# Le langage SQL

## Numérique & Science Informatique

# Le langage SQL

## Numérique & Science Informatique

**Langage normalisé**

**pour gérer**

**les bases de données relationnelles**

# Le langage SQL

## Numérique & Science Informatique

### Types de données

Le langage SQL utilise différents types de données :

- Entiers : INT, mais aussi SMALLINT, TINYINT et BIGINT
- Nombres à virgules : FLOAT, DOUBLE, DECIMAL, ...
- Chaînes de caractères : CHAR, VARCHAR, TINYTEXT, ...
- Dates : DATETIME, DATE, TIME
- etc.

# Le langage SQL

## Numérique & Science Informatique

### Création de table

```
CREATE TABLE Eleves (
   idEleve  INT UNSIGNED AUTO_INCREMENT,
   nom      VARCHAR(100) NOT NULL,
   prenom   VARCHAR(50)  NOT NULL,
   email    TINYTEXT,

   CHECK (nom <> '' AND prenom <> ''), -- contraintes
   PRIMARY KEY(idEleve),
   UNIQUE KEY(nom, prenom),

   INDEX idx_nom(nom)
   ) ENGINE=MyISAM
```

# Le langage SQL

## Numérique & Science Informatique

### La valeur NULL

NULL représente une valeur inconnue

- NULL ne vaut pas '0'
- NULL ne vaut pas 'chaîne vide'

=> ne peut être filtrée par les opérateurs de comparaison

Pour filtrer les champs d'une colonne il faut utiliser la syntaxe :

```
IS NULL
IS NOT NULL
```

# Le langage SQL

## Numérique & Science Informatique

### Modification de table

Attributs :

```
ALTER TABLE Eleves DROP email
ALTER TABLE Eleves ADD mail VARCHAR(40) AFTER prenom
ALTER TABLE Eleves CHANGE mail email TINYTEXT
```

Indexes :

```
ALTER TABLE Eleves ADD UNIQUE email (email)
ALTER TABLE Eleves DROP INDEX email
```

# Le langage SQL

## Numérique & Science Informatique

### Destruction de table

```
DROP TABLE Eleves
```

ou

```
DROP TABLE IF EXISTS Eleves
```

# Le langage SQL

## Numérique & Science Informatique

### Insertion de données

```
INSERT INTO Eleves
VALUES (1, 'Bajot', 'Julien', 'j.bajot@nsi.org')
```

ou

```
INSERT INTO Eleves (nom, prenom, email)
VALUES ('Bajot', 'Julien', 'j.bajot@nsi.org')
```

# Le langage SQL

## Numérique & Science Informatique

### Relation entre tables

```
FOREIGN KEY(idEleve)
REFERENCES Eleves(idEleve)
```

Relation Eleves

| idEleve | nom | prenom |
|---|---|---|
| 1 | Bajot | Julien |
| 2 | Balas | Aliocha |
| 3 | Bister | Cyril |
| ... | ... | ... |

Relation Log

| idLog | idEleve | date | details |
|---|---|---|---|
| 1 | 2 | 28/03/2016 17:28:32 | |
| 2 | 1 | 28/03/2016 18:45:01 | |
| 3 | 2 | 29/03/2016 00:05:57 | Erreur d'authentification |
| ... | ... | ... | ... |

# Le langage SQL

## Numérique & Science Informatique

### Recherche de données

```
SELECT nom, prenom
FROM Eleves
WHERE email IS NOT NULL AND nom LIKE 'a%'
ORDER BY nom ASC
```

Algèbre :

$$\Pi_{nom,prenom}\ \sigma_{(email = \neg NULL \wedge nom = 'a*')}\ Eleves$$

Calcul relationnel :

$$\{ x,y \mid \exists\ x,z\ (Eleves(x,y,z) \wedge (x='a*' \wedge z<>NULL)) \}$$

# Le langage SQL

## Numérique & Science Informatique

### Recherche avec jointure

```
SELECT Eleves.nom, Eleves.prenom, Log.details
FROM Eleves, Log
WHERE Eleves.idEleve = Log.idEleve
AND Eleves.details IS NOT NULL
ORDER BY Log.date DESC, Eleves.nom ASC
```

$$\Pi_{nom,prenom,details}\ \sigma_{(details = \neg NULL)}\ Eleves \bowtie_{idEleve} Log$$

$$\{\ x1,x2,y4\ \mid \exists\ x3,y1,y2,y3\ (\ Eleves(x1,x2,x3) \wedge Log(y1,y2,y3,y4) \wedge (x1=y2 \wedge y4 = \neg NULL))\ \}$$

# Le langage SQL

## Numérique & Science Informatique

### Autres syntaxes 1/2

```
SELECT e.nom, e.prenom, l.details
FROM Eleves AS e, Log AS l
WHERE e.idEleve = l.idEleve
AND e.details IS NOT NULL
ORDER BY l.date DESC, e.nom ASC
```

# Le langage SQL

## Numérique & Science Informatique

### Autres syntaxes 2/2

Jointure naturelle : clé primaire d'une table = clé étrangère de l'autre

```
SELECT nom, prenom, details
FROM Eleves
NATURAL JOIN Log
```

Si les attributs n'ont pas le même nom :

```
SELECT nom, prenom, details
FROM Eleves
JOIN Log
ON (idEleve = idStudent)
```

# Le langage SQL

## Numérique & Science Informatique

### Requête imbriquée

Avec quantification existentielle

```
SELECT nom
FROM Eleves
WHERE EXISTS (SELECT ''
    FROM Logs
    WHERE Eleves.idEleve = Log.idEleve
    AND details = 'Erreur d'authentification')
```

# Le langage SQL

## Numérique & Science Informatique

### Requête imbriquée

Dite corrélée (condition d'appartenance)

```
SELECT nom
FROM Eleves
WHERE idEleve IN (SELECT idEleve
    FROM Logs
    WHERE details = 'Erreur d'authentification')
```

# Le langage SQL

## Numérique & Science Informatique

### Recherche sur des valeurs NULL

Pour des valeurs NULL, la requête avec les jointures précédentes renvoie aucune valeur

Il faut alors utiliser l'opérateur left join

- Renvoie tous les n-uplets de la table directrice (gauche)
- Associe un n-uplet de la table gauche avec n-uplet table droite
- Le n-uplet de droite vaut NULL s'il n'existe pas

```
SELECT nom, details
FROM Eleves
LEFT JOIN Logs
ORDER BY nom ASC
```

# Le langage SQL

## Numérique & Science Informatique

### Cas extrême

Sans clause where, on effectue un produit cartésien

```
SELECT *
FROM Eleves
CROSS JOIN Logs
ORDER BY nom ASC
```

# Le langage SQL

## Numérique & Science Informatique

### La négation

On veut les élèves qui n'ont pas eu d'erreur d'identification.

```
SELECT DISTINCT nom
FROM Eleves, Log
WHERE Eleves.idEleve = Log.idEleve
AND details != 'Erreur d'authentification'
```

Cette requête est incorrecte !

Elle donne des noms pour des problèmes autre qu'une erreur d'identification

# Le langage SQL

## Numérique & Science Informatique

**La solution correcte de la négation**

Il ne doit pas exister d'erreur d'identification parmi les logs.

=> négation d'une condition existentielle

```
SELECT nom
FROM Eleves
WHERE NOT EXISTS (SELECT ''
    FROM Log
    WHERE Eleves.idEleve = Log.idEleve
    AND details = 'Erreur d'authentification')
```

Ou `NOT IN` pour une requête corrélée

# Le langage SQL

## Numérique & Science Informatique

### Fonctions d'agrégation

fonctions pour effectuer des statistiques sur une table

| | |
|---|---|
| AVG() | calcul de moyenne sur un ensemble d'enregistrement |
| COUNT() | comptage du nombre d'enregistrement |
| MAX() | récupère la valeur maximum d'une colonne |
| MIN() | récupère la valeur minimum d'une colonne |
| SUM() | calcul la somme sur un ensemble d'enregistrement |

# Le langage SQL

## Numérique & Science Informatique

### Partitionnement & Filtrage

Les fonctions d'agrégation prennent tout leur sens lorsqu'elles sont utilisées avec les commandes :

- GROUP BY partitionnement de valeurs
- HAVING filtrage sur fonction d'agrégation

```
SELECT idEleve, COUNT(*) AS total
FROM Log
WHERE details = 'Erreur d'authentification'
GROUP BY idEleve
HAVING COUNT(*) > 2
ORDER BY idEleve DESC
```

# Le langage SQL

## Numérique & Science Informatique

### Mise à jour & Suppression

```
UPDATE Eleves
SET nom='Barjot', email ='j.barjot@nsi.org'
WHERE IDeleve='1'

DELETE FROM Eleves
WHERE IDeleve='1'

DROP TABLE Eleves
```

# Le langage SQL

## Numérique & Science Informatique

**Couche d'accès aux données***

couche d'un programme informatique qui fournit un accès simplifié aux données stockées dans une base de données

Ex :
outils de mappage objet-rationnel (ORM)

- Doctrine (PHP)
- Sequelize (javascript)

*Data Access Layer (DAL)

# Le langage SQL

## Numérique & Science Informatique

### Object–Relational Mapping*

Traduction de la représentation logique des objets en une forme atomisée capable d'être stockée dans la base de données tout en préservant les propriétés des objets et leurs relations

- Sans ORM :

```
$stmt   = $pdo->query("SELECT * FROM users where id='10'") ;
$users  = $stmt->fetchAll();
$name = $users[0]['nom'];
```

- Avec API ORM :

```
$user   = repository.GetUser(10);
$name = user.GetName();
```



*ORM

# Le langage SQL

## Numérique & Science Informatique

### Exemple de DAL en PHP 1/2

```
class Bdd {
   private $dbh = NULL;         // database handler

   public function __construct($servname, $database, $user, $password) {
      $this->dbh = new PDO(
         "mysql:host=$servname;dbname=$database", $user, $password);

      $this->dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
      $this->dbh->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE,
            PDO::FETCH_ASSOC);
      }

   public function __destruct() {
      $this->dbh = NULL;
      }

   // ...
```

# Le langage SQL

## Numérique & Science Informatique

### Exemple de DAL en PHP 2/2

```
public function get_all_database() : array {
  $request = "SELECT * FROM TABLE_NAME";

  if ( ($result = $this->dbh->query($request)) !== false )
      return $result->fetchAll();
  }

public function get_data_from(int $id) : array {
  $request = "SELECT * FROM TABLE_NAME WHERE id = ?";

  if ( ($stmt = $this->dbh->prepare($request)) !== false )
    if  ( $stmt->execute( array($id)  );
      return $stmt->fetchAll();
  }

  // ...
}
```

# Le langage SQL

## Numérique & Science Informatique