Bases de données pour le Web

IO2

Internet et outils

Cristina Sirangelo
IRIF, Université Paris Diderot
cristina@irif.fr

Bases de données et Web

Toutes les applications Web manipulent des données persistantes

Exemples :

- Site de vente en ligne : descriptions et prix des articles vendus, le transitions de vente, …
- Site d'une compagnie aérienne : quels vols à quels horaires, ..
- Espace utilisateur de n'importe quel site : informations d'enregistrement des utilisateurs, …
- Réseaux sociaux : les profils des membres, les posts, qui suit qui, qui aime quoi…

Toutes ces données doivent être stockées de façon durable chez le serveur,

- pas uniquement pendant une session utilisateur

- pas uniquement pendant que le serveur est actif - elles doivent continuer à exister même si le serveur est arrêté

Ces données doivent donc exister ailleurs que dans la mémoire centrale du serveur (et elle ne serait pas suffisante de toute façon!)

La gestion des données persistantes

Un problème vieux comme l'informatique

Pour gérer une liste (par exemple de films),

pourrait-on utiliser un fichier texte?

```
> cat films.txt
Alien 1979 Scott
Vertigo 1958 Hitchcock
Psychose 1960 Hitchcock
Kagemusha 1980 Kurosawa
Volte-face 1997 Woo
```

La gestion des données persistantes

Ou une structure de données sérialisée sur disque?

- Définir une classe d'objets qu'on va mettre dans un tableau.

```
class Film {
   public $titre,
          $année,
          $réalisateur,
}
```

- Créer un tableau d'objets films le sérialiser dans un fichier sur sur le disque

Les problèmes des systèmes à base de fichiers

Accès difficile aux données

Ouvrir le fichier

Parcourir les “lignes”, ou les objets sérialisés

Effectuer les comparaisons

Format complexe (tenir compte des espaces…)

Format de fichiers non standards ⇒ partage des données difficile

Écriture d’un programme spécifique pour chaque interrogation/modification des données : coûts de développement élevés, coûts de maintenance élevés

Intégrité logique des données :

Comment garantir que les données restent cohérentes
(par exemple éviter :
`Vertigo 1958 Hitchcock`
`Vertigo 1962 Hitchcock`)

Les problèmes des systèmes à base de fichiers

Concurrence

Que se passe-t-il si plusieurs utilisateurs ajoutent, modifient ou suppriment des lignes simultanément ?

Gestion des pannes

Comment faire en sorte que les données ne soient pas laissées dans un état incohérent si le serveur va en panne pendant qu'il les manipule?
À programmer soi même (l'OS ne suffit pas)

Sécurité

Comment empêcher un programme erroné ou malveillant d'écrire des données autres (e.g. écraser le contenu du fichier) ?

etc.

SGBD

Les Systèmes de Gestion de Bases de Données ont été conçus pour apporter des (bonnes) réponses à ces problèmes

- Manipuler informations complexes de façon abstraite
  - sans se préoccuper de comment elle sont physiquement organisées sur disque (indépendance physique)
- Optimiser et rendre efficace l'accès aux données
- Garantir l'intégrité des données et la tolérance aux pannes
- Assurer un résultat cohérent quand plusieurs utilisateurs accèdent simultanément aux données

SGBD relationnels

Représentation de l'information sous forme de tables

La manière dont l'information est réellement stockée sur disque est inconnue de l'application

L'application ne voit que les "tables" présentées par le SGBD

Langage "standard" pour l'interrogation/manipulation de ces tables :
SQL (norme ANSI de 1992) - *Structured Query Language*

Il existe de très nombreux systèmes de gestion de bases de données relationnelles

Oracle, PostgreSQL, SQL Server, DB2, MySQL…

Un aperçu du modèle relationnel des données

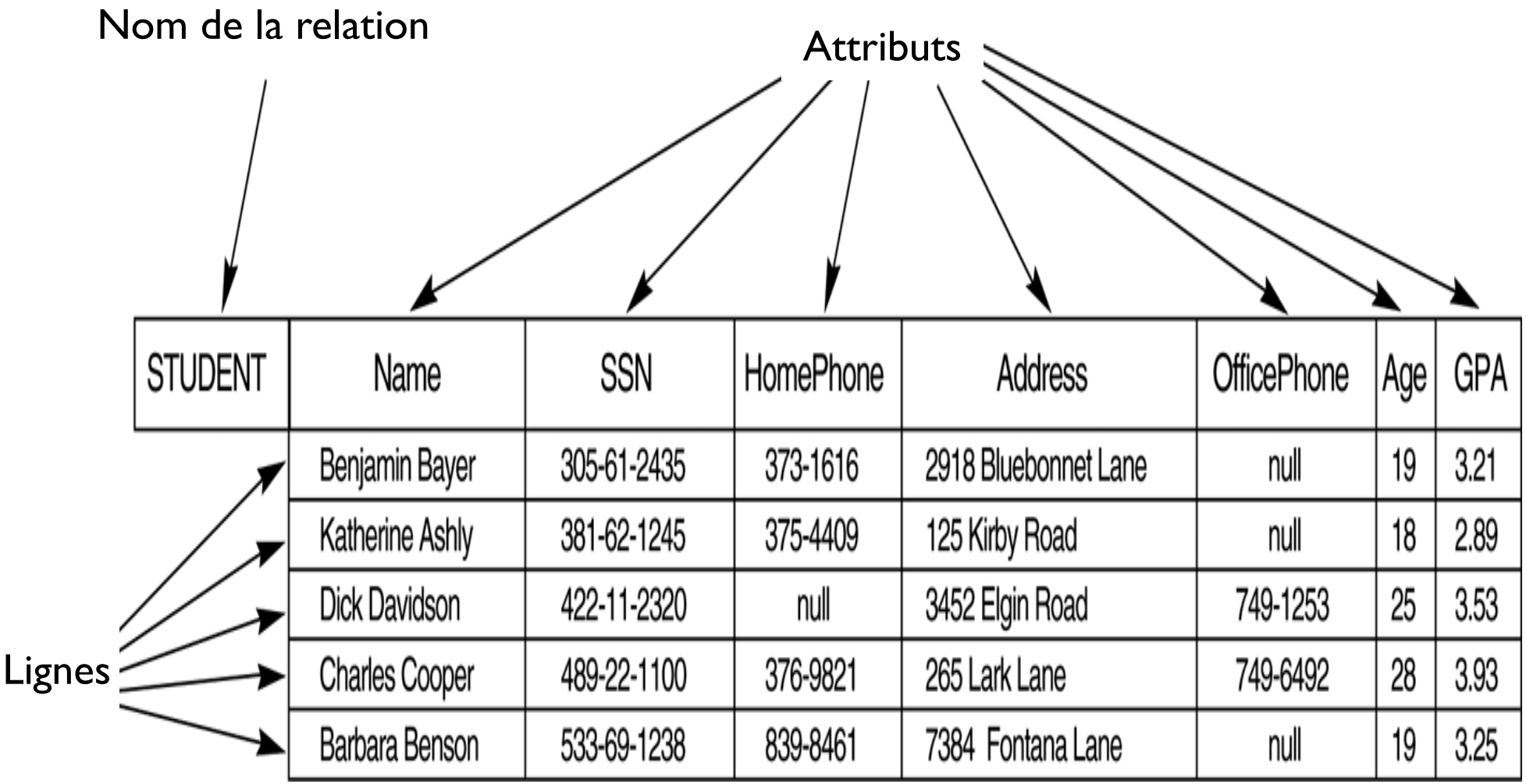
Modèle relationnel des données

- Une base de données consiste en plusieurs tables (relations)
- Chaque table a un nom

- Dans une table chaque colonne a un nom
- les noms des colonnes d'une table sont appelés attributs (ou champs)
- Chaque attribut a un domaine associé (i.e. l'ensemble des valeurs possibles pour cet attribut)

- Les données dans chaque table sont un ensemble de lignes (tuples)
  - une ligne fournit à chaque attribut une valeur de son domaine

Modèle relationnel des données



STUDENT	Name	SSN	HomePhone	Address	OfficePhone	Age	GPA
	Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	null	19	3.21
	Katherine Ashly	381-62-1245	375-4409	125 Kirby Road	null	18	2.89
	Dick Davidson	422-11-2320	null	3452 Elgin Road	749-1253	25	3.53
	Charles Cooper	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
	Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	null	19	3.25

Exemple : les films sous forme de table

Films

titre	*annee*	*realisateur*
Alien	1979	Scott
Vertigo	1958	Hitchcock
Psychose	1960	Hitchcock
Kagemusha	1980	Kurosawa
Volte-face	1997	Woo
Pulp Fiction	1995	Tarantino
Titanic	1997	Cameron
Sacrifice	1986	Tarkovski

Schéma et instance d'une table

- La structure des données est rigide (toutes les lignes ont les mêmes attributs)
  - Cette structure est appelé schéma de la table :

STUDENT

Name	SSN	Phone	Adress	OfficePhone	Age	GPA

  - L'ensemble des lignes est appelé instance de la table

- Le schéma est défini une fois pour toute
- L'instance varie dans le temps selon les données qu'on inséré et on supprime

- Des contraintes d'intégrité sont en général associées à chaque table, qui empêchent les instances non valides (valeurs requis, uniques, etc…)

Schéma et instance d'une base de données

- Une bases de données est formée par plusieurs tables

STUDENT

Name	SSN	Phone	Adress	OfficePhone	Age	GPA
Bayer	347294	333	Albyn Pl.	367	18	3.24
Ashly	5784673	466	Queen St.	390	20	3.53
Davidson	4357387	589	Princes St.	678	25	3.25

COURSE

Course-Id	Title
12	CS
34	DB

EXAM

Student-SSN	Course-Id
347294	12
5784673	12
4357387	34
347294	34

- Schéma de la base de donnée : l'ensembles de schémas de toutes ses tables
- Instance de la base de données : l'ensemble des instances de toutes ses tables (i.e les données)

SQL

Langage déclaratif

Permet de manipuler à la fois le schéma et les instances d'une bases de données

Composante DDL (Data Definition Language)

- définir le schéma de la bases de données

Composante DML (Data Manipulation Language)

- interroger (extraire de l'information de) la BD de manière déclarative :

```
SELECT titre          -- l'attribut recherché
FROM Films             -- la table interrogée
WHERE annee > 1980;   -- le critère de sélection
```

Quelques éléments de SQL DDL

Création d'une base de données

```
CREATE DATABASE ma_base; -- commande SQL
```

Au sein d'un serveur on peut créer plusieurs bases de données différentes

Chaque base à ses propres tables, mais une table appartient à une seule base

On peut associer des droits d'accès aux utilisateurs des bases, et des tables

Création d'une table (schéma)

Au sein de la base de données courante :

```
CREATE TABLE Film
(titre       VARCHAR(30),
 annee       INTEGER,
 realisateur  VARCHAR(30)
);
```

Les types des attributs
varient d'un SGBD à l'autre

Cette déclaration contient en général aussi la définition de plusieurs contraintes d'intégrité (donc on discutera plus loin)

À sa création, une table est "vide" (instance vide, on ne crée que le schéma)

Elle ne contient aucune donnée, i.e. aucune ligne

Détruire une table

Détruire la table (schéma et données):

```
DROP TABLE Film;
```

Supprimer toutes les données (mais garder le schéma):

```
TRUNCATE TABLE Film;
```

Quelques éléments de SQL - DML

Insérer des données

```
INSERT INTO Film
VALUES ('Pulp Fiction', 1995, 'Tarantino');
```

Note : on peut ne saisir que quelques attributs, et dans un ordre différent de celui défini pour la table.

```
INSERT INTO Film (realisateur, titre)
VALUES ('Allen', 'Match Point');
```

Insérer des données

Resultat :

Films

titre	*annee*	*realisateur*
Alien	1979	Scott
Vertigo	1958	Hitchcock
Psychose	1960	Hitchcock
Kagemusha	1980	Kurosawa
Volte-face	1997	Woo
Pulp Fiction	1995	Tarantino
Titanic	1997	Cameron
Sacrifice	1986	Tarkovski
Match Point	NULL	Allen

Attention : génère une erreur si les valeurs pas spécifiées sont requises dans la table

Supprimer des données

```
DELETE FROM Film WHERE annee <= 1960;
```

Supprime toutes les lignes pour lesquelles l'année est inférieure ou égale à 1960.

Modifier des données

```
UPDATE Film
SET realisateur = 'Wu'
WHERE realisateur = 'Woo';
```

Met à jour le champs `réalisateur` de toutes les lignes sélectionnées par la clause `WHERE`.

SQL - DML : Interrogation des données (aperçu)

Interrogation de bases de données

Interroger une base de données: "extraire" des données de la base

Requête: produire une nouvelle table à partir des tables dans la base

Exemples de requêtes:

quelles sont tous les films de 1954 ?

quels réalisateurs ont réalisé le plus de films?

SQL permet d'exprimer les requêtes de façon déclarative

on exprime **quelles sont les données** qu'on veut extraire et

NON PAS **comment les extraire**

Interroger une table

```
SELECT * FROM Film;
```

sélectionne toutes les colonnes (*) de la table Film

Resultat :

titre	**annee**	**realisateur**
Alien	1979	Scott
Vertigo	1958	Hitchcock
Psychose	1960	Hitchcock
Kagemusha	1980	Kurosawa
Volte-face	1997	Woo
Pulp Fiction	1995	Tarantino
Titanic	1997	Cameron
Sacrifice	1986	Tarkovski
Match Point	NULL	Allen

Interroger une table

On peut sélectionner les colonnes de son choix

```
SELECT titre, année FROM Film;
```

Resultat :

titre	annee
Alien	1979
Vertigo	1958
Psychose	1960
Kagemusha	1980
Volte-face	1997
Pulp Fiction	1995
Titanic	1997
Sacrifice	1986
Match Point	NULL

Interroger une table

La clause `WHERE` permet de définir un ensemble de conditions qui doivent être vérifiées par les lignes retenues

```
SELECT titre, annee
FROM Film
WHERE titre = 'Vertigo'
OR realisateur = 'Hitchcock'
OR (annee >= 1995
      AND annee < 2000);
```

Resultat :

titre	annee
Vertigo	1958
Psychose	1960
Volte-face	1997
Pulp Fiction	1995
Titanic	1997

Sélectionne les titres et les années des films dont le titre est 'Vertigo' ou le réalisateur est 'Hitchcock', ou l'année est entre 1995 et 2000

Introduction à MySQL

MySQL

Un système de gestion de bases de données relationnelles

Un des plus utilisés pour les applications Web

Relativement “léger” par rapport à d’autres SGBD

L’installation de MySQL fournit :

Un serveur : `mysqld`

le sgbd - seul capable d'accéder et manipuler les données

Des utilitaires

`mysql` : client permettant de se connecter au serveur de bases de données pour demander la manipuler et interrogation des données

`mysqldump` : client permettant de faire des sauvegardes

Connexion au serveur MySQL

Lancer le client mysql :

```
> mysql -h nom_serveur –u nom_utilisateur -p
Enter password: *******
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 8.0.11 MySQL Community Server - GPL
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> help;
```

Pour voir les bases de données disponibles :

```
mysql> show databases;
```

Connexion à une base de données MySQL

- Pour sélectionner une base de données :

```
mysql> USE ma_base
Database changed
```

- Pour se connecter et en même temps sélectionner la base `io2021` :

```
> mysql -h nom_serveur –u nom_utilisateur -p io2021
```

- On peut aussi utiliser le fichier de configuration (attention ! force à stocker le mot de passe en clair)

```
$HOME/.my.cnf
    [client]
    host = nom_serveur
    user = nom_utilisateur
    password = motdepasse
> mysql io2021
```

MySQL : utilisation du client mysql

Un fois la base de données sélectionnée, on peut taper à l'invite du shell mysql toute commande SQL pour cette base :

```
mysql> CREATE TABLE Film
    -> (titre      VARCHAR(30),
    ->  annee      INTEGER,
    ->  realisateur  VARCHAR(30)
    -> );
Query OK, 0 rows affected (0.05 sec)

mysql> INSERT INTO Film (realisateur, titre)
    -> VALUES ('Allen', 'Match Point');
Query OK, 1 row affected (0.00 sec)
```

MySQL : utilisation du client mysql

Un fois la base de données sélectionnée, on peut taper à l'invite du shell mysql toute commande SQL pour cette base :

```
mysql> SELECT *
    -> FROM Film
+-------------+-------+-------------+
| titre       | annee | realisateur |
+-------------+-------+-------------+
| Match Point |  NULL | Allen       |
+-------------+-------+-------------+
1 row in set (0.00 sec)
```

MySQL : exécution de scripts SQL

On peut entrer les commandes SQL depuis un fichier préalablement édité.

On utilise conventionnellement l'extension '`.sql`'

```
> mysql -h nom_serveur –u nom_utilisateur -p < script.sql
```

Ou depuis mysql :

```
mysql> SOURCE script.sql
```

script_films.sql

mysql> source script_films.sql

Quelques commandes MySQL

On peut également utiliser des commandes propres à MySQL par exemple :

décrire le schema d'une table :

```
mysql> DESCRIBE Film;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| titre       | varchar(30) | YES  |     | NULL    |       |
| annee       | int(11)     | YES  |     | NULL    |       |
| realisateur | varchar(30) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Quelques commandes MySQL

On peut également utiliser des commandes propres à MySQL par exemple :

Lister les bases de données

```
SHOW DATABASES;
```

Lister les tables de la base courante

```
SHOW TABLES;
```

Afficher le nom de la base courante

```
SELECT DATABASE();
```

Afficher le nom de l'utilisateur courant

```
SELECT USER();
```

Chargement de données

On peut aussi insérer des données depuis un fichier (insertion dite “en masse”)

Exemple :

```
mysql> LOAD DATA LOCAL INFILE
    -> 'films.txt' INTO TABLE Film
    -> FIELDS TERMINATED BY ',';
Query OK, 8 rows affected (0.00 sec)
Records: 8  Deleted: 0  Skipped: 0  Warnings: 0
```

films.txt

mysql> load data…

Types de données (My)SQL

Principaux types de données SQL ANSI supportés par MySQL :

Nombres :

- `SMALLINT` (2 octets)
- `INTEGER` (4 octets)
- `BIGINT` (8 octets)
- `FLOAT` (4 octets)
- `REAL` (8 octets)
- `DOUBLE PRECISION` (8 octets)

- `DATE` (3 octets)
- `TIME` (3 octets)

Types de données (My)SQL

Principaux types de données SQL ANSI supportés par MySQL :

Chaînes de caractères :

- `VARCHAR (M)` avec `M` < 256 taille variable
- `CHAR (M)` avec `M` < 256 taille fixe
- ⚠ MySQL only : `BLOB` / `TEXT`

  suite de octets / suite de caractères, taille < 2^{16}
- ⚠ MySQL only : `MEDIUMBLOB` / `MEDIUMTEXT` taille < 2^{24}
- ⚠ MySQL only : `LONGBLOB` / `LONGTEXT` taille < 2^{32}

Types de données (My)SQL

Principaux types de données SQL ANSI supportés par MySQL :

Dates :

- `DATE` (3 octets)
- `TIME` (3 octets)
- `DATETIME` (8 octets)
- `TIMESTAMP` (4 octets)
- `YEAR` (1 octets)

Ensembles :

- ⚠️ MySQL only : `ENUM ('val1', 'val2', ...)` (<= 2 octets)
- ⚠️ MySQL only : `SET ('val1', 'val2', ...)` (<= 8 octets)

Types de données (My)SQL

`INTEGER[(M)] [UNSIGNED] [ZEROFILL]`

taille max : `M` chiffres (seulement pour l'affichage)

Non signé si `UNSIGNED`

`ZEROFILL` : 3 ⇒ 0000000003

`VARCHAR(M) [BINARY]`

Par défaut, pas de différence entre minuscules et majuscules, i.e. `'aaa'` identique à `'AAA'`, sauf si `BINARY` est précisé