**PANORAMA SUR LES BASES DE DONNÉES**

**Formes normales :** Données atomiques et non répliquées

* Elles permettent d’éviter la réplication des données dans les tables

**Tables :**

Clé étrangère : doit être unique.

Plusieurs moyens de faire des clés étrangères :

* Simples 1 (id)
* Composites (12019) (id+annee)

Index :

Utilisation hashmap sur la colonne. Lors d’une recherche, cela évite de parcourir toute la table. (optimisation).

Attention, mettre un index sur une colonne, cela facilite les recherches mais à chaque INSERT ou DELETE, il va mettre à jour tous les index (lourd).

* Bonne pratique : mettre une colonne “id” dans chaque table pour rendre les lignes uniques sur un entier auto-incrémenté. → à faire même si on peut utiliser une colonne qui a déjà des valeurs uniques sur chaque ligne.

Contraintes :

On peut ajouter des contraintes pour l’insertion de nouvelles lignes, sur certaines colonnes telles que :

* unicité : données unique
* non nul : NOT NULL
* sur une colonne particulière : contrôler si la valeur respecte une règle (ex : âge inférieur à ..)

**Liaisons :**

Pour les mêmes questions de non duplication on crée plusieurs tables. On va donc avoir plusieurs entités avec des liens entres elles

ex: la table personne est en relation avec la table cours. On peut avoir des relations de la forme :

1-1 → Une personne à 1 cours (On choisis une des deux tables et on choisis une clé étrangère correspondant à l’id de l’autre table)

1-N → Une personne peut avoir plusieurs cours mais un cours ne peut avoir qu’une personne. (On met la clé étrangère de personne dans la table cours (relation contre intuitive))

N-N → Plusieurs personnes peuvent avoir plusieurs cours (On crée une table appelée table de liaison entre les deux tables. Cette table de liaison va avoir une clé étrangère de cours et une clé étrangère de personne. Si on a des couples de clés étrangères identiques dans cette table de liaison c’est seulement dans ce cas de figure que l’on peut penser à créer une clé combinée.)

* Quand on utilise souvent une jointure entre différentes tables, il peut être intéressant de dupliquer l’information dans une des tables, via un trigger à l’INSERT dans une des tables pour gagner en performances.
* On optimise jamais à priori mais à posteriori.

ORM (Object Relational Mapping)

* Permet de transformer automatiquement les données en base de données relationnelles, en organisation objet, dans le code d’un programme. Transforme les relations SQL en objet.
* Ex : eloquent, doctrine, Entity Framework, Dapper
* Plus pratique pour les raisons suivantes :
  + Si l'on souhaite changer des contraintes, on a juste à le faire à un endroit dans le code et mettre à jour la bdd, plutôt que de le faire en bdd.
  + Une courant de pensée actuel tend même à vouloir déléguer le contrôle des clés étrangères à l’ORM.

**NoSQL :**

Avant :

Propriétés “ACID” des bases de données, lié au mécanisme de “transactions” :

* Atomicité → contrôle de l’insertion : soit tout est ajouté, soit rien
* Cohérence → si une requête passe, toutes les règles de cohérences sont appliquées
* Isolation → Pendant qu’on inscrit les données, on ne peut pas voir l’état temporel dans lequel est la base de donnée. On voit soit la base non modifiée, soit la base modifiée mais pas entre les deux pendant la modification.
* Durabilité → Une fois les données validées / inscrites, on ne peut pas revenir en arrière sur la transaction donnée. (une fois “ABORD”)

(fun fact : git se base sur les propriétés ACID)

SQL est automatisé en transactions.

Depuis NoSQL :

Ce mouvement ne repose pas sur le principe ACID. Les transactions n’existent plus.

Ex : mongodb, Redis (performances importantes, pour faire des caches d’application), Hadoop (gestion de masses de données)...

La non cohérence des données ne pose pas de problème.

**INTERNET**

Tim Berners-Lee (CERN), a inventé les hyperliens permettant de récupérer directement les données/images. Il a donc inventé le protocole de communication permettant d’aller chercher le document URI combiné avec URL = l’identifiant, c’est la location.

Il a donc inventé un langage permettant d’utiliser ces protocoles, (HTML).

Création du Web = à la fois un client et un serveur. Vint Cerf a inventé Internet (IP)

Merci à pornhub pour avoir aidé à développer le HTML !

**Les bases du web**

****

* HTML va gérer le contenu d’une page web, et son agencement (layout). (XHTML = ancienne version)
* Utilisé à la base pour faire des documents de recherche scientifique
* CSS pour faire la composition et la partie graphique (layout et design). Il utilise un sélecteur (quel élément on sélectionne) et une règle (qu’est ce qu’on fait avec).
* JS est pour le comportement des interactions sur la page, réseau, boutons cliquables etc … (gestion des evts)

**Javascript**

* 90’s, NCSA Mosaic lance le premier logiciel uniquement client, premier vrai navigateur multi-plateforme
* mid 90’s, Netscape va créer un navigateur web (Netscape navigator), pour concurrencer Mosaic, qui va devenir progressivement Mozilla
* Microsoft : le web, c’est un enjeu : car cela peut concurrencer leur business model => développer un navigateur internet qui va leur permettre d’afficher facilement les publicités (Google fait de l’argent avec les données récupérées).
* La première version de Javascript a été publiée avec trop de défauts.

**Ajax**

* Permet d’avoir une page qui va chercher des informations de façon asynchrone suite à l’invention de XML HTTP REQUEST par Microsoft. = a véritablement permis l’émergence d’applications web.

**Fonctionnement d’une application web**

CLIENT -----------------------------> SERVEUR

* routeur (permet de décoder une url)
* Types de requêtes : GET : ne doit pas être utilisé pour écrire sur le serveur mais uniquement lire / POST / PUT / DELETE / OPTION / HEAD

**REST**

* Standard de communication, décrit les méthodes de communication entre client / serveur. Standardiser les URL, et les erreurs.
* Utilisation du JSON pour standardiser.

**Actions CRUD**

* Create / Read / Update / Delete

**MVC (Modèle Vue Contrôleur)**

* Design pattern permettant d’organiser le code en trois parties. Séparer le code gérant la vue (front), la logique de la gestion des données (Contrôleur) et les données (modèle).

**CORS**

RAPPELS SUR SQL

* **Caractères génériques** :

\_ % : donne le résultat qui match le reste de la chaîne de caractère associée :

Ex :

SELECT DISTINCT city

FROM crime\_scene\_report

WHERE city LIKE 'I%; // toutes les villes commençant par “I”

\_ BETWEEN / AND : permet de sélectionner quelque chose entre deux valeurs :

SELECT DISTINCT city

FROM crime\_scene\_report

WHERE city BETWEEN 'W%' AND 'Z%';

* **Fonctions d’agrégation :**

MAX

finds the maximum value

MIN

finds the minimum value

SUM

calculates the sum of the specified column values

AVG

calculates the average of the specified column values

COUNT​

counts the number of specified column values