GRENOBLE INP UGA Ensimag

École nationale supérieure d'informatique et de mathématiques appliquées

Langage SQL (LMD)

Ensimag 2A

Equipe pédagogique BD

Schéma de référence

Élèves	prénom	nom	e-mail	filière
	Luke	Skywalker	skywalker@imag.fr	MMIS
	Dark	Vador	vador@imag.fr	IF
	Han	Solo	solo@falcon.com	IF
	Leia	Solo	princess@falcon.com	MMIS
	Jabba	The Hut	jabba@imag.fr	ISSC

Notes	cours	prénom	nom	note
	sport	Dark	Vador	20
	sport	Jabba	The Hut	3
	pilotage	Han	Solo	15

Schéma SQL

```
CREATE TABLE Eleves (
    prenom varchar(30) NOT NULL,
    nom varchar (30) NOT NULL,
    email varchar (30),
    filiere varchar (5),
    PRIMARY KEY (prenom, nom)
);
CREATE TABLE Notes (
    cours varchar (30) NOT NULL,
    prenom varchar (30) NOT NULL,
    nom varchar(30) NOT NULL,
    note integer,
    PRIMARY KEY(cours, prenom, nom),
    FOREIGN KEY (prenom, nom) REFERENCES Eleves(prenom, nom)
);
```

GRENOBLE INP UGA Ensimag

École nationale supérieure d'informatique et de mathématiques appliquées

Insérer/supprimer/ modifier des données

```
INSERT INTO <relation>[(<attribut>*)]
    ( VALUES (<valeur>*)
    |  <requête>) ;

DELETE FROM <relation> [WHERE <condition>];

UPDATE <relation> SET (<attribut> = <expr>)*
[WHERE <condition>] ;
```

Relation Eleves

Élèves	prénom	nom	e-mail	filière
	Luke	Skywalker	skywalker@imag.fr	MMIS
	Dark	Vador	vador@imag.fr	IF
	Han	Solo	solo@falcon.com	IF
	Leia	Solo	princess@falcon.com	MMIS
	Jabba	The Hut	jabba@imag.fr	ISSC

```
INSERT INTO Eleves VALUES ('Luke', 'Skywalker', 'skywalker@imag.fr', 'MMIS');
INSERT INTO Eleves VALUES ('Dark', 'Vador', 'vador@imag.fr', 'IF');
INSERT INTO Eleves VALUES ('Han', 'Solo', 'solo@falcon.com', 'IF');
INSERT INTO Eleves VALUES ('Leia', 'Solo', 'princess@falcon.com', 'MMIS');
INSERT INTO Eleves VALUES ('Jabba', 'The Hut', 'jabba@tatooine.fr', 'ISSC');
```

Relation Notes

Notes	cours	prénom	nom	note
	sport	Dark	Vador	20
	sport	Jabba	The Hut	3
	pilotage	Han	Solo	15

INSERT INTO Notes VALUES (‘sport', 'Dark', 'Vador', '20');

INSERT INTO Notes VALUES (’sport', 'Jabba', 'The Hut', ’3');

INSERT INTO Notes VALUES (’pilotage', 'Han', 'Solo', ‘15');

GRENOBLE INP UGA Ensimag

École nationale supérieure d'informatique et de mathématiques appliquées

Sélection de données

```
SELECT [DISTINCT | ALL] (* | (<table> | <vue>).*
                        | <expr> [AS <alias>])*
FROM ((<table> | <vue>) [<alias>])*
[WHERE <condition>]
[GROUP BY (<expr>)* [HAVING <condition>]]
[(INTERSECT | UNION [ALL] | EXCEPT) <requête>]
[ORDER BY (<expr> [ASC | DESC])*;
```

```
SELECT [DISTINCT | ALL] (* | (<table> | <vue>).*
                          | <expr> [AS <alias_c>])*
FROM ((<table> | <vue>) [<alias_t>])*
[WHERE <condition>];
```

FROM : liste la ou les relations contenant les données interrogées

➔ **la relation (*résultat du produit cartésien*) sur laquelle portera les expressions des clauses SELECT (et WHERE)**

SELECT : liste d'expressions permettent de "construire" le schéma (attributs) de la relation résultat

WHERE : condition vérifiée par les n-uplets de la relation résultat

```
SELECT * FROM Eleves;

SELECT  prenom, nom FROM Eleves;

SELECT filiere FROM Eleves;

SELECT DISTINCT filiere FROM Eleves;

SELECT e.nom, e.prenom,
            e.e-mail AS “Adresse Electronique”, e.filiere
FROM Eleves e;

SELECT Eleves.nom, filiere, note
FROM Eleves, Notes;
```

Fonctions de regroupement

COUNT([DISTINCT] <expr>) :
comptage des valeurs (distinctes),
* pour compter les n-uplets.

AVG(<expr>) : moyenne

MIN(<expr>) : minimum

MAX(<expr>) : maximum

SUM(<expr>) : somme

```
SELECT COUNT(Eleves) FROM Eleves;
SELECT AVG(n.note) AS MoyNote FROM Notes n;
SELECT SUM(note) FROM Notes;
```

Clause WHERE

```
SELECT [DISTINCT | ALL] (* | (<table> | <vue>).*
                         | <expr> [AS <alias>])*
FROM ((<table> | <vue>) [<alias>])*
[WHERE <condition>];
```

WHERE <condition>

- Combinaison des conditions :
  - **NOT** <condition>
  - <cond1> **AND** <cond2>
  - <cond1> **OR** <cond2>
- opérateurs :
  - **= <> < > <= >=** (θ)
  - **BETWEEN LIKE IS NULL IN ALL ANY SOME**
  - **EXISTS UNIQUE**
- Opérande (expr) :
  - Constante,
  - nom d'attribut,
  - fonction,
  - résultat d'un SELECT imbriqué

Condition…

`<expr> IS [NOT] NULL`

Teste si <expr> (attribut) a ou non une valeur.

`<expr> θ (<expr1> | <constante>)`

Comparaison de valeurs d'<expr>.

`<expr> BETWEEN <expr1> AND <expr2>`

Teste l'appartenance à un intervalle fermé.

`<expr> LIKE <motif de chaîne>`

Comparaison avec un motif de chaîne de caractère

`%` : toute chaine de caractères

`_` : tout caractère

Exemples condition

-- Les adresses électroniques des étudiants de la filière ISSC

```
SELECT e-mail AS “Adresse Electronique”
FROM Eleves e
WHERE filiere = ‘ISSC’;
```

-- Les noms et prénoms des étudiants ayant eu moins de 10 à une matière

```
SELECT    prenom, nom
FROM      Notes
WHERE     note < 10 ;
```

Condition de jointure

-- e-mail des étudiants ayant eu moins de 10 à une matière

```
SELECT    e-mail
FROM      Eleves, Notes
WHERE     note < 10
AND Eleves.nom = Notes. nom
AND Eleves.prenom = Notes.prenom ;
```

```
SELECT    e-mail
FROM      Eleves JOIN Notes ON
                  Eleves.nom = Notes.nom
          AND Eleves.prenom = Notes.prenom
WHERE     note < 10 ;
```

Plus sophistiqué …

-- Le nombre d'élèves (et la moyenne des notes) dont le nom commence par S et qui ont des notes comprises entre 14 et 16

```
SELECT   COUNT (*), AVG (Notes)
FROM     Eleves e, Notes n
WHERE    e.nom LIKE 'S%'
AND      note BETWEEN 14 AND 16
AND      e.nom = e. nom
AND      e.prenom = e.prenom ;
```

GROUP BY

```
SELECT [DISTINCT | ALL] (* | (<table> | <vue>).*
                        | <expr> [AS <alias>])*
FROM ((<table> | <vue>) [<alias>])*
[WHERE <condition>]
[GROUP BY (<expr>)* [HAVING <condition>]]
[(INTERSECT | UNION [ALL] | EXCEPT) <requête>]
[ORDER BY (<expr> [ASC | DESC])*;
```

Exemple

-- Moyenne des notes par matière pour les matières ayant plus de 10 de moyenne

```
SELECT cours, AVG(note)
FROM   Notes
GROUP BY cours
HAVING  AVG(note) >=10;
```

Interrogation

```
SELECT [DISTINCT | ALL] (* | (<table> | <vue>).*
                        | <expr> [AS <alias>])*
FROM ((<table> | <vue>) [<alias>])*
[WHERE <condition>]
[GROUP BY (<expr>)* [HAVING <condition>]]
[(INTERSECT | UNION [ALL] | EXCEPT) <requête>]
[ORDER BY (<expr> [ASC | DESC])*;
```

Opérations ensemblistes

`UNION` [**`ALL`**] : Union ensembliste sans/avec conservation de doublons

`INTERSECT` : Intersection ensembliste

`EXCEPT` : difference

Attention : **`MINUS`** dans Oracle

Exemple Différence

Les noms et prénoms des élèves n'ayant que des notes supérieures ou à 10

```
SELECT    prenom, nom
FROM      Eleves
MINUS
SELECT    prenom, nom
FROM      Notes
WHERE     note < 10 ;
```

ORDER BY

```
SELECT [DISTINCT | ALL] (* | (<table> | <vue>).*
                        | <expr> [AS <alias>])*
FROM ((<table> | <vue>) [<alias>])*
[WHERE <condition>]
[GROUP BY (<expr>)* [HAVING <condition>]]
[(INTERSECT | UNION [ALL] | EXCEPT) <requête>]
[ORDER BY (<expr> [ASC | DESC])*;
```

Ordre d'évaluation

```
SELECT... FROM ... WHERE ...
GROUP BY ... HAVING... ORDER BY
```

1- **FROM** : on recherche les relations concernées

2- **WHERE** : on effectue les jointures et les restrictions
   => résultat intermédiaire

3- **GROUP BY** : partitionnement du résultat intermédiaire en *groupes*

4- **HAVING** : filtrage des *groupes* => ensemble de n-uplets du résultat final (F)

5- **SELECT** : projections sur F => F'

6- Opérations ensemblistes

7- **ORDER BY** : ordonnancement des n-uplets de F'

REQUETES imbriquées

`(SELECT FROM WHERE …)` dans

la clause `WHERE` (ou `HAVING`)

- <valeur>
- <n-uplet>
- <liste de valeurs>
- <liste de n-uplets>

SELECT pour <expr> dans SELECT

(SELECT FROM WHERE)

SELECT [**DISTINCT** | **ALL**] (* | (<table> | <vue>).*

| <expr> [**AS** <alias>])*

FROM ((<table> | <vue>) [<alias>])*

[**WHERE** <condition>]

[**GROUP BY** (<expr>)* [**HAVING** <condition>]]

[(**INTERSECT** | **UNION** [**ALL**] | **EXCEPT**) <requête>]

[**ORDER BY** (<expr> [**ASC** | **DESC**])*;

SELECT pour <table> dans FROM

SELECT [**DISTINCT** | **ALL**] (***** | (<table> | <vue>)**.***

| <expr> [**AS** <alias>])*

FROM ((<table> | <vue>) [<alias>])*

[**WHERE** <condition>]

(SELECT FROM WHERE)

[**GROUP BY** (<expr>)* [**HAVING** <condition>]]

[(**INTERSECT** | **UNION** [**ALL**] | **EXCEPT**) <requête>]

[**ORDER BY** (<expr> [**ASC** | **DESC**])*;

SELECT dans <condition>

SELECT [**DISTINCT** | **ALL**] (* | (<table> | <vue>).*

| <expr> [**AS**

FROM ((<table> | <vue>) [

[**WHERE** <condition>]

[**GROUP BY** (<expr>)* [**HAVING** <condition>]]

[(**INTERSECT** | **UNION** [**ALL**] | **EXCEPT**) <requête>]

[**ORDER BY** (<expr> [**ASC** | **DESC**])*;

SELECT FROM WHERE

SELECT FROM WHERE

REQUETES imbriquées / WHERE

`WHERE expr θ (SELECT ..)`

» Query Q –

1- Q retourne une valeur atomique

θ est = <> < > <= >=

2- Q retourne un ensemble de valeurs atomiques

3- Q retourne un n-uplet

4- Q retourne un ensemble de valeurs n-uplets

5- `EXISTS UNIQUE`

… Condition (rappel)

`<expr>` [**NOT**] **IN** `<liste de valeurs>`

Teste si la valeur résultat de `<expr>` apparaît ou non dans `<liste de valeurs>`

(`<expr>*`**)** [**NOT**] **IN** `<liste de n-uplets>`

Teste si la valeur n-uplet, résultat de `<expr>*` apparaît dans `<liste de n-uplets>`

`<expr>` θ (**ALL** | **ANY** | **SOME**) `<liste de valeurs>`

Compare une valeur resultat de `<expr>` avec `<liste de valeurs>`

- `ALL` : vrai si toutes les valeurs de la liste vérifient la condition.
- `ANY` et `SOME` : vrai si une / quelques valeurs de la liste vérifient la condition.

EXISTS `<liste de valeurs>`

Vrai si `<liste de valeurs>` possède au moins un élément.

UNIQUE `<liste de valeurs>`

Vrai si `<liste de valeurs>` ne contient qu'un seul élément.

REQUETES imbriquées (1)

`WHERE expr θ (SELECT ..)`

- Query Q –

Q retourne une valeur atomique

θ est =, <>, <, >, <= ou >=

-- Les noms et prénoms des étudiants ayant eu moins que Dark Vador en bases de données

```
SELECT *  FROM Notes
WHERE cours = 'Bases de Données'
AND note <= (  SELECT DISTINCT note
                    FROM Notes
                     WHERE cours = 'Bases de Données'
                    AND prenom = 'Dark' AND nom = 'Vador' );
```

Attention … SELECT FROM WHERE doit retourner une seule valeur

REQUETES imbriquées (2)

`WHERE expr θ (SELECT ..)`

- Query Q –

Q retourne un ensemble de valeurs atomiques

θ est IN, NOT IN

θ est ΦANY / ΦALL avec Φ est =, <>, <, >, <= ou >=

ΦANY vrai si la comparaison (Φ) est vérifiée pour au moins un élément de E
= *ANY équivalent à IN*

ΦALL : vrai si la comparaison (Φ) est vérifiée pour tous les éléments de E
<> *ALL équivalent à NOT IN*

-- Quel cours a la moyenne la plus basse ?

```
SELECT cours, AVG(note)
FROM Notes
GROUP BY cours
HAVING AVG(note) <= ALL (
                        SELECT AVG(note)
                        FROM notes
                        GROUP BY cours
);
```

REQUETES imbriquées (3)

```
WHERE (expr1, …, exprN)  θ (SELECT ..)
```

- Query Q -

Q retourne un n-uplet

θ est = ou <>

Q retourne un ensemble de n-uplets

θ est IN, NOT IN

ΦANY, ΦALL où Φ est =, <>

-- Les noms et prénoms des étudiants n'ayant que des notes supérieures ou égales à 10

```
SELECT    prenom, nom
FROM      Eleves
WHERE (prenom, nom) NOT IN
      ( SELECT  prenom, nom
      FROM      Notes
      WHERE     note < 10 ) ;
```

REQUETES imbriquées / EXISTS

```
WHERE EXISTS   (SELECT ...)
                    - Query Q -
```

La condition est vraie si il EXISTE au moins un n-uplet satisfaisant la condition de la sous-requête (Q)

-- Les noms et prénoms des étudiants n'ayant pas de note :

```
SELECT prenom, nom
      FROM Eleves e
      WHERE NOT EXISTS ( SELECT *
```

⇒ il n'existe pas de notes pour l'étudiant courant e

```
                              WHERE prenom = e.prenom
                        AND nom = e.nom );
```

Synchronisation : évaluation de Q pour chaque n-uplet de la requête principale (e)

DIVISION avec NOT EXISTS

R(A,B) / S(B)

```
SELECT A
FROM R r
WHERE NOT EXISTS (SELECT null
        FROM S s
        WHERE NOT EXISTS (SELECT null
                FROM R
                WHERE A = r.A  AND B= s.B ));
```

=> il n'existe pas de s ∄ S avec lequel R n'est pas en « relation ».

-- Les noms et prénoms des étudiants qui ont une note dans toutes les matières

```
SELECT Nom, Prenom
FROM Eleves  r
WHERE NOT EXISTS (SELECT null
        FROM Notes s
        WHERE NOT EXISTS (SELECT null
                FROM Eleve  e, Notes n
                WHERE e.nom = r.nom
                AND e.prenom = r.prenom
                AND e.nom=n.nom
                AND e.prenom = n.prenom
                AND n.cours = s.cours ));
```

=> il n'existe pas de s (Note) avec lequel r (Nom, Prenom) n'est pas en « relation ».

DIVISION avec COUNT

-- Les noms et prénoms des étudiants qui ont une note dans toutes les matières

```
SELECT    n.nom, n.prenom
FROM      Notes n
GROUP BY  n.nom, n.prenom
HAVING count(*) = (SELECT count (distinct cours)
                   FROM Notes);
```

GRENOBLE INP UGA Ensimag

École nationale supérieure d'informatique et de mathématiques appliquées

JOIN plus exotiques

OUTER JOIN : jointure externe

- FULL OUTER JOIN retourne tous les n-uplets vérifiant la condition + les n-uplets complétés par null si la condition n'est pas vérifiée
- LEFT OUTER JOIN
- RIGHT OUTER JOIN

FULL OUTER JOIN

Les élèves avec leurs notes (y compris ceux qui n'en n'ont pas)

```
SELECT *
FROM  Eleves e
        FULL OUTER JOIN Notes n
        ON e.nom = n.nom
        AND e.prenom = n.prenom;
```

Élèves 2	prénom	nom	e-mail	filière
	Dark	Vador	vador@imag.fr	IF
	Obi-Wan	Kenobi	kenobio@imag.fr	MMIS

Notes	cours	prénom	nom	note
	sport	Dark	Vador	20
	sport	Jabba	The Hut	3
	pilotage	Han	Solo	15

Él 2 ⋈ Notes	prénom	nom	e-mail	filière	cours	note
	Dark	Vador	v[...]g.fr	IF	sport	20
	Obi-Wan	Kenobi	k[...]g.fr	MMIS	**null**	**null**
	Jabba	The Hut	**null**	**null**	sport	3
	Han	Solo	**null**	**null**	pilotage	15

LEFT OUTER JOIN

Les élèves et leurs notes (y compris ceux qui n'en n'ont pas)

```
SELECT *
FROM  Eleves e
        LEFT OUTER JOIN Notes n
        ON e.nom = n.nom
        AND e.prenom = n.prenom;
```

Élèves 2	prénom	nom	e-mail	filière
	Dark	Vador	vador@imag.fr	IF
	Obi-Wan	Kenobi	kenobio@imag.fr	MMIS

Notes	cours	prénom	nom	note
	sport	Dark	Vador	20
	sport	Jabba	The Hut	3
	pilotage	Han	Solo	15

Él 2 ⋈ Notes	prénom	nom	e-mail	filière	cours	note
	Dark	Vador	vador@imag.fr	IF	sport	20
	Obi-Wan	Kenobi	kenobio@imag.fr	MMIS	**null**	**null**

SEMI JOIN

Les élèves qui ont une note (Semi - Jointure à gauche)

```
SELECT *
FROM Eleves e
WHERE EXISTS
(SELECT * FROM Notes
        WHERE e.nom  = Notes.nom
        AND e.prenom  = Notes.prenom );
```

ou

```
SELECT e.*
FROM Eleves e, Notes n
WHERE e.nom  = n.nom
        AND e.prenom  = n.prenom ;
```

Élèves 2	prénom	nom	e-mail	filière
	Dark	Vador	vador@imag.fr	IF
	Obi-Wan	Kenobi	kenobio@imag.fr	MMIS

Notes	cours	prénom	nom	note
	sport	Dark	Vador	20
	sport	Jabba	The Hut	3
	pilotage	Han	Solo	15

Él 2 ⋉ Notes	prénom	nom	e-mail	filière
	Dark	Vador	vador@imag.fr	IF