Sessions et sécurité

L'utilisation des sessions réduit les failles de sécurité du site (les problèmes liés aux deux premières solutions) :

- Les données de session qui peuvent contenir des informations sensibles sont stockées chez le serveur (pas dans l'url , ni dans des cookies)
  - normalement pas accessibles par d'autres utilisateurs
- La seule information sensible encore présente chez le client est l'identifiant de session (qui donne accès à toutes les données de session)
- Toutefois l'id de session est une information sensible uniquement pendent la session d'un utilisateur (de `session_start()` à `session_destroy()`)
- Après déconnexion de l'utilisateur (`session_destroy()`), cela ne donne plus accès à quoi que ce soit chez le serveur
  - ⇒ à ce moment là, la disponibilité chez le client de l'identifiant de session

    (dans un cookie pas effacé, ou dans historique du browser) n'est pas critique

Sessions et sécurité

La seule vulnérabilité des sessions : le vol de session

- Le seul problème encore possible : identifiant de session lu par un utilisateur malveillant (personne ou programme) pendant une session en cours
- Plus difficile que dans les deux premières solutions mais encore possible :
  - si l'id de session est passé dans l'url il est visible pendant la session
  - s'il est stocké dans un cookie, un script (Javascript) introduit par un site malveillant peut s'exécuter sur le client et lire ses cookies, pendant la session (XSS)
- Il existe des solutions pour lutter contre le vol de session également…(e.g. cookies stockés en mode httpOnly qui les rendent inaccessibles au Javascript, …)
- Une précaution habituelle (automatique dans la plus part des serveurs) : timeout de session

Sessions et sécurité

Timeout de session

- Avec les sessions la déconnexion de l'utilisateur devient une opération critique
- Si l'utilisateur quitte le site sans déconnexion, l'id de session reste disponible chez le client et encore actif chez le serveur
  - risque plus élevé de vol de session
- Solution : en général le serveur appelle automatiquement `session_destroy()` après une certaine période d'inactivité de la session
  - Inactivité : pas de pages demandés avec l'id de session
- Automatique pour les sessions PHP (et le délai peut être réglé)

Maintenir l'état de l'interaction client-serveur

Solution 3 : les sessions

- ⇒ On peut considérer les risques liés à l'utilisation des sessions relativement limités
- ⇒ Le sessions sont adaptées au partage d'information sensibles entre plusieurs pages
  - e.g. protection d'un site par authentification de l'utilisateur
  - rappel : une solution simpliste pour protéger une seule page ne nécessite pas de sessions

Protéger l'accès à un site par login avec les sessions

- Exemple d'organisation du site (réduit au minimum afin d'exemplifier)
  - une page principale (`home.php`) avec des liens vers les autres pages du site
  - trois pages indépendantes (`page1.php, page2.php, page3.php`) avec des liens vers le *home*
- Utilisation des sessions pour la protection du site - idée :
  - quand le login réussi une variable de session `'user'` est créé
  - ⇒ `isset($_SESSION('user'))` indique, dans n'importe quelle page du site, si l'utilisateur a bien effectué le login avant d'arriver à cette page
    - chaque page du site commence par faire cette vérification, si elle réussit la page est affichée, sinon le login est demandé par un formulaire
  - `$_SESSION('user')` pourrait être une variable booléenne, mais on peut l'utiliser pour transférer plus d'infos aux autres pages (e.g. le nom utilisateur)
- Pour simplifier: le formulaire de login est toujours traité par la page principale

Protéger l'accès à un site par login avec les sessions

- On aura un bibliothèque de fonctions pour chaque fonctionnalité.
  - les fonctions de validation de l'input (fonc_valid.php)
  - le fonction qui gèrent l'authentification de l'utilisateur (fonc_autent.php)
  - le fonctions d'affichage du contenu des pages (fonc_sortie.php)

Protéger l'accès à un site par login avec les sessions

- La fonction principale appelée par chaque page pour vérifier que l'utilisateur a effectué le login avec succès avant d'arriver sur la page :

```
//fonc_authent.php
//vérifie que une session est bien en cours
//c'est a dire qu'un login valide a eu lieu auparavant,
//s'il n'y a pas de session en cours affiche le formulaire de login
//(possiblement preremplí)
//et termine la page courante
function bloquer_sans_session($login) {
    if (! isset($_SESSION["user"])) {
        page_login($login); //formulaire preremplí avec $login, si dispo.
        exit;
    }
}
```

Protéger l'accès à un site par login avec les sessions

- Remarque : `exit` termine le script, pas uniquement la fonction

  ⇒ `bloquer_sans_session()` peut être appelé par chaque page avant d'afficher le contenu protégé

```
//page1.php
<?php
require_once("fonc_authent.php");
require_once("fonc_sortie.php");

session_start(); //avant tout HTML, pour accéder aux var. de session
bloquer_sans_session($login); //$login=undefined pour le formulaire
page1($_SESSION['user']); //affiche le contenu de la page 1
?>
```

- page2.php, page3.php : similaires

Protéger l'accès à un site par login avec les sessions

- La page principale (home.php)

  différente des autres pages parce que, en plus de protéger son contenu, elle gère le formulaire de login

  - Si la page est demandée avec une demande de login valide
    - effectue le login en créant `$_SESSION['user']`

  - Ensuite la page est protégée comme toutes les autres :
    - appelle `bloquer_sans_session()` pour vérifier qu'une session est bien en cours (créé par la phase de login, ou déjà en cours)

Protéger l'accès à un site par login avec les sessions

- La page principale

```
//home.php
<?php require_once("fonc_authent.php");
require_once("fonc_sortie.php");
session_start();

$login = $_POST["login"];
$mdp = $_POST["mdp"]; //provenants du formulaire de login

tenter_login($login, $mdp); //cree $_SESSION["user"] si login valide
bloquer_sans_session($login);

afficher_home($_SESSION['user']); //affiche le contenu de la page home
?>
```

Protéger l'accès à un site par login avec les sessions

- La phase de login :

```
//fonc_authent.php
//affecte une variable de session "user" pour l'utilisateur $login,
// si login possible
//(identifiants fournis et valides et utilisateur pas deja loggué)

function tenter_login ($login, $mdp){
     if (isset($login) && login_valid ($login, $mdp) &&
                                          !isset($_SESSION[“user"]) )
          $_SESSION["user"] = $login;
          //transfère aux autres pages l'info "login réussi"
}
```

Protéger l'accès à un site par login avec les sessions

- La déconnexion
  - toutes les pages affichent, avec le nom de l'utilisateur authentifié, un lien vers une page de déconnexion (logout.php)

```
//logout.php
<?php
require_once("fonc_authent.php");
require_once("fonc_sortie.php");
session_start();
$login = $_SESSION["user"];
if (isset($login)) {
      $_SESSION= array();
      session_destroy();
}
afficher_page_bye($login);
?>
```

le code complet

04-login-site/fonc_authent.php
/fonc_sortie.php
/home.php, page1,2,3.php
/logout.php

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 serialisés

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

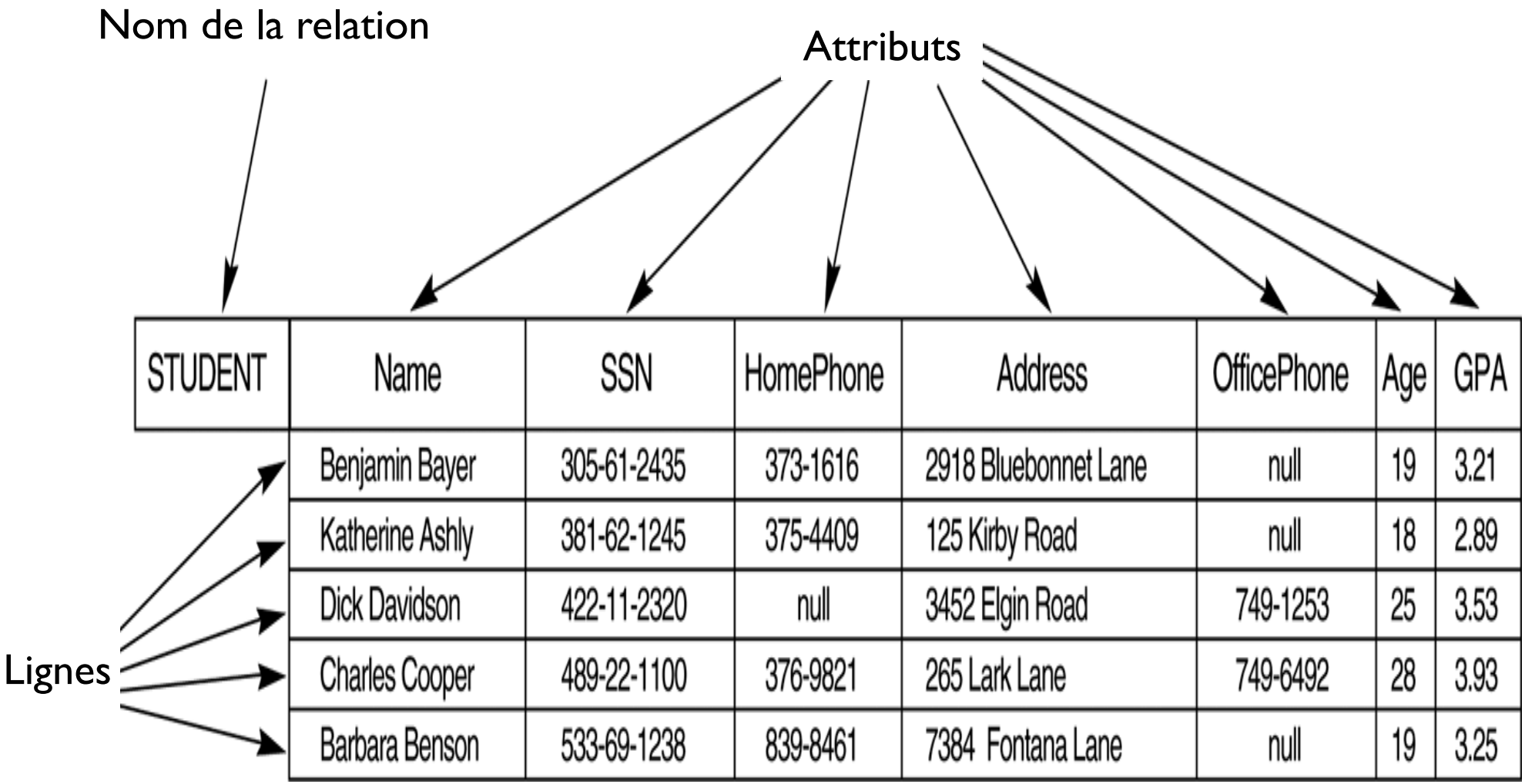
Modèle relationnel des données

- 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