Bases de données

Ecole Marocaine des Sciences de l'Ingénieur

© Yousra Lembachar

Plan

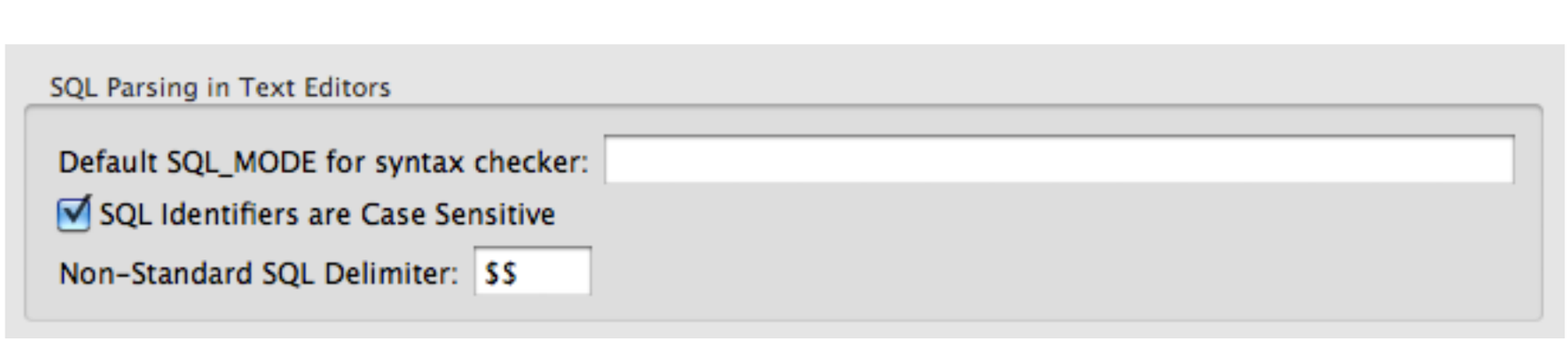
- Variables dans SQL
- Les fonctions d'agrégation
- Les opérateurs ensemblistes
- Les sous-requêtes SELECT
- Contraintes statiques et dynamiques

Rappel

- Requêtes CREATE/DROP de création/suppression de tables
- Requêtes UPDATE de mise à jour de lignes dans une ou plusieurs tables
- Requêtes DELETE de suppression de lignes dans une ou plusieurs tables
- Requêtes SELECT pour consultation des données

Remarques

- SQL n'est pas sensible à la casse... mais il est possible de le rendre sensible à la casse sur certains éditeurs



- La division entière se fait par l'opérateur DIV (c.f. Lien vers opérateurs arithmétiques)

Base de données

etudiant | matiere | note

Filter: Edit: File:

idEtudiant	nom
1	Samantha
2	Francis
3	Charles
4	Penelope
5	Craig
6	Steve
7	Dave
9	Cameron
10	Patricia
11	Patricia
	NULL

- universite
  - Tables
    - etudiant
    - matiere
    - note
  - Views
  - Routines

etudiant | matiere | note

Filter: Edit: File:

idMatiere	nomMatiere
1	Maths
2	Sociologie
3	Nutrition
4	Art Moderne
5	Art Contempo…
6	Art Classique
	NULL

etudiant | matiere | note

Filter: Edit: File:

idEtudiant	idMatiere	note
6	1	8
11	3	7
10	1	5
10	2	4
2	1	20
1	2	20
1	1	15
4	1	14
3	1	13
5	1	10
	NULL	NULL

Variables dans SQL

Variables dans SQL

```sql
SELECT
    Etudiant.idEtudiant, nom, note, nomMatiere
FROM
    ETUDIANT,
    MATIERE,
    NOTE
WHERE
    NOTE.idEtudiant = ETUDIANT.idEtudiant
        AND MATIERE.idMatiere = NOTE.idMatiere;
```

160% 48:9

Filter: File:

idEtudiant	nom	note	nomMatiere
1	Samantha	15	Maths
2	Francis	20	Maths
3	Charles	13	Maths
4	Penelope	14	Maths
5	Craig	10	Maths
6	Steve	8	Maths
10	Patricia	5	Maths
1	Samantha	20	Sociologie
10	Patricia	4	Sociologie
11	Patricia	7	Nutrition

Les identifiants, les noms, les notes et les matières respectives des étudiants ayant une note

Variables dans SQL

```sql
SELECT
    E.idEtudiant, nom, note, nomMatiere
FROM
    ETUDIANT E,
    MATIERE M,
    NOTE N
WHERE
    N.idEtudiant = E.idEtudiant
        AND M.idMatiere = N.idMatiere;
```

160% 39:9

Filter: File:

idEtudiant	nom	note	nomMatiere
1	Samantha	15	Maths
2	Francis	20	Maths
3	Charles	13	Maths
4	Penelope	14	Maths
5	Craig	10	Maths
6	Steve	8	Maths
10	Patricia	5	Maths
1	Samantha	20	Sociologie
10	Patricia	4	Sociologie
11	Patricia	7	Nutrition

Les identifiants, les noms, les notes et les matières respectives des étudiants ayant une note

Variables dans SQL

```
1  SELECT
2      E1.idEtudiant, E1.nom, E2.idEtudiant, E2.nom
3  FROM
4      ETUDIANT E1,
5      ETUDIANT E2
6  WHERE
7      E1.nom = E2.nom
8          AND E1.idEtudiant <> E2.idEtudiant
```

Les paires d'étudiants qui ont le même nom

160% 43:8

Filter: File:

idEtudiant	nom	idEtudiant	nom
11	Patricia	10	Patricia
10	Patricia	11	Patricia

```
1  SELECT
2      E1.idEtudiant, E1.nom, E2.idEtudiant, E2.nom
3  FROM
4      ETUDIANT E1,
5      ETUDIANT E2
6  WHERE
7      E1.nom = E2.nom
8          AND E1.idEtudiant < E2.idEtudiant
```

Les paires d'étudiants qui ont le même nom sans répétition

160% 28:8

Filter: File:

idEtudiant	nom	idEtudiant	nom
10	Patricia	11	Patricia

Fonctions d'agrégation

Fonctions d'agrégation

```
SELECT FONC_AGGR(COL),...
FROM nomTable1, nomTable2,...
WHERE condition
GROUP BY COL
HAVING condition
```

FONC_AGGR(COL) : MIN| MAX | AVG | SUM | COUNT

La clause GROUP BY

Query 1 | etudiant | matiere | note

```sql
SELECT
    idEtudiant, AVG(NOTE)
FROM
    NOTE
GROUP BY idEtudiant;
```

L'identifiant et la moyenne de chaque étudiant

idEtudiant	AVG(NOTE)
1	17.5000
2	20.0000
3	13.0000
4	14.0000
5	10.0000
6	8.0000
10	4.5000
11	7.0000

La clause GROUP BY

Query 1 | etudiant | matiere | note

```sql
SELECT
    ETUDIANT.idEtudiant, nom, AVG(NOTE)
FROM
    NOTE, ETUDIANT
WHERE NOTE.idEtudiant = ETUDIANT.idEtudiant
GROUP BY ETUDIANT.idEtudiant;
```

L'identifiant, le nom et la moyenne de chaque étudiant

Filter: File:

idEtudiant	nom	AVG(NOTE)
1	Samantha	17.5000
2	Francis	20.0000
3	Charles	13.0000
4	Penelope	14.0000
5	Craig	10.0000
6	Steve	8.0000
10	Patricia	4.5000
11	Patricia	7.0000

La clause HAVING

Query 1 | etudiant | matiere | note

```sql
SELECT
    NOTE.idEtudiant, nom, AVG(note)
FROM
    NOTE,
    ETUDIANT
WHERE
    NOTE.idEtudiant = ETUDIANT.idEtudiant
GROUP BY idEtudiant
HAVING AVG(note) > 10;
```

23:9

Filter: File:

idEtudiant	nom	AVG(note)
1	Samantha	17.5
2	Francis	20
3	Charles	13
4	Penelope	14

L'identifiant, le nom et la moyenne de chaque ayant une moyenne > 10

La clause COUNT

Query 1 | etudiant | matiere | note

```
1 • SELECT COUNT(NOTE)
2   FROM NOTE
```

Le nombre de notes

Filter: File:

COUNT(NOTE)
10

Query 1 | etudiant | matiere | note

```
1 • SELECT idEtudiant, COUNT(NOTE)
2   FROM NOTE
3   GROUP BY idEtudiant;
```

Le nombre de notes de chaque étudiant

21:3

idEtudiant	COUNT(NOTE)
1	2
2	1
3	1
4	1
5	1
6	1
10	2
11	1

Opérateurs ensemblistes

UNION

```
1 • SELECT nom as chaine FROM ETUDIANT
2   UNION
3   SELECT nomMatiere as chaine FROM MATIERE;
```

160% 42:3

Filter: File:

chaine
Samantha
Francis
Charles
Penelope
Craig
Steve
Dave
Cameron
Patricia
Maths
Sociologie
Nutrition
Art Moderne
Art Contempo…
Art Classique

Tous les noms des étudiants
et des matières

INTERSECT, EXCEPT

- Clauses non supportées par MySQL
- INTERSECT pour l'intersection de deux ensembles
- EXCEPT pour la différence entre deux ensembles

Sous-requêtes SELECT

Sous-requêtes SELECT

```sql
SELECT
    idEtudiant
FROM
    ETUDIANT
WHERE
    idEtudiant in (SELECT
            idEtudiant
        from
            note
        where
            note > 5);
```

Même résultat pour les deux requêtes

```sql
SELECT distinct
    Etudiant.idEtudiant
FROM
    ETUDIANT,
    NOTE
WHERE
    Etudiant.idEtudiant = Note.idEtudiant
        and note > 5
```

23:11

Filter: Edit: File:

idEtudiant
1
2
3
4
5
6

Les identifiants des étudiants ayant une note

Sous-requêtes SELECT

```
1  SELECT
2      nom
3  FROM
4      Etudiant
5  WHERE
6      idEtudiant in (SELECT
7              idEtudiant
8          from
9              NOTE)
```

160% 18:9

Filter: File:

nom
Samantha
Francis
Charles
Penelope
Craig
Steve
Patricia
Patricia

Les noms des étudiants ayant une note

etudiant | matiere

Filter:

idEtudiant	nom
1	Samantha
2	Francis
3	Charles
4	Penelope
5	Craig
6	Steve
7	Dave
9	Cameron
10	Patricia
11	Patricia
	NULL

etudiant | matiere | note

Filter: Edit: File:

idEtudiant	idMatiere	note
6	1	8
11	3	7
10	1	5
10	2	4
2	1	20
1	2	20
1	1	15
4	1	14
3	1	13
5	1	10
	NULL	NULL

Résultats différents

```
SELECT
    nom
FROM
    ETUDIANT,
    NOTE
WHERE
    Etudiant.idEtudiant
        = Note.idEtudiant
```

nom
Samantha
Samantha
Francis
Charles
Penelope
Craig
Steve
Patricia
Patricia
Patricia

```
SELECT distinct
    nom
FROM
    ETUDIANT,
    NOTE
WHERE
    Etudiant.idEtudiant
        = Note.idEtudiant
```

nom
Samantha
Francis
Charles
Penelope
Craig
Steve
Patricia

Sous-requêtes SELECT

```
1  SELECT
2      idEtudiant, nom
3  FROM
4      ETUDIANT
5  WHERE
6      idEtudiant not in (SELECT
7              idEtudiant
8          from
9              NOTE)
```

Les noms des étudiants n'ayant pas une note

160% 20:2

Filter: Edit: File:

idEtudiant	nom
7	Dave
9	Cameron

Sous-requêtes SELECT

```sql
SELECT
    idEtudiant, nom
FROM
    ETUDIANT E
WHERE
    not exists( SELECT
            idEtudiant
        from
            NOTE N
        where
            E.idEtudiant = N.idEtudiant)
```

Les noms des étudiants n'ayant pas une note

160% 41:11

Filter: Edit: File:

idEtudiant	nom
7	Dave
9	Cameron

Sous-requêtes SELECT

```sql
SELECT
    nom
FROM
    ETUDIANT
WHERE
    not (idEtudiant = any (SELECT idEtudiant FROM NOTE));
```

Les noms des étudiants n'ayant pas une note

160% 55:6

Filter: File:

nom
Dave
Cameron

Sous-requêtes SELECT

Les noms des étudiants ayant les notes les plus élevées

```
1  SELECT
2      nom
3  FROM
4      ETUDIANT,
5      NOTE
6  where
7      ETUDIANT.idEtudiant = NOTE.idEtudiant
8      and note >= all (SELECT note FROM NOTE);
```

160% 42:8

Filter: File:

nom
Samantha
Francis

Contraintes statiques et dynamiques

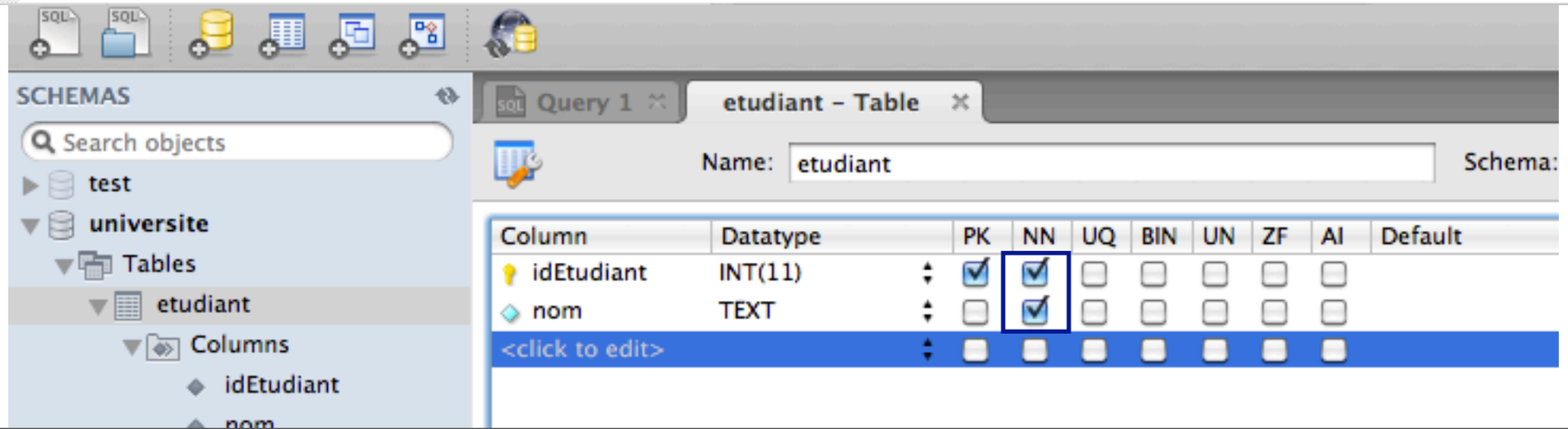
Types de contraintes

- Contraintes non null
- Contraintes de clés
- Contraintes sur les lignes

Contrainte non null - I

Query 1 | note – Table

```
1  CREATE TABLE etudiant (
2      idEtudiant INT NOT NULL,
3      nom TEXT NOT NULL,
4      CONSTRAINT pk1 PRIMARY KEY (idEtudiant)
5  )
```



Contrainte non null - 2

Query 1

```
1 • DESCRIBE ETUDIANT;
```

Query 1

Filter: File:

Field	Type	Null	Key	Default
idEtudiant	int(11)	NO	PRI	NULL
nom	text	NO		NULL

Query 1

```
1 • INSERT INTO ETUDIANT VALUES (8, NULL);
```

Contrainte non null violée!

39:1

Action Output

	Time	Action	Response
1	08:41:07	INSERT INTO ETUDIANT VALUES (8, NULL)	Error Code: 1048. Column 'nom' cannot be null

Contrainte de clé primaire

Query 1

```
1 • UPDATE ETUDIANT
2   set
3       idEtudiant = idEtudiant + 3
4   where
5       nom = 'Samantha';
```

22:5

etudiant

Filter:

idEtudiant	nom
1	Samantha
2	Francis
3	Charles
4	Penelope
5	Craig
6	Steve
7	Dave
NULL	

Action Output

	Time	Action	Response
1	12:02:53	UPDATE ETUDIANT set idEtudiant = idEtudiant + 3 where n...	Error Code: 1062. Duplicate entry '4' for key 'PRIMARY'

Contrainte d'intégrité (clé primaire) violée!

Contrainte de clé étrangère

Query 1

```
1 • INSERT INTO NOTE VALUES (17,1,13);
```

1:1

Action Output

		T	A	Response
✖	1	1	I.	Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`universite`.`note`, CONSTRAINT `fk1` FOREIGN KEY (`idEtudiant`)

Contrainte d'intégrité (clé étrangère) violée!

Contrainte de clé unique

```sql
DROP TABLE IF EXISTS MOYENNE;

CREATE TABLE MOYENNE (
    idEtudiant INT NOT NULL,
    moyenne INT NOT NULL,
    CONSTRAINT un1 unique (idEtudiant)
);

INSERT INTO MOYENNE VALUES (1,15),(1,20);
```

42:9

Action Output

		Time	Action	Response
✓	1	17:47:20	DROP TABLE IF EXISTS MOYENNE	0 row(s) affected
✓	2	17:47:20	CREATE TABLE MOYENNE (idEtudiant INT NOT NULL, moyenne INT NOT …	0 row(s) affected
✗	3	17:47:20	INSERT INTO MOYENNE VALUES (1,15),(1,20)	Error Code: 1062. Duplicate entry '1' for key 'un1'

Contrainte d'unicité violée!

Contrainte CHECK

```sql
DROP TABLE IF EXISTS MOYENNE;

CREATE TABLE MOYENNE (
    idEtudiant INT NOT NULL,
    moyenne INT NOT NULL
        check (moyenne = 20 and moyenne >= 0),
    CONSTRAINT un1 unique (idEtudiant)
);

INSERT INTO MOYENNE VALUES (1,15);

INSERT INTO MOYENNE VALUES (2,25);
```

Contrainte check violée!*

*Les contraintes check sont supportées par SQLite, mais non par MySQL.