Introduction générale aux bases de données

Table des matières

Cours

Les *BD*∗ sont nées à la fin des années 1960 pour combler les lacunes des systèmes de fichiers et faciliter la gestion qualitative et quantitative des données informatiques. Les *SGBD*∗ sont des applications informatiques permettant de créer et de gérer des BD (comme Oracle ou PostgreSQL par exemple)

Les BD relationnelles, issues de la recherche de Codd, sont celles qui ont connu le plus grand essor depuis les années, et qui reste encore aujourd'hui les plus utilisées. On utilise des *SGBDR*∗ pour les implémenter. Le langage *SQL*∗ est le langage commun à tous les SGBDR, ce qui permet de concevoir des BD relativement indépendamment des systèmes utilisés.

Les usages de BD se sont aujourd'hui généralisés pour entrer dans tous les secteurs de l'entreprise, depuis les petites bases utilisées par quelques personnes dans un service pour des besoins de gestion de données locales, jusqu'aux bases qui gèrent de façon centralisée des données partagées par tous les acteurs de l'entreprise.

L'accroissement de l'utilisation du numérique comme outil de manipulation de toutes données (bureautique, informatique applicative, etc.) et comme outil d'extension des moyens de communication (réseaux) ainsi que les évolutions technologiques (puissance des PC, Internet, etc.) ont rendu indispensable, mais aussi complexifié la problématique des BD.

Les conséquences de cette généralisation et de cette diversification des usages se retrouvent dans l'émergence de solutions conceptuelles et technologiques nouvelles, les bases de données du mouvement NoSQL particulièrement utilisées par les grands acteurs du web.

1. Notions fondamentales : donnée, base de données, système de gestion de bases de données

1.1. Base de données

Logiciel et données

Un logiciel informatique est composé de programmes, c'est à dire d'instructions données à l'ordinateur, et de données auxquelles s'appliquent ces instructions.

Par exemple un logiciel de traitement de texte est composé de fonctions - ouvrir, copier, coller, insérer une image, changer la police, enregistrer... - et de fichiers sur lesquels elles s'appliquent. Dans ce cas les fichiers de traitement de texte sont les données.

Définition lâche de base de données : un ensemble de données ***Définition***

On appelle parfois base de données tout ensemble de données stocké numériquement et pouvant servir à un ou plusieurs programme. De ce point de vue des fichiers sur un disque dur, un fichier de tableur, voire un fichier de traitement de texte peuvent constituer des bases de données.

Définition restreinte de base de données : un ensemble de données structuré ***Définition***

On appellera base de données un ensemble de données numériques qui possède une structure ; c'est à dire dont l'organisation répond à une **logique** systématique.

On parlera de modèle logique de données pour décrire cette structure.

Base de données relationnelle *Exemple*

Une base de données relationnelle permet d'organiser les données en tableaux (appelés relations).

Base de données de classification classique des espèces animales

espèce	eucaryote	multicellulaire	propriété
bactéries	false	false	
archées	false	false	
protistes	true	false	
champignons	true	true	décompose
végétaux	true	true	photosynthétise
animaux	true	true	ingère

Fonctions d'une base de données *Fondamental*

Une base de données est structurée afin de pouvoir mieux répondre à des fonctions fondamentales en informatique, telles que :

- Stocker l'information de façon fiable (c'est à dire être capable de restituer l'information entrée dans le système)
- Traiter de grands volumes de données (massification)
- Traiter rapidement les données (optimisation)
- Sécuriser les accès aux données (gérer les autorisations selon les utilisateurs)
- Contrôler la qualité des données (par exemple la cohérence par rapport à un modèle pré-établi)
- Partager les données (entre plusieurs applications dédiées à plusieurs métiers)
- Rendre accessible les données en réseau (gérer la concurrence des accès parallèles)
- ...

1.2. Système de gestion de bases de données (SGBD)

La conception d'un système d'information se doit d'adopter un certain nombre de principes, tels que :

- une description des données indépendante des traitements,
- une gestion automatique de la cohérence de données,
- le recours à des langages non procéduraux structurants.

Système de Gestion de Bases de Données *Définition*

Un *SGBD*∗ est un logiciel qui prend en charge la structuration, le stockage, la mise à jour et la maintenance d'une base de données. Il est l'unique interface entre les informaticiens et les données (définition des schémas, programmation des applications), ainsi qu'entre les utilisateurs et les données (consultation et mise à jour).

Exemples de SGBD

Exemple

- Oracle est un SGBD relationnel et relationnel-objet très utilisé pour les applications professionnelles.
- PostgreSQL est un SGBD relationnel puissant qui offre une alternative libre (licence BSD) aux solutions commerciales comme Oracle ou IBM.
- Access est un SGBD relationnel Microsoft, qui offre une interface graphique permettant de concevoir rapidement des applications de petite envergure ou de réaliser des prototypes.
- MongoDb est un SGBD non-relationnel libre (licence Apache) orienté document. Il permet de gérer facilement de très grandes quantités de données - dans un format arborescent JSON - réparties sur de nombreux ordinateurs.

Objectifs des SGBD

Fondamental

- **Indépendance physique des données**

  Le changement des modalités de stockage de l'information (optimisation, réorganisation, segmentation, etc.) n'implique pas de changements des programmes.
- **Indépendance logique des données**

  L'évolution de la structure d'une partie des données n'influe pas sur l'ensemble des données.
- **Manipulation des données par des non-informaticiens**

  L'utilisateur n'a pas à savoir comment l'information est stockée et calculée par la machine, mais juste à pouvoir la rechercher et la mettre à jour à travers des IHM ou des langages assertionnels simples.
- **Administration facilitée des données**

  Le SGBD fournit un ensemble d'outils (dictionnaire de données, audit, tuning, statistiques, etc.) pour améliorer les performance et optimiser les stockages.
- **Optimisation de l'accès aux données**

  Les temps de réponse et de débits globaux sont optimisés en fonctions des questions posées à la BD.
- **Contrôle de cohérence (intégrité sémantique) des données**

  Le SGBD doit assurer à tout instant que les données respectent les règles d'intégrité qui leurs sont imposées.
- **Partageabilité des données**

  Les données sont simultanément consultables et modifiables.
- **Sécurité des données**

  La confidentialité des données est assurée par des systèmes d'authentification, de droits d'accès, de chiffrement des mots de passe, etc.
- **Sûreté des données**

  La persistance des données, même en cas de panne, est assurée, grâce typiquement à des sauvegardes et des journaux qui gardent une trace persistante des opérations effectuées.

Complément

Pourquoi des SGBD ? *(cf. p.25)*

SGBD relationnel et non-relationnel *Complément*

Les SGBR relationnels (SGBDR) sont les plus courants des SGBD ; jusqu'au début des années 2000, la plupart des bases de données étaient relationnelles.

Mais avec l'arrivée des géants du web, ces entreprises qui gèrent des quantités énormes de données comme Google, Amazon ou Facebook, s'est développé un mouvement important de développement de bases de données non-relationnelles, également appelées NoSQL.

1.3. Application de base de données

Une base de données seule n'est pas directement utilisable par un utilisateur humain ; elle n'est utilisable que par les informaticiens qui connaissent son langage de programmation et par les applications qui ont été programmées pour s'en servir.

Définition

On appelle application de base de données un logiciel informatique permettant à un utilisateur final de manipuler (lire ou écrire) les données d'une base de données.

Application web *Exemple*

Une application web est composée d'interfaces en HTML qui permettent d'écrire et de lire des données dans une base de données, via un langage applicatif, comme par exemple PHP.

Twitter *Exemple*

L'application Twitter est composée d'interfaces web permettant d'entrer des données (saisir son profil, *twitter*, *retwitter*, ...) et de sortir des données (consulter un fil *twitter*, faire une recherche sur un *hashtag*...) d'une base de données.

Cette base de données est stockée sur les serveurs de Twitter et elle contient tous les profils de tous les utilisateurs, tous les *tweets*, tous les *hashtags*...

Compagnie aérienne *Exemple*

Une base de données de gestion de l'activité d'une compagnie aérienne concerne les voyageurs, les vols, les avions, le personnel, les réservations...

Une application à partir d'une telle base de données pourra permettre la gestion des réservations, des disponibilités des avions en fonction des vols à effectuer, des affectations des personnels volants...

Application de bureau Access *Exemple*

Avec un logiciel comme Access on peut réaliser à la fois une base de données et une application permettant de manipuler cette base de données pour des besoins bureautiques simples.

1.4. Donnée (en relationnel) : table, objet, propriété, domaine, atomicité

Base de données relationnelle *Rappel*

Une base de données relationnelle permet d'organiser les données en tables (appelés relations).

Chaque case de la table contient une information atomique.

Objet (ligne) *Définition*

Chaque ligne de la table correspond à un objet que l'on veut gérer dans la base de données : une voiture, une personne, une espèce...

Fondamental

Toutes les lignes d'une même table correspondent à des objets du même type, donc dans une table, on met soit des voitures, soit des personnes, mais on ne mélange pas les deux.

Propriété et domaine (colonne) *Définition*

Chaque colonne de la table correspond à une propriété des objets qui se trouvent dans la table ; tous les objets de la table partagent donc les mêmes propriétés.

Domaine *Fondamental*

Chaque colonne de la table est associée à un domaine de valeur fixé **a priori**, par exemple : entier, texte, booléen...

Donnée en relationnel (cellule) *Définition*

Une donnée en relationnel, c'est une cellule d'une table, qui correspond à la propriété d'un objet.

Une table ou relation (en relationnel)

propriété 1 **domaine : d1**	**propriété 2** **domaine : d2**	**...**
objet1, donnée 1	objet1, donnée 2	...
objet2, donnée 1	objet2, donnée 2	...
...	...	...

Exemple

Exemple de relation instanciée

espèce **domaine : texte**	**eucaryote** **domaine : booléen**	**...**
bactéries	false	...
archées	false	...
...	...	...

Atomicité (contre-exemple)

Attention

Pour que la base de données fonctionne correctement on veille à ne mettre qu'une seule donnée par case, c'est le principe d'atomicité en relationnel.

Un mauvais exemple de relation : les données ne sont pas atomiques (il y a plusieurs données par case de la table)

espèce, domaine : texte
bactéries : procaryotes unicellulaires
archées : procaryotes unicellulaires
protistes : eucaryotes unicellulaires
champignons : eucaryotes multicellulaires qui décomposent
végétaux : eucaryotes multicellulaires qui photosynthétisent
animaux : eucaryotes multicellulaires qui ingèrent

1.5. Langage de données : l'exemple du langage SQL

Langage de données

Définition

Un langage de données est un langage informatique permettant de décrire et de manipuler les schémas et les données d'une *BD*∗.

Synonymes : Langage orienté données

SQL

Fondamental

SQL∗ est le langage consacré aux SGBD relationnels et relationnels-objet.

Il permet de :

- créer des tables, en définissant le domaine de chaque colonne ;
- insérer des lignes dans les tables
- lire les données entrées dans la base de données

Création de table en SQL (définition du schéma de données) *Exemple*

```
CREATE TABLE etudiant (
 numetu INTEGER PRIMARY KEY,
nom VARCHAR,
 ville VARCHAR)
```

Cette instruction permet de créer une relation "etudiant" comportant les propriétés "numetu", "nom" et "ville" de domaines, respectivement, entier, texte et texte.

Insertion de ligne en SQL (création de données) *Exemple*

```
INSERT INTO etudiant (numetu, nom, ville) VALUES (1, 'Holmes', 'Londres')
```

Cette instruction permet de créer l'étudiant numéro 1, de nom Holmes qui habite la ville de Londres.

Manipulation de données en SQL (exploitation des données) *Exemple*

```
SELECT nom
FROM etudiant
WHERE ville = 'Compiègne'
```

Cette instruction permet de rechercher les noms de tous les étudiants habitant la ville de Compiègne.

Autres langages de données *Complément*

- XQuery est un langage de données mobilisé dans les bases de données arborescentes XML.
- Les bases NoSQL proposent des langages de données spécifiques, souvent inspirés du SQL. Par exemple le langage de MongoDB permet de manipuler une base de contenus JSON.

2. Approche générale pour la conception des bases de données

2.1. Exercice : Étapes de la conception d'une base de données relationnelle

Mettre dans l'ordre les étapes de conception suivantes.

1. Modélisation conceptuelle en UML ou E-A
2. Élaboration du modèle logique en relationnel
3. Création du code SQL pour un SGBDR
4. Analyse de la situation existante et des besoins

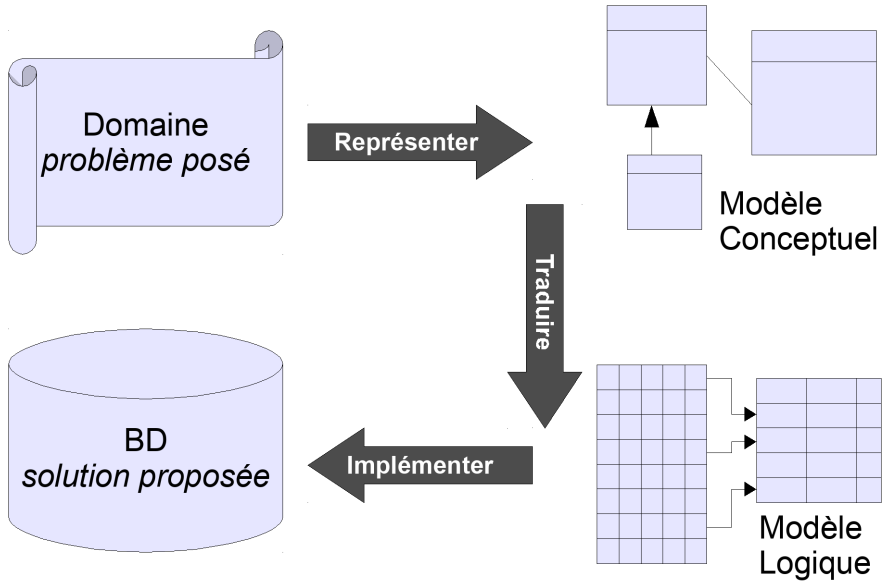
Réponse :

2.2. Méthodologie générale de conception d'une base de données

Étapes de la conception d'une base de données *Méthode*

1. **Analyse** de la situation existante et des besoins (clarification)
2. Création d'un **modèle conceptuel** qui permet de représenter tous les aspects importants du problème

3. Traduction du modèle conceptuel en **modèle logique** (et normalisation de ce modèle logique)
4. Implémentation d'une **base de données** dans un SGBD, à partir du modèle logique (et optimisation)



Processus de conception d'une base de données

On distingue quatre étapes dans la conception d'une base de données :

- **L'analyse**

  Elle consiste à étudier le problème et à consigner dans un document, la note de clarification, les besoins, les choix, les contraintes.

- **La modélisation conceptuelle**

  Elle permet de décrire le problème posé, de façon non-formelle (en générale graphique), en prenant des hypothèses de simplification. Ce n'est pas une description du réel, mais une représentation simplifiée d'une réalité.

- **La modélisation logique**

  Ele permet de décrire une solution, en prenant une orientation informatique générale (type de SGBD typiquement), formelle, mais indépendamment de choix d'implémentation spécifiques.

- **L'implémentation**

  Elle correspond aux choix techniques, en terme de SGBD choisi et à leur mise en œuvre (programmation, optimisation...).

Fondamental

- **Bien analyser** le problème posé en amont
- **Bien modéliser** le problème au niveau conceptuel avant de passer au niveau logique et à l'implémentation

L'importance de l'étape d'analyse ***Conseil***

La première étape de la conception repose sur l'analyse de l'existant et des besoins. De la qualité de la réalisation de cette première étape dépendra ensuite la pertinence de la base de données par rapports aux usages. Cette première étape est donc essentielle et doit être menée avec soins.

Si la première étape est fondamentale dans le processus de conception, elle est aussi la plus délicate. En effet, tandis que des formalismes puissants existent pour la modélisation conceptuelle puis pour la modélisation logique, la perception de l'existant et des besoins reste une étape qui repose essentiellement sur l'expertise d'analyse de l'ingénieur.

L'importance de l'étape de modélisation conceptuelle *Conseil*

Étant donnée une analyse des besoins correctement réalisée, la seconde étape consiste à la traduire selon un modèle conceptuel. Le modèle conceptuel étant formel, il va permettre de passer d'une spécification en langage naturel, et donc soumise à interprétation, à une spécification non ambigüe. Le recours aux formalismes de modélisation tels que *E-A∗* ou *UML∗* est donc une aide fondamentale pour parvenir à une représentation qui ne sera plus liée à l'interprétation du lecteur.

La traduction d'un cahier des charges spécifiant l'existant et les besoins en modèle conceptuel reste néanmoins une étape délicate, qui va conditionner ensuite l'ensemble de l'implémentation informatique. En effet les étape suivantes sont plus mécaniques, dans la mesure où un modèle logique est déduit de façon systématique du modèle conceptuel et que l'implémentation logicielle est également réalisée par traduction directe du modèle logique.

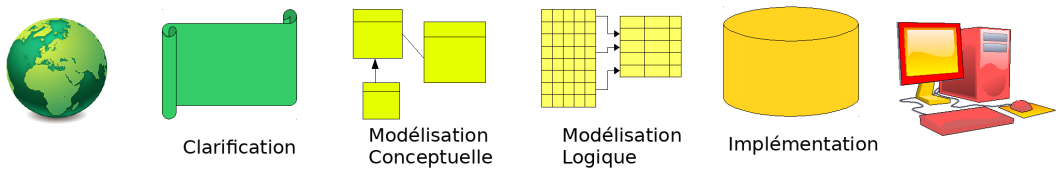
Les étapes de traduction logique et d'implémentation *Remarque*

Des logiciels spécialisés sont capables à partir d'un modèle conceptuel d'appliquer des algorithmes de traduction qui permettent d'obtenir directement le modèle logique, puis les instructions pour la création de la base de données dans un langage orienté données tel que SQL. L'existence de tels algorithmes de traduction montre que les étapes de traduction logique et d'implémentation sont moins complexes que les précédentes, car plus systématiques.

Néanmoins ces étapes exigent tout de même des compétences techniques pour optimiser les modèles logiques (normalisation), puis les implémentations en fonction d'un contexte de mise en œuvre matériel, logiciel et humain.

2.3. Quatre étapes pour réduire la complexité

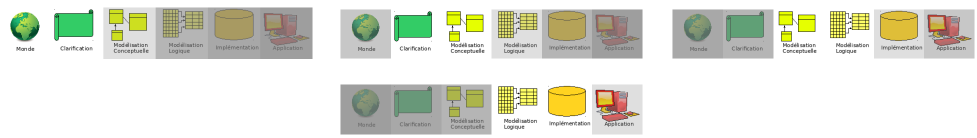
La réalisation d'une base de données est une tâche complexe, le découpage en quatre étapes permet de gérer cette complexité.



Conception en quatre étapes

L'idée générale est de ne pas mélanger la nature des tâches : typiquement, on ne veut pas en même temps se poser des questions générales relatives aux besoins (ce que je veux faire) et des questions techniques très spécifiques (comment représenter telle information).

Méthode



Conception en quatre étapes

L'approche est donc :

- de "regarder à gauche" : on regarde le monde quand on clarifie, on regarde la note de clarification quand on fait le MCD, on regarde le MCD quand on fait le MLD, on regarde le MLD quand on implémente ;
- et de ne "pas regarder trop loin" : on anticipe déjà le MCD pendant la clarification, mais on ne se préoccupe pas de questions d'implémentation ; quand on fait le MLD, on ne se pose plus de question sur le monde, la clarification et le MCD ont dû déjà répondre ; quand on implémente on ne se pose plus de question de modélisation, on s'occupe des programmes, de la machine.

Fondamental

On peut toujours revenir sur une étape, la conception est itérative, mais on ne doit pas tout le temps se poser tous les problèmes en même temps.

2.4. Éléments pour l'analyse de l'existant et des besoins

La phase d'analyse de l'existant et des besoins est une phase essentielle et complexe. Elle doit aboutir à des spécifications générales qui décrivent en langage naturel les données manipulées, et les traitements à effectuer sur ces données.

On se propose de donner une liste **non exhaustive** d'actions à mener pour rédiger de telles spécifications.

L'analyse de documents existants

Méthode

La conception d'une base de données s'inscrit généralement au sein d'usages existants. Ces usages sont généralement, au moins en partie, instrumentés à travers des documents électroniques ou non (papier typiquement). Il est fondamental d'analyser ces documents et de recenser les données qu'ils manipulent.

Exemples de document existants à analyser

Exemple

- Fichiers papiers de stockage des données (personnel, produits, etc.)
- Formulaires papiers d'enregistrement des données (fiche d'identification d'un salarié, fiche de description d'un produit, bon de commande, etc.)
- Documents électroniques de type traitement de texte (lettres, mailing, procédures, etc.)
- Documents électroniques de type tableurs (bilans, statistiques, calculs, etc.)
- Bases de données existantes, à remplacer ou avec lesquelles s'accorder (gestion des salaires, de la production, etc.)
- Intranet d'entreprise (information, téléchargement de documents, etc.)
- etc.

Le recueil d'expertise métier

Méthode

Les données que la base va devoir manipuler sont toujours relatives aux métiers de l'entreprise, et il existe des experts qui pratiquent ces métiers. Le dialogue avec ces experts est une source importante d'informations. Il permet également de fixer la terminologie du domaine.

Exemples d'experts à consulter

Exemple

- Praticiens (secrétaires, ouvrier, contrôleurs, etc.)
- Cadres (responsables de service, contre-maîtres, etc.)
- Experts externes (clients, fournisseurs, etc.)
- etc.

Le dialogue avec les usagers

Méthode

La base de données concerne des utilisateurs cibles, c'est à dire ceux qui produiront et consommeront effectivement les données de la base. Il est nécessaire de dialoguer avec ces utilisateurs, qui sont les détenteurs des connaissances relatives aux besoins réels, liés à leur réalité actuelle (aspects de l'organisation fonctionnant correctement ou défaillants) et à la réalité souhaitée (évolutions, lacunes, etc.).

Exemples d'utilisateurs avec qui dialoguer

Exemple

- Personnes qui vont effectuer les saisies d'information (à partir de quelles sources ? Quelle est leur responsabilité ? etc.)
- Personnes qui vont consulter les informations saisies (pour quel usage ? pour quel destinataire ? etc.)
- Personnes qui vont mettre à jour les informations (pour quelles raisons ? comment le processus est enclenché ? etc.)
- etc.

L'étude des autres systèmes informatiques existants

Méthode

la base de données va généralement (et en fait quasi systématiquement aujourd'hui) s'insérer parmi un ensemble d'autres logiciels informatiques travaillant sur les données de l'entreprise. Il est important d'analyser ces systèmes, afin de mieux comprendre les mécanismes existants, leurs forces et leurs lacunes, et de préparer l'intégration de la base avec ces autres systèmes. Une partie de ces systèmes seront d'ailleurs souvent également des utilisateurs de la base de données, tandis que la base de données sera elle même utilisatrice d'autre systèmes.

Exemples d'autres systèmes coexistants à étudier

Exemple

- Autres bases de données (les données sont elle disjointes ou partiellement communes avec celles de la base à concevoir ? quelles sont les technologies logicielles sur lesquelles reposent ces *BD*$*$? etc.)
- Systèmes de fichiers classiques (certains fichiers ont-ils vocations à être supplantés par la base ? à être générés par la base ? à alimenter la base ? etc.)
- Applications (ces applications ont elles besoins de données de la base ? peuvent-elles lui en fournir ? etc.)
- etc.

Méthodes d'analyse *Complément*

Il existe des outils et méthodes d'analyse en ingénierie, comme l'analyse fonctionnelle (AF), qui dépassent le cadre de la conception des bases de données, mais sont tout à fait complémentaires.

2.5. Note de clarification

Note de clarification (NDC) *Définition*

La note de clarification est une reformulation du cahier des charges, qui précise, ajoute et supprime des éléments (en justifiant). C'est cette référence qui sera utilisée pour la suite du projet. Si le périmètre d'un projet est modifié, la NDC doit être ajouté en conséquence.

Méthode

Il existe plusieurs façons de réaliser une NDC. En base de données, on se focalisera sur l'explicitation des points suivants :

- Liste des objets qui devront être gérés dans la base de données
- Liste des propriétés associées à chaque objet
- Liste des contraintes associées à ces objets et propriétés
- Liste des utilisateurs (rôles) appelés à modifier et consulter les données
- Liste des fonctions que ces utilisateurs pourront effectuer

On formulera explicitement les hypothèses qui ont permis de faire des choix, lorsque le CDC

Attention

La NDC est la reformulation du cahier des charge sur laquelle on se base pour réaliser le *MCD*∗. Si aucune NCD n'est associée à un MCD, cela signifie que :

- soit le *CDC*∗ est réputé sans erreur et parfaitement clair (ce qui est rare) ;
- soit on est pas capable de savoir à quoi se rapporte la modélisation.

Fondamental

On modélise toujours quelque chose dans un contexte, on pose que c'est ce qui est consigné dans la NDC. Si on ne sait pas ce qui est modélisé, la validité du modèle n'est pas évaluable.

Méthode

Un NDC peut revêtir plusieurs formats, mais on privilégiera :

- un document facile à mettre à jour (un projet évolue) et dont les versions sont traçables (on doit pouvoir remonter des modifications pour retrouver des bifurcations dans les attendus) ;
- un texte très clair, avec une rédaction minimaliste mais non ambiguë, des mots précis (par exemple on utilisera un seul mot pour un seul concept, la répétition dans ce contexte étant toujours préférable à un synonyme).

2.6. Modélisation conceptuelle de données

L'objection du modèle conceptuel est de représenter le problème à l'aide de représentations graphiques et partiellement formelles.

Les principales caractéristiques du modèle conceptuel sont :

- Une représentation graphique simple
- Une puissance d'expression élevée pour un nombre de symboles raisonnables
- Une lecture accessible à tous et donc un bon outil de dialogue entre les acteurs techniques et non techniques
- Une formalisation peu ambiguë et donc un bon outil de spécification détaillée

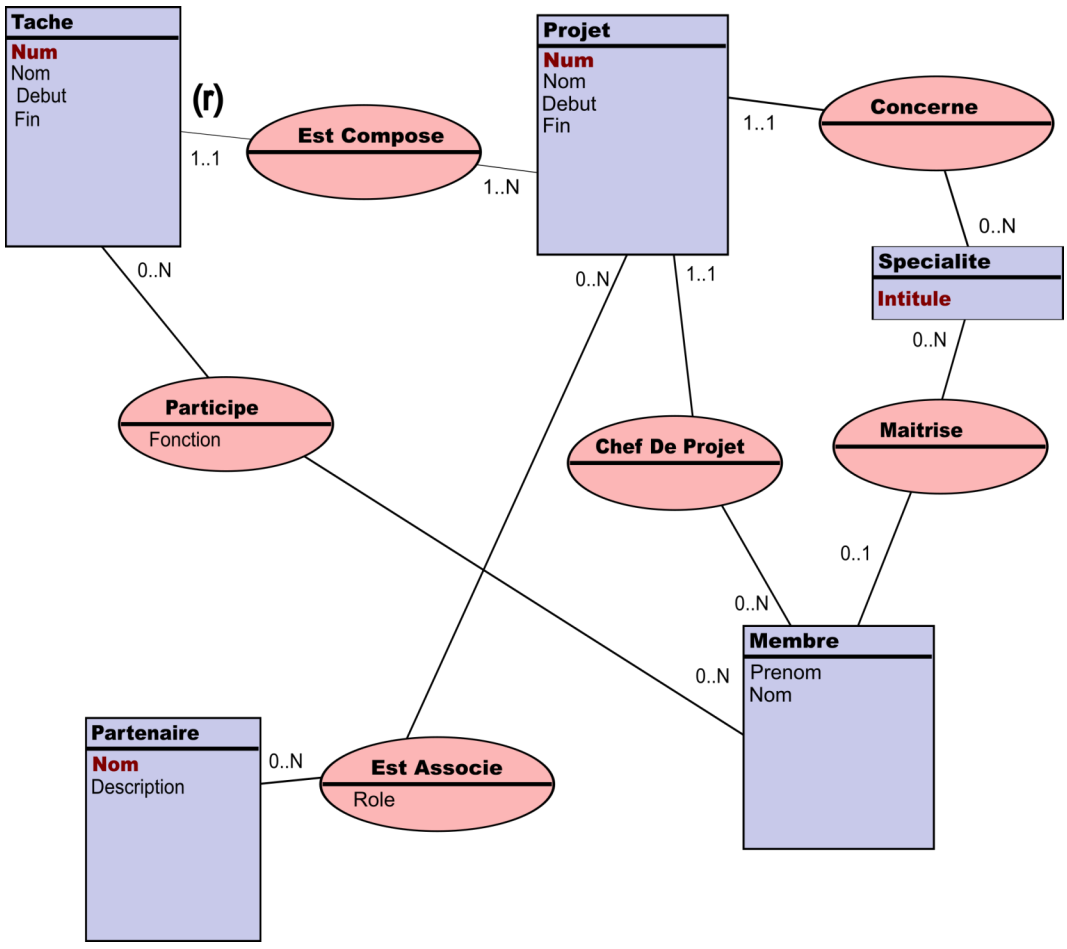
Remarque

Le modèle n'est pas encore formel, donc certaines représentations peuvent être équivoques, même si on a levé de très nombreuses ambiguïtés.

E-A

La modélisation conceptuelle en bases de données relationnelle était à l'origine faite avec le formalisme *E-A*∗ de la méthode MERISE.

Exemple



Modèle E-A "gestion de projets"

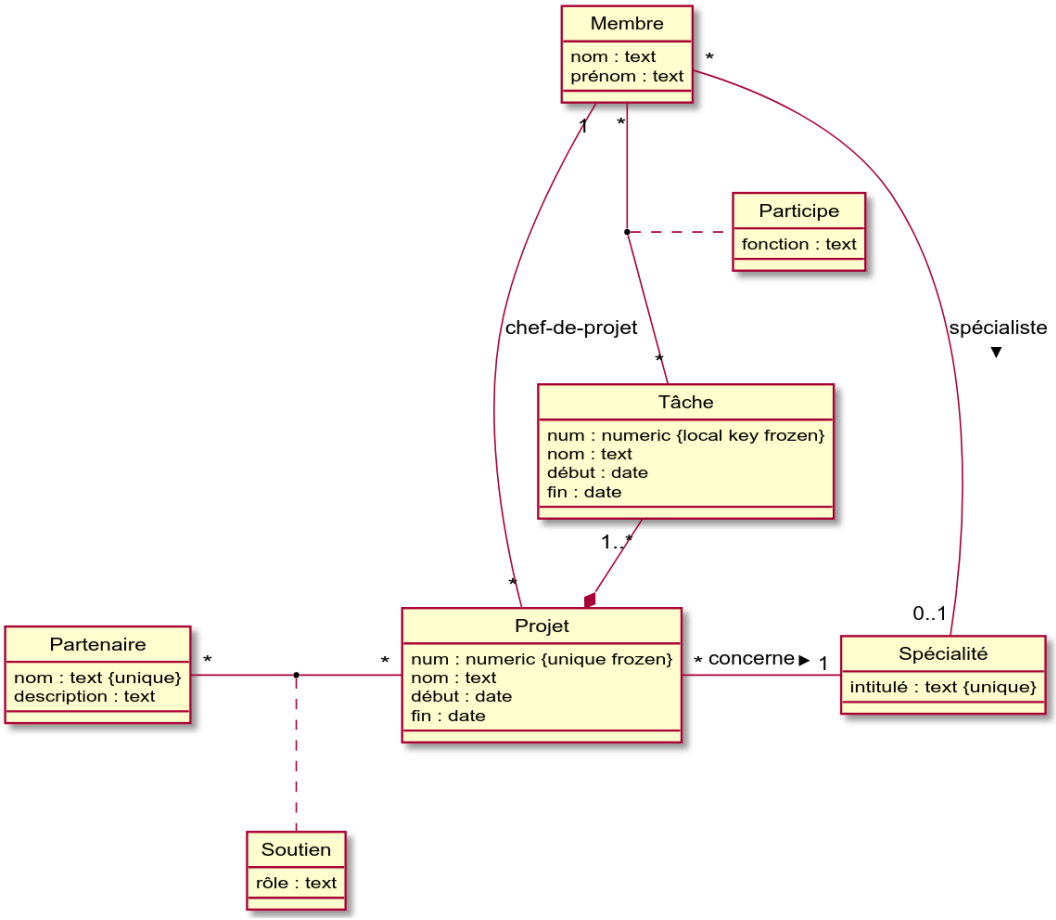
UML

UML ∗ est un autre langage de modélisation, plus récent que E-A et couvrant un spectre plus large que les bases de données. En tant que standard de l'*OMG* ∗ et en tant que outil très utilisé pour la programmation orientée objet, il a supplanté la modélisation E-A.

Remarque

En BD on utilise uniquement le **diagramme de classe** d'UML pour modéliser conceptuellement les données.

Exemple



Modèle UML "gestion de projets"

2.7. Modélisation logique de données

Modèle logique de données

Définition

Un modèle logique de données est une description, au moyen d'un langage formel, d'un ensemble de données.

Un schéma permet de décrire la structure d'une base de données, en décrivant l'ensemble des types de données de la base. Une instance de base de données est constituée d'un ensemble de données qui respectent le schéma de la base.

Synonyme : schéma de données, schéma

Modèle logique de données relationnel *Exemple*

Un modèle logique de données relationnel permet de représenter une base de données relationnelles, c'est à dire : des tables, des propriétés, des domaines...

Schéma d'une relation *Exemple*

```
Espece(nom:chaîne, eucaryote:booléen, multicellulaire:booléen, propriété:chaîne)
```

Schéma d'une base de données avec plusieurs relations *Exemple*

```
Etudiant (num:entier, nom:chaîne, ville:chaîne)
Module(num:entier, titre:chaîne)
Inscription(numetu:entier, nummod:entier, année:entier(4))
```

Instance de base de données *Exemple*

Etudiant

172	Dupont	Lille
173	Durand	Paris
174	Martin	Isabelle

Module

1	SGBD
2	OS

Inscription

172	1	2016
172	2	2016
173	1	2015
174	2	2017

Exemple de formalismes de modélisation logique *Complément*

- Le modèle CODASYL, antérieur au modèle relationnel est un modèle hiérarchique (*Tardieu, 1983*∗).
- Le modèle relationnel (tabulaire) est le modèle dominant à la base des SGBDR.
- Le modèle relationnel-objet (adaptation des modèles relationnels et objets au cadre des SGBD) est actuellement en croissance.
- D'autres modèles (document, graphe, ...) se développent dans le cadre du mouvement NoSQL.

Voir aussi *Complément*

Définition du mouvement NoSQL (cf. p.26)

2.8. Synthèse : Les trois niveaux de conception

- Niveau Conceptuel

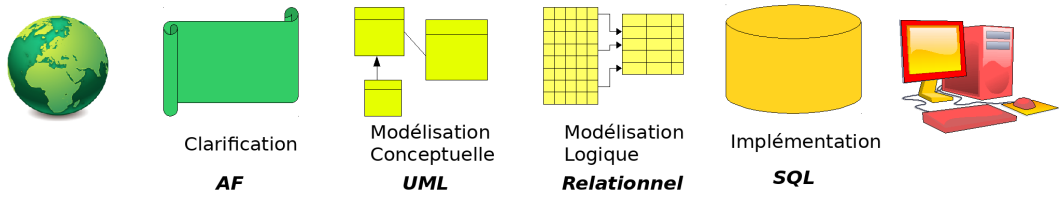
  Modèle conceptuel graphique
  - Exemples
    - E-A
    - UML
- Niveau Logique

  Schéma logique indépendant d'un SGBD

    - Exemples
        - Relationnel
        - Objet
        - Relationnel-Objet
        - Graphe
        - Document
- Niveau Informatique

  Implémentation pour un SGBD particulier
    - Exemples
        - Oracle
        - MySQL
        - PostgreSQL
        - DB2
        - Access
        - SQLServer
        - MongoDB
        - Cassandra

? Exemple



Conception en quatre étapes : exemple de formalismes

3. Découverte d'une base de données relationnelle

Objectifs

- Découvrir le modèle relationnel
- Découvrir un SGBDR
- Découvrir le langage SQL

Cette série d'exercices est destinée à faire expérimenter un *SGBDR*∗, afin de se familiariser avec les concepts classiques des bases de données relationnelles.

Pour la réalisation de cet exercice, se connecter sur le site *Db Disco*∗ et conserver la fenêtre du navigateur ouverte.

3.1. Exercice : Notion de table

Créer sa première table

Une base de données **relationnelle** est principalement constituée de **tables** (ou « relations » d'où le nom de relationnel). Une table est basiquement un élément d'organisation de l'information constitué de colonnes (ou attributs) et de lignes (ou enregistrements).

Nous allons dans un premier temps créer le **schéma** d'une table, c'est à dire définir ses colonnes. Pour cela nous utiliserons l'instruction *SQL∗ LDD∗* « CREATE ».

Question 1

Exécuter l'instruction suivante et décrire ce qu'elle fait.

```
1 CREATE TABLE tEtu (
2 pk_numSecu CHAR(13) PRIMARY KEY,
3 k_numEtu VARCHAR(20) UNIQUE NOT NULL,
4 nom VARCHAR(50),
5 prenom VARCHAR(50));
```

Alimenter la table

Une fois les colonnes de la table définies, nous pouvons en déclarer les lignes. Nous utilisons pour cela l'instruction SQL *LMD∗* « INSERT ».

Question 2

Exécuter les deux instructions suivantes et décrire ce qu'elles font.

```
1 INSERT INTO tEtu (pk_numSecu, k_numEtu, nom, prenom)
2 VALUES ('1800675001066', 'AB3937098X', 'Dupont', 'Pierre');
3 INSERT INTO tEtu (pk_numSecu, k_numEtu, nom, prenom)
4 VALUES ('2820475001124', 'XGB67668', 'Durand', 'Anne');
```

Interroger la table

Une fois une table créée, il est possible à tout moment d'en inspecter le contenu. Nous utilisons pour cela l'instruction SQL LMD « SELECT ».

Question 3

Exécuter l'instruction suivante et décrire ce qu'elle fait.

```
1 SELECT pk_numSecu, k_numEtu, nom, prenom
2 FROM tEtu;
```

Question 4

Exécuter l'instruction suivante et décrire ce qu'elle fait.

```
1 SELECT nom, prenom
2 FROM tEtu
3 WHERE pk_numSecu='2820475001124';
```

3.2. Exercice : Notion de contraintes

Contrainte de domaine

Lorsque l'on définit une table, on définit également des contraintes sur cette table, qui serviront à contrôler son **intégrité**, par rapport à des règles que l'on aura fixées.

C'est notamment le cas des contraintes de domaine, qui permettent de vérifier qu'une colonne prend ses valeurs parmi un ensemble déterminé (les chaînes de 10 caractères au plus, les entier de 1 à 1000, etc.).

Question 1

Exécuter l'instruction suivante et expliquer pourquoi le système renvoie une erreur.

```
1 INSERT INTO tEtu (pk_numSecu, k_numEtu, nom, prenom)
2 VALUES ('XXXXXXXXXXXXXXX', 'XXXXXX', 'Dupont', 'Pierre');
```

Question 2

Donner un exemple de contrainte qui n'est pas formulée dans la définition de la table *tEtu* et que l'on aurait pu souhaiter.

Indice :

Pour indiquer qu'un élément est obligatoire, on ajoute la clause `NOT NULL` après la définition de son domaine dans l'instruction `CREATE TABLE`.

Contraintes de clé

Les contraintes de clé se composent de contraintes d'**unicité** et de contraintes de **non nullité**. Elles permettent d'assurer que toutes les valeurs d'une colonne seront différentes pour chaque ligne.

Question 3

Exécuter les trois instructions suivantes (les unes après les autres) et expliquer ce qui se passe.

```
INSERT INTO tEtu (pk_numSecu, k_numEtu, nom, prenom)
VALUES ('1800675001066', 'HGYT67655Y', 'Dupont', 'Pierre');
INSERT INTO tEtu (pk_numSecu, k_numEtu, nom, prenom)
VALUES ('2810592012232', 'XGB67668', 'Durand', 'Anne');
INSERT INTO tEtu (pk_numSecu, k_numEtu, nom, prenom)
VALUES ('2810592012232', 'HGYT67655Y', 'Duchemin', 'Aline');
```

Question 4

Explorer le contenu de votre table en exécutant l'instruction suivante, et vérifier vos explications précédentes.

```
SELECT *
FROM tEtu;
```

Question 5

Pourrait-on insérer dans la table une seconde personne qui aurait le prénom "Aline" et le nom "Duchemin" ? Pourquoi ?

3.3. Exercice : Notion de références

Clé étrangère

Une base de données est en général constituée de plusieurs tables. Ces tables se référencent entre elles en utilisant une **clé étrangère** : c'est à dire qu'une des colonnes de la table est utilisée pour faire référence à la colonne d'une autre table.

On va à présent créer une seconde table, qui permettra d'associer des Unités de Valeurs (UVs) aux étudiants, puis insérer des valeurs dans cette table.

```
CREATE TABLE tUv (
pk_code CHAR(4) NOT NULL,
fk_etu CHAR(13) NOT NULL,
PRIMARY KEY (pk_code, fk_etu),
FOREIGN KEY (fk_etu) REFERENCES tEtu(pk_numSecu));
```

```
INSERT INTO tUV (pk_code, fk_etu)
VALUES ('NF17', '1800675001066');
INSERT INTO tUV (pk_code, fk_etu)
VALUES ('NF26', '1800675001066');
INSERT INTO tUV (pk_code, fk_etu)
VALUES ('NF29', '1800675001066');
```

Question 1

Expliciter ce qu'exprime le contenu de la table tUv .

Contraintes d'intégrité référentielle

Lorsque nous avons défini la table tUv, nous avons défini une contrainte supplémentaire, dite d'intégrité référentielle : contrainte de type FOREIGN KEY.

Question 2

En exécutant les instructions suivantes, expliquer quel est le rôle d'une contrainte d'intégrité référentielle.

```
1 INSERT INTO tUV (pk_code, fk_etu)
2 VALUES ('NF17', '2810592012232');
3 INSERT INTO tUV (pk_code, fk_etu)
4 VALUES ('NF17', '1700792001278');
```

3.4. Exercice : Projection, restriction et jointure

L'instruction SELECT du langage SQL LMD nous donne de larges possibilités pour interroger les tables d'une base de données. Cette instruction se fonde notamment sur les opérations mathématiques de l'algèbre relationnelle, dont les principales sont la projection, la restriction, le produit et la jointure.

Question 1

Exécuter l'instruction suivante et expliquer pourquoi c'est une **projection**.

```
1 SELECT nom, prenom
2 FROM tEtu;
```

Question 2

Exécuter l'instruction suivante et expliquer pourquoi c'est une **restriction**.

```
1 SELECT *
2 FROM tEtu
3 WHERE nom='Dupont';
```

Question 3

Exécuter l'instruction suivante et expliquer pourquoi c'est un **produit** (cartésien).

```
1 SELECT *
2 FROM tEtu,tUv;
```

Question 4

Exécuter l'instruction suivante et expliquer pourquoi c'est une **jointure**.

```
1 SELECT *
2 FROM tEtu JOIN tUv ON pk_numSecu=fk_etu;
```

Question 5

Exécuter l'instruction suivante et montrer qu'une jointure est la composition d'un produit et d'une restriction.

```
1 SELECT *
2 FROM tEtu,tUv
3 WHERE pk_numSecu=fk_etu;
```

3.5. Exercice : Fonctions et agrégats

L'instruction SELECT permet également d'effectuer des calculs qui portent sur plusieurs lignes, ce que l'on appelle des agrégats.

Question 1

Exécuter la requête SQL suivante et expliquer le résultat obtenu.

```
1 SELECT COUNT(pk_code)
2 FROM tUv
3 WHERE fk_etu='1800675001066';
```

Question 2

Exécuter la requête SQL suivante et expliquer le résultat obtenu.

```
SELECT fk_etu, COUNT(pk_code)
FROM tUv
GROUP BY fk_etu;
```

Question 3

Compléter la requête SQL suivante afin qu'elle renvoie, pour chaque UV, le nombre d'étudiants inscrits.

```
SELECT _______, COUNT(______)
FROM tUv
GROUP BY _______
```

À l'issue de cette série d'exercices, vous devez savoir définir les termes suivants :

- table ou relation
- schéma relationnel
- domaine
- clé
- clé étrangère
- opérations de projection, restriction, jointure, produit

Exercices

1. Exercice : Lab 0

Description du problème

[30 min]

Un laboratoire souhaite gérer les médicaments qu'il conçoit.

Un médicament est décrit par un nom, qui permet de l'identifier. En effet il n'existe pas deux médicaments avec le même nom. Un médicament comporte une description courte en français, ainsi qu'une description longue en latin. On gère aussi le conditionnement du médicament, c'est à dire le nombre de pilules par boîte (qui est un nombre entier).

Ce problème est un exemple très simple que l'on pourra modéliser avec une base de données relationnelle à une seule table.

Vous pouvez tester vos instructions SQL ici : *Db Disco*∗

Question 1

Proposer une clarification du problème (par exemple sous la forme d'une liste des propriétés de la relation visée). On fera l'hypothèse qu'un seul type d'utilisateur existe et qu'il peut à la fois modifier et consulter toutes les données.

Question 2

Proposer un exemple de données.

Question 3

Dessiner un modèle conceptuel de données en UML. Il ne contient qu'une seule classe.

Indice :

Modélisation conceptuelle de données *(cf. p.15)*

Question 4

Proposer un modèle logique de données sous forme de schéma relationnel. Il ne contient qu'une seule relation.

Indice :

Modélisation logique de données *(cf. p.16)*

Question 5

Proposer une implémentation en SQL standard de votre modèle relationnel. Il ne contient qu'une seule instruction CREATE TABLE.

Utiliser *Db Disco*∗ pour valider votre code.

Indice :

Langage de données : l'exemple du langage SQL *(cf. p.8)*

Question 6

Écrivez les instructions SQL permettant d'insérer vos données de test dans votre base de données.

Utiliser *Db Disco∗* pour valider votre code.

Indice :

Langage de données : l'exemple du langage SQL *(cf. p.8)*

2. Exercice : Site de livres électroniques sous licence libre

[30 min]

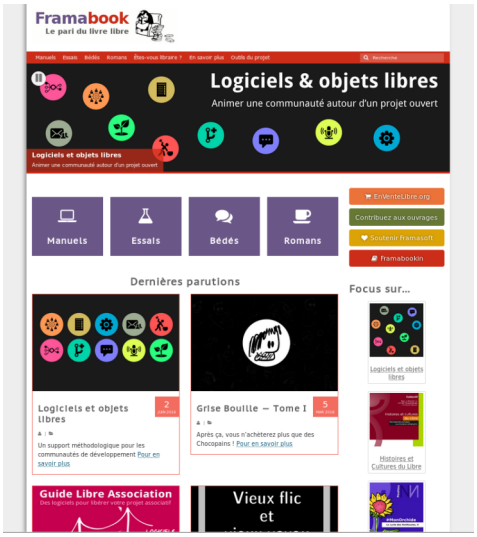
Vous avez comme projet de proposer un site de référencement de livres électroniques au format EPUB sous licence libre.

Question

Proposer, **librement**, une petite note de clarification d'environ une page (500 mots), qui identifiera :

- cinq ou six objets dotés de cinq ou six propriétés chacun
- deux ou trois rôles
- une dizaine de fonctions

Vous vous inspirerez en priorité du site similaire Framabook[1], dont la simplicité vous aidera à démarrer, mais vous pouvez chercher des éléments complémentaires dans votre propre expérience, ou bien sur d'autres sites similaires comme inlibroveritas.net[2], sur des sites commerciaux grands publics comme Amazon ou la Fnac, ou des sites plus spécialisés comme C&F éditions[3] ou Epagine[4].



[1] http://framabook.org/
[2] http://www.inlibroveritas.net/
[3] http://cfeditions.com
[4] http://www.epagine.fr/

Contenus annexes

1. Pourquoi des SGBD ?

Jadis...

Avant l'avènement des SGBD, chaque application informatique dans l'entreprise impliquait sa propre équipe de développement, ses propres supports physiques, ses propres fichiers, ses propres normes, ses propres langages, etc.

Conséquences...

L'existence conjointe et croissante de ces applications indépendantes a des effets négatifs, tels que :

- La multiplication des tâches de saisie, de développement et de support informatique ;
- La redondance anarchique des informations dans les fichiers ;
- L'incohérence des versions simultanées de fichiers ;
- La non-portabilité des traitements en raison des différences dans les formats et langages ;
- La multiplication des coûts de développement et de maintenance des applications.

Problèmes...

Les conséquences précédemment citées se répercutent sur l'entreprise en générant des problèmes humains et matériels ;

- Coûts en personnels qualifiés et en formations ;
- Remise des pouvoirs de décision entre les mains de spécialistes informatiques ;
- Tout changement matériel ou logiciel a un impact sur les applications ;
- Tout changement de la structure des données nécessite de modifier les programmes.

Or...

En réalité les applications ne sont jamais totalement disjointes, des données similaires (le cœur de l'information d'entreprise) sont toujours à la base des traitements.

On peut citer typiquement :

- Les données comptables
- Les données clients et fournisseurs
- Les données relatives à la gestion des stocks
- Les données relatives aux livraisons
- Les données marketing et commerciales
- Les données relatives au personnel
- ...

2. Définition du mouvement NoSQL

Définition

« Le NoSQL regroupe de nombreuses bases de données, récentes pour la plupart, qui se caractérisent par une logique de représentation de données non relationnelle et qui n'offrent donc pas une interface de requêtes en SQL.

http://blog.xebia.fr/2010/04/21/nosql-europe-tour-dhorizon-des-bases-de-donnees-nosql/ »

Attention

NoSQL signifie *Not Only SQL* et non pas *No SQL*, il s'agit de compléments aux SGBDR pour des besoins spécifiques et non de solutions de remplacement.

Exemple

- BD orientée clé-valeur
- BD orientée graphe
- BD orientée colonne
- BD orientée document

Complément

http://blog.xebia.fr/2010/04/21/nosql-europe-tour-dhorizon-des-bases-de-donnees-nosql/

Glossaire

Cahier des charges (CDC)

Document qui est fourni dans le cadre de la réalisation d'un projet pour en décrire les attendus.

Abréviations

BD : Base de Données
E-A : Entité-Association
LDD : Langage de Définition de Données
LMD : Langage de Manipulation de Données
MCD : Modèle Conceptuel de Données
OMG : Object Management Group
SGBD : Système de Gestion de Bases de Données
SGBDR : Système de Gestion de Bases de Données Relationnelles
SGDBR : Système de Gestion de Bases de Données Relationnelles
SQL : Structured Query Language
UML : Unified Modeling Language

Références

dbdisco *DB Disco* est une application web permettant de créer une base de données temporaire (les données sont effacées à la déconnexion).

http://pic.crzt.fr/dbdisco

Bibliographie

Tardieu H., Rochfeld A., Colleti R., *Méthode MERISE Tome 1 : Principes et outils*, Les Editions d'Organisation, 1983.

Index

Crédits des ressources

Conception en quatre étapes p. 11

http://creativecommons.org/licenses/by-sa/4.0/fr/, Stéphane Crozat (Monde : Adrien Facélina https://commons.wikimedia.org)