POO (Programmation orienté objet)

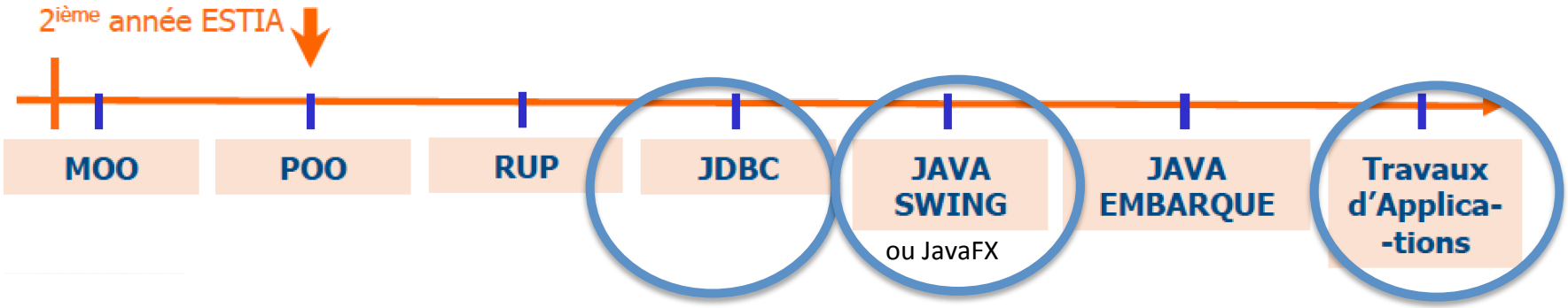
JDBC –
Java Database Connectivity

Patrick Reuter

Maître de conférences (associate professor)
Bordeaux University
Inria Bordeaux Sud-Ouest

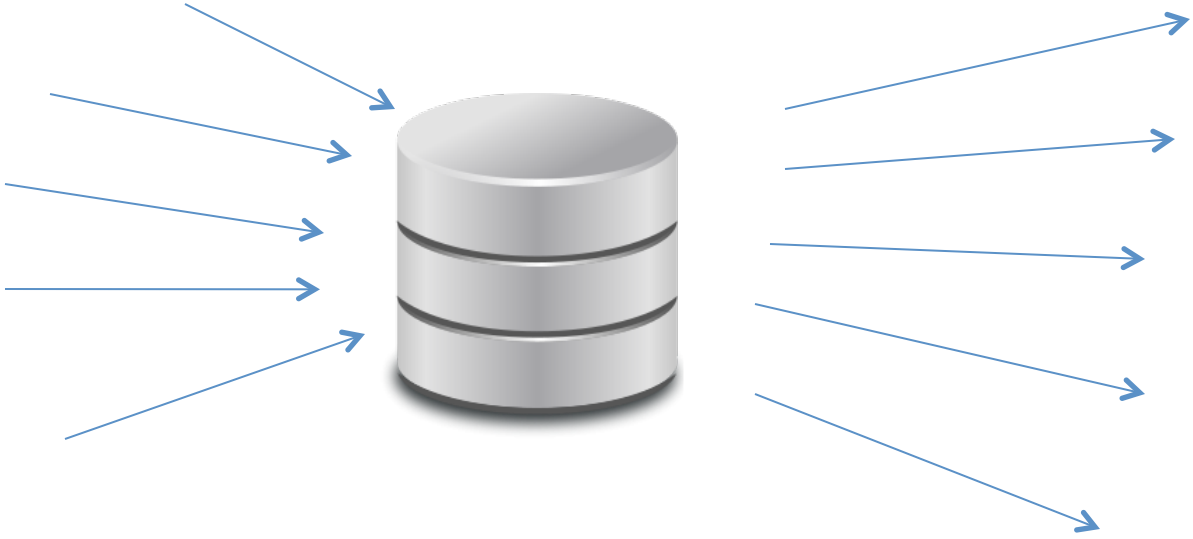
October 9th, 2018

Lecture integration



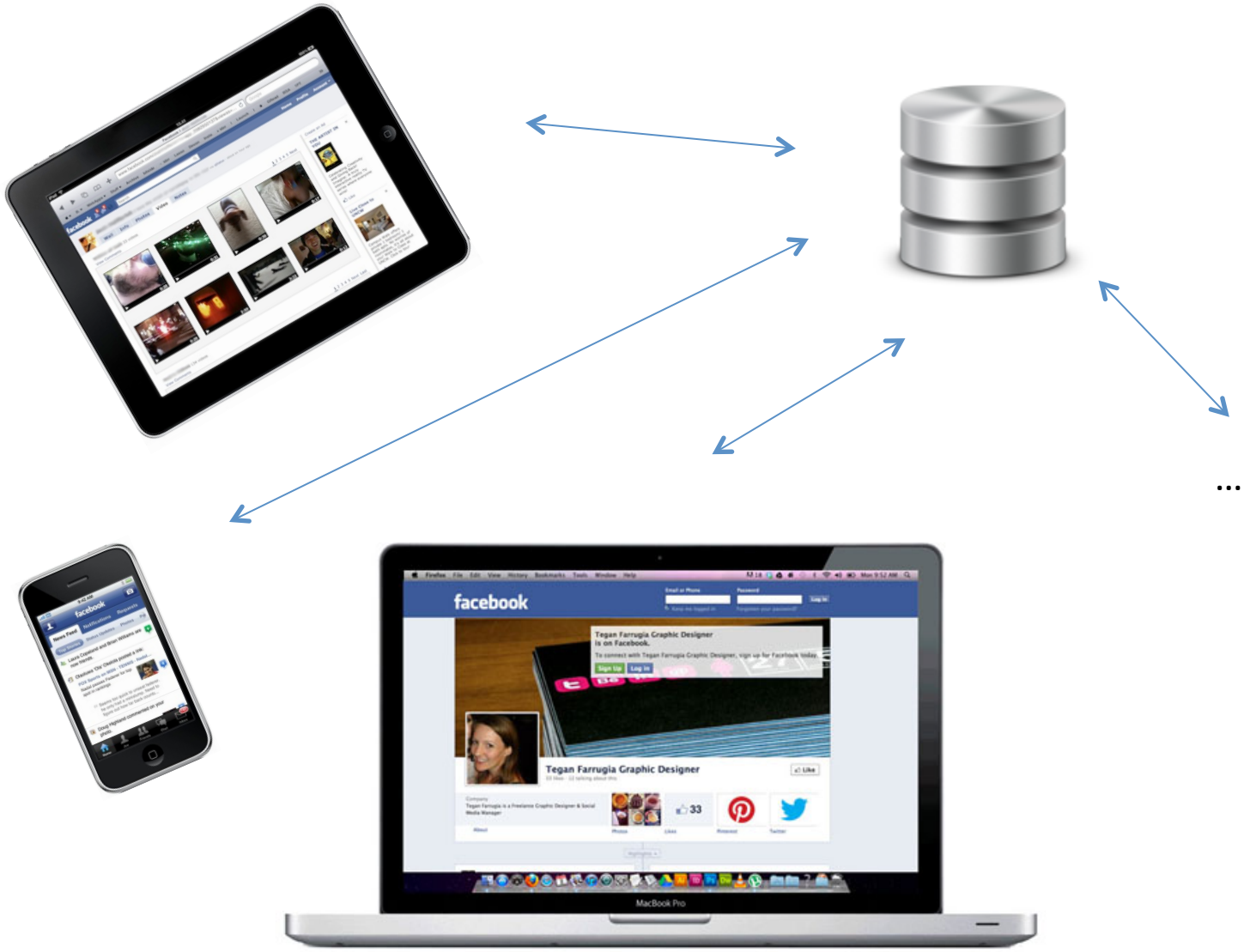
Base de données - Fonctionnement

Insérer et récupérer des données

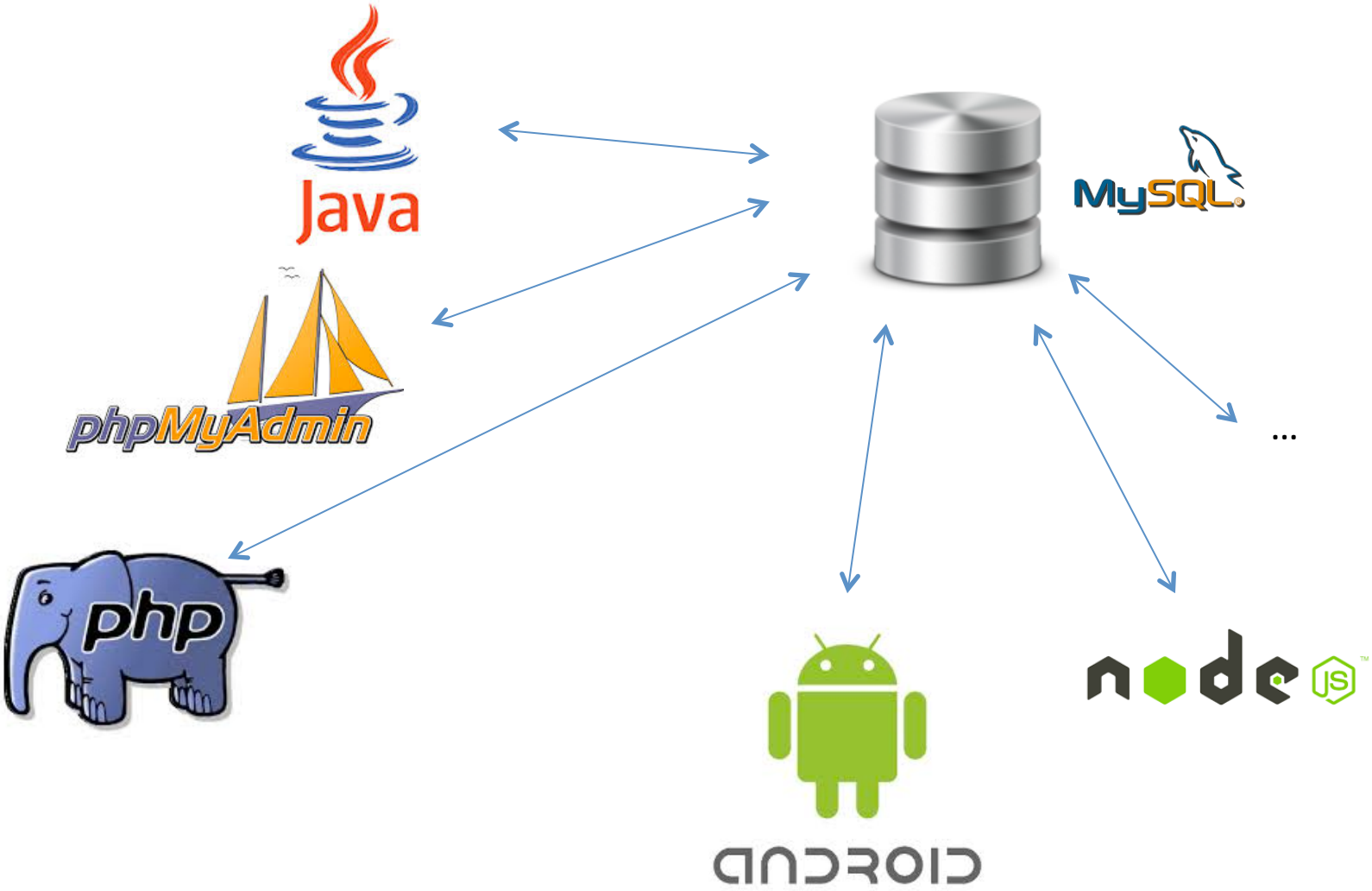


...

Database access



Database access



Databases - Terminology

- Databases *(base de données)*
- Tables *(tables)*
- Fields, columns *(champs, colonnes)*
- Records, rows *(enregistrements, lignes)*

Principles

- *Distinction : Structure / data*
- *No data duplication*

personne				
id	nom	prenom	date_naiss	id_ville
1	Elissalde	Amaia	01-01-1950	1
2	Lagarde	Xabi	29-04-1999	2
3	Leclerc	André	26-12-2000	1

ville				
id	nom	population	superficie	region
1	Paris	2 250 000	105,4	1
2	Lyon	445 000	47,9	2
3	Grenoble	800 000	240	1

...

- *Example :*
  - *Structure : **1 database** with **2 tables** consisting of **5 fields/columns** each*
  - *Data : **3 records/rows** per table*

JDBC

- API for the Java programming language
- Methods for database access :
  - querying data
  - updating data

TP JDBC

Installation :

- To access the DBMS driver :
  - Eclipse : Build-Path -> Add External Archives -> mysql-connector-java-x.x.x-bin.jar
  - Command line : add the following value to the environment variable CLASSPATH :

```
.;chemin_du_connector_mysql\mysql-connector-java-x.x.x-bin.jar
```

Standard SQL queries

- SELECT
  - Select records
- INSERT
  - Insert records
- UPDATE
  - Upate records/fields of records
- DELETE
  - Delete records

Database connection

```
import java.sql.*;

// import java.sql.Connection;
// import java.sql.DriverManager;
// import java.sql.Statement;

String nomUser = "root"; // Utilisateur de la BD
String passwd = "root"; // Password de l'utilisateur de la BD
String url = "jdbc:mysql://localhost/"; // Serveur de la BD
String nomBase = "estiajdbc"; // Nom de la BD sur laquelle nous allons acceder


private Connection _conn;
_conn = null;

try
{
    Class.forName("com.mysql.jdbc.Driver");
    _conn=DriverManager.getConnection(url+nomBase, nomUser, passwd);

}
catch (SQLException ex1)
{
    System.out.println("J'ai detecté une erreur de type SQL: " + ex1.getMessage());
}
catch (Exception ex2)
{
    System.out.println("J'ai detecté une erreur de type lang: " + ex2.getMessage());
}
```

QUERIES

- CREATE
- INSERT
- UPDATE
- **SELECT**

Query : SELECT

```
SELECT nom, login, mdp FROM employe
```

Query : SELECT

```
try
{
     String requete = new String("SELECT nom, login, mdp FROM employe");

     Statement stmt = _conn.createStatement();
     ResultSet rs = stmt.executeQuery(requete);

     while (rs.next ())
     {
            System.out.println(rs.getString("nom"));
            System.out.println(rs.getString("login"));
            System.out.println(rs.getString("mdp"));

     }
     rs.close();
     stmt.close();
}
catch (SQLException ex3)
{
     while (ex3 != null)
     {
       System.out.println(ex3.getSQLState());
       System.out.println(ex3.getMessage());
       System.out.println(ex3.getErrorCode());
       ex3=ex3.getNextException();
     }
}
```

Query : SELECT

Requêtes

```
SELECT nom, login, mdp FROM employe
```

Chercher un login + mdp particulier

Requête: 1ère solution (vérification par Java) – BAD (no prepared statement, verification by Java, no MD5 or other)

```
boolean trouve = false;
try
{
    String requete = new String("SELECT nom, login, mdp FROM employe");

    Statement stmt = _conn.createStatement();
    ResultSet rs = stmt.executeQuery(requete);

    while (rs.next ())
    {
     if (rs.getString("login").equals(login)) && rs.getString("motdepasse").equals(mdp))
          trouve = true;
    }
    rs.close();
    stmt.close();
}
catch (SQLException ex3)
{
    while (ex3 != null)
    {
     System.out.println(ex3.getSQLState());
     System.out.println(ex3.getMessage());
     System.out.println(ex3.getErrorCode());
     ex3=ex3.getNextException();
    }
}
```

QUERY : SELECT

Requêtes

```
SELECT nom, login, mdp FROM employe
```

Requêtes

```
SELECT nom, login, mdp FROM employe WHERE login='Thierry'
   AND mdp = md5('Ofkefji88_i98')
```

Requête: 2ème solution (mieux : vérification par MySQL) – BAD (no prepared statement, no MD5 or other)

```
try
{
    String requete = new String("SELECT nom, login, mdp FROM employe WHERE login =
    '"+login+"' AND mdp= '"+mdp+"'");
    Statement stmt = _conn.createStatement();
    ResultSet rs = stmt.executeQuery(requete);
    if (rs.next ()) /* s'il y a au moins un enregistrement */
    {
     trouve = true;
    }
    rs.close();
    stmt.close();
}
catch (SQLException ex3)
{
    while (ex3 != null)
    {
     System.out.println(ex3.getSQLState());
     System.out.println(ex3.getMessage());
     System.out.println(ex3.getErrorCode());
     ex3=ex3.getNextException();
    }
}
```

Requête: 3ème solution (prepared statement) -BAD (no MD5 or other)

```
try
{
     String requete = new String("SELECT id, nom, login FROM employe WHERE login=? AND mdp=?;");

     PreparedStatement stmt = _conn.prepareStatement(requete);

     stmt.setString(1, login);
     stmt.setString(2, mdp);

     ResultSet rs = stmt.executeQuery();

     if (rs.next ()) /* s'il y a au moins un enregistrement */
     {
       trouve = true;
     }
     rs.close();
     stmt.close();
}
catch (SQLException ex3)
{
     while (ex3 != null)
     {
       System.out.println(ex3.getSQLState());
       System.out.println(ex3.getMessage());
       System.out.println(ex3.getErrorCode());
       ex3=ex3.getNextException();
     }
}
```

Requête: 4ème solution (prepared statement) -GOOD

```
try
{
     String requete = new String("SELECT id, nom, login FROM employe WHERE login=? AND mdp=MD5(?);");

     PreparedStatement stmt = _conn.prepareStatement(requete);

     stmt.setString(1, login);
     stmt.setString(2, mdp);

     ResultSet rs = stmt.executeQuery();

     if (rs.next ()) /* s'il y a au moins un enregistrement */
     {
       trouve = true;
     }
     rs.close();
     stmt.close();
}
catch (SQLException ex3)
{
     while (ex3 != null)
     {
       System.out.println(ex3.getSQLState());
       System.out.println(ex3.getMessage());
       System.out.println(ex3.getErrorCode());
       ex3=ex3.getNextException();
     }
}
```

QUERIES

- CREATE
- **INSERT**
- UPDATE
- SELECT

Query : INSERT

Requête: 1ère solution - MAUVAIS

```
try
{
     String requete = new String(  "INSERT INTO employe(id, nom, login,mdp) VALUES (NULL, '"+nom+"',
     '"+login+"', MD5('"+mdp+"'));" );

     Statement stmt = _conn.createStatement();

     // à noter : executeUpdate
     int nombre = stmt.executeUpdate(requete);

}
catch (SQLException ex3)
{
     while (ex3 != null)
     {
       System.out.println(ex3.getSQLState());
       System.out.println(ex3.getMessage());
       System.out.println(ex3.getErrorCode());
       ex3=ex3.getNextException();
     }
}
```

Query : INSERT

Requête: 2ème solution - CORRECTE

```
try
{
     String requete = "INSERT INTO employe(id,        nom,     login,     mdp,       date) VALUES
     (NULL, ? , ? , MD5(?) , NOW() );";


     System.out.println(requete);
     PreparedStatement stmt = _conn.prepareStatement(requete);
     stmt.setString (1, nom);
     stmt.setString (2, login);
     stmt.setString (3, mdp);

     // à noter : executeUpdate sans paramètres !
     int nombre = stmt.executeUpdate();

}
catch (SQLException ex3)
{
     while (ex3 != null)
     {
       System.out.println(ex3.getSQLState());
       System.out.println(ex3.getMessage());
       System.out.println(ex3.getErrorCode());
       ex3=ex3.getNextException();
     }
}
```

Types de relations

- Relation de type 1:1
- Relation de type 1:n
- Relation de type n:m

Relation de type 1:1

Une entité est partenaire de **exactement** une autre entité

Exemples:

- Mariage : Une personne est marié avec exactement une autre personne

  ⇒ **Personne 1:1 Personne**
- Immatriculation : Une véhicule à exactement une immatriculation

  ⇒ **Véhicule 1:1 Immatriculation**
- Stade : Dans notre exemple, un club a exactement un stade

  ⇒ **Club 1:1 Stade**

Relation 1:1

RELATION 1 :1

Table voiture

id	marque	couleur
1	Swift	verte
2	Mini	rouge
3	206	bleu
4	Polo	anthracitezze

Table immatriculation

id	numero	id_voiture
1	2643 LH 33	4
2	9826 AA 64	3
3	8650 CD 75	2
4	9245 AB 78	1

→ Pas de duplication de valeurs dans les clés étrangères

→ On pourrait fusionner les 2 tables en 1

- Relation 1 : n

Relation de type 1:n

Une entité d'un côté est en relation avec un ou plusieurs entités de l'autre côté.

Exemples :

- Art : Un tableau se trouve (à un moment donné) dans un musée. Par contre, dans un musée il y a plusieurs tableaux.

  ⇒ **Musée 1:n Tableaux**

- Famille: Un enfant à exactement une mère biologique. Mais une mère peut avoir plusieurs enfants.

  ⇒ **Mère 1:n Enfants**

Relation 1:n

FAUX :

Questionquiz

Id	**question**	**reponse1**	**reponse2**	**reponse3**	**bonnreponse**
1	Quelle est la capitale d'Australie ?	Sydney	Melbourne	Canberra	3
2	Quelle est la capitale de Madagaskar ?	Paris	Tanarido	Madagaskar	2

CORRECT :

question

Id	**question**	id_bonne_reponse
1	Quelle est la capitale d'Australie ?	3
2	Quelle est la capitale de Madagaskar ?	5

reponse

id	**reponse**	**id_question**
1	Sydney	1
2	Melbourne	1
3	Canberra	1
4	Paris	2
5	Tanarido	2
6	Madagaskar	2

Relation n:m

Dans les deux côtés de la relation il y a un nombre arbitraire d'entités.

Attention : Ne pas écrire n:n ! (n ≠ m)
Exemples :

- Etudiant ↔ Professeur: Un professeur enseigne plusieurs étudiants. Un étudiant va en cours de plusieurs professeurs.
  ⇒ **Professeur m:n Etudiant**

- joueurs ↔ clubs anciens : Un joueur peut avoir plusieurs clubs anciens. Un club a plusieurs joueurs.
  ⇒ **Joueur n:m clubs**

Relation n:m - Exemple

Table : etudiants

id	prenom	nom
1	Patrick	Reuter
2	Pierrick	Legrand
3	Ion	Lapteacru
4	Brigitte	Patouille
5	Michel	Langlais

Table : professeurs

id	prenom	nom
1	Lison	Lafitte
2	Eloise	Etalon
3	Amandine	Grec
4	Arnaud	Filère
5	Romain	Roussel

Table de jointure : participe

id	id_professeur	id_etudiants
1	1	1
2	1	2
3	1	3
4	1	4
5	2	1
6	2	2
7	2	3
8	2	4
9	3	4