Stage Liesse Ensimag juin 2014

Introduction aux Systèmes de Gestion de Bases de Données Relationnels

Sylvain Bouveret & Noha Ibrahim

prénom.nom@grenoble-inp.fr

26 juin 2014

1 Première partie

Introduction aux SGBD

Pourquoi des SGBD ?

Système de Gestion de Bases de Données (SGBD) : système permettant de stocker et d'accéder à de l'information.

Un simple ensemble de fichiers en fait ?

Élèves Enseignants Cours Notes Filières

Pas tout-à-fait...

Problèmes de stockage de l'information

Accès aux données :

1. Accès complexe à une information disparate et potentiellement stockée à plusieurs endroits.
   → **besoin d'un mécanisme d'interrogation complexe**
2. Accès inefficace si quantité de données conséquente
   → **besoin d'un accès optimisé aux données**
3. Données potentiellement sensibles (en terme de sécurité)
   → **besoin de mécanismes de contrôle d'accès**

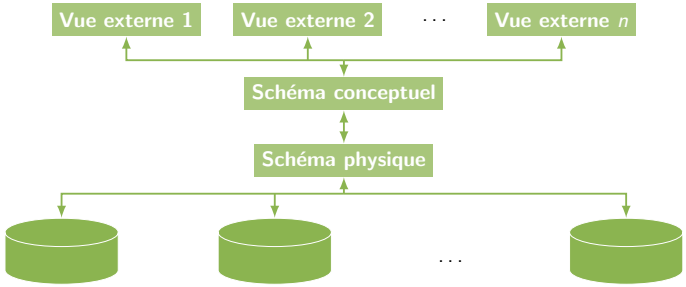
Problèmes de stockage de l'information

Cohérence de l'information :

4. Pas de vérification de la cohérence de l'information entrée dans la base
   → **Expression et de vérification de contraintes nécessaire**
5. Information potentiellement redondante ⇒ risque d'incohérence
   → **redondance subie ✗**
   → **redondance contrôlée (sauvegarde) ✓**
6. Accès concurrents de plusieurs utilisateurs ⇒ problèmes de cohérence
   → **besoin de mécanismes de gestion des accès concurrents**
7. Panne pendant une mise-à-jour massive ?
   → **besoin de garantir l'atomicité des mises-à-jour**

Problèmes de stockage de l'information

8. Stockage des données (fichiers, format) : influe sur leur interrogation.
   → **besoin d'assurer l'indépendance physique des données**
9. Différents types d'utilisateurs ⇒ différents points de vue sur les données.
   → **besoin d'assurer l'indépendance logique des données**



SGBD, modèle de données

Système de Gestion de Bases de Données

Un SGBD est un système de stockage de données, qui permet de résoudre tous ces problèmes.

Tout SGBD s'appuie sur un **modèle de données**, permettant de représenter en mémoire des données du monde réel.

Modèle de données

Modèle = représentation formelle abstraite (simplifiée) de la réalité.

Modèle de données = représentation formelle abstraite des données du monde réel à manipuler.

Permet de décrire :

1. **la structure des données** (types d'objets, relations entre ces types, etc.)
   Exemple : « un étudiant est un objet qui possède un numéro INSEE, un nom, un prénom, une adresse... »
2. **les occurrences des données**
   Exemple : « Dark Vador est un étudiant possédant le numéro INSEE 1770901142555, le nom 'Vador', le prénom 'Dark', l'adresse `vador@imag.fr` »

Le modèle relationnel

Plusieurs modèles de données existent (hiérarchique, réseau, etc.). Aujourd'hui, nous nous focaliserons uniquement sur

le modèle relationnel

- développé par Codd en 1970
- très simple conceptuellement (tableaux 2D)
- très puissant grâce à l'**Algèbre Relationnelle**
- modèle sur lequel s'appuie la norme **SQL2**, standard encore à l'heure actuelle

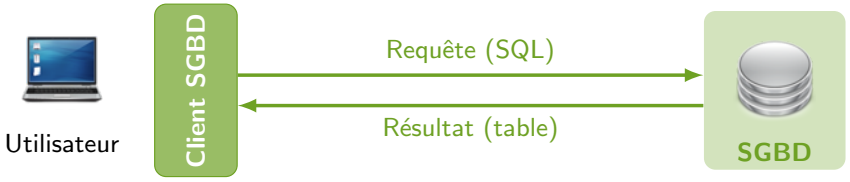
Introduction aux SGBD

Qu'est-ce qu'un Système de Gestion de Bases de Données ?

À propos de modèles de données

Les SGBD en pratique...

Les SGBD en pratique



Concrètement, pour utiliser une base de données :

- On lance un client
- On fournit les identifiants de connexion : nom de la base, localisation (serveur), nom d'utilisateur, mot de passe
- Une fois connecté, on envoie des requêtes et on récupère le résultat

Client SGBD console

Exemple, avec un client *PostgreSQL* en ligne de commande :

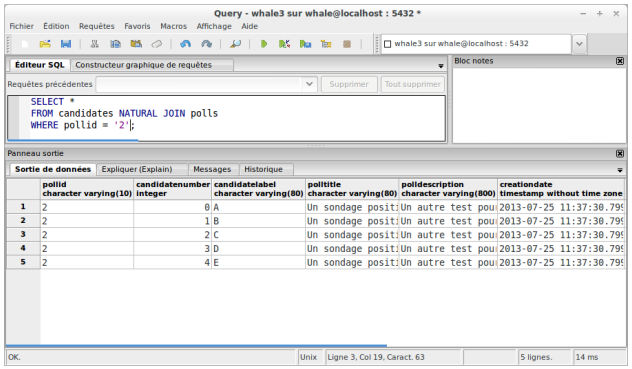
```
[sylvain@msnordlys]~ $ psql -U sylvain -d mabase -h monserveur

Password for user sylvain: ********
psql (9.1.8)
Type "help" for help.

mabase=# SELECT name, place, ST_XMin(way), ST_YMin(way)
mabase-# FROM planet_osm_point WHERE name='Grenoble' AND place='city';
   name   | place |     st_xmin      |     st_ymin
----------+-------+------------------+-----------------
 Grenoble | city  | 638583.190179611 | 5650917.09875511
(1 row)
```

Client SGBD graphique

Je n'aime pas la console... Suis-je condamné à l'utiliser quand même ?
Non... La plupart des SGBD ont des clients graphiques permettant d'administrer et d'interroger des bases
Exemple : pgAdmin III, pour le SGBD Postgres.



Un SGBD chez moi...

Concrètement, comment fait-on pour installer un SGBD ?
La plupart des SGBD (sauf SQLite) fonctionnent sur le mode classique client-serveur. C'est le cas par exemple pour :

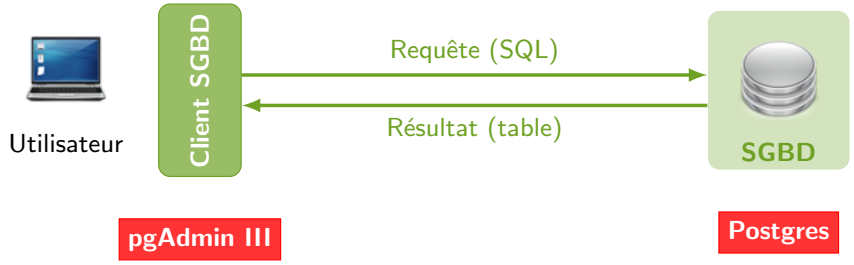
- **PostgreSQL** (libre) ;
- **MySQL** (libre ou propriétaire) → fourni par exemple dans la suite WAMP (*Windows*, *Apache*, *MySQL*, *PHP*) ou LAMP (*Linux*...) fournissant un serveur Web PHP + BD clef en main.
- **Oracle** (propriétaire)
- ...

Pour ces SGBD, il faut :

- installer la partie « serveur » du SGBD
- démarrer le serveur
- utiliser le client fourni, ou en trouver un qui nous convient.

Postgres + pgAdmin III

Aujourd'hui, nous allons utiliser le SGBD **Postgres**, avec le client graphique **pgAdmin III**.



Le programme de la journée

1. D'abord (surtout) de la théorie :
   - 1.1 **Le modèle relationnel**
   - 1.2 **L'algèbre relationnelle**
2. Ensuite (surtout) de la pratique :
   - 2.1 interrogation avec **SQL**
   - 2.2 interface **SQL - Python** : construction d'une petite application

2 Deuxième partie

Le modèle relationnel

Relation

Notion centrale du modèle relationnel : la **relation**...

R	$att_1 : \mathcal{D}_1$	...	$att_j : \mathcal{D}_j$	...	$att_n : \mathcal{D}_n$
	$t_{1,1}$	...	$t_{1,j}$	...	$t_{1,n}$
	...	...	...	...	...
	$t_{i,1}$	...	$t_{i,j}$	...	$t_{i,n}$
	...	...	...	...	...
	$t_{m,1}$	...	$t_{m,j}$	...	$t_{m,n}$

Relation : détails

R	$att_1 : \mathcal{D}_1$	...	$att_j : \mathcal{D}_j$	...	$att_n : \mathcal{D}_n$
	$t_{1,1}$	...	$t_{1,j}$	...	$t_{1,n}$
	...	...	...	...	...
	$t_{i,1}$	...	$t_{i,j}$	...	$t_{i,n}$
	...	...	...	...	...
	$t_{m,1}$	...	$t_{m,j}$	...	$t_{m,n}$

Relation : détails

R	$att_1 : \mathcal{D}_1$	...	$att_j : \mathcal{D}_j$	...	$att_n : \mathcal{D}_n$
	$t_{1,1}$	...	$t_{1,j}$	...	$t_{1,n}$
	...	...	...	...	...
	$t_{i,1}$	...	$t_{i,j}$	...	$t_{i,n}$
	...	...	...	...	...
	$t_{m,1}$	...	$t_{m,j}$	...	$t_{m,n}$

→ **schéma de relation**

- Schéma = **intension** de la relation (définition abstraite)
- **R** = **nom** de la relation (et du schéma)
- n = **arité** de la relation (relation n-aire)

ECOLE NATIONALE SUPERIEURE D'INFORMATIQUE ET DE MATHEMATIQUES APPLIQUEES 19 / 95

Relation : détails

R	$\text{att}_1 : \mathcal{D}_1$	...	$\text{att}_j : \mathcal{D}_j$	...	$\text{att}_n : \mathcal{D}_n$
	$t_{1,1}$	...	$t_{1,j}$	...	$t_{1,n}$
	...	...	...	...	...
	$t_{i,1}$	...	$t_{i,j}$	...	$t_{i,n}$
	...	...	...	...	...
	$t_{m,1}$	...	$t_{m,j}$	...	$t_{m,n}$

→ **tuple (ligne)**

- Ensemble des tuples = **extension** de la relation (**données**)
- Une relation est un **ensemble** de tuples. Conséquences :
  - lignes non ordonnées
  - pas de doublons (tuples tous différents)
- m = **cardinal** de la relation (= nombre de tuples)

Relation : détails

R	$\text{att}_1 : \mathcal{D}_1$	...	$\text{att}_j : \mathcal{D}_j$	...	$\text{att}_n : \mathcal{D}_n$
	$t_{1,1}$	...	$t_{1,j}$	...	$t_{1,n}$
	...	...	...	...	...
	$t_{i,1}$	...	$t_{i,j}$	...	$t_{i,n}$
	...	...	...	...	...
	$t_{m,1}$	...	$t_{m,j}$	...	$t_{m,n}$

↓

attribut (colonne)

- $\text{att}_j : \mathcal{D}_j$ = nom et domaine d'attribut (intension)
- $\{t_{1,j}, \ldots, t_{m,j}\}$ = projection de **R** sur l'attribut att_j (extension)
- Attributs nommés, mais ordre des colonnes sans importance

Exemple

Élèves 1	prénom	nom	e-mail	filière
	Luke	Skywalker	skywalker@imag.fr	MMIS
	Dark	Vador	vador@imag.fr	IF
	Han	Solo	solo@falcon.com	IF
	Leia	Solo	princess@falcon.com	MMIS
	Jabba	The Hut	jabba@imag.fr	TELECOM

dont le schéma est...

Élèves 1 : {prénom : String, nom : String, e-mail : String, filière :{MMIS, IF, TELECOM, SLE, ISI}}.

(Remarque : pour simplifier on peut omettre le domaine des attributs)

Le modèle relationnel

Contraintes d'intégrité

- Dans le modèle relationnel, le concepteur de la base de données peut définir des **contraintes** que doivent respecter les données
- La définition d'une contrainte se fait sur le **schéma** des tables (intension).
- Ce sont les **données** (les tuples, l'extension) qui doivent se conformer aux contraintes
- Trois types de contraintes :
  - clef primaire ;
  - référence (clef étrangère) ;
  - valeur ou domaine.

Contrainte de clef primaire

- **Définition :** un sous-ensemble $\{\text{att}_1, \ldots, \text{att}_k\}$ d'attributs d'un schéma de relation **R**.
- **Sémantique formelle :** pour tout $(t_1, t_2) \in \mathbf{R}$, si t_1 et t_2 coïncident sur $\{\text{att}_1, \ldots, \text{att}_k\}$ alors $t_1 = t_2$.

  $\rightsquigarrow$ on dit que $\{\text{att}_1, \ldots, \text{att}_k\}$ est une **clef** de **R**.
- **Sémantique informelle :** imposer un identifiant à chaque relation.

Contrainte de clef primaire : exemple

Élèves 1 : {prénom, nom, e-mail, filière}

Élèves 1	prénom	nom	e-mail	filière
→	**Dark**	**Vador**	vador@imag.fr	**IF**
	Obi-Wan	Kenobi	kenobio@imag.fr	MMIS
	Han	Solo	solo@falcon.com	IF
→	**Dark**	**Vador**	vador@blackstar.com	**IF**

$\rightsquigarrow$ viole la contrainte de clef primaire !

Contraintes de référence

Élèves 1	prénom	nom	e-mail	filière
	Dark	Vador	vador@imag.fr	IF
	Obi-Wan	Kenobi	kenobio@imag.fr	MMIS
	Han	Solo	solo@falcon.com	IF

Notes	cours	prénom	nom	note
	sport	Dark	Vador	20
→	**sport**	**Jabba**	**The Hut**	**3**
	pilotage	Han	Solo	15

Comment imposer le fait que tout élève apparaissant dans la table **Notes** *apparaisse aussi dans la table* **Élèves** *?* $\rightsquigarrow$ **Contrainte de référence**

Contraintes de référence

- **Définition :** une **correspondance** entre des attributs $\{att_1, \ldots, att_k\}$ d'un schéma **R** et des attributs $\{att'_1, \ldots, att'_k\}$ d'un autre schéma **R'**.
- **Exemple :** **Notes**(prénom, nom) référence **Élèves**(prénom, nom)
- **Sémantique formelle :**
  1. Pour tout tuple $t \in \mathbf{R}$, il existe un tuple $t' \in \mathbf{R'}$ tel que $t[att_1, \ldots, att_k] = t'[att'_1, \ldots, att'_k]$.
  2. $\{att'_1, \ldots, att'_k\}$ est une clef de **R'**
- **Sémantique informelle :** tout « objet » de **R** doit correspondre à un « objet » existant de **R'**.

Contraintes de valeur / domaine

- **Définition :** une condition booléenne sur les attributs d'une relation.
- **Exemple :** note ≥ 0 et note ≤ 20.
- **Sémantique :** tous les tuples de la relation doivent vérifier la condition booléenne.

Le modèle relationnel

Base de données relationnelle

Schéma de base de données relationnelle

- Un ensemble de **schémas de relations**
- Une **contrainte de clef primaire** par schéma de relation
- Un ensemble de **contraintes de clefs étrangères**
- Un ensemble de **contraintes de valeurs**

Base de données relationnelle

- Un **schéma** de bases de données relationnelle
- Des **tuples** pour peupler ce schéma, vérifiant toutes les contraintes d'intégrité.

Le modèle relationnel

À propos de SQL

- *Structured Query Language* (SQL2) est un langage **déclaratif** (décrit le *quoi* sans se préoccuper du *comment*)
- Comprend (entre autres) :
  - un langage de **description de données** (définition du schéma)
  - un langage de **manipulation de données** (insertion / suppression / mise-à-jour de données, interrogation)
- Fondé sur **le modèle et l'algèbre relationnels** mais introduit des opérateurs supplémentaires
- Objets manipulés : **tables** (≈ relations), **colonnes** (≈ attributs) et **lignes** (tuples)

Contrairement à l'algèbre relationnelle, la sémantique est celle des **multi-ensembles** (optimisation de requêtes)
⇒ doublons autorisés dans les tables.

Création de tables en SQL

Un exemple :

```
1  CREATE TABLE Notes (
2      nom VARCHAR(80),
3      prénom VARCHAR(80),
4      cours VARCHAR(80),
5      note INT
6  ); -- Crée une table Notes à 4 attributs
```

Quelques types de données (pour les attributs) :

- `CHAR(n)`, `VARCHAR(n)` → chaînes de caractères de longueur fixe ou variable
- `INT` ou `INTEGER` → entiers
- `FLOAT` → « réels » (flottants)
- `DATE`, `TIME`, `TIMESTAMP` → estampilles temporelles

Contraintes

- Clefs primaires : `PRIMARY KEY(...)` [Note : une seule par table !]
- Références (clefs étrangères) : `FOREIGN KEY(...)REFERENCES R(...)`

Exemple :

```
1  CREATE TABLE Notes (
2      nom VARCHAR(80),
3      prénom VARCHAR(80),
4      cours VARCHAR(80),
5      note INT,
6      PRIMARY KEY(nom, prénom, cours),
7      FOREIGN KEY(nom, prénom) REFERENCES
8          Élèves(nom, prénom),
9      FOREIGN KEY(cours) REFERENCES Cours(intitulé)
10 );
```

Compléments

Contraintes de valeur : `CHECK condition`
Exemple :

```
1  CREATE TABLE Notes (
2      ... -- le reste de la déclaration précédente
3      CHECK (note >= 0) AND (note <= 20)
4  );
```

Destruction de table :

```
1  DROP TABLE R; -- Supprime la table R
```

Remarque : Attention à l'ordre des créations et destructions de tables si clefs étrangères.

Insertion et suppression de tuples en SQL

Insertion de tuples : `INSERT INTO ... VALUES (...);`
Exemple :

```
1  INSERT INTO Notes
2      VALUES ('Vador', 'Dark', 'Sport', 20);
```

Suppression de tuples : `DELETE FROM ... WHERE (...);`
Exemple :

```
1  DELETE FROM Notes
2      WHERE nom = 'Vador' AND prénom = 'Dark';
3  -- supprime toutes les notes de Dark Vador
```


À vous de jouer...

Nous allons passer un peu à la pratique.

Objectifs :

- Prendre en main Postgres, et l'interface graphique pgAdmin III
- Manipuler le langage de description de données SQL pour la création d'une petite base très simple
- Insérer quelques données et illustrer le mécanisme de vérification de contraintes d'intégrité du SGBD.

À vous de jouer...

Création de tables

1. Ouvrez l'interface pgAdmin III.
2. Créez une connexion avec les paramètres suivants :
   - **Hôte** : `ensibd.imag.fr`
   - **Nom d'utilisateur** : votre nom de famille et la première lettre de votre prénom, en minuscules et sans espace
   - **Mot de passe** : idem
3. Vérifiez que la base est bien ouverte : cliquez dans l'arborescence de gauche sur la base qui porte votre nom d'utilisateur (ci-dessus), puis Schéma ▶ public ▶ tables. Pas de message d'erreur si tout se passe bien.

À vous de jouer...

4. Ouvrez l'éditeur SQL.
5. Écrivez les requêtes correspondant à la création du schéma suivant, exécutez-les.
6. Revenez dans la fenêtre principale, rafraîchissez la liste des tables (clic droit sur Tables, puis « Rafraîchir »).
7. Vérifiez que les tables sont bien créées.

Artiste : {nom, nationalité}
Disque : {titre, artiste, année, style} ;

artiste référence **Artiste**(nom)

À vous de jouer...

Insertion de données

1. Insérez un artiste de votre choix (disons 'A') avec sa nationalité dans la table **Artiste**.
2. Parcourez la table pour vérifier s'il a bien été inséré dans la table.
3. Renouvelez la même opération avec le même artiste, mais une nationalité différente. Que se passe-t-il ?

À vous de jouer...

Insertion de données

1. Insérez dans la table **Disque** un disque de votre choix de l'artiste 'A'.
2. Parcourez la table pour vérifier s'il a bien été inséré dans la table.
3. Renouvelez la même opération avec un artiste qui n'existe pas dans la relation **Artiste**. Que se passe-t-il ?
4. Essayez de supprimer dans la table **Artiste** l'artiste 'A'. Que se passe-t-il ?

3 Troisième partie

L'algèbre relationnelle

Des opérateurs pour le modèle relationnel

Algèbre relationnelle : une algèbre au sens large (mathématique) du terme...

- **un ensemble** : l'ensemble $\mathcal{R}$ des **relations**
- **des opérateurs** : des **lois internes** sur les relations :
  - Opérateurs unaires ($f : \mathcal{R} \to \mathcal{R}$) : projection, sélection
  - Opérateurs binaires ($f : \mathcal{R} \times \mathcal{R} \to \mathcal{R}$) : union, intersection, différence, produit, jointure, division.

Projection

Notation :

$$\pi_{att_1,\ldots,att_n}(\mathbf{R})$$

- **R** une relation ;
- $att_1, \ldots, att_n$ un sous-ensemble d'attributs de **R**.

Résultat : La **projection** d'une relation sur un **ensemble d'attributs** ne garde que les colonnes correspondant à ces attributs.

Projection : exemple

Projection sur l'attribut nom de la relation **Élèves**...

Élèves	prénom	nom	e-mail	filière
	Luke	Skywalker	skywalker@imag.fr	MMIS
	Dark	Vador	vador@imag.fr	IF
	Han	Solo	solo@falcon.com	IF
	Leia	Solo	princess@falcon.com	MMIS
	Jabba	The Hut	jabba@imag.fr	TELECOM

π_{nom}(Élèves)	nom
	Skywalker
	Vador
	Solo
	The Hut

Sélection

Notation :

$$\sigma_P(\mathbf{R})$$

- **R** une relation ;
- P une condition booléenne (un **critère**) sur les attributs de **R**.

Résultat : La **sélection** d'une relation selon un **critère** ne garde que les tuples satisfaisant ce critère.

Sélection : exemple

Sélection dans la relation **Notes** des tuples dont la note est inférieure à 10.

Notes	cours	prénom	nom	note
	sport	Dark	Vador	20
	sport	Jabba	The Hut	3
	pilotage	Han	Solo	15

$\sigma_{\text{note}<10}(\textbf{Notes})$	cours	prénom	nom	note
	sport	Jabba	The Hut	3

Passage à la pratique

(R1) Quelle est la liste des noms des morceaux de l'artiste numéro '22164' ?

À vous de jouer...

Passage à la pratique

(R2) Quel est le nom exact de l'artiste numéro '22164' ?

À vous de jouer...

Opérateurs ensemblistes

Notation :

$$\mathbf{R}_1 \cup \mathbf{R}_2 \quad ; \quad \mathbf{R}_1 \cap \mathbf{R}_2 \quad ; \quad \mathbf{R}_1 - \mathbf{R}_2$$

$\mathbf{R}_1$ et $\mathbf{R}_2$ deux relations **de même schéma**.

Résultat :

- **Union** : tous les tuples contenus dans $\mathbf{R}_1$ **ou** $\mathbf{R}_2$
- **Intersection** : tous les tuples contenus dans $\mathbf{R}_1$ **et** $\mathbf{R}_2$
- **Différence** : tous les tuples contenus dans $\mathbf{R}_1$ **mais pas** dans $\mathbf{R}_2$

Différence : exemple

Différence entre **Élèves 1** et **Élèves 2**...

Élèves 1	prénom	nom	e-mail	filière
	Luke	Skywalker	skywalker@imag.fr	MMIS
	Dark	Vador	vador@imag.fr	IF
	Han	Solo	solo@falcon.com	IF
	Leia	Solo	princess@falcon.com	MMIS

Élèves 2	prénom	nom	e-mail	filière
	Dark	Vador	vador@imag.fr	IF
	Obi-Wan	Kenobi	kenobio@imag.fr	MMIS

Élèves 1 – Élèves 2	prénom	nom	e-mail	filière
	Luke	Skywalker	skywalker@imag.fr	MMIS
	Han	Solo	solo@falcon.com	IF
	Leia	Solo	princess@falcon.com	MMIS

Passage à la pratique

(R6) Quel est la liste des identifiants d'artistes n'ayant sorti que des *Singles* ?

À vous de jouer...

Produit cartésien

Notation :

$$\mathbf{R}_1 \times \mathbf{R}_2$$

$\mathbf{R}_1$ et $\mathbf{R}_2$ deux relations (de schémas quelconques).

Résultat : Toutes les combinaisons possibles (t_1, t_2) de tuples $t_1 \in \mathbf{R}_1$ et $t_2 \in \mathbf{R}_2$.

Produit cartésien : exemple

Élèves 2	prénom	nom	e-mail	filière
	Dark	Vador	vador@imag.fr	IF
	Obi-Wan	Kenobi	kenobio@imag.fr	MMIS

Notes	cours	prénom	nom	note
	sport	Dark	Vador	20
	sport	Jabba	The Hut	3
	pilotage	Han	Solo	15

R	Él2.pré	Él2.nom	e-mail	filière	cours	N.pré	N.nom	note
	Dark	Vador	va[...]fr	IF	sport	Dark	Vador	20
	Dark	Vador	va[...]fr	IF	sport	Jabba	The Hut	3
	Dark	Vador	va[...]fr	IF	pilotage	Han	Solo	15
	Obi-Wan	Kenobi	ke[...]fr	MMIS	sport	Dark	Vador	20
	Obi-Wan	Kenobi	ke[...]fr	MMIS	sport	Jabba	The Hut	3
	Obi-Wan	Kenobi	ke[...]fr	MMIS	pilotage	Han	Solo	15

Produit cartésien : remarques

- Produit possible entre tables ayant des attributs communs (de mêmes noms)
- Auto-produit ($\mathbf{R} \times \mathbf{R}$) possible aussi (usage : renommer les attributs pour éviter la confusion).

Jointure conditionnelle

Notation :

$$\mathbf{R}_1 \bowtie_P \mathbf{R}_2$$

- $\mathbf{R}_1$ et $\mathbf{R}_2$ deux relations (de schémas quelconques).
- P une condition booléenne (un **critère**) sur les attributs de $\mathbf{R}_1$ et $\mathbf{R}_2$.

Résultat : Complètement équivalent à un produit cartésien suivi d'une sélection : $\mathbf{R}_1 \bowtie_P \mathbf{R}_2 = \sigma_P(\mathbf{R}_1 \times \mathbf{R}_2)$.

Jointure naturelle

Notation :

$$\mathbf{R}_1 \bowtie \mathbf{R}_2$$

- $\mathbf{R}_1$ et $\mathbf{R}_2$ deux relations ayant **des attributs en commun** (de même nom).

Résultat : Toutes les combinaisons possibles (t_1, t_2) de tuples $t_1 \in \mathbf{R}_1$ et $t_2 \in \mathbf{R}_2$ pour lesquelles t_1 et t_2 ont **les mêmes valeurs sur les attributs communs**. Les colonnes correspondant à ces attributs communs sont fusionnées dans la relation résultante.

Jointure naturelle : exemple

Jointure entre **Élèves 2** et **Notes** : informellement, liste des élèves (avec leurs données) et leurs notes.

Élèves 2	prénom	nom	e-mail	filière
	Dark	Vador	vador@imag.fr	IF
	Obi-Wan	Kenobi	kenobio@imag.fr	MMIS

Notes	cours	prénom	nom	note
	sport	Dark	Vador	20
	sport	Jabba	The Hut	3
	pilotage	Han	Solo	15

Él 2 ⋈ Notes	prénom	nom	e-mail	filière	cours	note
	Dark	Vador	vador@imag.fr	IF	sport	20

Passage à la pratique

(R3) Quels sont les albums du groupe Radiohead (donnez juste leur nom, sans doublon) ?

À vous de jouer...

Passage à la pratique

(R4) Par qui a été composée la chanson *Straumnes* ?

À vous de jouer...

Passage à la pratique

(R5) Sur quel album la chanson *Straumnes* est-elle enregistrée, et quels sont les morceaux de cet album ?

À vous de jouer...

Pour aller plus loin... La division

Notation :

$$\mathbf{R}_1 \div \mathbf{R}_2$$

$\mathbf{R}_1$ et $\mathbf{R}_2$ deux relations telles que **l'ensemble des attributs de $\mathbf{R}_2$ est inclus dans l'ensemble des attributs de $\mathbf{R}_1$.**

- Att_2 : ensemble des attributs de $\mathbf{R}_2$
- $Att_1 = Att_2 \cup Att_3$: ensemble des attributs de $\mathbf{R}_1$

Résultat : Tous les tuples t_3 sur Att_3 tels que : pour chaque tuple $t_2 \in \mathbf{R}_2$ (sur Att_2), le tuple (t_2, t_3) (sur Att_1) existe dans $\mathbf{R}_1$.

Division : exemple

Élèves 3	nom	filière
	Skywalker	MMIS
	Vador	IF
	Solo	IF
	Solo	MMIS
	The Hut	TELECOM

÷

R_2	filière
	MMIS
	IF

Informellement : trouver tous les noms d'élèves qui apparaissent à la fois dans la filière MMIS et dans la filière IF.

Élèves 3 ÷ R_2	nom
	Solo

Passage à la pratique

(R8) Quels artistes ont des disques dans tous les pays dans lesquels The Beatles en ont ?

À vous de jouer...

Pour ceux qui avancent très vite...

À vous de jouer sur les requêtes restantes

—

(NB : certaines ne sont pas exprimables en algèbre relationnelle)

4 Quatrième partie

Structured Query Language

Avant de commencer...

Dans cette partie, nous allons écrire des requêtes d'interrogation sur une base préexistante. Avant de commencer :

1. Ouvrez l'interface pgAdmin III.
2. Créez une connexion avec les paramètres suivants :
   - **Hôte** : `ensibd.imag.fr`
   - **Nom d'utilisateur** : `musicbrainz`
   - **Mot de passe** : `musicbrainz`
3. Vérifiez que la base est bien ouverte : cliquez sur le nom de la base dans l'arborescence de gauche, puis Schémas ▶ musicbrainz ▶ Vues. Vous devriez y trouver l'ensemble des tables **mb_*** de la feuille de TD. [1]

1. À notre niveau, Vue ≈ Table.

SELECT FROM WHERE

Notes : {<u>prénom</u>, <u>nom</u>, <u>cours</u>, note, filière}

Un exemple :

```
1 SELECT nom, notes
2 FROM Notes; -- sélectionne les noms des élèves ainsi
      que leur notes
```

```
1 SELECT nom, notes
2 FROM Notes
3 WHERE notes >= 10; -- sélectionne que les élèves qui
      ont une note au dessus de 10
```


Passage à la pratique

(R1) Quelle est la liste des noms des morceaux de l'artiste numéro '22164' ?

À vous de jouer...

Passage à la pratique

(R2) Quel est le nom exact de l'artiste numéro '22164' ?

À vous de jouer...

Passage à la pratique

(R3) Quels sont les albums du groupe Radiohead (donnez juste leur nom, sans doublon) ?

À vous de jouer...

Passage à la pratique

(R4) Par qui a été composée la chanson *Straumnes* ?

À vous de jouer...

Passage à la pratique

(R5) Sur quel album la chanson *Straumnes* est-elle enregistrée, et quels sont les morceaux de cet album ?

À vous de jouer...

SOUS REQUÊTES

Notes : {prénom, nom, cours, note, filière}
Élèves : {prénom, nom, e-mail, filière}

Un exemple :

```sql
SELECT nom
FROM Élèves
WHERE nom NOT IN (SELECT nom FROM Notes);
 -- sélectionne les noms des élèves qui n'ont pas de
    note
```

```sql
SELECT nom
FROM Élèves e
WHERE NOT EXISTS (SELECT * FROM Notes n
                  WHERE e.nom = n.nom);
 -- sélectionne les noms des élèves qui n'ont pas note
```

Passage à la pratique

(R6) Quel est la liste des identifiants d'artistes n'ayant sorti que des *Singles* ?

À vous de jouer...

GROUP BY

Notes : {prénom, nom, cours, note, filière}

Un exemple :

```
1 SELECT prénom, nom, AVG(notes)
2 FROM Notes
3 GROUP BY prénom, nom; -- calcule la moyenne des notes
      de tous les cours pour chaque élève
```

Autres opérateurs d'agrégation : `SUM`, `MAX`, `MIN`, `COUNT`

GROUP BY ... HAVING

Notes : {prénom, nom, cours, note, filière}

Un exemple :

```
1 SELECT prénom, nom, AVG(notes)
2 FROM Notes
3 GROUP BY prénom, nom
4 HAVING AVG(notes) > 10; -- garde les élèves qui ont
      une moyenne supérieure à 10
```


Passage à la pratique

(R7) Donnez la liste des albums du groupe U2, avec leur durée.

À vous de jouer...

Passage à la pratique

(R8) Donnez la liste des albums du groupe U2 de plus d'une heure, avec leur durée.

À vous de jouer...

Passage à la pratique

(R9) Quels artistes ont des disques dans tous les pays dans lesquels The Beatles en ont ?

À vous de jouer...

Pour ceux qui avancent très vite...

À vous de jouer sur les requêtes restantes

—

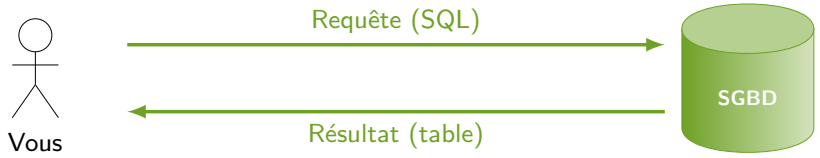
(NB : toutes sont faisables en SQL)

5 Cinquième partie

Utilisation d'une BD dans une application

Architecture *n*-tiers

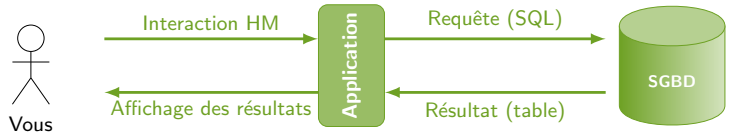
Voici (approximativement) ce que l'on a fait jusqu'à présent :



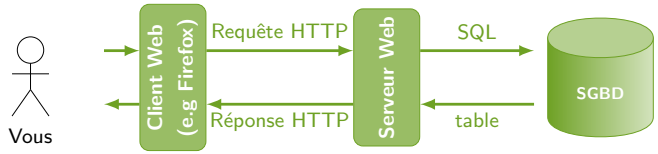
Architecture *n*-tiers

Dans la vraie vie, l'utilisateur final n'accède jamais **directement** au SGBD (*qui a déjà écrit une requête SQL pour réserver un billet de train ?*)

En pratique, c'est donc plutôt ça :



Et même plutôt ça :



Accéder à un SGBD depuis un programme

Schéma classique d'accès à un SGBD à partir d'un langage de programmation quelconque :

1. Récupérer une connexion à la base
2. Exécuter une (ou des) requête(s) en utilisant cette connexion
3. Récupérer le résultat des requêtes (et le traiter)
4. Fermer la connexion

La plupart des SGBD possèdent des pilotes d'accès (ou modules) pour la plupart des langages.
Exemple : pour une application développée en Python et le SGBD Postgres :

```
import psycopg2
```


Accéder à Postgres avec Python

Un exemple avec Python : je veux créer une fonction prenant en paramètre un prénom et un nom, et affichant toutes les notes de l'élève, récupérées depuis la base de données.

Déclaration de la fonction :

```
1  def getNotes(prenom, nom):
2      # Ici le corps de la fonction...
```

Accéder à Postgres avec Python

1. Récupérer une connexion à la base

```
connection = psycopg2.connect(
  database="notes",
  user="utilisateur",
  password="motdepasse",
  host="ensibd.imag.fr")
```

2. Exécuter une (ou des) requête(s) en utilisant cette connexion

```
cursor = connection.cursor()
req = 'SELECT cours, note FROM Notes WHERE prenom
    =\'' + prenom + '\' AND nom=\'' + nom + '\';'
cursor.execute(req)
```


Accéder à Postgres avec Python

3. Récupérer le résultat des requêtes (et le traiter)

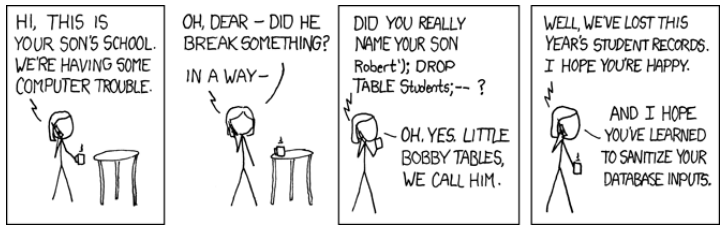
```
data = cursor.fetchall()
for r in data: # pour chaque tuple
  print(r)     # on l'affiche simplement
```

4. Fermer la connexion

```
connection.close()
```


Construction de requête, remarque

Remarque : Dans la pratique, ce n'est pas une bonne idée de construire une requête directement par concaténation de chaîne dans un programme...



Source : `http://xkcd.com/327/` (licence CC-by-NC)

La plupart des langages proposent des méthodes de construction de requêtes plus robustes, résistantes en particulier à l'injection de code SQL.

À vous de jouer...

Objectif : créer une petite application Python qui affiche la liste des disques pour un artiste donné :

- d'abord dans l'interpréteur Python classique ;
- ensuite à l'aide d'une page Web affichée dans votre navigateur préféré (Firefox pour aujourd'hui).

Le dossier `python` de votre archive contient (entre autres) :

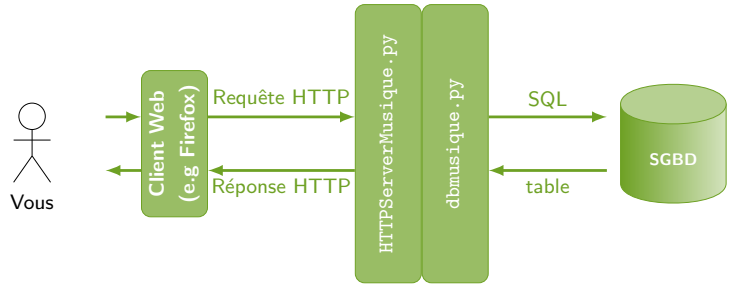
- un fichier `dbmusique.py` (à compléter), qui contient le code d'accès à la base de données ;
- un fichier `musique.py`, faisant appel à `dbmusique.py`, qui contient le code d'affichage pour l'application version « classique » ;
- un fichier `HTTPServerMusique.py`, faisant appel à `dbmusique.py`, qui contient le code du serveur chargé de la version « web » de l'application.

Le code d'accès au SGBD (`dbmusique.py`) est commun aux deux applications.

Architecture de l'application Python

Application, version classique

Application, version Web



Application, version « classique »

1. Ouvrez dans votre éditeur préféré le fichier `dbmusique.py` du dossier `python` de votre archive
2. Ce fichier contient la fonction définie par

```
1  def getReleasesByArtist(artist):
```

   Cette fonction interroge la base de données pour renvoyer la liste des disques pour l'artiste `artist`.
   Complétez la ligne commençant par `requete = ...` en mettant en partie droite une chaîne de caractère représentant la requête SQL adéquate.
3. Exécutez le fichier `musique.py` du dossier `python` : il demande à l'utilisateur le nom d'un artiste et affiche tous les disques correspondant grâce à `dbmusique.py`.

Application, version « web »

1. Exécutez le fichier `HTTPServerMusique.py` grâce à votre interpréteur Python : il lance le serveur chargé de répondre aux requêtes HTTP. Laissez le programme lancé.
2. Ouvrez votre navigateur Firefox et tapez dans la barre d'adresse :

   `http://localhost:4242/`
3. Testez l'application en tapant un nom d'artiste dans la boîte prévue à cet effet.

Et ça marche !

6 Sixième partie

Conclusion

Conclusion

Aujourd'hui, nous avons vu (entre autres) :

- **Côté théorique :**
  - Modèle relationnel
  - Algèbre relationnelle
- **Côté langage :**
  - SQL, description de données (création / destuctions de tables)
  - SQL, manipulation de données (insertion / suppression de tuples)
  - SQL, manipulation de données (interrogation)
- **Côté technique :**
  - Qu'est-ce qu'un SGBD
  - Utilisation de Postgres en pratique
  - Architecture *n*-tiers (couplage SGBD / Python)

...Il y aurait plein d'autres choses à dire... Autres aspects de SQL (vues, contraintes d'accès,...), formes normales de relation, dépendances fonctionnelles, entité association, optimisation de requêtes, relationnel-objet, bases de données réparties...

Merci de votre participation...