Les sous requêtes

- On peut imbriquer un **SELECT** dans une autre requête.
- On distingue
  - les sous requêtes qui sélectionnent *une seule colonne* et renvoient *une seule valeur* ;
  - les sous requêtes qui sélectionnent *une seule colonne* et renvoient *une liste de valeurs* ;
  - les sous requêtes qui sélectionnent *plusieurs colonnes*.
- La sous requête se met entre parenthèses.
- La sous requête ne comprend pas de **ORDER BY**.

Sous requête qui renvoie une seule valeur

- On utilise les opérateurs =, <>, <, <=, >, >=.

```
SELECT MIN(amount2) FROM easy_drinks;
```

```
+--------------+
| MIN(amount2) |
+--------------+
|         0.75 |
+--------------+
```

```
SELECT drink_name FROM easy_drinks
  WHERE amount2=
    (SELECT MIN(amount2) FROM easy_drinks);
```

```
+------------+
| drink_name |
+------------+
| Lone Tree  |
| Blue Moon  |
| Lime Fizz  |
+------------+
```

```
SELECT * FROM easy_drinks;
+-----------------+--------------+---------+------------------+---------+
| drink_name      | main         | amount1 | second           | amount2 |
+-----------------+--------------+---------+------------------+---------+
| Kiss on the Lips | cherry juice |     2.0 | apricot nectar   |    7.00 |
| Hot Gold        | peach nectar |     3.0 | orange juice     |    6.00 |
| Lone Tree       | soda         |     1.5 | cherry juice     |    0.75 |
| Greyhound       | soda         |     1.5 | grapefruit juice |    5.00 |
| Indian Summer   | apple juice  |     2.0 | hot tea          |    6.00 |
| Bull Frog       | iced tea     |     1.5 | lemonade         |    5.00 |
| Soda and It     | soda         |     2.0 | grape juice      |    1.00 |
| Blackthorn      | tonic water  |     1.5 | pineapple juice  |    1.00 |
| Blue Moon       | soda         |     1.5 | blueberry juice  |    0.75 |
| Oh My Gosh      | peach nectar |     1.0 | pineapple juice  |    1.00 |
| Lime Fizz       | Sprite       |     1.5 | lime juice       |    0.75 |
+-----------------+--------------+---------+------------------+---------+
```

Sous requête qui renvoie une liste de valeurs

```
SELECT main FROM easy_drinks WHERE amount2=
   (SELECT MIN(amount2) FROM easy_drinks);
```

```
+--------+
| main   |
+--------+
| soda   |
| soda   |
| Sprite |
+--------+
```

```
SELECT drink_name FROM easy_drinks
  WHERE main IN
     (SELECT main FROM easy_drinks WHERE amount2=
        (SELECT MIN(amount2) FROM easy_drinks
        )
     );
```

```
+-------------+
| drink_name  |
+-------------+
| Lone Tree   |
| Greyhound   |
| Soda and It |
| Blue Moon   |
| Lime Fizz   |
+-------------+
```

```
SELECT * FROM easy_drinks;
+-----------------+-------------+---------+-----------------+---------+
| drink_name      | main        | amount1 | second          | amount2 |
+-----------------+-------------+---------+-----------------+---------+
| Kiss on the Lips | cherry juice |     2.0 | apricot nectar  |    7.00 |
| Hot Gold        | peach nectar |     3.0 | orange juice    |    6.00 |
| Lone Tree       | soda         |     1.5 | cherry juice    |    0.75 |
| Greyhound       | soda         |     1.5 | grapefruit juice |    5.00 |
| Indian Summer   | apple juice  |     2.0 | hot tea         |    6.00 |
| Bull Frog       | iced tea     |     1.5 | lemonade        |    5.00 |
| Soda and It     | soda         |     2.0 | grape juice     |    1.00 |
| Blackthorn      | tonic water  |     1.5 | pineapple juice |    1.00 |
| Blue Moon       | soda         |     1.5 | blueberry juice |    0.75 |
| Oh My Gosh      | peach nectar |     1.0 | pineapple juice |    1.00 |
| Lime Fizz       | Sprite       |     1.5 | lime juice      |    0.75 |
+-----------------+-------------+---------+-----------------+---------+
```

Comparer aux valeurs d'une liste avec ALL

ALL

- **ALL** permet d'utiliser les opérateurs de comparaison =, <>, <, <=, >, >= avec une liste de valeurs.
- **SELECT** ... **FROM** ... **WHERE** *myColumn myComparaisonOperator* **ALL** (**SELECT** ...)
- sélectionne les valeurs qui vérifient la condition pour toutes les valeurs renvoyées par la sous requête.

```
SELECT drink_name FROM easy_drinks
  WHERE amount1 < ALL
    (SELECT amount1 FROM easy_drinks WHERE amount2 BETWEEN 2 AND 6);
```

```
mysql> SELECT amount1 FROM easy_drinks WHERE amount2 BETWEEN 2 AND 6;
+---------+
| amount1 |
+---------+
|     3.0 |
|     1.5 |
|     2.0 |
|     1.5 |
+---------+
mysql> SELECT drink_name FROM easy_drinks WHERE amount1 < ALL (SELECT amount1 FROM easy_drinks WHERE amount2 BETWEEN 2 AND 6);
+------------+
| drink_name |
+------------+
| Oh My Gosh |
+------------+
```

```
SELECT * FROM easy_drinks;
+-----------------+--------------+---------+-----------------+---------+
| drink_name      | main         | amount1 | second          | amount2 |
+-----------------+--------------+---------+-----------------+---------+
| Kiss on the Lips | cherry juice |     2.0 | apricot nectar  |    7.00 |
| Hot Gold        | peach nectar |     3.0 | orange juice    |    6.00 |
| Lone Tree       | soda         |     1.5 | cherry juice    |    0.75 |
| Greyhound       | soda         |     1.5 | grapefruit juice |    5.00 |
| Indian Summer   | apple juice  |     2.0 | hot tea         |    6.00 |
| Bull Frog       | iced tea     |     1.5 | lemonade        |    5.00 |
| Soda and It     | soda         |     2.0 | grape juice     |    1.00 |
| Blackthorn      | tonic water  |     1.5 | pineapple juice |    1.00 |
| Blue Moon       | soda         |     1.5 | blueberry juice |    0.75 |
| Oh My Gosh      | peach nectar |     1.0 | pineapple juice |    1.00 |
| Lime Fizz       | Sprite       |     1.5 | lime juice      |    0.75 |
+-----------------+--------------+---------+-----------------+---------+
```

Comparer aux valeurs d'une liste avec ANY (SOME)

ANY (SOME)

- **ANY** permet d'utiliser les opérateurs de comparaison =, <>, <, <=, >, >= avec une liste de valeurs.
- **SOME** est équivalent à **ANY**.
- **SELECT** ... **FROM** ... **WHERE** *myColumn myComparaisonOperator* **ANY** (**SELECT** ...)
- sélectionne les valeurs qui vérifient la condition pour au moins une valeur renvoyée par la sous requête.

```
SELECT drink_name FROM easy_drinks
  WHERE amount1 < ANY
    (SELECT amount1 FROM easy_drinks WHERE amount2 BETWEEN 2 AND 6);
```

```
mysql> SELECT amount1 FROM easy_drinks WHERE amount2 BETWEEN 2 AND 6;
+---------+
| amount1 |
+---------+
|     3.0 |
|     1.5 |
|     2.0 |
|     1.5 |
+---------+
mysql> SELECT drink_name FROM easy_drinks WHERE amount1 < ANY (SELECT amount1 FROM easy_drinks WHERE amount2 BETWEEN 2 AND 6);

+------------------+                 | Bull Frog        |
| drink_name       |                 | Soda and It      |
+------------------+                 | Blackthorn       |
| Kiss on the Lips |                 | Blue Moon        |
| Lone Tree        |                 | Oh My Gosh       |
| Greyhound        |                 | Lime Fizz        |
| Indian Summer    |                 +------------------+
```

```
SELECT * FROM easy_drinks;
+-----------------+-------------+---------+-----------------+---------+
| drink_name      | main        | amount1 | second          | amount2 |
+-----------------+-------------+---------+-----------------+---------+
| Kiss on the Lips | cherry juice |     2.0 | apricot nectar  |    7.00 |
| Hot Gold        | peach nectar |     3.0 | orange juice    |    6.00 |
| Lone Tree       | soda        |     1.5 | cherry juice    |    0.75 |
| Greyhound       | soda        |     1.5 | grapefruit juice |    5.00 |
| Indian Summer   | apple juice |     2.0 | hot tea         |    6.00 |
| Bull Frog       | iced tea    |     1.5 | lemonade        |    5.00 |
| Soda and It     | soda        |     2.0 | grape juice     |    1.00 |
| Blackthorn      | tonic water |     1.5 | pineapple juice |    1.00 |
| Blue Moon       | soda        |     1.5 | blueberry juice |    0.75 |
| Oh My Gosh      | peach nectar |     1.0 | pineapple juice |    1.00 |
| Lime Fizz       | Sprite      |     1.5 | lime juice      |    0.75 |
+-----------------+-------------+---------+-----------------+---------+
```

Sous requête avec EXISTS ou NOT EXISTS

EXISTS

- ... **WHERE EXISTS** (*mySubQuery*) renvoie TRUE si la sous requête renvoie au moins 1 n-uplet et FALSE sinon.

```
SELECT toy FROM toys
  WHERE EXISTS (SELECT * FROM boys WHERE boys.toy_id=toys.toy_id);
```

```
+-----------------+
| toy             |
+-----------------+
| hula hoop       |
| balsa glider    |
| toy soldiers    |
| harmonica       |
| baseball cards  |
+-----------------+
```

```
SELECT toy FROM toys
  WHERE NOT EXISTS (SELECT * FROM boys WHERE boys.toy_id=toys.toy_id);
```

```
+----------------+
| toy            |
+----------------+
| tinker toys    |
| etch-a-sketch  |
| slinky         |
+----------------+
```

```
mysql> SELECT * FROM boys;
+--------+--------+--------+
| boy_id | boy    | toy_id |
+--------+--------+--------+
|      1 | Davey  |      3 |
|      2 | Bobby  |      5 |
|      3 | Beaver |      2 |
|      4 | Richie |      1 |
|      6 | Johnny |      4 |
|      5 | Billy  |      2 |
|      7 | Tony   |   NULL |
+--------+--------+--------+
```

```
mysql> SELECT * FROM toys;
+--------+----------------+
| toy_id | toy            |
+--------+----------------+
|      1 | hula hoop      |
|      2 | balsa glider   |
|      3 | toy soldiers   |
|      4 | harmonica      |
|      5 | baseball cards |
|      6 | tinker toys    |
|      7 | etch-a-sketch  |
|      8 | slinky         |
+--------+----------------+
```

Les vues

Vue

Une vue est une table virtuelle stockée sous forme de requête.

- Confidentialité : permet de restreindre l'accès aux données à certains utilisateurs.
- Lisibilité : une vue construite à partir de plusieurs tables est plus compréhensible.
- Adéquation : permet de fournir aux utilisateurs les données dont ils ont besoin avec des noms d'attributs qui leur conviennent.

Vue

- **CREATE VIEW** *myView* **AS** *myQuery*

Exemple de vue

```
CREATE VIEW toys_of_boys
  AS SELECT boy,toy FROM boys INNER JOIN toys on boys.toy_id=toys.toy_id;
```

```
mysql> DESC toys_of_boys;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| boy   | varchar(20) | YES  |     | NULL    |       |
| toy   | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

```
mysql> SELECT * FROM toys_of_boys;
+--------+----------------+
| boy    | toy            |
+--------+----------------+
| Davey  | toy soldiers   |
| Bobby  | baseball cards |
| Beaver | balsa glider   |
| Richie | hula hoop      |
| Johnny | harmonica      |
| Billy  | balsa glider   |
+--------+----------------+
```

```
mysql> SELECT * FROM boys;
+--------+--------+--------+
| boy_id | boy    | toy_id |
+--------+--------+--------+
|      1 | Davey  |      3 |
|      2 | Bobby  |      5 |
|      3 | Beaver |      2 |
|      4 | Richie |      1 |
|      6 | Johnny |      4 |
|      5 | Billy  |      2 |
|      7 | Tony   |   NULL |
+--------+--------+--------+
```

```
mysql> SELECT * FROM toys;
+--------+----------------+
| toy_id | toy            |
+--------+----------------+
|      1 | hula hoop      |
|      2 | balsa glider   |
|      3 | toy soldiers   |
|      4 | harmonica      |
|      5 | baseball cards |
|      6 | tinker toys    |
|      7 | etch-a-sketch  |
|      8 | slinky         |
+--------+----------------+
```

Les index

- Lorsqu'on définit une clé primaire ou que l'on indique une contrainte d'unicité sur un attribut, le SGBD crée automatiquement un index pour accéder rapidement aux enregistrements.
- On peut demander la création d'index supplémentaires sur une table ou sur une colonne pour augmenter les performances des requêtes pour des tables de grande taille

```
mysql> SHOW INDEX FROM boys;
+-------+------------+----------+--------------+-------------+-----------+-------------+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality |
+-------+------------+----------+--------------+-------------+-----------+-------------+
| boys  |          0 | PRIMARY  |            1 | boy_id      | A         |           7 |
| boys  |          1 | toy_id   |            1 | toy_id      | A         |           6 |
+-------+------------+----------+--------------+-------------+-----------+-------------+
+----------+--------+------+------------+---------+---------------+
| Sub_part | Packed | Null | Index_type | Comment | Index_comment |
+----------+--------+------+------------+---------+---------------+
|     NULL | NULL   |      | BTREE      |         |               |
|     NULL | NULL   | YES  | BTREE      |         |               |
+----------+--------+------+------------+---------+---------------+
```

Les contraintes d'intégrité référentielle

Elles assurent la cohérence de la base de données lorsque les opérations concernent des clés étrangères.

Insertion

Vérifier qu'une valeur de clé étrangère existe bien comme clé primaire dans la table référencée.

- Pour les suppressions et les mises à jour, on va détailler plusieurs comportements possibles.

Effacer un enregistrement référencé par une clé étrangère

Par défaut : ON DELETE NO ACTION

Le comportement par défaut consiste à empêcher la suppression d'un enregistrement tant qu'il est référencé par une clé étrangère.

```
mysql> DELETE from toys WHERE toy_id=1;
ERROR 1451 (23000): Cannot delete or update a parent row:
a foreign key constraint fails
(‘courses‘.‘boys‘, CONSTRAINT ‘boys_ibfk_1‘
FOREIGN KEY (‘toy_id‘) REFERENCES ‘toys‘ (‘toy_id‘))
```

```
mysql> SELECT * FROM boys;
+--------+--------+--------+
| boy_id | boy    | toy_id |
+--------+--------+--------+
|      1 | Davey  |      3 |
|      2 | Bobby  |      5 |
|      3 | Beaver |      2 |
|      4 | Richie |      1 |
|      6 | Johnny |      4 |
|      5 | Billy  |      2 |
|      7 | Tony   |   NULL |
+--------+--------+--------+
```

```
mysql> SELECT * FROM toys;
+--------+----------------+
| toy_id | toy            |
+--------+----------------+
|      1 | hula hoop      |
|      2 | balsa glider   |
|      3 | toy soldiers   |
|      4 | harmonica      |
|      5 | baseball cards |
|      6 | tinker toys    |
|      7 | etch-a-sketch  |
|      8 | slinky         |
+--------+----------------+
```

Effacer un enregistrement référencé par une clé étrangère

ON DELETE SET NULL

La référence vers l'objet effacé devient NULL.

```
mysql> ALTER TABLE boys DROP FOREIGN KEY fk_toys;
mysql> ALTER TABLE boys ADD CONSTRAINT fk_toys
         FOREIGN KEY (toy_id) REFERENCES toys(toy_id)
           ON DELETE SET NULL;

mysql> DELETE from toys WHERE toy_id=1;
Query OK, 1 row affected (0,01 sec)
```

```
mysql> SELECT * FROM toys;
+--------+----------------+
| toy_id | toy            |
+--------+----------------+
|      2 | balsa glider   |
|      3 | toy soldiers   |
|      4 | harmonica      |
|      5 | baseball cards |
|      6 | tinker toys    |
|      7 | etch-a-sketch  |
|      8 | slinky         |
+--------+----------------+
```

```
mysql> SELECT * FROM boys;
+--------+--------+--------+
| boy_id | boy    | toy_id |
+--------+--------+--------+
|      1 | Davey  |      3 |
|      2 | Bobby  |      5 |
|      3 | Beaver |      2 |
|      4 | Richie |   NULL |
|      5 | Johnny |      4 |
|      6 | Billy  |      2 |
|      7 | Tony   |   NULL |
+--------+--------+--------+
```

Effacer un enregistrement référencé par une clé étrangère

ON DELETE CASCADE

Tous les enregistrements contenant la référence effacée sont également effacés.

- ATTENTION, cela peut être dangereux. Dans l'exemple, si on efface `balsa glider`, on perd `Beaver` et `Billy`.

```
mysql> ALTER TABLE boys DROP FOREIGN KEY fk_toys;
mysql> ALTER TABLE boys ADD CONSTRAINT fk_toys
         FOREIGN KEY (toy_id) REFERENCES toys(toy_id)
           ON DELETE CASCADE;

mysql> DELETE from toys WHERE toy_id=2;
Query OK, 1 row affected (0,01 sec)
```

```
mysql> SELECT * FROM toys;
+--------+----------------+
| toy_id | toy            |
+--------+----------------+
|      3 | toy soldiers   |
|      4 | harmonica      |
|      5 | baseball cards |
|      6 | tinker toys    |
|      7 | etch-a-sketch  |
|      8 | slinky         |
+--------+----------------+
```

```
mysql> SELECT * FROM boys;
+--------+--------+--------+
| boy_id | boy    | toy_id |
+--------+--------+--------+
|      1 | Davey  |      3 |
|      2 | Bobby  |      5 |
|      4 | Richie |   NULL |
|      5 | Johnny |      4 |
|      7 | Tony   |   NULL |
+--------+--------+--------+
```

METTRE à un enregistrement référencé par une clé étrangère

Par défaut : ON UPDATE NO ACTION

Le comportement par défaut consiste à empêcher la modification d'un enregistrement lorsqu'il est référencé par une clé étrangère et que la modification ferait que la contrainte ne soit plus respectée (la nouvelle valeur n'est pas une clé de la table référencée).

```
mysql> UPDATE boys SET toy_id=9 WHERE boy='Davey';
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails
(`courses`.`boys`, CONSTRAINT `fk_toys` FOREIGN KEY (`toy_id`)
REFERENCES `toys` (`toy_id`))

mysql> UPDATE boys SET toy_id=1 WHERE boy='Davey'
Query OK, 1 row affected (0,01 sec)
mysql> SELECT * FROM boys;
+--------+--------+--------+
| boy_id | boy    | toy_id |
+--------+--------+--------+
|      1 | Davey  |      1 |
|      2 | Bobby  |      5 |
|      3 | Beaver |      2 |
|      4 | Richie |      1 |
|      6 | Johnny |      4 |
|      5 | Billy  |      2 |
|      7 | Tony   |   NULL |
+--------+--------+--------+
```

```
mysql> SELECT * FROM boys;
+--------+--------+--------+
| boy_id | boy    | toy_id |
+--------+--------+--------+
|      1 | Davey  |      3 |
|      2 | Bobby  |      5 |
|      3 | Beaver |      2 |
|      4 | Richie |      1 |
|      6 | Johnny |      4 |
|      5 | Billy  |      2 |
|      7 | Tony   |   NULL |
+--------+--------+--------+
```

```
mysql> SELECT * FROM toys;
+--------+----------------+
| toy_id | toy            |
+--------+----------------+
|      1 | hula hoop      |
|      2 | balsa glider   |
|      3 | toy soldiers   |
|      4 | harmonica      |
|      5 | baseball cards |
|      6 | tinker toys    |
|      7 | etch-a-sketch  |
|      8 | slinky         |
+--------+----------------+
```

Mettre à jour un enregistrement référencé par une clé étrangère

ON UPDATE SET NULL

La référence vers l'objet effacé devient NULL.

```
mysql> ALTER TABLE boys DROP FOREIGN KEY fk_toys;
mysql> ALTER TABLE boys ADD CONSTRAINT fk_toys
         FOREIGN KEY (toy_id) REFERENCES toys(toy_id)
           ON UPDATE SET NULL;

mysql> UPDATE toys SET toy_id=9 WHERE toy_id=3;
Query OK, 1 row affected (0,01 sec)
```

```
mysql> SELECT * FROM toys;
+--------+----------------+
| toy_id | toy            |
+--------+----------------+
|      2 | balsa glider   |
|      9 | toy soldiers   |
|      4 | harmonica      |
|      5 | baseball cards |
|      6 | tinker toys    |
|      7 | etch-a-sketch  |
|      8 | slinky         |
+--------+----------------+
```

```
mysql> SELECT * FROM boys;
+--------+--------+--------+
| boy_id | boy    | toy_id |
+--------+--------+--------+
|      1 | Davey  |   NULL |
|      2 | Bobby  |      5 |
|      3 | Beaver |      2 |
|      4 | Richie |   NULL |
|      5 | Johnny |      4 |
|      6 | Billy  |      2 |
|      7 | Tony   |   NULL |
+--------+--------+--------+
```

```
mysql> SELECT * FROM toys;
+--------+----------------+
| toy_id | toy            |
+--------+----------------+
|      2 | balsa glider   |
|      3 | toy soldiers   |
|      4 | harmonica      |
|      5 | baseball cards |
|      6 | tinker toys    |
|      7 | etch-a-sketch  |
|      8 | slinky         |
+--------+----------------+
```

```
mysql> SELECT * FROM boys;
+--------+--------+--------+
| boy_id | boy    | toy_id |
+--------+--------+--------+
|      1 | Davey  |      3 |
|      2 | Bobby  |      5 |
|      3 | Beaver |      2 |
|      4 | Richie |   NULL |
|      5 | Johnny |      4 |
|      6 | Billy  |      2 |
|      7 | Tony   |   NULL |
+--------+--------+--------+
```

Mettre à jour un enregistrement référencé par une clé étrangère

ON UPDATE CASCADE

Tous les enregistrements contenant la référence mise à jour sont également mis à jour.

```
mysql> ALTER TABLE boys DROP FOREIGN KEY fk_toys;
mysql> ALTER TABLE boys ADD CONSTRAINT fk_toys
         FOREIGN KEY (toy_id) REFERENCES toys(toy_id)
           ON UPDATE CASCADE;

mysql> UPDATE toys SET toy_id=9 WHERE toy_id=3;
Query OK, 1 row affected (0,01 sec)
```

```
mysql> SELECT * FROM toys;
+--------+----------------+
| toy_id | toy            |
+--------+----------------+
|      9 | toy soldiers   |
|      4 | harmonica      |
|      5 | baseball cards |
|      6 | tinker toys    |
|      7 | etch-a-sketch  |
|      8 | slinky         |
+--------+----------------+
```

```
mysql> SELECT * FROM boys;
+--------+--------+--------+
| boy_id | boy    | toy_id |
+--------+--------+--------+
|      1 | Davey  |      9 |
|      2 | Bobby  |      5 |
|      4 | Richie |   NULL |
|      5 | Johnny |      4 |
|      7 | Tony   |   NULL |
+--------+--------+--------+
```


```
mysql> SELECT * FROM toys;
+--------+----------------+
| toy_id | toy            |
+--------+----------------+
|      3 | toy soldiers   |
|      4 | harmonica      |
|      5 | baseball cards |
|      6 | tinker toys    |
|      7 | etch-a-sketch  |
|      8 | slinky         |
+--------+----------------+
```

```
mysql> SELECT * FROM boys;
+--------+--------+--------+
| boy_id | boy    | toy_id |
+--------+--------+--------+
|      1 | Davey  |      3 |
|      2 | Bobby  |      5 |
|      4 | Richie |   NULL |
|      5 | Johnny |      4 |
|      7 | Tony   |   NULL |
+--------+--------+--------+
```