JDBC et Projet BDCO

ENSIMAG 2ème année

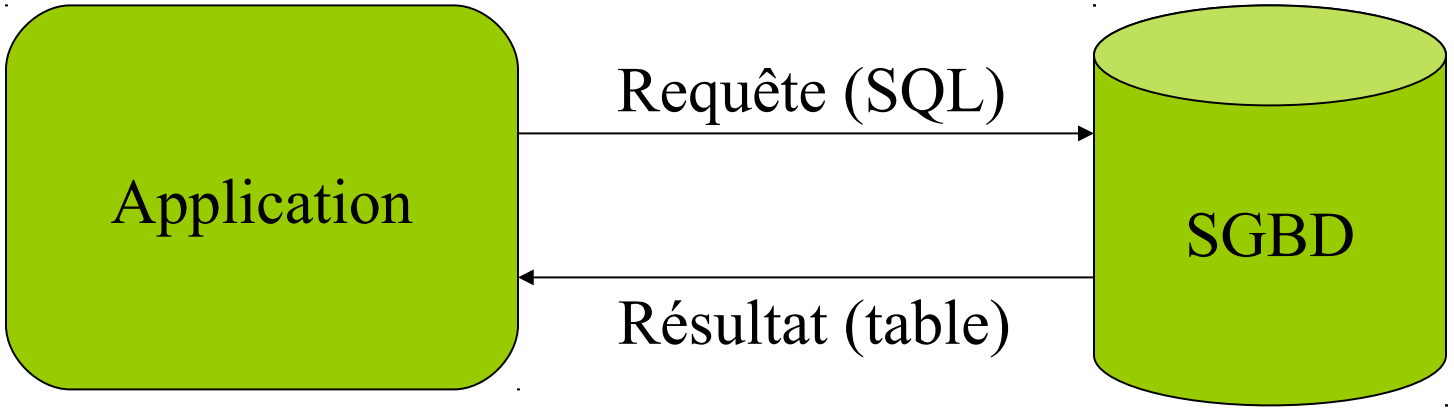
2016-2017

Plan

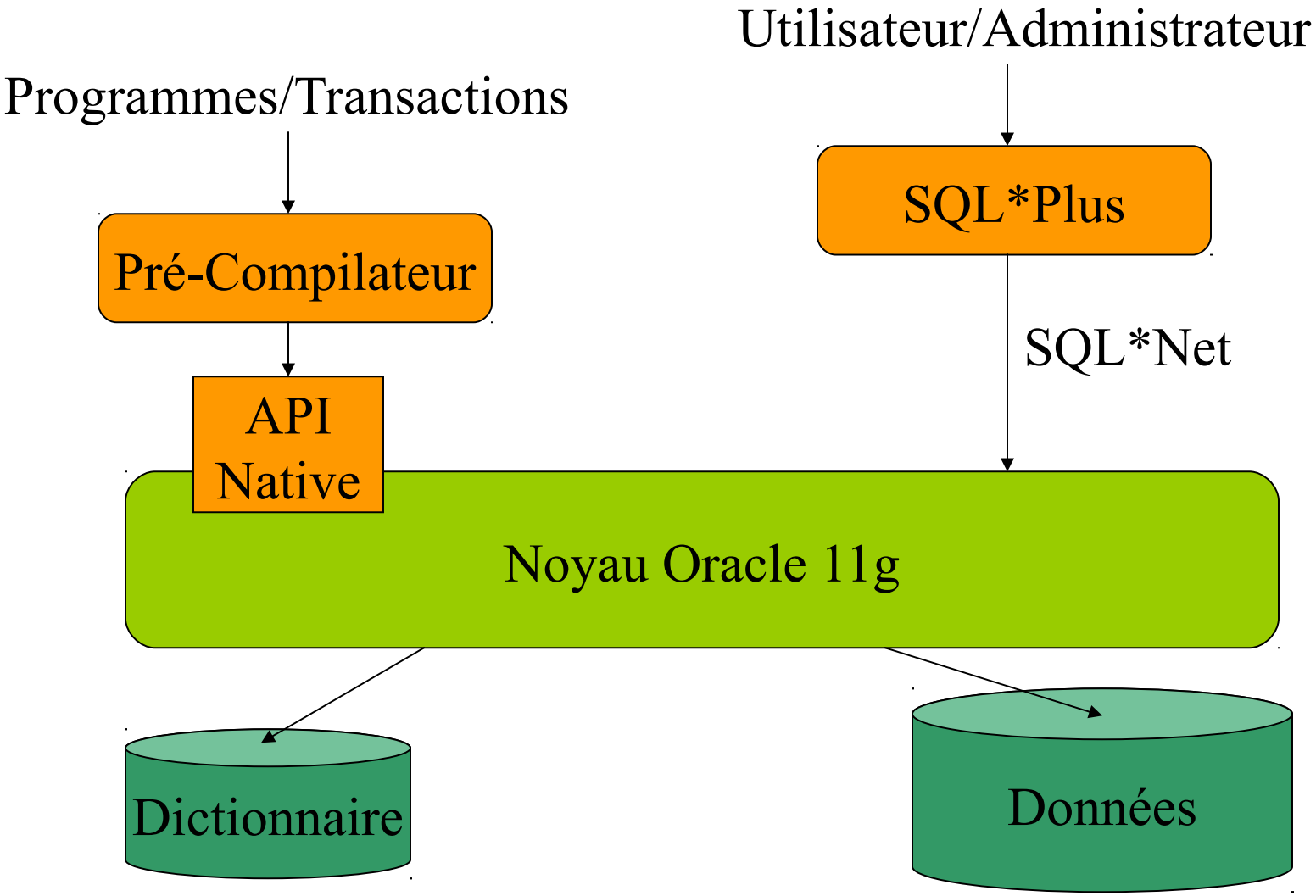
- **Architecture Oracle 12c**
- **JDBC**
  - Principe
  - Drivers
  - Connexion
  - Transactions
  - Requêtes
- **Projet BDCO**
  - Enseignants
  - Objectifs
  - Outils
  - Organisation

Architecture Client-Serveur

- Le SGBD est ici vu comme un Serveur de Requêtes
- Communication directe entre l'application et le SGBD



Oracle 12c


Utilisateur/Administrateur
Programmes/Transactions
SQL*Plus
Pré-Compilateur
SQL*Net
API
Native
Noyau Oracle 11g
Dictionnaire
Données

Plan

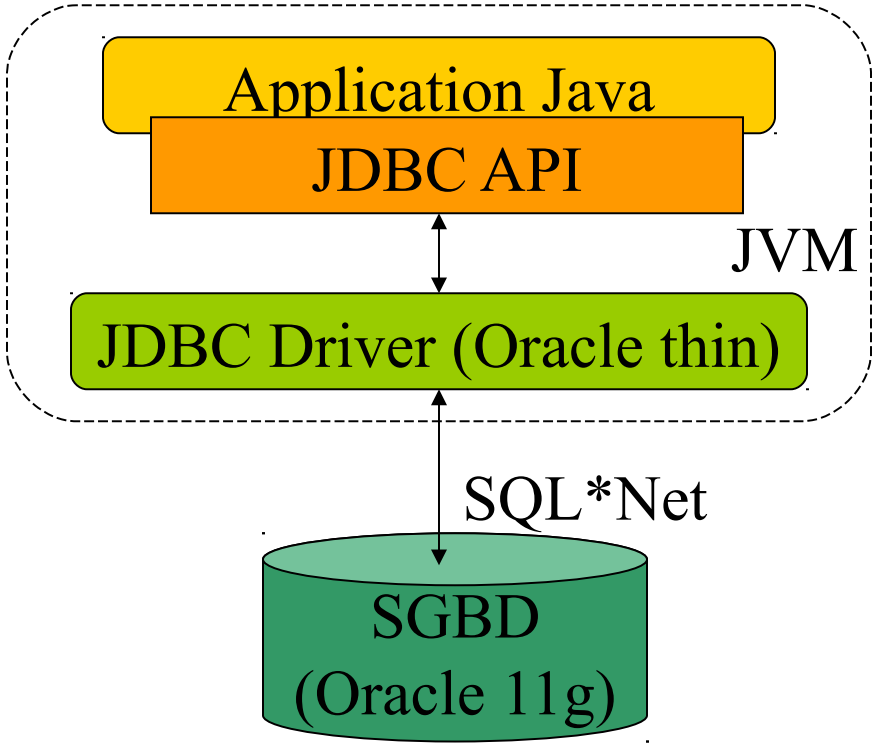
- Architecture Oracle 12c
- JDBC
  - Principe
  - Drivers
  - Connexion
  - Transactions
  - Requêtes
- Projet BDCO
  - Enseignants
  - Objectifs
  - Outils
  - Organisation

JDBC : Description

- **Ensemble de classes et d'interfaces permettant à des applications Java d'utiliser les services proposés par un ou plusieurs SGBD.**
- **Une API pur Java (package *java.sql*)**
- **Supporte SQL2**
  - Indépendamment d'un SGBD particulier

Que peut-on faire avec JDBC ?

- Se connecter à un ou plusieurs SGBD
  - Driver et modèle de connexion
- Envoyer des requêtes SQL
- Récupérer une structure contenant le résultat.



Manipuler une BD avec JDBC

- Initialiser la JVM (Driver JDBC)
- Ouvrir une connexion
- Créer des requêtes SQL
- Exécuter les requêtes
  - Sélection
  - Modification
  - Suppression
- Fermer la connexion

Drivers JDBC

- **Indispensable pour pouvoir interagir avec un SGBD**
- **Différents drivers, même pour un seul SGBD**
- **Charger plusieurs drivers permet de se connecter à plusieurs SGBD**
- **Chargement d'un driver :**
  - Création d'une instance du driver
  - Enregistrement auprès du *DriverManager*
- **Connexion à un SGBD**
  - *DriverManager* utilise le premier driver compatible

Exemple (driver Oracle 11g)

```
import java.sql.*;
…
try {
   DriverManager.registerDriver(
   new oracle.jdbc.driver.OracleDriver());
   …
}
…
```

Connexion à un SGBD

- **Une connexion est représentée par une instance de la classe *Connection*.**
- **Pour se connecter, il faut :**
  - Une URL : « jdbc:<sous-protocole>:<base> »
  - Un login
  - Un mot de passe
- **Une application peut avoir plusieurs connexions sur la même ou sur différentes bases de données.**

Exemple de connexion

```
import java.sql.*;
…
static final String url =
  « jdbc:oracle:thin:@ensioracle1.imag.fr:1521:ensioracle1 »;
```

Sous-protocole

Base

```
…
try {
  …
  Connection con = DriverManager.getConnection(url, « login »,
« passwd »);
  …
}
…
```

Connexions et Transactions…

- **A l'ouverture d'une connexion, une transaction démarre**
- **Par défaut, JDBC est en *autocommit* : commit après chaque requête.**
  - Peut (doit !) être changé par la méthode *void setAutoCommit*(*boolean*) de la classe *Connection* (**FORTEMENT** recommandé)
- **Terminaison de transactions**
  - Méthodes *void commit*() et *void rollback*() de la classe *Connection*.
  - Toute terminaison entraîne le démarrage d'une nouvelle transaction.

…Connexions et Transactions

- **Points de sauvegarde**
  - Création
    - Méthode *Savepoint setSavepoint(String name)* de la classe *Connection*
  - Abandon partiel
    - Méthode *void rollback(Savepoint savepoint)* de la classe *Connection*

Requêtes SQL simples

- **L'interface *Statement* permet d'envoyer des requêtes au SGBD**
- **Une requête SQL simple est représentée par une instance d'objet implantant l'interface *Statement,* créé à partir d'une instance de *Connection*.**
- **L'exécution d'une requête est typée :**
  - Sélection
  - Mise à jour

Sélection de données

```
import java.sql.*;
…
try {
   …
   Statement stmt = con.createStatement();
   ResultSet rs = stmt.executeQuery(« select * from emp; »);
   …
```

- **Résultat de requête : instance de la classe *ResultSet***
  - Permet de récupérer les différents attributs (méthodes *getInt*(*i*), *getFloat*(*i*), *getString*(*i*), …, où *i* est le numéro ou le nom d'un attribut)
  - Permet la navigation dans le résultat (méthodes *next*(), *previous*(), etc).

Mise à jour de données

```
import java.sql.*;
…
try {
   …
   Statement stmt = con.createStatement();
   int nb = stmt.executeUpdate(« insert into… »);
   …
```

- La méthode *executeUpdate*() permet l'exécution de requêtes de mise à jour (*insert*, *update*, *delete*, *create*, *alter* et *drop*)
- La valeur de retour contient le nombre de tuples mis à jour.

Requêtes paramétrées

- Intéressant si la même requête est exécutée plusieurs fois, mais avec des valeurs de paramètres différents.

```
import java.sql.*;
…
try {
  …
  PreparedStatement pstmt = con.prepareStatement(
    « update emp set sal = ? where ename = ? ; »);
  …
  pstmt.setFloat(1, 10000.0);
  pstmt.setString(2, « Dupont »);
  int nb = pstmt.executeUpdate();
  …
```

Fermeture des objets

- **Le ramasse-miettes de la JVM ferme automatiquement les objets *Statement* et *PreparedStatement*.**
- **Néanmoins, il vaut mieux les fermer soi-même (méthode *close()*)**
  - Libération des ressources du SGBD
  - Libération rapide de mémoire

Méta-données

- **On peut récupérer les méta-données pour:**
  - Le résultat d'une requête (*ResultsetMetaData* obtenu à partir d'un *ResultSet*)
  - La base de données entière (*DatabaseMetadata* obtenu à partir d'une *Connection*)
- **Dans tous les cas, l'objet obtenu contient :**
  - Le nom des attributs
  - Le type des attributs
  - Etc…

Plan

- Architecture Oracle 12c
- JDBC
  - Principe
  - Drivers
  - Connexion
  - Transactions
  - Requêtes
- **Projet BDCO**
  - Enseignants
  - Objectifs
  - Outils
  - Organisation

Enseignants

- **Responsable : Maxence Ahlouche**
- **Email : *maxence.ahlouche@gmail.com***

- **Rôles des encadrants**
  - Client
    - Connaît bien l'application
    - Ne connaît rien en bases de données ni en conception Objet
  - Expert en bases de données ou en conception objet
    - Ne connaît rien de l'application
  - Jamais les deux à la fois !
    - Posez bien vos questions

Objectifs du projet…

- **Mettre en application vos connaissances en Bases de Données**
  - **Analyse**
    - Dépendances fonctionnelles (DF)
    - Contraintes de valeur, de multiplicité et autres
  - **Conception**
    - Schéma Entités/Associations (notation UML) complet
    - DF et contraintes non incluses dans le schéma
  - **Analyse des accès à la base de données**
    - Ensemble complet de requêtes (y compris pour la vérification des contraintes)
    - Transactions pour garantir la cohérence globale (définie par le schéma Entités/Associations et les DF et contraintes associées)
  - **Implantation dans Oracle 12g**
    - Schéma relationnel complet
    - Contraintes SQL
    - DF et contraintes non vérifiables en relationnel
    - Requêtes et transactions à tester via SQL*Plus

…Objectifs du projet

- **Concevoir une application utilisant une BD et des interfaces utilisateur**
  - **Analyse**
    - Analyse des besoins fonctionnels,
    - Modélisation objets du domaine
  - **Architecture de l'application intégrant :**
    - Des interfaces utilisateur
    - Les liens avec la base de données
  - **Conception Objet**
    - Diagramme de classes logicielles
    - Mise en place de patrons de conception
  - **Implémentation en Java**
    - Utilisant l'API JDBC
    - Implantant les requêtes et transactions

Outils à mettre en place / A (re?)découvrir

- **JDBC -> exemples sur le *kiosk***
- **Java -> cours et TD du cours POO**

- **Interfaces graphiques**
  - GUI-Builder (Netbeans, plugin Eclipse) – *Fortement conseillé !!*

- **Partage des sources / gestionnaire de versions**

- **La découverte et la mise en place de ces outils**
  - sont **indispensables** au projet
  - **ne doivent pas** être faites pendant les séances encadrées

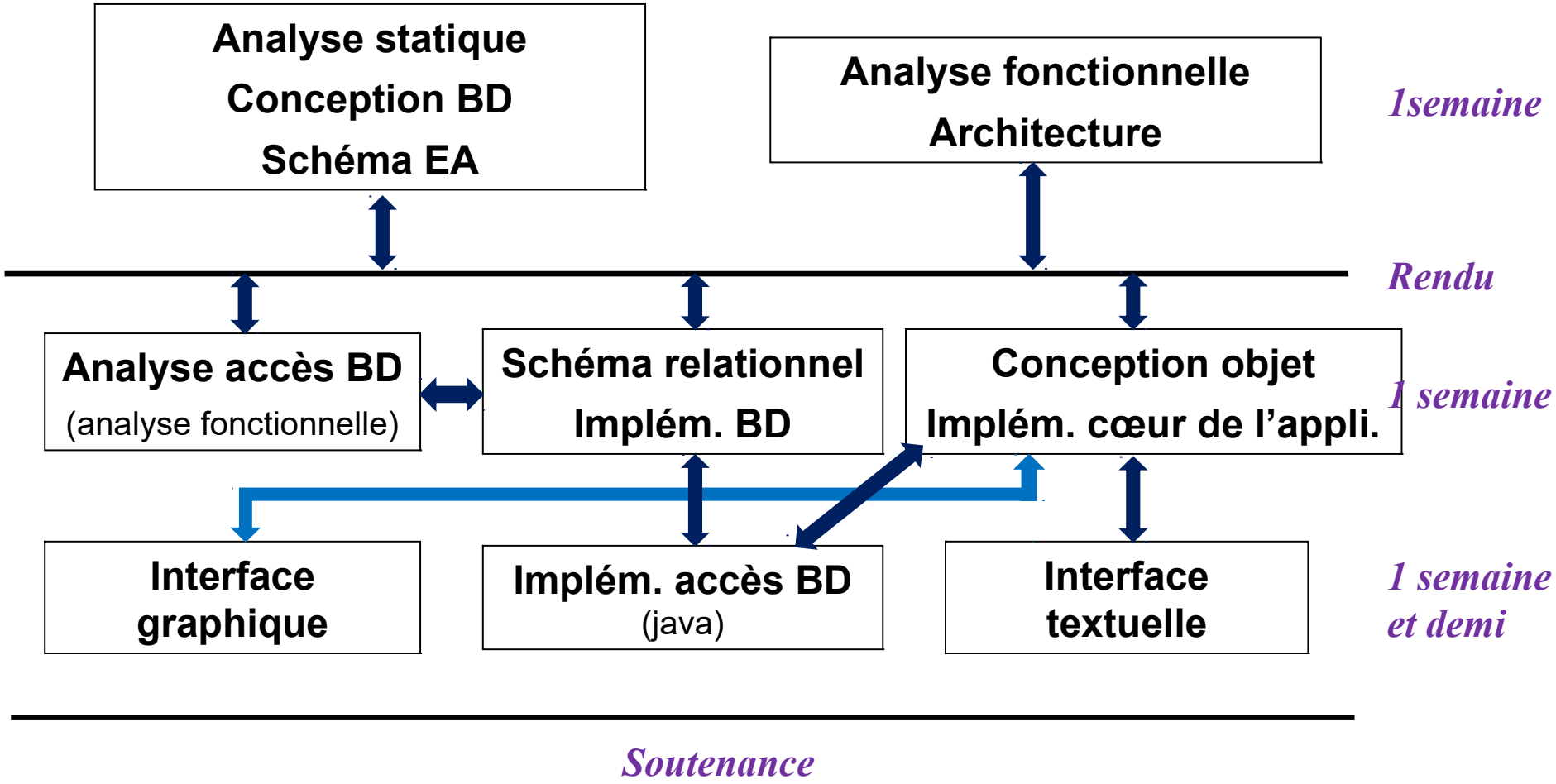
Déroulement

- Projet à réaliser en équipe de 5 ou 6

- 9 heures encadrées / personne
- + 6 heures non encadrées / personne

- Attention au calendrier : cela va très vite !!

Travail à faire et organisation en tâches



Rendus

- **Rendu intermédiaire : le 24 mai 2017**
  - schéma entité association de votre base de données, *(pdf, A4)*
  - schéma ou les schémas de l'architecture de votre logiciel, *(pdf, A4)*

- **Journal d'avancement** (au jour le jour)

- **Rendus pour la soutenance**
  - Schéma entités/associations + schéma relationnel
  - Diagramme de classes
  - Rapport présentant les choix de conception + déroulement du projet + bilan
  - Code source (java, sql, éventuels scripts)

- **Soutenance le *2 juin 2017 de 14h à 17h***
  - 30 minutes par équipe :
    - **Présentation + démonstration**
    - Objectif = vendre le produit **et** l'équipe
  - + 10 minutes de questions et retour à l'équipe

Projet BDCO

- **Pensez à consulter régulièrement les pages projet de Chamillo**
  - Documentation
  - Questions fréquentes
  - Dates
  - …
- **N'hésitez pas à poser des questions à vos encadrants**

Bon courage à tous !