Cursus JAVA

M2I Formations 2022

Olivier Blaivie

MODULE SQL

Langage SQL ou PL/SQL

Olivier Blaivie

1.

Base de données : Les concepts

Base de données

- **Ensemble structuré d'informations** (*d'une entreprise ou organisation*), **mémorisé** sur une machine (*serveur*).

- **Données stockées** et **organisées** sous forme de fichiers ou ensemble de fichiers.

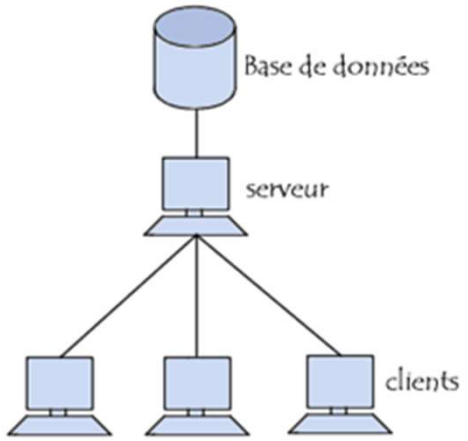
- Une BD sert à **créer**, **enregistrer**, **récupérer** et **manipuler** des **données communes**.

SGBD

- **Système de Gestion de Base de Données** (ou DBMS: Data Base Management System)

- **Ensemble cohérant** de **services** (*logiciels*) permettant aux utilisateurs **d'accéder**, **mettre à jour** ou **administrer** une DB

- Fonctionne sur le modèle **client/serveur** (requêtes/traitements)



SGBD

Pourquoi utiliser un SGBD ? Quels objectifs ?

- Indépendance physique
- Indépendance logique
- Accès / partage des données
- Administration centralisée
- Non redondance des données
- Cohérence des données
- Sécurité des données
- Résistance aux pannes

Chronologie

Années 60:

- Apparition des premiers SGBD

Années 70:

- Ted Codd propose le modèle relationnel => 2ème generation Codd définit l'algèbre relationnelle (prémices du SQL)

Années 80:

- SGBD relationnel commercialisé (Oracle, SysBase, DB2…)

Années 90:

- SGBD relationnel dominent le marché
- Début des SGBD orientés objets

2.

Base de données : Modélisation

Modélisation

- **Pourquoi modéliser ?**
  - Avoir une représentation graphique de la structure
  - Connaître les propriétés attendues d'une donnée
  - Connaître les relations entre les données

- **Comment modéliser ?**
  - Effectuer un design conceptuel
  - Insérer des cardinalités
  - Effectuer un modèle logique

Modélisation : Extraire l'information

- Lorsque vous allez recevoir un cahier des charges, il faudra tout d'abord réussir à en extraire les informations qui nécessitent un stockage persistant.
- Ces information seront regroupées dans un documents appelé le "**dictionnaire de données**".
- Exemple :

Nom donnée	Type donnée	Référence	Commentaire
nom_client	TEXT	Client	Contient le nom des clients

- **Objectif** : Avoir une structure des **tables** pour notre futur base de données.

Modélisation : Les types de données

Dans une base de données il existe plusieurs **types** de données utilisables en fonction de la nature de l'information que l'on souhaite stocker :

Nombre entier :

Type	Nombre d'octets	Minimum	Maximum
TINYINT	1	-128	127
SMALLINT	2	-32768	32767
MEDIUMINT	3	-8388608	8388607
INT	4	-2147483648	2147483647
BIGINT	8	-9223372036854775808	9223372036854775807

Chaîne de caractères :

Type	Longueur maximale	Mémoire occupée
TINYTEXT	2^8 octets	Longueur de la chaîne + 1 octet
TEXT	2^16 octets	Longueur de la chaîne + 2 octets
MEDIUMTEXT	2^24 octets	Longueur de la chaîne + 3 octets
LONGTEXT	2^32 octets	Longueur de la chaîne + 4 octets

Modélisation : la clef primaire

- Le stockage d'informations en base doit être **unique** (une donnée = une entrée)
- On parle alors de **"clef primaire"** (ou primary Key)
- Généralement au format numérique et nommée "**id**".

id	prenom	nom
1	Chandler	Bing
2	Phoebe	Buffay
3	Monica	Geller
4	Ross	Geller
5	Chandler	Bing

Modélisation : les relations

- Une relation veut dire que des données sont liées.
- Il existe 3 types de relations :
  - un à un (one-to-one)
  - un à plusieurs (one-to-many) ou plusieurs à un (many-to-one)
  - plusieurs à plusieurs (many-to-many)

Chacun de ces types engendre une conséquence différente sur le modèle de données.

Modélisation : les relations

Afin d'éviter les doublons de données il est possible de mettre en place des **relations** entre nos différentes **tables/entités**.

Par exemple :

id	nom	modele	annee_sortie
1	Jean Manchzeck	Kawasaki 750	2015
2	Edouard Bracame	Kawasaki 750	2015

Pourrait devenir :

id	nom
1	Jean Manchzeck
2	Edouard Bracame

0,1 0,1

id	modele	annee_sortie
1	Kawasaki 750	2015

Modélisation : les relations

One-to-One : Une relation one-to-one implique la création d'une **clé étrangère** dans l'une des deux tables. Cette clé représente la référence de la seconde table. Si on reprend l'exemple précédent :

id	nom
1	Jean Manchzeck
2	Edouard Bracame

0,1 — 0,1

id	modele	annee_sortie	conducteur
1	Kawasaki 750	2015	1
2	Kawasaki 750	2015	2

Modélisation : les relations

One-to-Many : Une relation one-to-many implique également la création d'une **clé étrangère** dans l'une des deux tables. Par contre dans ce cas nous n'avons pas le choix de la table qui portera la référence. Si on reprend l'exemple précédent :

id	nom
1	Jean Manchzeck
2	Edouard Bracame

0,1 — n

id	modele	annee_sortie	conducteur
1	Kawasaki 750	2015	1
2	Kawasaki 750	2015	2

Modélisation : les relations

Many-to-Many : Une relation many-to-many engendrera la création d'une table de correspondance. Si on reprend l'exemple précédent :

id	nom
1	Jean Manchzeck
2	Edouard Bracame

conducteur	moto
1	1
2	2

id	modele	annee_sortie
1	Kawasaki 750	2015
2	Kawasaki 750	2015

Les règles à respecter

- **Normalisation des tables**
  - Ne contient pas d'espace, d'accents ni de caractères spéciaux
  - Tout doit être écrit en lowercase (minuscule)
  - Les espaces sont remplacés par des underscores : "_"

- **Les relations**
  - Une relation permet de définir le type de lien entre 2 entités
  - Les relations "0,n"
  - Les relations "1,n"
  - Les relations "n, n"

- **Un identifiant**
  - Un champs d'identification unique est obligatoire, la clef primaire (souvent nommé "id").

La modélisation UML

- **Comment utiliser UML**
  - Le langage UML permet de schématiser les différentes entités
  - UML à ses propres règles et codes pour représenter ces entités
  - On parle de diagramme de Classe

- **Quels outils ?**
  - Il existe plusieurs outils pour modéliser (représenter) des entités
  - Draw.io est un outil gratuit en ligne et adapté à l'UML

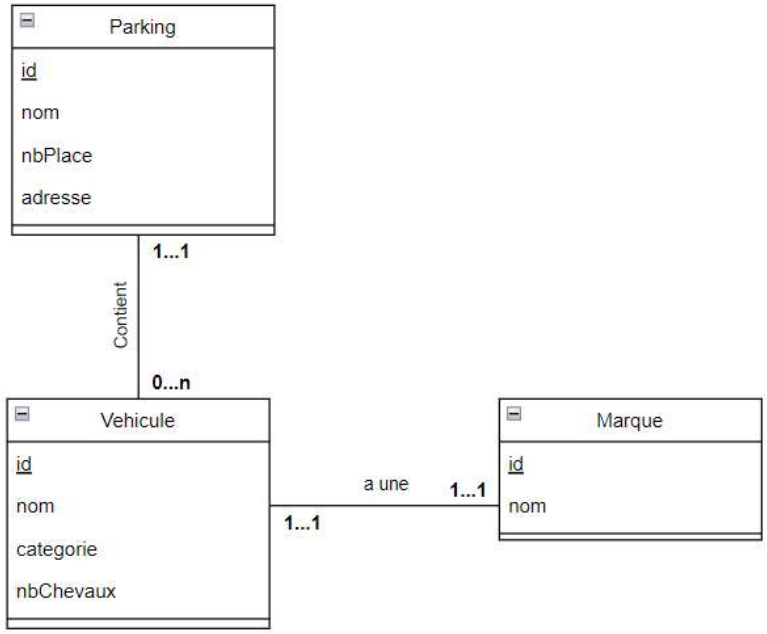
- **Les étapes ?**
  - L'objectif est de passer d'un texte (une spec, un cahier des charges...) à une représentation graphique

Exemple: du besoin à la modélisation

Besoin client :

Je souhaite avoir une modélisation représentant la gestion d'un parking de voiture.

Un parking à un nom, un nombre de place et une adresse. Le parking contient plusieurs véhicules, de différentes tailles.



3.

Base de données : BDD relationnelles

Introduction

- **Qu'est-ce qu'une BDD relationnelle ?**
  - Concept basé sur le modèle de relation des données (Tables)
  - Organisation en colonnes et lignes
  - Attributs, données, tuples ...

- **SGBD couramment utilisés ?**
  - MySQL, PostgreSQL, Oracle, SQLite, Microsoft SQL Server

Propriété 1	Propriété 2	Propriété N...
Objet1, donnée1	Objet1, donnée2	Objet1, donnée N
Objet2, donnée1	Objet2, donnée2	...

Avantages

- **Avantages du modèle relationnel :**
  - Simplicité de représentation
  - Indépendance physique
  - Indépendances logique (Vues)
  - Maintient de l’intégrité (Contraintes)

Type de données et contraintes

- **Types de données divisés en 3 catégories :**
  - Alphanumériques : VARCHAR, TEXT, BLOB ...
  - Numériques : INTEGER, FLOAT, DECIMAL, BOOLEAN ...
  - Temporels : DATE, DATETIME, TIME ...

- **Contraintes de données :**
  - Clés primaires ➔ Unicité (SERIAL)
  - Tailles de champs ➔ Min, max
  - Contrôles de champs ➔ Obligatoire, Défaut, facultatif
  - Clé étrangères ➔ Références aux autres tables
  - Et bien d'autres ...

Exemple

- De la modélisation UML à la création de table:

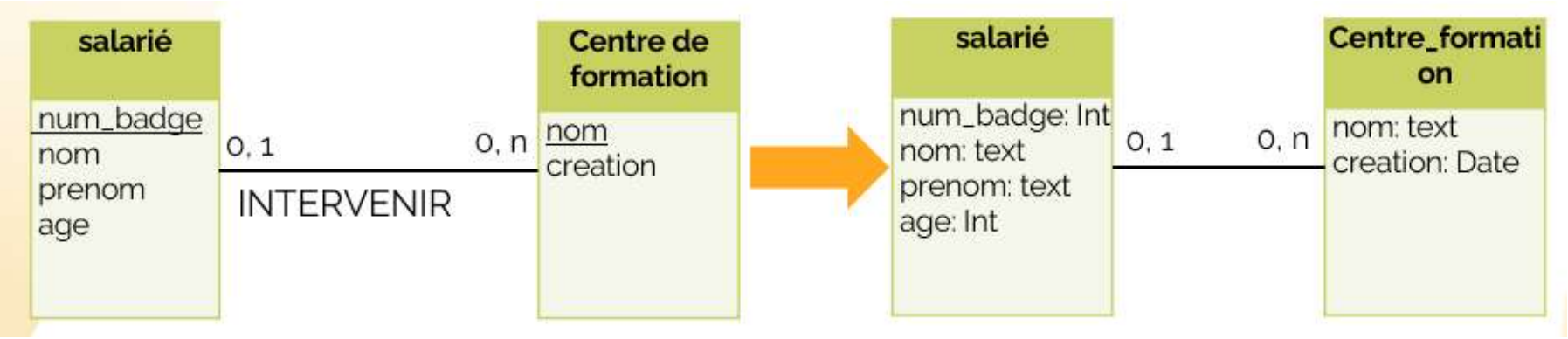
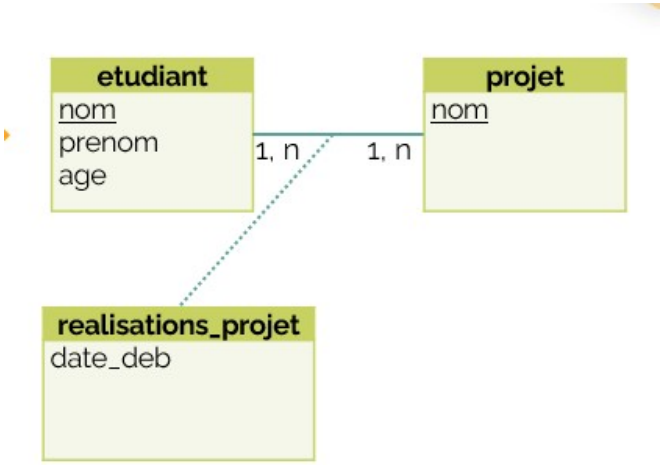


Table d’association

- **Qu’est-ce qu’une table d’association ?**
  - On utilise une table d’association pour ajouter des informations entre 2 tables
  - La clé primaire d’une table d’association est le couple des clés primaires des 2 tables
  - Règle: La relation n-n crée une table d’association

- **Exemple :**



4.

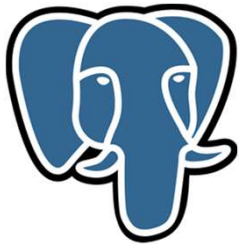
Base de données : Le SQL

Introduction

- SQL (Structured Query Language): Langage permettant d'interagir avec une base de données relationnelle.
- Langage introduit par IBM et commercialisé uniquement par Oracle dans un premier temps.
- Héritage du langage SEQUEL en 1977 reposant sur la théorie relationnelle proposée par Ted Codd.
- Terminologie définit par SQL:
  - *Table*
  - *Colonne*
  - *Ligne*
  - *Primary Key*
  - *Foreign Key*
  - ...

Type de SGBD – Les principaux

Installation de PostGres



- Installation de PostgreSQL 14.4 (+ pgAdmin)
  https://www.enterprisedb.com/downloads/postgres-postgresql-downloads

  ➔ **Notez bien le mot de passe admin !**

- Docs SQL :

https://sql.sh/

https://docs.postgresql.fr/

Connexion à PostGres

- Avant de pouvoir manipuler les tables et les données, la première étape est de se **connecter** au SGBD
- Toute instruction se termine par un **point virgule** pour être interprétée et exécutée !
- La connexion au SGBD se fait depuis une invite de commande avec la commande **psql** et l'utilisateur crée par défaut **postgres**
- Connexion au SGBD PostGres :

  psql -U *userName*

  psql -U *userName* -d *baseName*
- Exemple :

```
C:\Users\(      >psql -U postgres
Mot de passe pour l'utilisateur postgres :
psql (13.1)
Attention : l'encodage console (850) diffère de l'encodage Windows (1252).
            Les caractères 8 bits peuvent ne pas fonctionner correctement.
            Voir la section « Notes aux utilisateurs de Windows » de la page
            référence de psql pour les détails.
Saisissez « help » pour l'aide.

postgres=#
```

Commandes de gestion

- L'interface Postgres permet de **manipuler**, **lister** ou faire des **opérations** sur le SGBD.

- Quelques commandes de gestion de base :

    - Lister les bases existantes ➔ \l
    - Se connecter à une base ➔ \c *database username*
    - Lister les tables ➔ \dt
    - Quitter le SGDB ➔ \q

SQL: Les requêtes (CRUD)

Il existe plusieurs instructions possibles sur les **données** d'une base :

- **INSERT** pour ajouter des lignes à une table
- **UPDATE** pour modifier des lignes d'une table
- **DELETE** pour supprimer des lignes d'une table
- **SELECT** pour extraire des données à partir de tables existantes

Commandes de création

- Créer une base de données :
  CREATE DATABASE *nom_base;*

- Créer un utilisateur :
  CREATE USER *nom_user;*
  CREATE USER *nom_user* WITH ENCRYPTED PASSWORD *'password'*;

- Accorder des droits :
  GRANT *privilèges* ON DATABASE *nom_base* TO *user;*
  Exemple1 : GRANT all privileges ON DATABASE mabase TO user1;
  Exemple2 : GRANT SELECT, UPDATE ON table1 TO user2;

- Créer une table :
  CREATE TABLE *nom_table (nom1 type, nom2 type, …);*
  Exemple : CREATE TABLE table_1 (id INT PRIMARY KEY, nom VARCHAR);

Insertions et modifications

- Modifier une table :
  ALTER TABLE *nom_table* ADD nom type; (DROP)
  ALTER TABLE *nom_table* ALTER COLUMN nom TYPE type;

- Insérer des données dans une table :
  INSERT INTO *nom_table* VALUES (value1, value2, valueN,…);
  INSERT INTO *nom_table* (col1, col3) VALUES (value1, value2);

- Supprimer des données dans une table :
  DELETE FROM *nom_table* WHERE *conditions;*
  Exemple : DELETE FROM table1 WHERE prenom = ‘Jean’;

- Modifier des données dans une table :
  UPDATE *nom_table* SET *col1=new_val, col2=new_val2, …* WHERE *conditions;*
  Exemple : UPDATE table1 SET prenom=‘Jean-Michel’ WHERE id = 1;

Types et contraintes

- Déclaration d'un type de données :
  VARCHAR(*size*) , INTEGER, DECIMAL(*size*, *decimalSize*), TEXT, DATE, BOOLEAN

- Contraintes de données :
  Clé primaire/Auto incrément : **PRIMARY KEY / SERIAL**
  Valeur par défaut: **DEFAULT**
  Non null: **NOT NULL**
  Contraintes: **CONSTRAINT**
  Conditions: **CHECK** *conditions*
  Clé étrangère: **FOREIGN KEY** *fk_name* **REFERENCES** *table(champs);*

- Exemple :
  ```
  CREATE TABLE table1 (id SERIAL, prix INTEGER(5), CONSTRAINT prixMin CHECK (prix > 0);
  ```

Requêtes (1/2)

- Requêter une table :
  SELECT *nom_champ1, nom_ch2…* FROM *nom_table*
  SELECT * FROM *nom_table*

- Ajouter une condition à une requête :
  SELECT *nom_champ1, nom_ch2…* FROM *nom_table* WHERE *nom_champ* = x

- Ajouter des restrictions (SELECT… FROM …):
  LIMIT 3 ➔ Limiter résultat
  ORDER BY Nom ➔ Ordonner pour un champs donné
  ORDER BY N DESC ➔ Ordonner pour un champs donné inversé

Requêtes (2/2)

- Modifier une table :

  ALTER TABLE *nom_table* ADD nom type; (DROP)
  ALTER TABLE *nom_table* ALTER COLUMN nom TYPE type;

- Opérateurs classiques : SELECT *FROM Table…

  WHERE Fonction = "Représentant"
  WHERE Fonction <> "Représentant"
  WHERE Fonction = "Représentant"
  AND Titre = "M."
  AND nombres < 8;
  WHERE UPPER(Ville) = "SEATTLE";
  WHERE Fonction BETWEEN 3 AND 8;
  WHERE Titre IN ('Mlle', 'Mme');
  WHERE Nom LIKE 'a%';
  WHERE Nom LIKE '%a%';

- Sélection pour un champs renommé :

  SELECT ville AS liste_des_villes FROM Ville;