Université Paris-Saclay
M2 DCN & DIPI

Approche de l'élaboration et du fonctionnement des logiciels

Alain Delaët

http://alain.delaet.info

web/bases de données

1. Web
   - 1.1 le protocole HTTP
   - 1.2 JavaScript
   - 1.3 sécurisation des communications
2. bases de données
   - 2.1 modèle relationnel
   - 2.2 le langage SQL

le Web

le **Web** (abréviation de l'anglais **World Wide Web**), ce sont des documents reliés entre eux par des **liens hypertexte**
par extension, le Web désigne

- les langages informatique permettant d'écrire les documents hypertextes, à savoir **HTML** et **CSS**
- une architecture client-serveur utilisant le protocole **HTTP** (*Hypertext Transfer Protocol*) ou sa version sécurisée **HTTPS** (*HTTP Secure*)
- des langages de programmation côté serveur, comme PHP
- des langages de programmation côté client, comme JavaScript

le protocole HTTP

un réseau informatique global conçu en 1970
fonctionne grâce à un ensemble de protocoles, organisés en quatre couches

nº	nom	exemple de protocole
4	application	HTTP, POP, IMAP
3	transport	TCP, UDP
2	réseau	IPv4, IPv6, ICMP
1	liaison	Ethernet, Wi-Fi

on accède à un site Web au moyen d'une adresse appelée **URL** (pour *Uniform Resource Locator*), de la forme

protocole`://`*nom-ou-adresse*`/`*document*

le protocole peut être `http` ou `https`
exemple : `https://www.ined.fr/index.html`

le protocole **HTTP** (pour *HyperText Transfert Protocol*) est un protocole de type client-serveur

- le **serveur Web** attend des messages via des connexions réseaux TCP et y répond
- les clients sont les **navigateurs Web**

les messages envoyés par le client sont appelés des **requêtes** ;
ceux envoyés par le serveur sont appelés des **réponses**

1. le navigateur Web isole le nom de machine `www.ined.fr`
2. le navigateur Web effectue une **requête DNS** pour obtenir l'adresse IP du serveur Web hébergeant le site (par exemple `193.49.36.4`).
3. le navigateur Web se connecte à la machine dont l'adresse IP est `193.49.36.4`, en utilisant le protocole TCP, sur le **port** 80
4. une fois la connexion établie, le navigateur Web demande la ressource `index.html`
5. le serveur Web se trouvant à l'adresse `193.49.36.4` envoie en réponse le contenu du fichier
6. le navigateur Web peut ensuite parcourir le fichier et afficher la page correspondante

la requête envoyée par le navigateur est un texte de la forme

```
GET /index.html HTTP/1.1
Host: www.ined.fr
```

ici, le navigateur indique

- qu'il veut communiquer en utilisant la version 1.1 du protocole HTTP
- qu'il demande la ressource `/index.html`

la réponse du serveur est alors :

```
HTTP/1.1 200 OK
Date: Wed, 22 Feb 2023 10:32:05 GMT
Server: Apache
Content-Type: text/html; charset=utf-8

<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="utf-8"/>
  ...
</head>
<body>
  ...
```

si en revanche la page n'existe pas, alors la réponse est

```
HTTP/1.1 404 Not Found
Date: Wed, 22 Feb 2023 10:35:43 GMT
Server: Apache
Content-Type: text/html; charset=utf-8

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>404 Not Found</title>
  </head>
  <body>
    <center>
      <h1>404 Not Found</h1>
      ...
```

l'URL peut inclure des **paramètres** de requête

$$\textit{proto}://\textit{nom-ou-adresse}:\textit{port}/\textit{document}?n_1=v_1\&\ldots\&n_k=v_k$$

exemple :
`https://fr.wikipedia.org/w/index.php?search=Informatique`

sur une requête comme

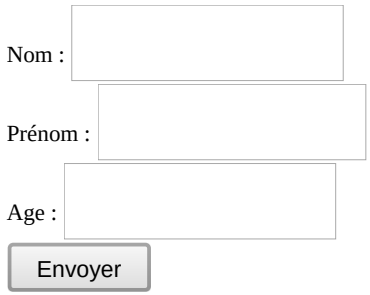
`https://fr.wikipedia.org/w/index.php?search=Informatique`

le serveur **exécute un programme** qui écrit la réponse HTML

- ici, il s'agit du programme `index.php` en langage PHP
- ce programme dispose des paramètres (ici `search` qui vaut `"Informatique"`) et peut produire une réponse qui dépend de ces paramètres, par exemple en consultant/modifiant une base de données

les paramètres peuvent être passés via des **formulaires**

```
<form action="https://www.nsi-premiere.fr/form">
  Nom : <input type="text" name="nom" />
  Prénom : <input type="text" name="prenom" />
  Age : <input type="text" name="age" />
  <button type="submit">Envoyer</button>
</form>
```



en cliquant sur le bouton, on crée la requête

`https://www.nsi-premiere.fr/form?nom=...&prenom=...&age=...`

(avec les valeurs saisies) dont le résultat est du HTML utilisant ces paramètres, par exemple

- La valeur du nom est **Lovelace**
- La valeur du prénom est **Ada**
- La valeur de l'âge est **36**

un site Web peut vouloir identifier ses clients et les différencier
l'adresse IP du client n'est pas une solution pour cela, car elle peut changer, voire être partagée par plusieurs clients
pour différencier leurs clients, les sites Web utilisent des **cookies**

un **cookie** est une petite quantité de donnée, comportant

- un nom
- une valeur
- optionnellement une date d'expiration

choisis et envoyés par le serveur
exemple :

```
HTTP/1.1 200 OK
Server: nginx/1.10.3 (Ubuntu)
Date: Mon, 29 May 2019 17:24:32 GMT
Content-Type: text/html
Content-Length: 1023
Set-Cookie: test=42; Expires=Mon, 30 May 2019 18:10:12 GMT;
...
```

le navigateur **stocke** le cookie, en l'associant au nom de domaine du site visité
lorsque le navigateur effectue une nouvelle requête, possiblement longtemps après, il **renvoie** le cookie au serveur dans sa requête, si la date d'expiration n'est pas dépassée

```
GET /page.php HTTP/1.1
Host: www.site.fr
Cookie: test=42;
```

ainsi, la page renvoyée peut être personnalisée par le serveur

un navigateur peut envoyer un cookie **uniquement** au domaine qui l'a déposé initialement, ceci pour interdire à des sites tiers d'espionner les utilisateurs
mais cela peut être contourné avec les **cookies de tierce partie** ou cookies tiers

- le site `www.news.com` diffuse des articles variés
- sous chaque article, il y a un bouton « partager cet article sur `reseau-social.com` »
- le bouton est une ressource du type `https://www.reseau-social.com/bouton?origin=123`
- si on clique sur le bouton
  - le site `reseau-social.com` peut déposer un cookie
  - il peut savoir quelle URL a été utilisée pour la requête
  - il peut stocker toutes ces informations dans le cookie
- lorsqu'on va plus tard sur `reseau-social.com`, le site peut relire son cookie et afficher de la publicité en rapport avec l'article lu

interdire les cookies tiers

les navigateurs permettent de limiter cette pratique avec l'option

interdire les cookies de tierce partie

le navigateur refusera alors les cookies venant d'un domaine qui n'est pas celui affiché dans la barre d'adresse de l'URL

depuis 2018, le **RGPD** (règlement général sur la protection des données, du Parlement Européen) encadre l'utilisation des cookies
l'utilisateur doit donner son consentement explicite et positif sur l'utilisation qu'un site fait des cookies, par catégories
ainsi, on peut par exemple

- n'accepter que les cookies techniques
- refuser les cookies de publicité, télémétrie, etc.

les sites se servent notamment des cookies pour implémenter des **sessions HTTP**, c'est-à-dire un ensemble de requêtes où un utilisateur

- s'authentifie
- navigue plus ou moins longtemps sur le site
- se déconnecte

pendant toute la session, le serveur conserve des données relatives à l'utilisateur, dans une base de données
la clé est un cookie particulier (par exemple `ID`)

- un site contenant un formulaire d'authentification ne fonctionne pas si les cookies sont désactivés
- si on s'authentifie sur un site avec un navigateur, puis qu'on le visite avec un second navigateur, on ne sera pas authentifié sur le second, car les cookies sont stockés par le navigateur
- si on utilise un ordinateur qui n'est pas le sien, on veut probablement effacer les cookies

le mode de **navigation privée** du navigateur supprime, lors de la fermeture de la fenêtre, tous les cookies qui ont été créés depuis le démarrage de la navigation privée, même s'ils n'ont pas expiré

JavaScript

avec ce qu'on a vu, l'interaction avec un site se fait

- par l'envoi de paramètres dans les requêtes HTTP
- par l'envoi de pages en retour

c'est un cycle lent, notamment à cause des aller-retours via le réseau
par exemple : communication en antartique
JavaScript est une réponse à ce problème, en calculant **côté client**, c'est-à-dire dans le navigateur

JavaScript est un langage de programmation, créé en 1995 et standardisé en 1997
un navigateur Web inclut un interprète JavaScript
le langage JavaScript permet notamment

- d'associer des actions à des événements (cliquer sur un bouton, passer la souris sur une zone de la page, appuyer sur une touche, etc.)
- de modifier la structure du document HTML

```
<html>
  <head><title>Age</title>
  <script type="text/javascript">
    let compteur = 0;
    function suivant () {
      compteur = compteur + 1;
      let v = document.getElementById("valeur");
      v.innerHTML = compteur;
    }
 </script>
 </head>
 <body>
   <button onclick="suivant();">Clic!</button>
   <span id="valeur">0</span>
 </body>
</html>
```

- l'attribut `onclick` du bouton est un morceau de code JavaScript (l'appel de la fonction `suivant`)
- le code de cette fonction est défini dans l'élément `<script>`
- ce code incrémente la variable globale `compteur`, puis **modifie le document HTML**, ici pour changer le texte de l'élément `<span>` identifié par `id="valeur"`

sécurisation des communications

toute l'activité sur le Web, et notamment des informations sensibles (banque, impôts, santé, etc.), transite sous la forme de paquets IP passant de machines en machines
ces paquets sont lisibles par tout le monde

- comment sécuriser le contenu des communications, afin qu'il ne soit lisible que par la source et la destination ?
- comment garantir que le serveur auquel on se connecte est bien celui de la personne et ou de l'entité auquel on pense se connecter ?
- comment garantir les deux propriétés ci-dessus en réutilisant l'infrastructure d'Internet, à savoir les communications TCP/IP ?

la **cryptographie** nous offre des outils permettant de **chiffrer** et **signer** des messages de façon sûre

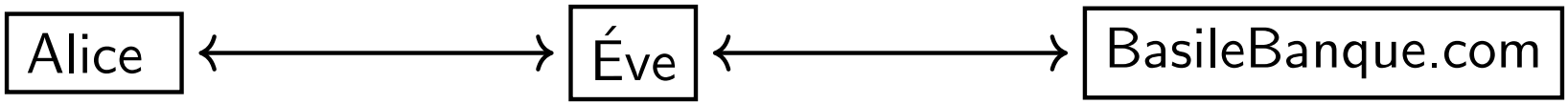
- celui qui possède la **clé de chiffrement** peut aisément déchiffrer un message ou valider une signature
- à l'inverse, celui qui ne la connaît pas n'est pas en mesure de le faire, même avec une grosse capacité de calcul

Alice reçoit un mail au format HTML, contenant

```
<html>
 ...
   <body>
      Veuillez vous connecter d'urgence au site
      <a href="http://eve.com">BasileBanque.com</a>.
   </body>
</html>
```

le lien s'affiche comme `BasileBanque.com` et Alice clique sur le lien

Alice pense communiquer avec sa banque, mais elle communique en réalité avec Ève, même si les communications sont chiffrées !



on appelle cela l'**attaque de l'homme du milieu** (en anglais *man in the middle attack*)

un site qui veut prouver son identité présente un **certificat**
ce dernier doit être délivré par une autre entité, en laquelle les deux participants ont confiance, que l'on appelle un **tiers de confiance**

une **autorité de certification** (AC) est une entité habilitée à délivrer des certificats
parmi les différentes AC, on trouve

- des entreprises spécialisées
- des associations à but non lucratif, comme l'initiative **Let's Encrypt**, libre et gratuite
- des états

les navigateurs sont capables de vérifier les certificats
les développeurs des navigateurs reconnaissent les autorités de certification, chacune d'elles répondant à des critères très stricts, vérifiés par des audits réguliers dont les résultats sont publics

présence et examen du certificat

Lecteur du certificat : discuss.ocaml.org

Général Détails

Émis pour

Nom commun (CN)	discuss.ocaml.org
Organisation (O)	<Ne fait pas partie du certificat>
Unité d'organisation (OU)	<Ne fait pas partie du certificat>

Émis par

Nom commun (CN)	R3
Organisation (O)	Let's Encrypt
Unité d'organisation (OU)	<Ne fait pas partie du certificat>

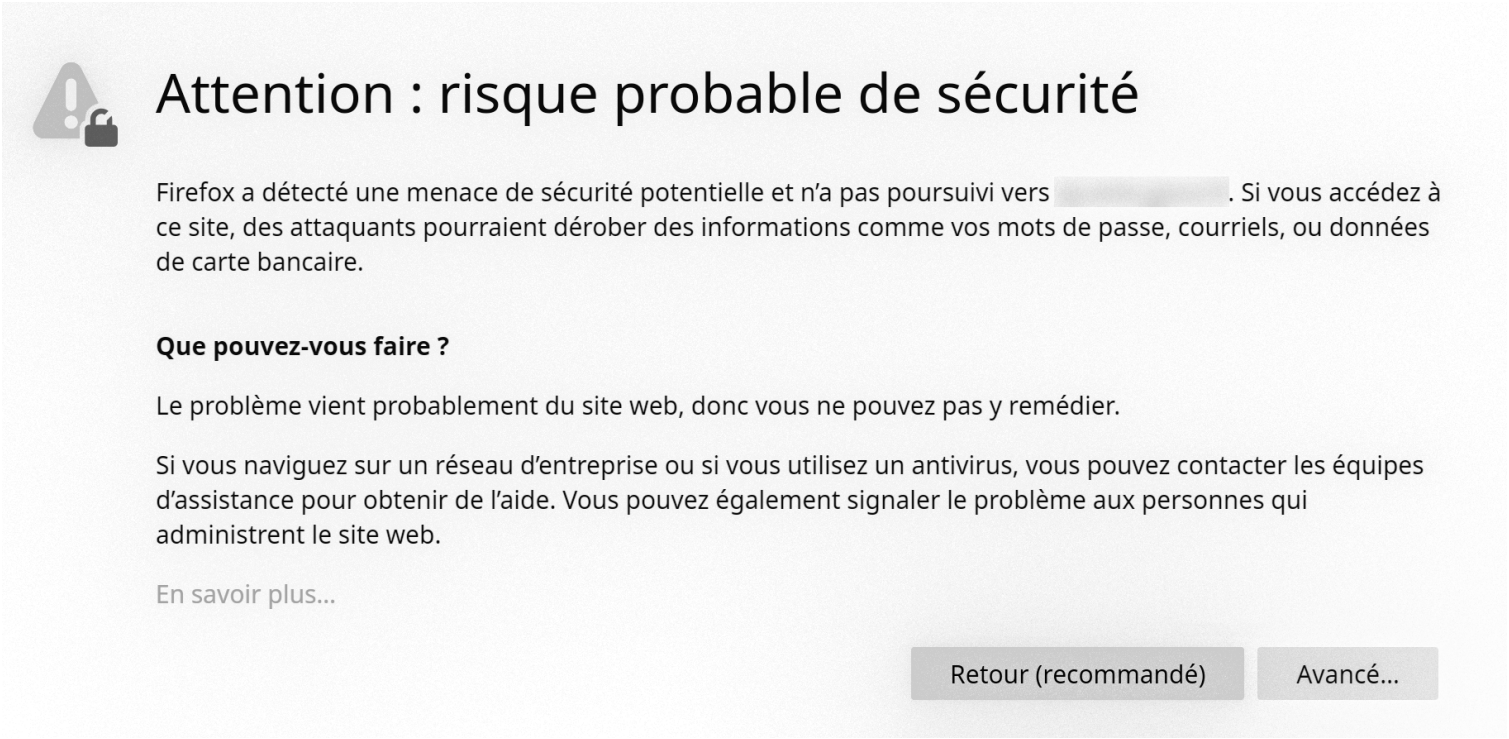
Durée de validité

Émis le	jeudi 5 janvier 2023 à 01:00:22
Expire le	mercredi 5 avril 2023 à 02:00:21

Empreintes

Empreinte SHA-256	15 E0 C2 F2 3A DF 63 43 BC A8 56 6A 02 CB 32 A6 C7 C2 77 89 2F 7A D6 23 17 51 51 AA 88 3E 94 5F
Empreinte SHA-1	28 FB 25 A1 85 39 BC 87 77 12 03 00 E7 BD 90 55 64 61 F5 E6

les navigateurs indiquent les certificats non valables (par ex. expirés)



dans ce cas, la communication est chiffrée, mais pas authentifiée

considérons une médiathèque où des usagers peuvent emprunter des livres

- chaque usager a une carte avec un code barre
- des bornes permettent de scanner une carte puis d'emprunter/rendre un livre

un **système d'information** est un système informatique et humain permettant de gérer de l'information
en particulier, le système informatique contient une description

- d'objets (les livres)
- de personnes physiques (les usagers)
- de processus (emprunt et restitution de livres)

c'est une approximation de la réalité

- livres : seulement le titre, les auteurs, l'éditeur et l'ISBN
- usagers : seulement nom, prénom, adresse, email

on appelle cela une **modélisation**

outre le stockage des données (livres, usagers, emprunts), le système doit **garantir**

- qu'un livre rendu peut être emprunté de nouveau
- qu'un livre non encore rendu ne peut être emprunté

le système doit aussi permettre des **niveaux d'accès**

- les documentalistes peuvent ajouter / retirer des livres
- les usagers ne le peuvent pas

enfin, si la médiathèque offre plusieurs bornes pour emprunter / restituer les livres, ce ne sont que des **clients** et les données sont stockées dans un **même endroit**

le modèle relationnel

dans le **modèle relationnel** (Edgar F. Codd, 1970), un objet est modélisé par un *n*-uplet de valeurs scalaires
exemple : un livre modélisé avec des chaînes de caractères et un entier

```
('La Programmation en pratique', 'B. Kernighan et R. Pike',
 'Vuibert', 2017, '978-2711786701')
```

on appelle cela une **entité**

l'ensemble des livres de la médiathèque est alors représenté par **un ensemble** de tels n-uplets :

$$
\begin{aligned}
\mathit{Livre} = \{ & \\
& (\texttt{'La Programmation en pratique'}, \texttt{'B. Kernighan, R. Pike'}, \\
& \texttt{'Vuibert'}, 2017, \texttt{'978-2711786701'}), \\
& (\texttt{'Le Bouclier Arverne'}, \texttt{'René Goscinny, Albert Uderzo,} \\
& \texttt{'Dargaud'}, 1968, \texttt{'978-2012101432'}), \\
& (\texttt{'Zadig et autres contes'}, \texttt{'Voltaire'}, \texttt{'Folio'}, \\
& 1992, \texttt{'978-2070384815'}), \\
& \quad \vdots \\
\} &
\end{aligned}
$$

on appelle cela une **relation**

une autre relation est l'ensemble des livres actuellement empruntés :

$$
\begin{aligned}
&\mathit{Emprunt} = \{ \\
&\quad (\texttt{'La Programmation en pratique'}, \texttt{'Alice'}, 01/04/2020), \\
&\quad (\texttt{'Le Bouclier Arverne'}, \texttt{'Charlie'}, 15/05/2020), \\
&\quad (\texttt{'Le Guide du routard galactique'}, \texttt{'Charlie'}, 08/04/2020), \\
&\qquad \vdots \\
&\}
\end{aligned}
$$

chaque relation se conforme à un **schéma**, qui décrit chaque composante des n-uplets par

- son nom
- son domaine (une chaîne, un entier, etc.)

exemple :

Livre(*titre* `String`, *auteur* `String`, *éditeur* `String`, *année* `Int`, *isbn* `String`)

une **base de données** est un **ensemble de relations**

Livre(*titre* `String`, *auteur* `String`, *éditeur* `String`, *année* `Int`, *isbn* `String`)
Usager(*nom* `String`, *prénom* `String`, *code_barre* `String`)
Emprunt(*titre* `String`, *code_barre* `String`, *date* `String`)

à ce niveau-là, on pourrait imaginer réaliser la base de données comme trois feuilles Excel

titre	auteurs	éditeur	année	ISBN
La Programmation...	B. Kernighan...	Vuibert	2017	978-271178
Le Bouclier Arverne	René Goscinny...	Dargaud	1968	978-201210
Zadig et autres...	Voltaire	Folio	1992	978-207038
...				

nom	prénom	code barre
Avril	Alice	767627683
Mense	Charlie	418346761
...		

titre	code barre	date
La Programmation...	767627683	01/04/2020
Le Bouclier Arverne	418346761	15/05/2020
Le Guide du...	418346761	08/04/2020
...		

ou encore avec des tableaux Python

```
livre = [('La Programmation...', 'B. Kernighan...',
'Vuibert', 2017, '978-2711786701'),
('Le Bouclier Arverne', 'René Goscinny...',
'Dargaud', 1968, '978-2012101432'),
('Zadig et autres...', 'Voltaire',
'Folio', 1992, '978-2070384815'),
...]
usager = [('Avril', 'Alice', 767627683),
('Mense', 'Charlie', 418346761),
...]
emprunt = [('La Programmation...', 767627683, '01/04/2020'),
('Le Bouclier Arverne', 418346761, '15/05/2020'),
('Le Guide du...', 418346761, '08/04/2020'),
...]
```

représenter ainsi notre base de données par des feuilles Excel ou des tableaux Python serait toutefois un peu **naïf**, car

- certaines opérations seraient **inefficaces**, comme par exemple
  - rechercher tous les emprunts d'une même personne
  - rechercher les emprunts en retard
- il serait difficile de maintenir la **cohérence** des données, par exemple
  - un emprunt mentionne un livre qui existe et un usager qui existe
  - un même livre n'est emprunté qu'une seule fois

une **contrainte d'intégrité** est une propriété logique, préservée à tout instant par la base de données, qui garantit la **cohérence des données**

chaque usager est identifié de manière unique par le code barre de sa carte
on le note comme ceci :

$$\textit{Usager}(\textit{nom}\ \texttt{String}, \textit{prénom}\ \texttt{String}, \textit{adresse}\ \texttt{String}, \textit{cp}\ \texttt{String}, \textit{ville}\ \texttt{String}, \textit{email}\ \texttt{String}, \underline{\textit{code\_barre}\ \texttt{String}})$$

cela signifie que, dans la relation *Usager*, chaque entité a un code barre **unique** ; on appelle cela une **clé primaire**
on garantit ainsi que deux usagers ne peuvent avoir le même code barre

pour les livres, on peut utiliser le numéro ISBN comme clé primaire :

Livre(*titre* `String`, *auteur* `String`, *éditeur* `String`, *année* `Int`, *isbn* `String`)

les clés primaires peuvent servir de **références** dans d'autres relations
exemple :

$$\mathit{Emprunt}(\underline{\mathit{code\_barre}}\ \texttt{String}, \underline{\mathit{isbn}}\ \texttt{String}, \mathit{retour}\ \texttt{Date})$$

ici, *code_barre* et *isbn* sont des **clés étrangères**
par ailleurs, *isbn* est aussi la clé primaire de la relation *Emprunt*, ce qui garantit qu'un même livre ne peut être emprunté qu'une seule fois

un même personne peut emprunter plusieurs livres car la clé étrangère *code_barre* n'a pas à être unique dans la relation *Emprunt*

maintenant qu'on connaît le principe des clés primaires et étrangères, on peut proposer une meilleure modélisation des auteurs

1. on ne mentionne plus les auteurs dans les livres

$$\mathit{Livre}(\mathit{titre}\ \texttt{String}, \mathit{éditeur}\ \texttt{String}, \mathit{année}\ \texttt{Int}, \underline{\mathit{isbn}\ \texttt{String}})$$

2. on crée une nouvelle relation pour les auteurs

$$\mathit{Auteur}(\underline{\mathit{a\_id}\ \texttt{Int}}, \mathit{nom}\ \texttt{String}, \mathit{prénom}\ \texttt{String})$$

   (où *a_id* est un identifiant unique d'auteur)

3. on associe des auteurs à des livres avec une troisième relation

$$\mathit{Auteur\_de}(\underline{\mathit{a\_id}\ \texttt{Int}, \mathit{isbn}\ \texttt{String}})$$

avec cette nouvelle modélisation,

- un même auteur **et** un même livre n'apparaissent qu'une seule fois dans la relation *Auteur_de* : un même auteur ne peut pas être mentionné deux fois pour le même livre
- un même auteur peut être mentionné pour des livres différents
- un livre peut avoir plusieurs auteurs

attention, rien n'empêche en revanche qu'un livre n'ait aucun auteur

le modèle relationnel est mis en œuvre dans des **systèmes de gestion de bases de données** (SGBD)
quelques exemples :

- Oracle Database (Oracle Corporation)
- DB2 (IBM)
- Microsoft SQL Server
- MySQL, aujourd'hui MariaDB (logiciel libre)
- PostgreSQL (logiciel libre)

les SGBD utilisent presque tous le **langage SQL**

le langage SQL

SQL (*Structured Query Language*) est un langage déclaratif permettant notamment de

- créer des relations
- ajouter / supprimer des données
- récupérer des données selon des critères particuliers (**requête**)

SQL est défini par le standard ISO/IEC 9075

langage déclaratif

on dit que le langage SQL est **déclaratif** car on exprime **ce que l'on veut faire** mais pas comment le faire
c'est le SGBD qui choisit des structures de données et des algorithmes pour mettre en œuvre les ordres SQL

```
CREATE TABLE usager (
    nom VARCHAR(90), prenom VARCHAR(90),
    adresse VARCHAR(300), cp VARCHAR(5),
    ville VARCHAR(60), email VARCHAR(60),
    code_barre CHAR(15) PRIMARY KEY);

CREATE TABLE livre (
    titre VARCHAR(300),
    editeur VARCHAR(90),
    annee INT,
    isbn CHAR(14) PRIMARY KEY);

CREATE TABLE auteur (
    a_id INT PRIMARY KEY,
    nom VARCHAR(90),
    prenom VARCHAR(90));
```

```
INSERT INTO auteur VALUES
    (97, 'Ritchie', 'Dennis'),
    (98, 'Voltaire', ''),
    (103, 'Toriyama', 'Akira');
```

en particulier, le SGBD assure les **contraintes d'intégrité** :

```
# INSERT INTO auteur (97, 'Dumas', 'Alexandre');
ERROR:  duplicate key value violates unique constraint
"auteur_pkey"
DETAIL:  Key (a_id)=(97) already exists.
```

si on essaye de supprimer la table `usager`, on obtient une erreur :

```
# DROP TABLE usager;

ERROR:  cannot drop table utilisateurs because other
  objects depend on it
DETAIL:  constraint emprunt_code_barre_fkey on table
  emprunt depends on table usager
```

il faudrait d'abord supprimer la table `emprunt`
là encore, le SGBD garantit les contraintes d'intégrité

maintenant que les tables sont créées (avec `CREATE`) et remplies (avec `INSERT`), on souhaite pouvoir répondre à des questions comme

- étant donné un code barre, quels sont les livres empruntés par l'utilisateur correspondant ?
- étant donné un ISBN, le livre correspondant est-il emprunté ?
- quels sont les utilisateurs en retard, c'est-à-dire ceux dont la date de retour est inférieure à une date donnée ?
- quels sont tous les livres écrits par Voltaire qui ne sont pas empruntés ?
- quel est le nombre total de livres empruntés ?

on le fait avec des **requêtes SQL**

avec la requête

```
SELECT titre FROM livre WHERE annee >= 1990;
```

on demande « les titres des livres édités à partir de 1990 »

le résultat est une **table**, ici avec une seule colonne

titre
Les Aventures de Huckleberry Finn
Fondation et Empire
Akira
Les Robots
Astérix chez les Pictes
Les Monades urbaines
Les Voyages de Gulliver
Lolita
La Nuit des temps
Ravage
⋮ (112 résultats)

la forme générale est

`SELECT` *colonnes* `FROM` *table* `WHERE` *condition*

la condition peut être plus complexe

```
SELECT titre FROM livre
WHERE annee >= 1970 AND
      annee <= 1980 AND
      editeur = 'Dargaud';
```

titre
Astérix chez les Belges

la condition peut inclure un **motif**, par exemple pour chercher les titres contenant le mot `Astérix`

```
SELECT titre FROM livre WHERE titre LIKE '%Astérix%';
```

titre
Astérix et les Normands
Le Tour de Gaule d'Astérix
Astérix chez les Pictes
Astérix chez les Belges
Astérix et Cléopâtre
Astérix légionnaire
Astérix et la Transitalique
Astérix chez les Bretons
Astérix en Corse
L'Odyssée d'Astérix

on peut spécifier plusieurs colonnes

```
SELECT titre, isbn FROM livre WHERE annee >= 1990;
```

titre	isbn
Les Aventures de Huckleberry Finn	978-2081509511
Fondation et Empire	978-2207249123
Akira	978-2723428262
Les Robots	978-2745989857
Astérix chez les Pictes	978-2864972662
Les Monades urbaines	978-2221197691
Les Voyages de Gulliver	978-2335008586
Lolita	978-0141391601
La Nuit des temps	978-2258116429
Ravage	978-2072534911
⋮ (112 résultats)	

et les renommer

```
SELECT titre AS le_titre, isbn AS num_serie
FROM livre
WHERE annee >= 1990;
```

le_titre	num_serie
Les Aventures de Huckleberry Finn	978-2081509511
Fondation et Empire	978-2207249123
Akira	978-2723428262
⋮ (112 résultats)	

on peut sélectionner toutes les colonnes avec *

```
SELECT * FROM livre WHERE annee >= 1990;
```

titre	editeur	annee	isbn
Les Aventures de H. . .	Flammarion	2020	978-2081509511
Fondation et Empire	Éditions Denoël	1999	978-2207249123
Akira	Glénat	2000	978-2723428262
Les Robots	Éditions Milan	2017	978-2745989857
Astérix chez les Pictes	Éditions Alb. . .	2013	978-2864972662
Les Monades urbaines	Robert Laffont	2016	978-2221197691
Les Voyages de Gulliver	Primento	2015	978-2335008586
Lolita	Penguin UK	2012	978-0141391601
La Nuit des temps	Presses de la Cité	2014	978-2258116429
Ravage	Éditions Gallimard	2014	978-2072534911
⋮ (112 résultats)			

enfin, on peut appliquer des **fonctions d'agrégation** à l'ensemble des valeurs du résultat de la requête

- `COUNT` : le nombre
- `AVG` : la moyenne
- `SUM` : la somme
- `MIN` : la valeur minimale
- `MAX` : la valeur maximale

```
SELECT COUNT(titre) AS total FROM livre
WHERE titre LIKE '%Astérix%';
```

total
10

note : on peut aussi écrire `COUNT(*)`

```
SELECT MIN(annee) AS inf FROM livre;
SELECT MAX(annee) AS sup FROM livre;
```

inf	sup
1933	2020

on le fait en ajoutant `ORDER BY`

```
SELECT titre FROM livre
WHERE annee >= 1990
ORDER BY titre ASC
```

titre
Akira
Algorithms
Anna Karénine
Astérix chez les Bretons
Astérix chez les Pictes
Astérix en Corse
Astérix et Cléopâtre
Astérix et la Transitalique
⋮ (112 résultats)

(`ASC` pour l'ordre croissant / `DESC` pour l'ordre décroissant)

```
SELECT DISTINCT annee FROM livre;
```

annee
2020
1999
2000
2017
⋮
(38 résultats)

```
SELECT COUNT(DISTINCT annee) FROM livre;
```

count(distinct annee)
38

jointures

supposons que l'on veuille les titres et dates de retour de tous les livres actuellement empruntés ?
il n'y a pas de moyen simple de l'obtenir car

- le titre est dans la table `livre`
- la date est dans la table `emprunt`

on veut **croiser** les tables `livre` et `emprunt`
en SQL, on appelle cela une **jointure**

```
SELECT *
FROM livre JOIN emprunt ON livre.isbn = emprunt.isbn;
```

titre	editeur	annee	isbn	code_	isbn	retour
Mrs Dalloway	Flammarion	2015	978-...	421...	978-...	2020-04-28
Fondation...	Denoël	1999	978-...	654...	978-...	2020-03-17
Le Journal...	Bibebook	2013	979-...	346...	979-...	2020-04-21
⋮ (17 résultats)						

ici,

- a on croisé les deux tables `livre` et `emprunt`, c'est-à-dire considéré **toutes les paires** *livre*/*emprunt* (`JOIN`)
- mais **sous la condition** que la valeur de `isbn` soit identique dans le livre et dans l'emprunt (`ON`)

on obtient la requête que l'on voulait en ne conservant que deux champs

```
SELECT livre.titre, emprunt.retour
FROM livre
JOIN emprunt ON livre.isbn = emprunt.isbn;
```

(noter comment on préfixe avec le nom des tables)

le_titre	num_serie
Mrs Dalloway	2020-04-28
Fondation et Empire	2020-04-28
Le Journal d'un fou	2020-04-28
Guerre et Paix	2020-02-20
Les Voyages de Gulliver	2020-02-20
⋮ (17 résultats)	

on pourrait de plus trier avec `ORDER BY`

si on veut en plus le nom de l'emprunteur, il faut aussi croiser avec la table `usager`

```
SELECT usager.nom, usager.prenom, livre.titre, emprunt.retour
FROM emprunt
JOIN livre ON emprunt.isbn = livre.isbn
JOIN usager ON usager.code_barre = emprunt.code_barre;
```

nom	prenom	titre	retour
PETIT	SÉBASTIEN	Anna Karénine	2020-01-01
SIMON	SANDRINE	Astérix et la Transitalique	2020-02-17
MOREAU	ALAIN	Fondation et Empire	2020-04-28
⋮		(17 résultats)	

et par ailleurs, on peut filtrer avec `WHERE`

```
SELECT livre.titre, emprunt.retour
FROM emprunt
JOIN livre ON emprunt.isbn = livre.isbn
WHERE emprunt.retour < '2020-02-01';
```

titre	retour
La Planète des singes	2020-01-01
Anna Karénine	2020-01-01

on a ici deux conditions :

- `ON` pour la jointure
- `WHERE` pour filtrer les résultats

requêtes imbriquées

le résultat de `SELECT` **est une table**, sur laquelle on peut donc effectuer de nouveau une requête
on appelle cela une **requête imbriquée**

`SELECT ... FROM (` *requête* `) AS` *nom* `WHERE ...`

```
SELECT t.titre, t.annee
FROM (SELECT * FROM livre WHERE annee >= 2000) AS t
WHERE t.titre LIKE 'Les %';
```

titre	annee
Les Aventures de Huckleberry Finn	2020
Les Robots	2017
Les Monades urbaines	2016
⋮ (14 résultats)	

(même si ici on pouvait le faire plus simplement avec `AND`)

dans le cas où la requête a un **unique résultat** (une seule case), on peut l'utiliser comme une valeur scalaire dans une expression
exemple : la requête

```
SELECT MAX(annee) FROM livre;
```

renvoie une seule valeur

MAX(annee)
2020

et on peut donc l'utiliser dans une expression

tous les livres édités la même année que le livre dont la date d'édition est la plus récente

```
SELECT titre FROM livre
WHERE annee = (SELECT MAX(annee) FROM livre)
```

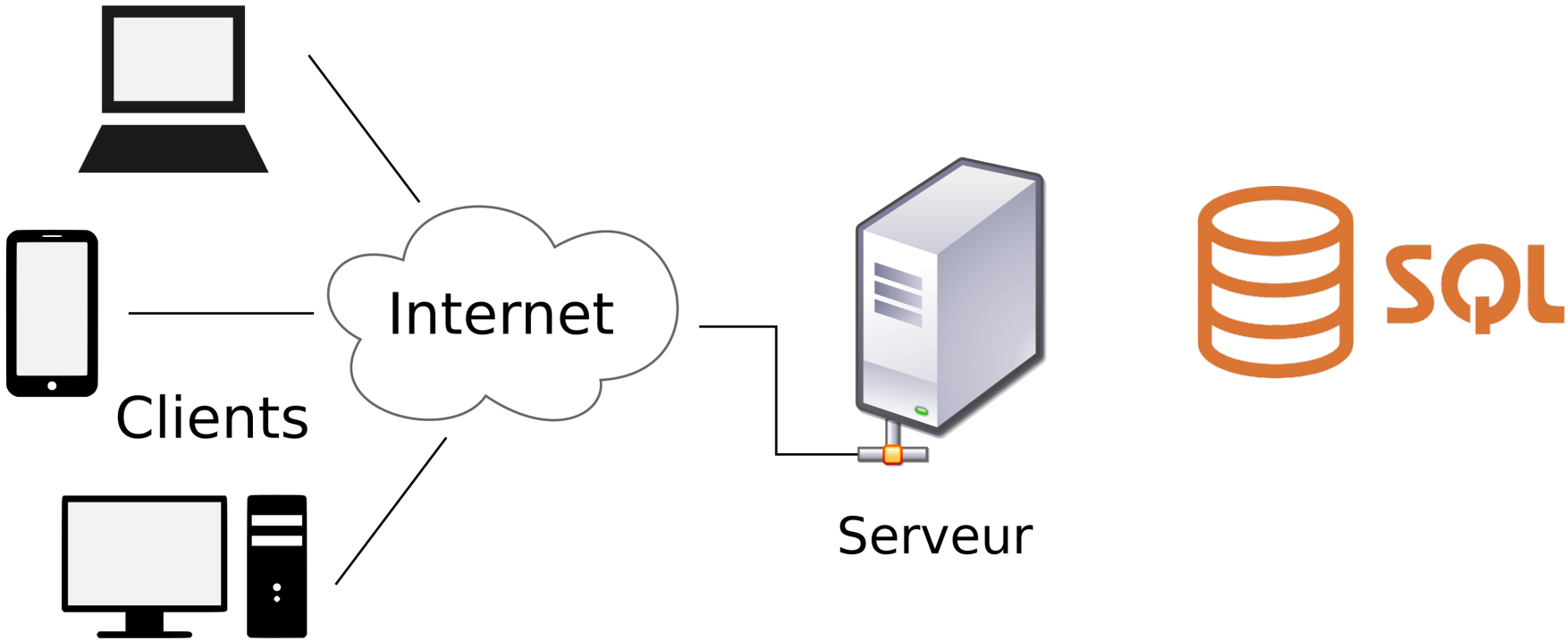
titre
Les Aventures de Huckleberry Finn
Romancero gitano

interaction entre un programme et un SGBD

un SGBD peut être utilisé depuis un programme informatique (par exemple en Python)
le programme

1. se connecte au SGBD
2. envoie des ordres SQL (par ex. des chaînes de caractères)
3. récupère les résultats (par ex. dans des tableaux Python)
4. exécute du code Python sur les données

un exemple typique est un serveur, par exemple un serveur web



exemples : eCampus, Wikipedia, votre réseau social préféré, etc.

ça ressemble à ça :

```
import psycopg2 as sgbd

cnx = sgbd.connect(host="localhost", user="alice", \
password="secret")

c = cnx.cursor()

c.execute("SELECT * FROM livre WHERE annee < 1990")

for l in c.fetchall():
    print(l[0], l[2])

cnx.close()
```

en particulier, on peut construire une requête qui résulte de l'interaction avec l'utilisateur

naïvement, on pourrait écrire par exemple

```
texte = input("Texte à rechercher dans le titre :")
motif = "'%" + texte + "'%"
c.execute("SELECT * FROM livre WHERE titre LIKE " + motif)
```

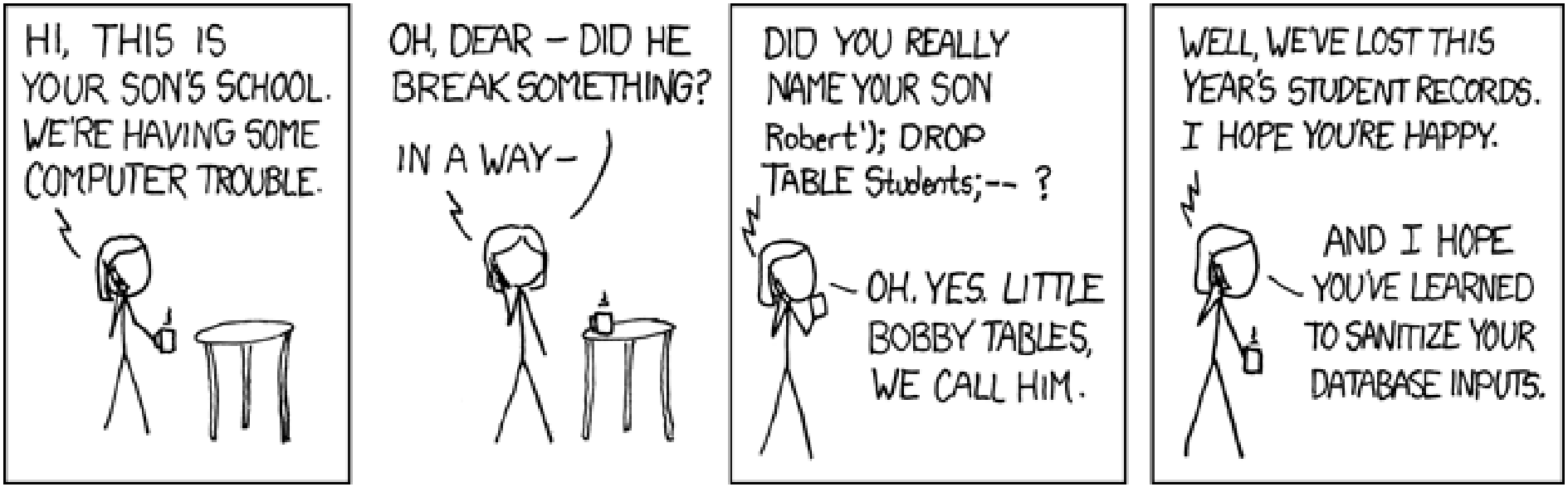
mais un utilisateur maladroit ou malicieux pourrait saisir un texte comme

```
'; DROP TABLE emprunt; SELECT * FROM livre WHERE titre = '
```

et on se retrouvera alors à exécuter ces trois ordres !

```
SELECT * FROM livre WHERE titre LIKE '%';
DROP TABLE emprunt;
SELECT * FROM livre WHERE titre = '%';
```

on appelle cela une **injection SQL**
c'est une faille de sécurité de nombreux sites web


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

il faut prendre soin d'échapper correctement les caractères comme ' qui se trouveraient dans le texte saisi par l'utilisateur
la fonction execute le fait si on l'appelle correctement

```
texte = input("Texte à rechercher dans le titre :")
motif = "%" + texte + "%"
c.execute("SELECT * FROM livre WHERE titre LIKE %s", [motif])
```

on a maintenant une requête inoffensive

```
SELECT * FROM livre WHERE titre LIKE '%'';DROP TABLE emprunt;
SELECT * FROM livre WHERE titre = ''%';
```

objectif : traduire des requêtes en SQL, sur deux bases de données

- la médiathèque
- des films extraits de IMDB

on le fait dans un navigateur :

`https://tchou.github.io/sqlsb/`
`http://alain.delaet.info/`

examen vendredi 21 mars 14h–16h
documents autorisés :

- glossaire
- aide-mémoire Python, HTML/CSS, SQL
- une feuille recto-verso a4 manuscrite