Base de données et structures de données

CM4 : Langage de manipulation de données (LMD)

Mickaël Martin Nevot

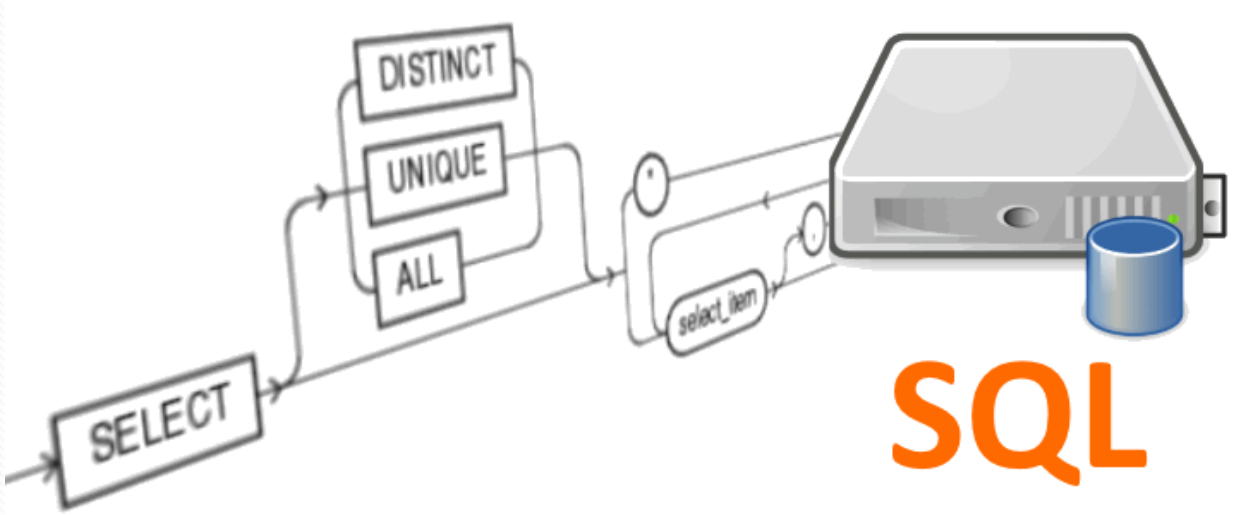
V1.0.4

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

BY-NC-SA

LMD

- **Langage de manipulation de données** (LMD)
- Recherche des données d'une BD
- Mise à jour des données d'une BD



Le résultat d'une requête relationnelle est une relation

BY-NC-SA

Recherche de données

- Sélection :
  - SELECT : définit les attributs de la relation résultante
  - FROM : spécifie les relations sur lesquelles porte la recherche
  - WHERE : indique des conditions de restriction
  - Les autres clauses seront traitées ultérieurement

Syntaxe :

```
SELECT [DISTINCT] ...
FROM ...
[WHERE ...]
[GROUP BY ...]
[HAVING ...]
[ORDER BY ...]
```

SELECT, FROM, WHERE, etc. sont des clauses

BY-NC-SA

Opérateurs SQL

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

Opérateurs ensemblistes

Opérateurs primitifs

Opérateurs dérivés

(cc) BY-NC-SA

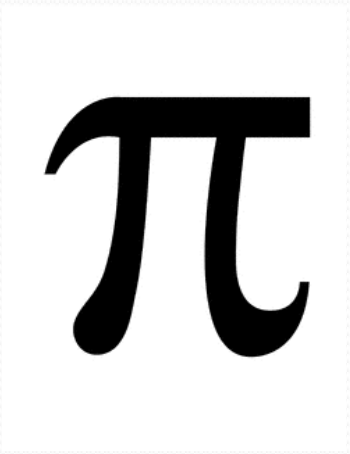
Projection

- Préserve (projette) certains **attributs** d'une relation

```
SELECT nom, prenom
FROM Etudiant;
```

```
SELECT *
FROM Convention;
```

Tous les attributs sont préservés



CC BY-NC-SA

Sélection

- Filtre certains **tuples** d'une relation (avec condition[s])

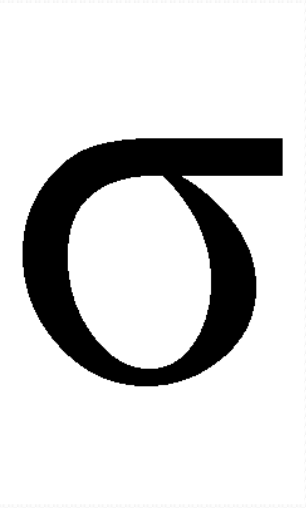
```
SELECT nom, prenom FROM Etudiant WHERE annee = 2;

SELECT * FROM Etudiant WHERE annee = 1 AND sexe = 'F';
```

- Condition :
  - Attribut(s)
  - Constantes : 2, 'F', etc.
  - Opérateur(s) :
    - De comparaison : =, <>, >, >=, <, <=, etc.
    - Booléens : AND, OR, NOT, etc.
    - IS NULL, IN, ALL, ANY, BETWEEN, EXISTS, LIKE, etc.

Une condition est un prédicat

Expression logique vraie ou fausse

Les opérateurs de comparaison peuvent être utilisés avec des attributs textuels



(cc) BY-NC-SA

Opérateurs de sélection

- IN : comparaison avec un ensemble

```
SELECT nom, prenom
FROM Etudiant
WHERE annee IN (2, 3);
```

- BETWEEN : intervalle de valeur

```
SELECT nom, prenom
FROM Etudiant
WHERE annee BETWEEN 1 AND 3; -- Equivalent à : annee >= 1 AND annee <= 3
```

- LIKE : comparaison de chaîne de caractères

```
SELECT nom, raisons
FROM Societe
WHERE adresse LIKE '_ar%';
```

Jokers : _ (un caractère)
% (n'importe quel nombre de caractères)

- IS (NOT) NULL : traitement des valeurs nulles

```
SELECT ide, ids
FROM Convention
WHERE date_deb IS NULL;
```

BY-NC-SA

Projection et sélection

- Projection et sélection sont combinables

```
SELECT nom, prenom
FROM Etudiant
WHERE sexe = 'F' AND annee <= 2;

SELECT nom, raisons
FROM Societe
WHERE adresse LIKE '%ar%'
  AND activite BETWEEN 7 AND 15;
```

Et ce ne sont pas les seuls !

BY-NC-SA

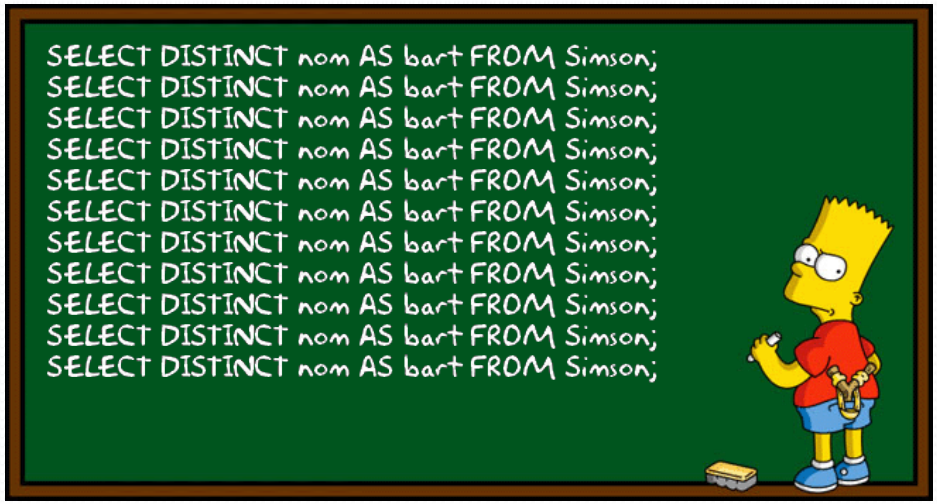
Autres mots clef de SELECT

- DISTINCT : élimination des doublons

```
SELECT DISTINCT nom
FROM Etudiant
WHERE prenom <> 'Lulu';
```

- AS : renommage :

```
SELECT raisons AS formej, adresse AS "Adresse postale"
FROM Societe
WHERE ids = 8;
```



BY-NC-SA

ORDER BY

- Tri des résultats :
  - ASC : ascendant (par défaut)
  - DESC : descendant

```
SELECT nom
FROM Etudiant
WHERE prenom <> 'Lulu'
ORDER BY nom ASC;

SELECT DISTINCT nom
FROM Etudiant
WHERE annee = 2
ORDER BY nom DESC;
```

BY-NC-SA

Fonctions d'agrégation

- Agrège les tuples (un seul résultat)
- Uniquement dans clause `SELECT` (ou `HAVING`)
- Fonctions :
  - `SUM(...)` : somme
  - `AVG(...)` : moyenne algébrique
  - `MIN(...)`, `MAX(...)` : valeur min., max.
  - `COUNT(...)` : dénombrement
  - Etc.

Uniquement pour des attributs numériques

Aussi appelées fonctions de calcul intégrées

Pas de fonction d'agrégat en paramètre d'une fonction d'agrégat

BY-NC-SA

Fonctions mathématiques

- Uniquement dans clause `SELECT` (ou `HAVING`) et `WHERE`
- Opérateurs : +, –, *, /, %, etc.
- Fonctions :
  - `ABS(x)` : valeur absolue
  - `ROUND(x, y)` : arrondi
  - `FLOOR(x)`, `CEIL(x)` : arrondi à l'inférieur, supérieur
  - `TRUNC(x)` : troncature
  - `POWER(x, y)` : puissance
  - `SIGN(x)` : signe
  - `SQRT(x)` : racine carré
  - Etc.

BY-NC-SA

Fonctions de chaînes

- Uniquement dans clause `SELECT` (ou `HAVING`) et `WHERE`
- Opérateurs : `||`
- Fonctions :
  - `LOWER(str)`, `UPPER(str)` : conversion en minuscule, majuscule
  - `REPLACE(txt, str1, str2)` : remplacement de chaîne
  - `SUBSTR(txt, i, n)` : extrait une sous-chaîne
  - `LENGTH(str)` : taille de la chaîne
  - `INITCAP(str)` : premier caractère en majuscule
  - Etc.

BY-NC-SA

Fonctions date/heure

- NOW() : date et heure courantes
- TO_CHAR(..., txt) : conversion de date en chaîne

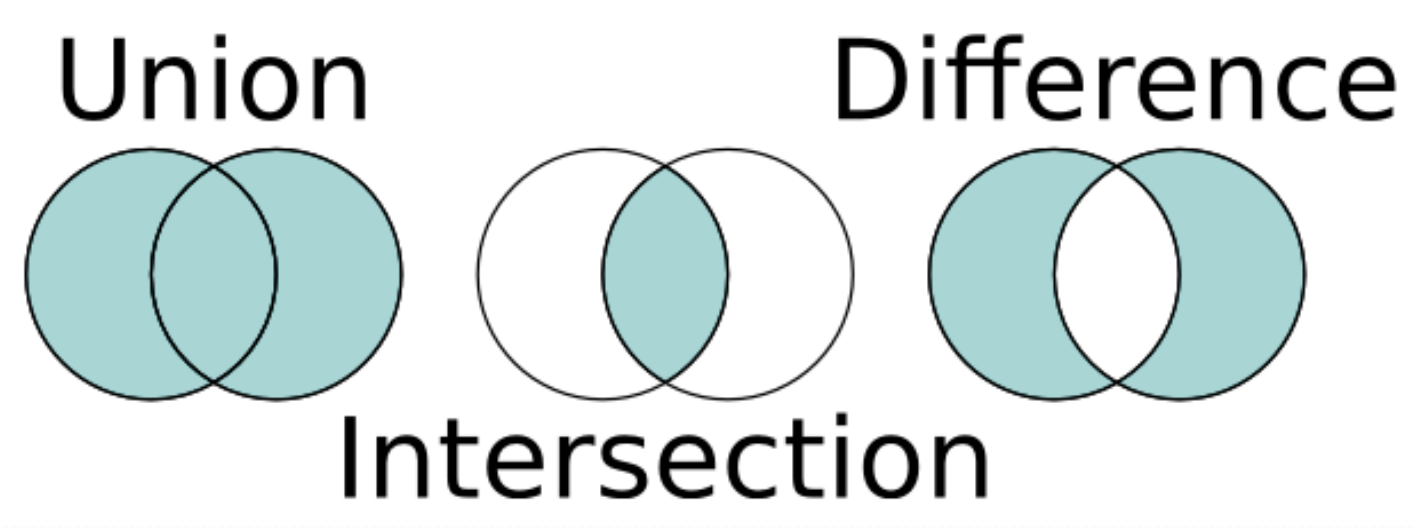
```
SELECT TO_CHAR(current_timestamp, 'HH12:MI:SS') FROM T;
```

- TO_DATE(str1, str2) : conversion de chaîne en date

```
INSERT INTO T VALUES (TO_DATE('05 Dec 2020', 'DD Mon YYYY'));
```

BY-NC-SA

Opérateurs ensemblistes



BY-NC-SA

Opérateurs ensembliste

- Union :

```
SELECT ids FROM Personnel
WHERE nom = 'Durand'
UNION
SELECT ids FROM Convention
WHERE duree = 4;
```

Les domaines des attributs correspondants dans les deux relations doivent être les mêmes (on parle de relations unicompatibles)

- Différence :

```
SELECT ide FROM Etudiant
EXCEPT
SELECT ide FROM Convention
WHERE intitule LIKE '%logiciel%';
```

- Intersection :

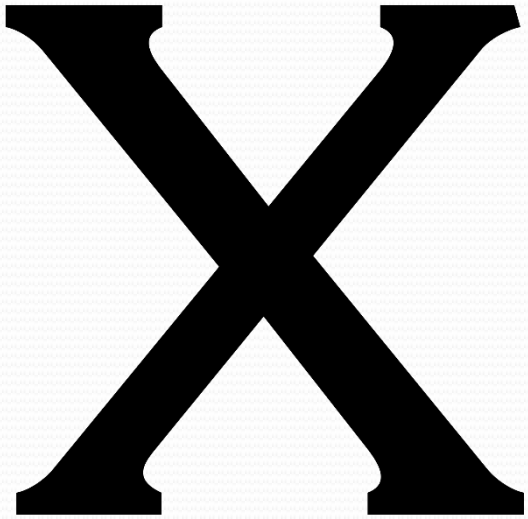
```
SELECT ide FROM Etudiant
WHERE prenom = 'Lulu'
INTERSECT
SELECT ide FROM Convention
WHERE duree <= 3;
```

`UNION ALL`, `EXCEPT ALL`, `INTERSECT ALL` : préservent toutes les valeurs, même les dupliquées

BY-NC-SA

Produit cartésien

```
SELECT *
FROM Etudiant, Societe;
```



Très peu utile (seul)

BY-NC-SA

Jointure

Permet de retrouver des données organisée sur plusieurs relation indépendantes

- Jointure ensembliste : imbrication de sous-requête(s)

```
SELECT nom
FROM Etudiant
WHERE ide IN (
    SELECT ide
    FROM Convention
    WHERE duree >= 3);
```

IN, ALL, ANY, etc.

- Jointure prédicative : utilisation de prédicats

```
SELECT nom
FROM Etudiant INNER JOIN Convention
  ON Etudiant.ide = Convention.ide
WHERE duree >= 3;
```

Version normalisée

Qualification

```
SELECT nom
FROM Etudiant E, Convention C
WHERE E.ide = C.ide AND duree >= 3;
```

Alias (à ne pas confondre avec AS)

```
SELECT E.nom, raisons
FROM Etudiant E, Convention C, Societe S
WHERE E.ide = C.ide
  AND S.ids = C.ids;
```

Généralement on effectue des équi-jointure (=), mais pas toujours (>, <, etc.)

(cc) BY-NC-SA

Auto-jointure

```
SELECT ETR.nom
FROM Etudiant ETR NATURAL JOIN Etudiant ETS
WHERE ETS.nom = 'Dupond';

SELECT ETR.nom
FROM Etudiant ETR INNER JOIN Etudiant ETS
  ON ETR.daten = ETS.daten
WHERE ETS.nom = 'Dupond';

SELECT nom
FROM Etudiant
WHERE daten IN (
    SELECT daten
    FROM Etudiant
    WHERE nom = 'Dupond');
```

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**

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

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

BY-NC-SA

Autres opérateurs multilignes

- ANY : n'importe quel résultat de l'ensemble (imbriqué) vrai
- ALL : tous les résultats de l'ensemble (imbriqué) vrais
- =ANY : équivalent à IN
- <>ALL : équivalent à NOT IN

ETUDIANT	Nom	DateN
	Dupond	16-03-81
	Duvent	20-04-81
	Dupond	17-04-82
	Durand	18-04-82

```
SELECT nom, daten
FROM Etudiant
WHERE daten > ALL (SELECT daten
                   FROM Etudiant
                   WHERE nom = 'Dupond');
```

ETUDIANT	Nom	DateN
	Durand	18-04-82

```
SELECT nom, daten
FROM Etudiant
WHERE daten > ANY (SELECT daten
                   FROM Etudiant
                   WHERE nom = 'Dupond');
```

ETUDIANT	Nom	DateN
	Duvent	20-04-81
	Dupond	17-04-82
	Durand	18-04-82

BY-NC-SA

Expressions conditionnelles

- CASE : recomposition contextuelle des données

```
SELECT CASE DATE_PART('month', datec)
         WHEN 3 THEN 'Mars'
         WHEN 4 THEN 'Avril'
         WHEN 5 THEN 'Mai'
       END AS mois,
       COUNT(DISTINCT ide) AS nombre
FROM Convention
WHERE DATE_PART('month', datec) IN (3, 4, 5)
GROUP BY datec;
```

Branchement

- COALESCE : renvoie le premier argument non nul

```
SELECT ide, ids, COALESCE(date_deb, '2021-6-1')
FROM Convention;
```

BY-NC-SA

Division

BY-NC-SA

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

BY NC SA