

```
-- Employés (directs et indirects) du personnel n°2.
SELECT employe, level
FROM Organigramme
CONNECT BY responsable = PRIOR employe
START WITH employe = 2;
```

Propriété système donnant le numéro des niveaux

```
-- Employés du personnel n°2 à l'exception de l'employé n°4.
SELECT LPAD('-', 2 * level, ' ') || employe
FROM Organigramme
WHERE employe <> 4
CONNECT BY responsable = PRIOR employe
START WITH employe = 2;
```

Affichage indenté

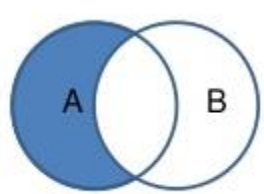
```
-- Employés du personnel n°2 à l'exception de l'employé n°4 et de ses employés.
SELECT employe
FROM Organigramme
CONNECT BY responsable = PRIOR employe
AND employe <> 4
START WITH employe = 2;
```

(cc) BY-NC-SA

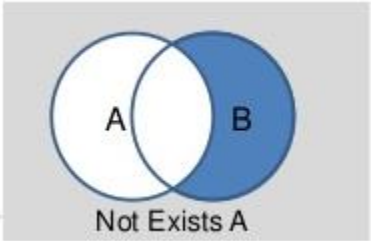
Existence

- 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);
```



Not exists B



Not Exists A

(cc) BY-NC-SA

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 ∀ n'existe pas
  - Il est remplacé par une double négation :
    ∀ x, P(x) ⇔ ⌉(∃ x, ⌉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 Convention C1
WHERE NOT EXISTS (
    SELECT ide FROM Etudiant E
    WHERE NOT EXISTS (
        SELECT * FROM Convention C2
        WHERE C2.ids = C1.ids AND C2.ide = E.ide));
```

(cc) BY-NC-SA

Division (par cardinalités)

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

Cette approche n'est pas toujours possible

```
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);
```

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

(cc) BY-NC-SA

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;
```

(cc) BY-NC-SA

Vue

- Nouveaux attributs

```
CREATE OR REPLACE VIEW A3 (nom, age) AS
SELECT nom, TRUNC(MONTHS_BETWEEN(sysdate, daten) / 12)
FROM Etudiant
WHERE annee = 3;
```

Un attribut 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 attribut 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 (boite noire), limiter ou sécuriser l'accès à des données

Ne devrait pas être utilisé comme table « temporaire »

(cc) BY-NC-SA

Vue de mise à jour

- Permet la vérification de l'intégrité de relations

Lors de l'insertion, la mise à jour ou la suppression des données via une vue

```
CREATE OR REPLACE VIEW Vue2 AS
SELECT *
FROM Personnel
WHERE salaire > 2000
WITH CHECK OPTION;

-- Ne provoque pas d'erreur.
INSERT INTO Vue2 (idP, nom, prenom, idS, salaire)
VALUES (111, 'Magnus', 'Maximus', 8, 5000);

-- Provoque une erreur.
INSERT INTO Vue2 (idP, nom, prenom, idS, salaire)
VALUES (666, 'Masyaf', 'Theophraste', 8, 1500);
```

(cc) BY-NC-SA

Table commune

- **Relation précalculée** avant la requête principale

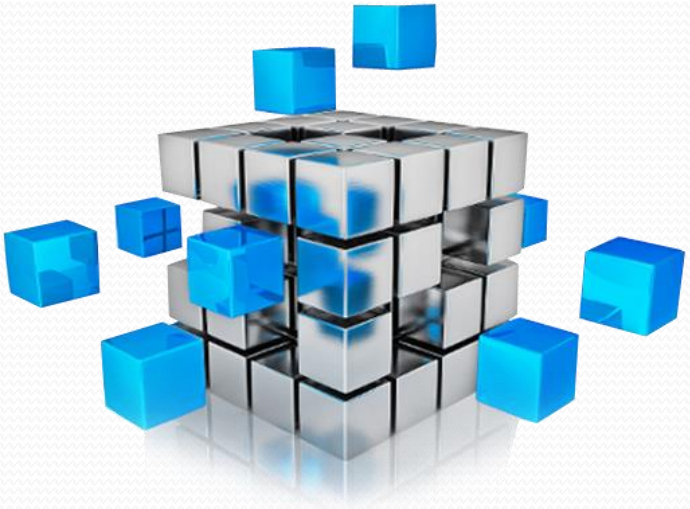
```
WITH
T1 (nom, nb_ade) AS (
     SELECT nom, COUNT(DISTINCT adresse) AS nb_ade
     FROM Etudiant
     GROUP BY nom
),
T2 (nom, nb_ads) AS (
     SELECT P.nom, COUNT(DISTINCT adresse) AS nb_ads
     FROM Personnel P INNER JOIN Societe S
       ON P.ids = S.ids
     GROUP BY P.nom
)
SELECT T1.nom,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

Plusieurs tables communes peuvent être imbriquées

(cc) BY-NC-SA

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



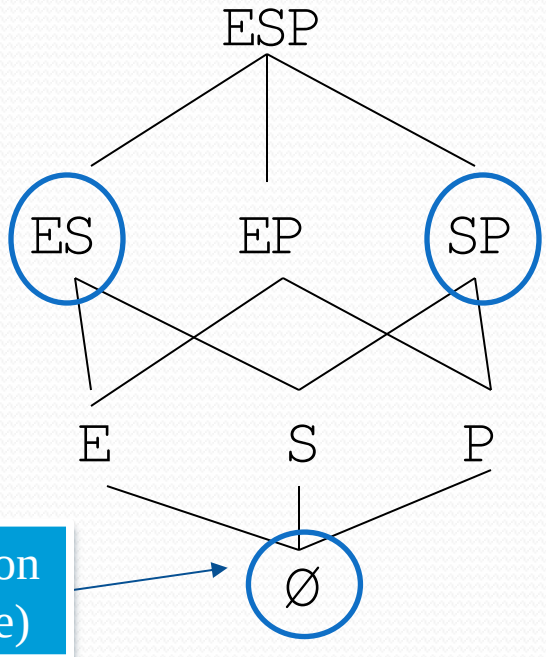
OLAP *(online analytical processing)* : traitement analytique en ligne (couramment utilisé en informatique décisionnelle) permet l'analyse sur-le-champ d'informations selon plusieurs axes, dans le but d'obtenir des rapports de synthèse

(cc) BY-NC-SA

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)

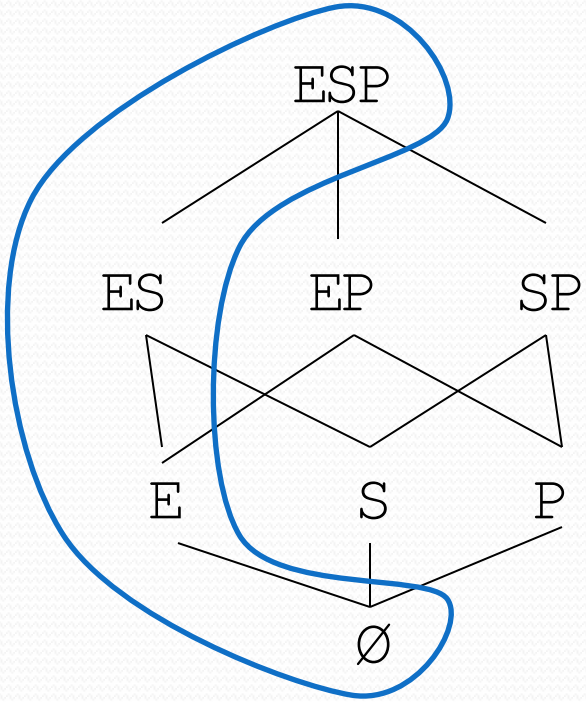
E : ide
S : ids
P : idp
∅ : absence de partition

(cc) BY-NC-SA

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

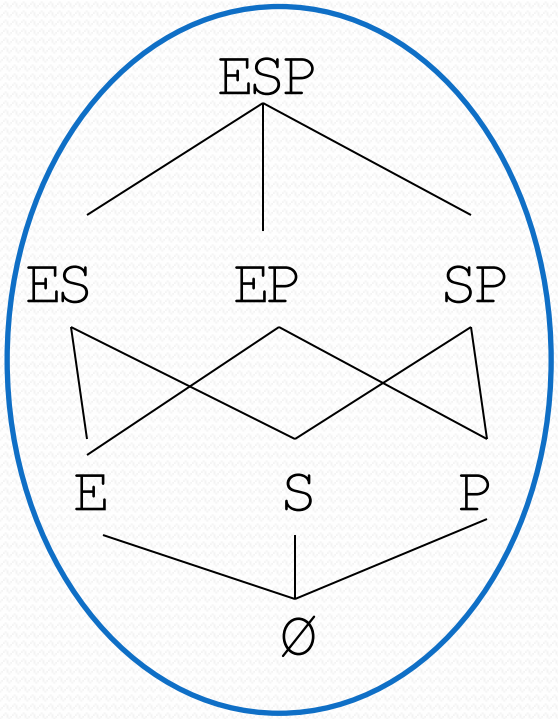


E : ide
S : ids
P : idp
∅ : absence de partition

BY-NC-SA

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);
```



45 résultats

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

(cc) BY-NC-SA

Ordonnancement

- Connaître le rang d'une donnée

```
-- Donner par année le classement des étudiants par rapport aux notes de stages en
-- entreprise obtenues.
SELECT TO_CHAR(datec, 'YYYY') AS annee,
       ide,
       note,
       RANK() OVER (PARTITION BY TO_CHAR(datec, 'YYYY') 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

Crédits

Auteur

Mickaël Martin-Nevot
mmartin.nevot@gmail.com

- Laurent Carmignac

Carte de visite électronique

Relecteurs

Cours en ligne sur : www.mickael-martin-nevot.com