stph.scenari-community.org/bdd

Conception des bases de données I

Introduction aux bases de données relationnelles

bdd1.pdf

utc Formation

Stéphane Crozat

Publiée le 14 février 2017

Table des matières

I

Introduction générale aux bases de données

A. 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

a) 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 : Définition lâche de base de données : un ensemble de données

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

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.

Exemple : Base de données relationnelle

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

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

Tableau 1 Base de données de classification classique des espèces animales

Fondamental: Fonctions d'une base de données

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)**
- **...**

b) Système de gestion de bases de données (SGBD)

Définition : Système de Gestion de Bases de Données

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).

Exemple : Exemples de SGBD

- 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.

Complément : SGBD relationnel et non-relationnel

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.

c) 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.

Exemple : Application web

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.

Exemple : Twitter

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 (Twitter utilise une base de données NoSQL Apache Cassandra).

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*...

Exemple : Compagnie aérienne

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...

Exemple : Application de bureau Access

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.

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

Rappel : Base de données relationnelle

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

Chaque case de la table contient une information atomique.

Définition : Objet (ligne)

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.

Définition : Propriété et domaine (colonne)

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.

Fondamental: Domaine

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

Définition : Donnée en relationnel (cellule)

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

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

Tableau 2 Une table ou relation (en relationnel)

Exemple

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

Tableau 3 Exemple de relation instanciée

Attention : Atomicité (contre-exemple)

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.

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

Tableau 4 Un mauvais exemple de relation : les données ne sont pas atomiques (il y a plusieurs données par case de la table)

e) Langage de données : l'exemple du langage SQL

Définition : Langage de données

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

Fondamental: SQL

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**

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

```
1  CREATE TABLE Etudiant (
2    NumEtu : integer PRIMARY KEY,
3    Nom : varchar,
4    Ville : varchar)
```

Cette instruction permet de créer une relation "Etudiant" comportant les propriétés "NumEtu", "Nom" et "Ville" de domaines, respectivement, entier, texte et texte.

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

```
1  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.

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

```
1  SELECT Nom
2  FROM Etudiant
3  WHERE Ville = 'Compiègne'
```

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

Complément : Autres langages de données

- 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

a) Exercice : Étapes de la conception d'une base de données relationnelle

Mettre dans l'ordre les étapes de conception suivantes.

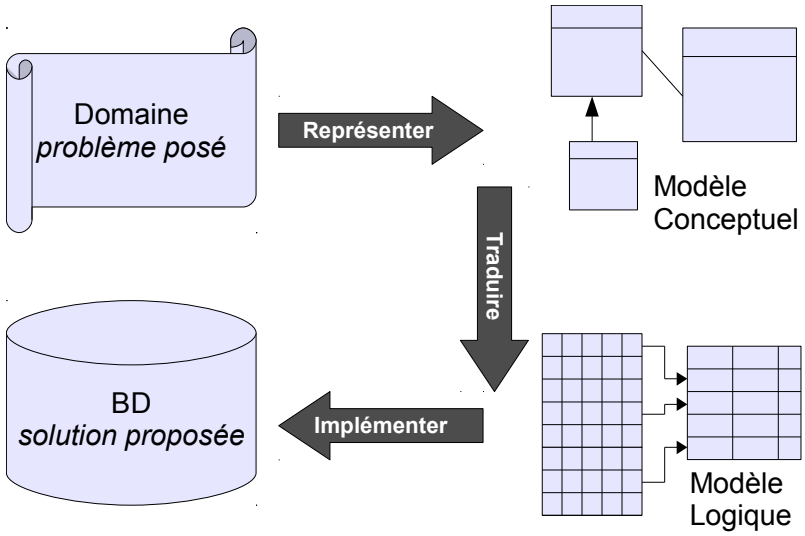
1. Modélisation conceptuelle en UML ou E-A
2. Création du code SQL pour un SGBDR
3. Analyse de la situation existante et des besoins
4. Élaboration du modèle logique en relationnel

Réponse : ___ ___ ___ ___

b) Méthodologie générale de conception d'une base de données

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

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)



Graphique 1 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**

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

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.

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

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.

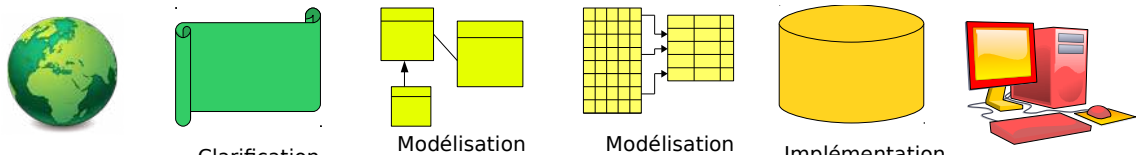
Remarque : Les étapes de traduction logique et d'implémentation

Des logiciels spécialisés (par exemple Objecteering [w_objecteering]) 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.

c) 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).

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.

d) 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.

Méthode : L'analyse de documents existants

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.

Exemple : Exemples de document existants à analyser

- 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.

Méthode : Le recueil d'expertise métier

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.

Exemple : Exemples d'experts à consulter

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

Méthode : Le dialogue avec les usagers

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.).

Exemple : Exemples d'utilisateurs avec qui dialoguer

- 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.

Méthode : L'étude des autres systèmes informatiques existants

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.

Exemple : Exemples d'autres systèmes coexistants à étudier

- 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.

Complément : Méthodes d'analyse

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.

e) 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

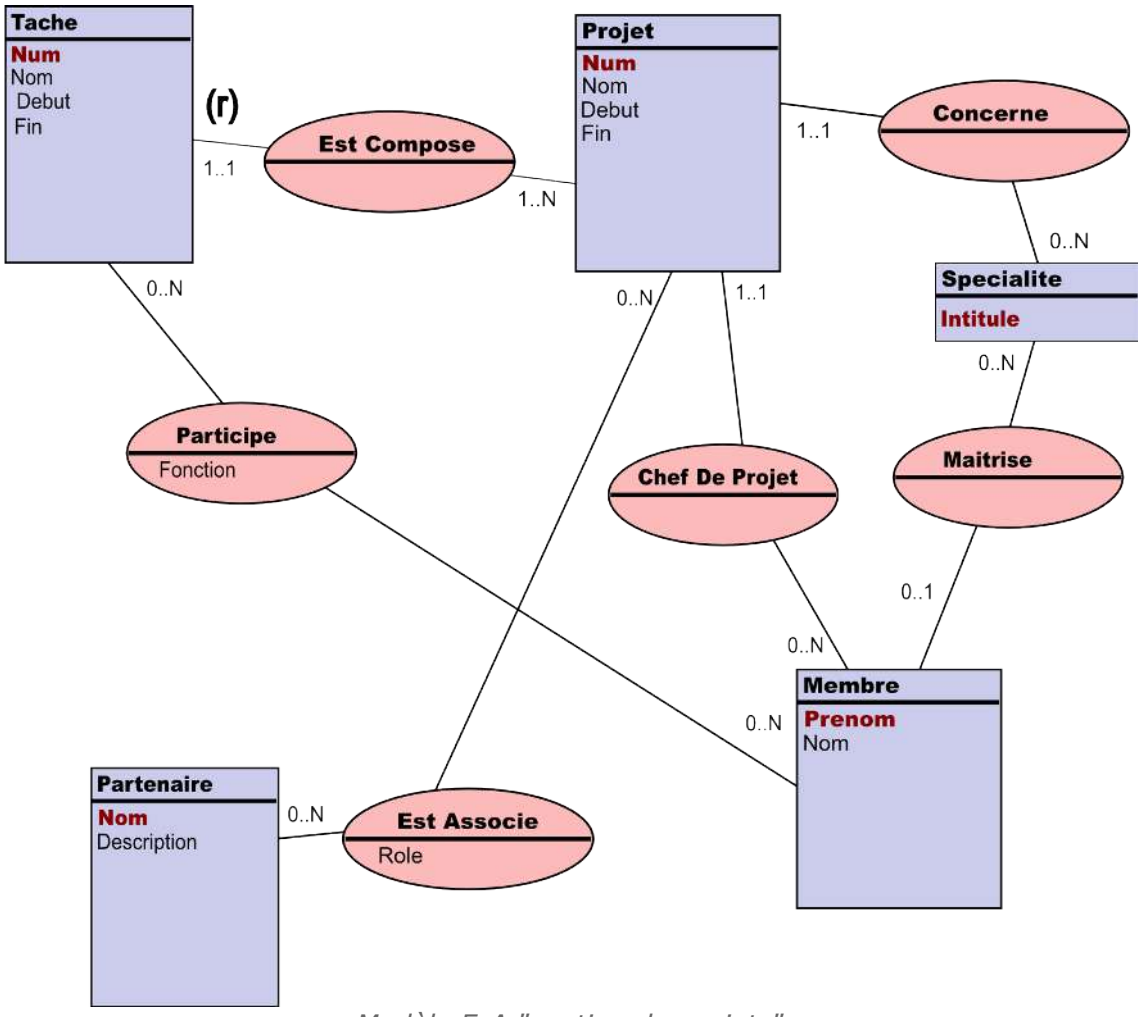
Remarque

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"

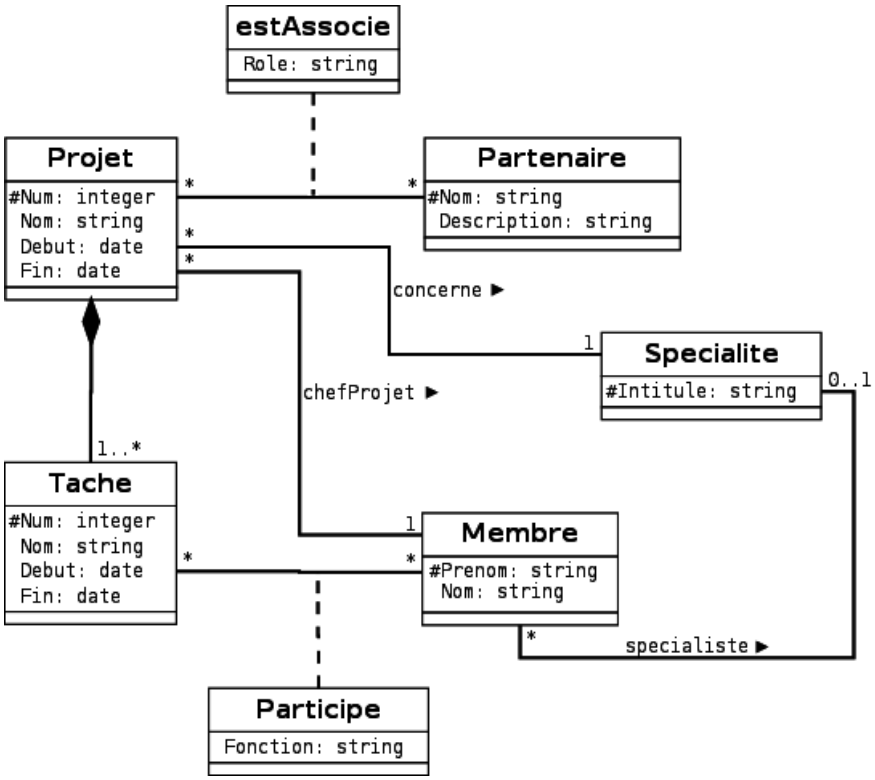
UML

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"

f) Modélisation logique de données

Définition : Modèle logique de données

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

Exemple : Modèle logique de données relationnel

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

Exemple : Schéma d'une relation

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

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

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

Exemple : Instance de base de données

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

Complément : Exemple de formalismes de modélisation logique

- Le modèle CODASYL, antérieur au modèle relationnel est un modèle hiérarchique (Tardieu, 1983 [Tardieu83]).
- 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.

Complément : Voir aussi

Brève introduction aux bases de données NoSQL

g) Synthèse : Les trois niveaux de conception

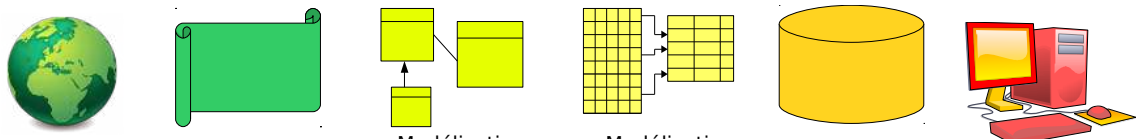
- Niveau Conceptuel

  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

B. Exercices

1. 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 *dbdisco.crzt.fr*, et conserver la fenêtre du navigateur ouverte.

- **Db Disco**[dbdisco.crzt.fr]

a) 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';
```

b) 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.

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

Question 4

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

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

Question 5

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

c) 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.

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

```
1  INSERT INTO tUV (pk_code, fk_etu)
2  VALUES ('NF17', '1800675001066');
3  INSERT INTO tUV (pk_code, fk_etu)
4  VALUES ('NF26', '1800675001066');
5  INSERT INTO tUV (pk_code, fk_etu)
6  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');
```

d) 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;
```

e) 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.

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

Question 3

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

```
1  SELECT _______, COUNT(______)
2  FROM tUv
3  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

2. 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[dbdisco.crzt.fr]

Question 1

Proposer une clarification du problème (par exemple sous la forme d'une liste des propriétés de la relation visée).

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

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

Question 5

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

Indice :

Langage de données : l'exemple du langage SQL

Question 6

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

Indice :

Langage de données : l'exemple du langage SQL

Introduction à la modélisation conceptuelle de données avec UML

II

A. Cours

La **modélisation conceptuelle** est l'étape **fondatrice** du processus de conception de BD✯. Elle consiste à abstraire le problème réel posé pour en faire une reformulation qui trouvera une solution dans le cadre technologique d'un SGBD✯.

Si le modèle dominant en conception de bases de données a longtemps été le modèle E-A✯, le modèle UML✯ se généralise de plus en plus. Nous proposons ici une introduction au diagramme de classes à travers la représentation de classes et d'associations simples (il existe d'autres diagrammes UML, par exemple le diagramme de cas, et d'autres primitives de représentation dans le diagramme de classe, par exemple l'héritage).

1. Notion de modèle

Objectifs

Savoir ce qu'est un modèle
Savoir ce qu'est le langage UML

a) Exercice : Centre médical

[5 min]

Soit le modèle conceptuel suivant représentant des visites dans un centre médical. Quelles sont les assertions vraies selon ce schéma ?

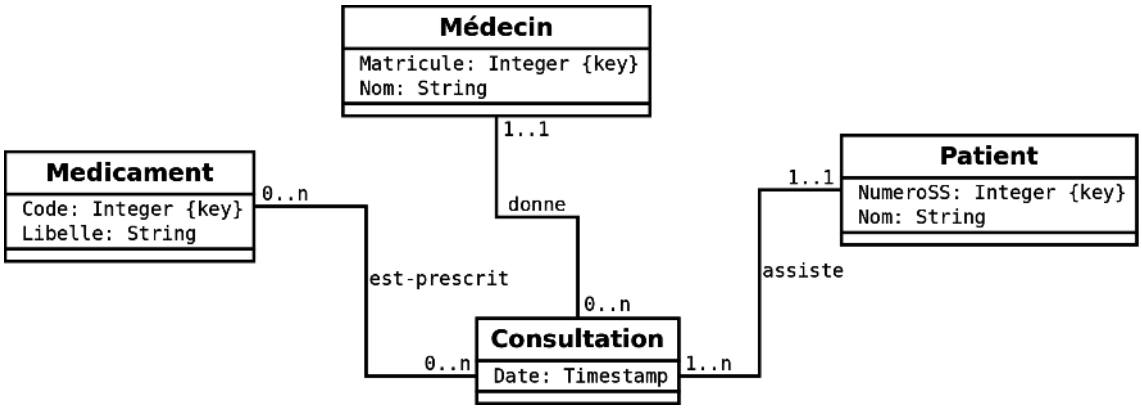


Image 1

☐	Un patient peut effectuer plusieurs visites.
☐	Tous les patients ont effectué au moins une consultation.
☐	Un médecin peut recevoir plusieurs patients pendant la même consultation.
☐	Un médecin peut prescrire plusieurs médicaments lors d'une même consultation.
☐	Deux médecins différents peuvent prescrire le même médicament.

b) Qu'est ce qu'un modèle ?

Définition : Modèle

« Modeling, in the broadest sense, is the cost-effective use of something in place of something else for some cognitive purpose. It allows us to use something that is simpler, safer or cheaper than reality instead of reality for some purpose. A model represents reality for the given purpose; the model is an abstraction of reality in the sense that it cannot represent all aspects of reality. » (Rothenberg, 1989 [Rothenberg et al., 1989], cité par Arribe, 2014 [Arribe, 2014])

« Système physique, mathématique ou logique représentant les structures essentielles d'une réalité et capable à son niveau d'en expliquer ou d'en reproduire dynamiquement le fonctionnement. » (TLFi)

Fondamental: Modèle

Un modèle est une représentation simplifiée de la réalité en vue de réaliser quelque chose.

c) Qu'est ce qu'un modèle en informatique ?

Définition : Modèle informatique

Un modèle informatique est une représentation simplifiée de la réalité en vue de réaliser un traitement avec un ordinateur.

Complément : Numérisation et abstraction : Toute information numérique a été codée selon un modèle donné

« Tout numérisation est une représentation de la réalité sous la forme d'une modélisation numérique. Cette modélisation procède d'une abstraction au sens où c'est une séparation d'avec le réel, au sens où c'est une construction destinée à la manipulation (algorithmique en l'occurrence) et au sens où c'est une simplification de la réalité. »

http://aswemay.fr/co/tropism-pres.html[1]

1 - http://aswemay.fr/co/tropism-pres.html

d) Qu'est ce qu'un bon modèle ?

Attention

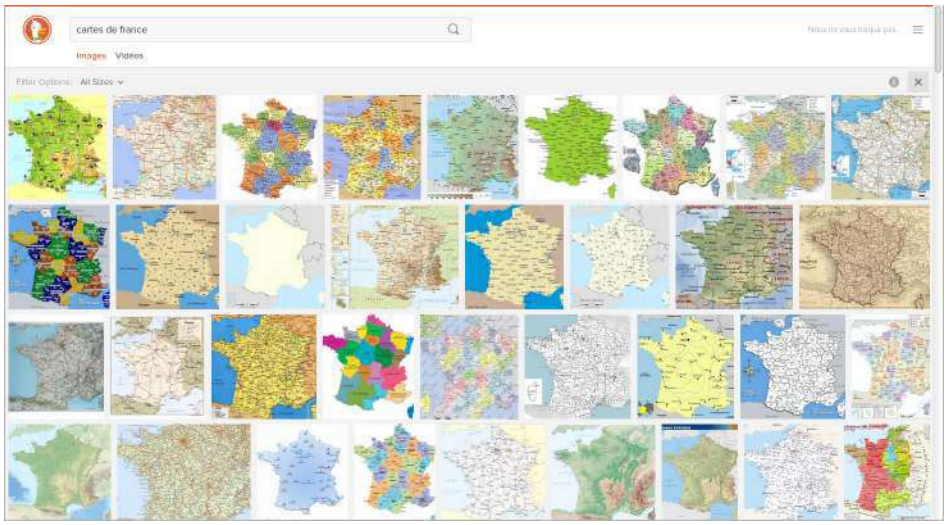
Un modèle est une abstraction, une simplification de la réalité, ce n'est pas la réalité, il n'est jamais complètement fidèle par construction.

Le seul modèle complètement fidèle à la réalité est la réalité elle-même, et ce n'est donc pas un modèle.

Exemple : La carte et le territoire

Une carte est un modèle d'un territoire. Elle est une représentation simplifiée destiné à un usage particulier :

- randonner à pied, en vélo ;
- se diriger en voiture sur des grands axes, sur des axes secondaires ;
- voler en avion de tourisme, en avion de ligne ;
- naviguer sur fleuve, sur mer ;
- étudier les frontières d'une région, d'un pays, de la terre ;
- étudier la démographie, l'économie... ;
- ...



Fondamental

À partir de cet exemple on notera que :

1. **Un modèle est orienté par un usage.**
   Chacune de ces cartes est très différente selon ce que l'on veut faire.
2. **Un modèle ne cherche pas à être proche de la réalité.**
   Chacune de ces cartes est très différente de la réalité qu'elle représente.
3. **Un modèle adresse un niveau d'information qui existe mais qui n'est pas accessible dans la réalité.**
   Chacune de ces cartes permet quelque chose que ne permet pas l'accès direct à la réalité.

Méthode : Le rasoir d'Ockham : Entre deux modèles donnés le meilleur modèle est-il toujours le plus fourni ?

La méthode de raisonnement connue sous le nom de rasoir d'Ockham (du nom du philosophe éponyme) consiste à préférer les solutions les plus simples aux plus complexes, lorsqu'elles

semblent permettre également de résoudre un problème donné ; entre deux théories équivalentes, toujours préférer la plus simple.

Ce principe s'applique très bien à la modélisation : étant donné un objectif et plusieurs modèles possibles, il ne faut pas choisir a priori celui qui représente le plus de choses, mais préférer le plus simple dès qu'il couvre le besoin.

C'est un principe d'économie (il coûte moins cher à produire) et d'efficacité (car les éléments inutiles du modèle plus fourni nuiront à l'efficacité de la tâche).

Exemple

Ainsi, pour naviguer en voiture, il est plus simple de ne **pas** avoir sur la carte les chemins de randonnées qui ne sont pas praticables en voiture.

2. Introduction au diagramme de classes UML : classes et associations

Objectifs

Savoir faire un modèle conceptuel.
Savoir interpréter un modèle conceptuel.

a) Lab I

Description du problème

[15 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).
- À chaque médicament on associe une liste de contre-indications, généralement plusieurs, parfois aucune. Une contre-indication comporte un code unique qui l'identifie, ainsi qu'une description. Une contre-indication est toujours associée à un et un seul médicament.

Exemple de données

Afin de matérialiser notre base de données, nous obtenons les descriptions suivantes :

- Le **Chourix** a pour description courte *« Médicament contre la chute des choux »* et pour description longue *« Vivamus fermentum semper porta. Nunc diam velit, adipiscing ut tristique vitae, sagittis vel odio. Maecenas convallis ullamcorper ultricies. Curabitur ornare. »*. Il est conditionné en boîte de 13.

  Ses contre-indications sont :
  - CI1 : Ne jamais prendre après minuit.
  - CI2 : Ne jamais mettre en contact avec de l'eau.
- Le **Tropas** a pour description courte *« Médicament contre les dysfonctionnements intellectuels »* et pour description longue *« Suspendisse lectus leo, consectetur in tempor sit amet, placerat quis neque. Etiam luctus porttitor lorem, sed suscipit est rutrum non. »*. Il est conditionné en boîte de 42.

  Ses contre-indications sont :
  - CI3 : Garder à l'abri de la lumière du soleil

Question 1

Réaliser le modèle conceptuel de données en UML du problème.

Question 2

Étendre le modèle conceptuel UML afin d'ajouter la gestion des composants. Un composant est identifié par un code unique et possède un intitulé. Tout médicament possède au moins un composant, souvent plusieurs. Tout composant peut intervenir dans la fabrication de plusieurs médicaments. Il existe des composants qui ne sont pas utilisés pour fabriquer des médicaments et que l'on veut quand même gérer.

b) Présentation d'UML

UML✯ est un langage de représentation destiné en particulier à la modélisation objet. UML est devenu une norme OMG✯ en 1997.

UML propose un formalisme qui impose de "penser objet" et permet de rester indépendant d'un langage de programmation donné. Pour ce faire, UML normalise les concepts de l'objet (énumération et définition exhaustive des concepts) ainsi que leur notation graphique. Il peut donc être utilisé comme un moyen de communication entre les étapes de spécification conceptuelle et les étapes de spécifications techniques.

Fondamental: Diagramme de classe

Le diagramme de classes est un sous ensemble d'UML qui s'attache à la description statique d'un modèle de données représentées par des classes d'objets.

Remarque

Dans le domaine des bases de données, UML peut être utilisé à la place du modèle E-A✯ pour modéliser le domaine. De la même façon, un schéma conceptuel UML peut alors être traduit en schéma logique (relationnel ou relationnel-objet typiquement).

c) Classes

Définition : Classe

Une classe est un type abstrait caractérisé par des propriétés (attributs et méthodes) communes à un ensemble d'objets et permettant de créer des instances de ces objets, ayant ces propriétés.

Syntaxe

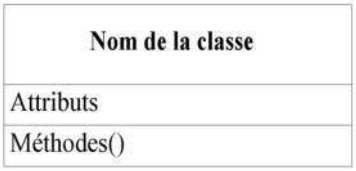


Image 2 Représentation UML d'une classe

Exemple : La classe Voiture

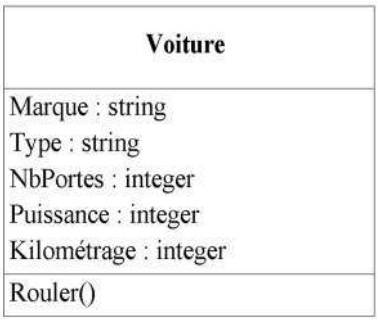


Image 3 Exemple de classe représentée en UML

Exemple : Une instance de la classe Voiture

L'objet V1 est une instance de la classe Voiture.

V1 : Voiture

- Marque : 'Citroën'
- Type : 'ZX'
- Portes : 5
- Puissance : 6
- Kilométrage : 300000

Complément

La modélisation sous forme de diagramme de classes est une modélisation statique, qui met en exergue la structure d'un modèle, mais ne rend pas compte de son évolution temporelle. UML propose d'autres types de diagrammes pour traiter, notamment, de ces aspects.

d) Attributs

Définition : Attribut

Un attribut est une information élémentaire qui caractérise une classe et dont la valeur dépend de l'objet instancié.

Un attribut est typé : Le domaine des valeurs que peut prendre l'attribut est fixé a priori.

- **Un attribut peut être multivalué** : Il peut prendre plusieurs valeurs distinctes dans son domaine.
- **Un attribut peut être dérivé** : Sa valeur alors est une fonction sur d'autres attributs de la classe
- **Un attribut peut être composé** (ou composite) : Il joue alors le rôle d'un groupe d'attributs (par exemple une adresse peut être un attribut composé des attributs numéro, type de voie, nom de la voie). Cette notion renvoie à la notion de variable de type `Record` dans les langages de programmation classiques.

Attention : On utilise peu les attributs dérivés et composés en UML

- **En UML on préfère l'usage de méthodes aux attributs dérivés. On utilisera toujours des méthodes dès que la valeur de l'attribut dérivé dépend d'autres attributs extérieurs à sa classe.**
- **En UML on préfère l'usage de compositions aux attributs composés. On utilisera toujours des compositions pour les attributs composés et multivalués.**

Syntaxe

```
attribut:type
attribut_multivalué[nbMinValeurs..nbMaxValeurs]:type
/attribut_dérivé:type
attribut_composé
    - sous-attribut1:type
    - sous-attribut2:type
    - ...
```

Exemple : La classe Personne

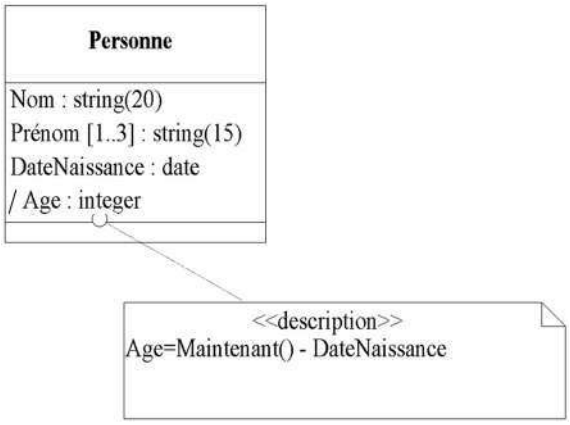


Image 4 Représentation d'attributs en UML

Dans cet exemple, les attributs Nom, Prénom sont de type *string*, l'un de 20 caractères et l'autre de 10, tandis que DateNaissance est de type *date* et Age de type *integer*. Prénom est un attribut multivalué, ici une personne peut avoir de 1 à 3 prénoms. Age est un attribut dérivé, il est calculé par une fonction sur DateNaissance.

Complément : Voir aussi

Méthodes

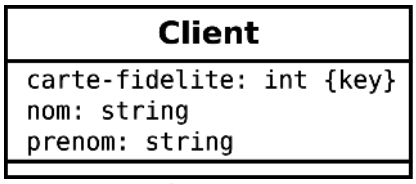
Composition

e) Repérage des clés

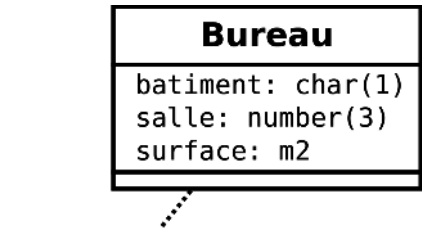
Un attribut ou un groupe d'attributs peut être annoté comme étant **clé** s'il permet d'identifier de façon unique un objet de la classe.

On ajoute le symbole `{key}` à côté du ou des attributs concernés.

Exemple



Clé en UML



Clé composée de deux attributs

Méthode

Le repérage des clés n'est pas systématique en UML (la définition des clés se fera essentiellement au niveau logique). On cherchera néanmoins à repérer les clés rendues évidentes par la phase de clarification.

Attention

On n'ajoutera jamais de clé artificielle au niveau du MCD. Si aucune clé n'est évidente, on laisse la classe sans clé.

Attention : Attribut souligné et #

On trouvera dans ce cours des exemples d'attributs soulignés ou précédés de # pour exprimer l'unicité. Ce n'est pas une pratique standard et la notation `{key}` devrait lui être substituée.

- **Un attribut souligné est normalement un attribut de classe, ou *static*, en UML,**
- **Un attribut précédé de # est normalement un attribut protégé en UML.**

Mais les concepts d'attribut de classe et d'attribut protégé ne sont pas utilisés dans le cadre des bases de données.

f) Méthodes

Définition : Méthode

Une méthode (ou opération) est une fonction associée à une classe d'objet qui permet d'agir sur les objets de la classe ou qui permet à ces objets de renvoyer des valeurs (calculées en fonction de paramètres).

Syntaxe

```
1 methode(paramètres):type
```

Remarque : Méthodes et modélisation de BD

Pour la modélisation des bases de données, les méthodes sont surtout utilisées pour représenter des données calculées (à l'instar des attributs dérivées) ou pour mettre en exergue des fonctions importantes du système cible. Seules les méthodes les plus importantes sont représentées, l'approche est moins systématique qu'en modélisation objet par exemple.

Remarque : Méthodes, relationnel, relationnel-objet

Lors de la transformation du modèle conceptuel UML en modèle logique relationnel, les méthodes **ne seront généralement pas implémentées**. Leur repérage au niveau conceptuel sert donc surtout d'aide-mémoire pour l'implémentation au niveau applicatif.

Au contraire, un modèle logique relationnel-objet permettra l'implémentation de méthodes directement associées à des tables. Leur repérage au niveau conceptuel est donc encore plus important.

Complément

Transformation des méthodes par des vues

g) Associations

Définition : Association

Une association est une relation logique entre deux classes (association binaire) ou plus (association n-aire) qui définit un ensemble de liens entre les objets de ces classes.

Une association est **nommée**, généralement par un verbe. Une association peut avoir des propriétés (à l'instar d'une classe). Une association définit le nombre minimum et maximum d'instances autorisée dans la relation (on parle de cardinalité).

Syntaxe

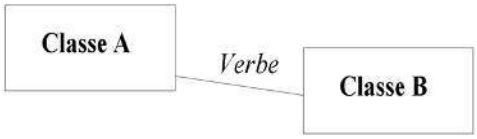


Image 5 Notation de l'association en UML

Attention

Le nom de l'association (verbe qui la décrit) est obligatoire, au même titre que le nom d'une classe ou d'un attribut.

Remarque

Une association est généralement bidirectionnelle (c'est à dire qu'elle peut se lire dans les deux sens). Les associations qui ne respectent pas cette propriété sont dites unidirectionnelles ou à navigation restreinte.

Exemple : L'association Conduit

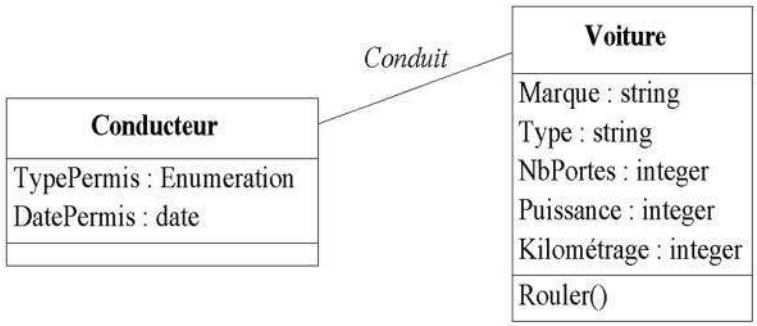


Image 6 Représentation d'association en UML

L'association Conduit entre les classes Conducteur et Voiture exprime que les conducteurs conduisent des voitures.

Complément : Voir aussi

- *Cardinalité*
- *Explicitation des associations*
- *Associations ternaires*
- *Contraintes sur les associations*

h) Cardinalité

Définition : Cardinalité d'une association

La cardinalité d'une association permet de représenter le nombre minimum et maximum d'instances qui sont autorisées à participer à la relation. La cardinalité est définie pour les deux sens de la relation.

Syntaxe

Si min_a (resp. max_a) est le nombre minimum (resp. maximum) d'instances de la classe A autorisées à participer à l'association, on note sur la relation, à côté de la classe A : $min_a..max_a$.

Si le nombre maximum est indéterminé, on note n ou *.

Attention

La notation de la cardinalité en UML est opposée à celle adoptée en E-A. En UML on note à gauche (resp. à droite) le nombre d'instances de la classe de gauche (resp. de droite) autorisées dans l'association. En E-A, on note à gauche (resp. à droite) le nombre d'instances de la classe de droite (resp. de gauche) autorisées dans l'association.

Remarque

Les cardinalités les plus courantes sont :

- 0..1 (optionnel)
- 1..1 ou 1 (un)
- 0..n ou 0..* ou * (plusieurs)
- 1..n ou 1..* (obligatoire)

Exemple : La cardinalité de l'association Possède

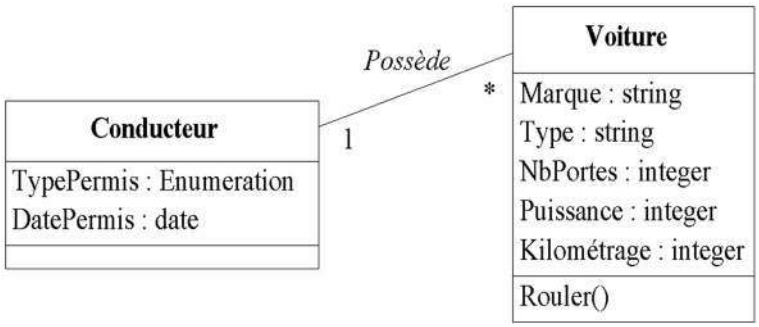


Image 7 Représentation de cardinalité en UML

Ici un conducteur peut posséder plusieurs voitures (y compris aucune) et une voiture n'est possédée que par un seul conducteur.

i) Classe d'association

Définition : Classe d'association

On utilise la notation des classes d'association lorsque l'on souhaite ajouter des propriétés à une association.

Syntaxe : Notation d'une classe d'association en UML

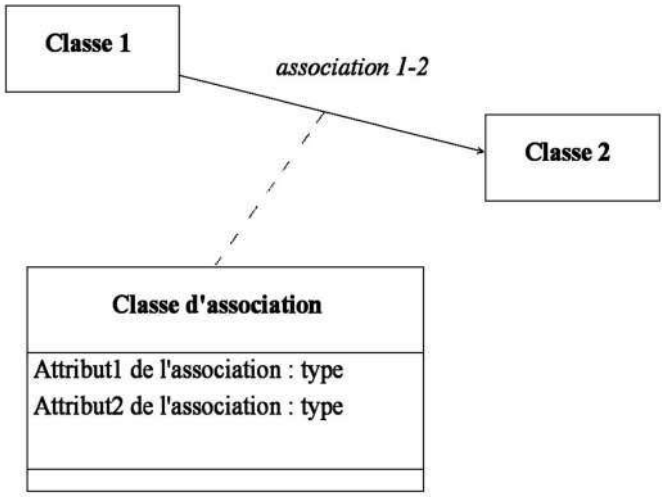


Image 8 Notation d'une classe d'association en UML

Exemple : Exemple de classe d'association

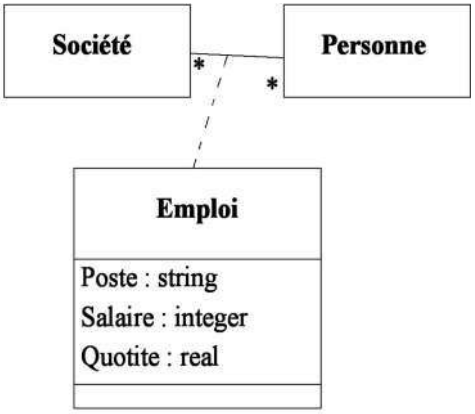


Image 9 Emplois

B. Exercices

1. Exercice : Lire l'UML

[15 min]

Tennis

Le schéma suivant représente les rencontres lors d'un tournoi de tennis. Quelles sont les assertions vraies selon ce schéma ?

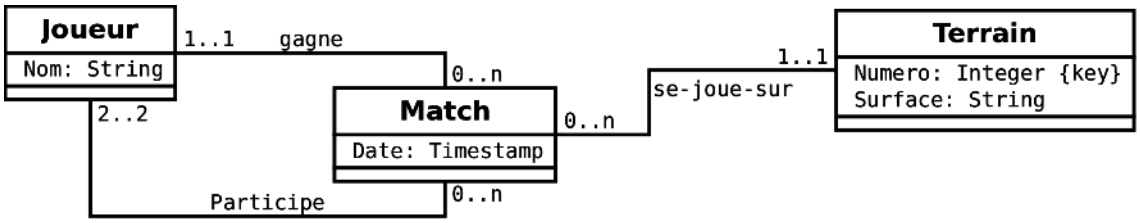


Image 10

- [ ] On peut jouer des matchs de double.
- [ ] Un joueur peut gagner un match sans y avoir participé.
- [ ] Il peut y avoir deux matchs sur le même terrain à la même heure.
- [ ] Connaissant un joueur, on peut savoir sur quels terrains il a joué.

Journal

Voici le schéma conceptuel du système d'information (très simplifié) d'un quotidien. Quelles sont les assertions vraies selon ce schéma ?

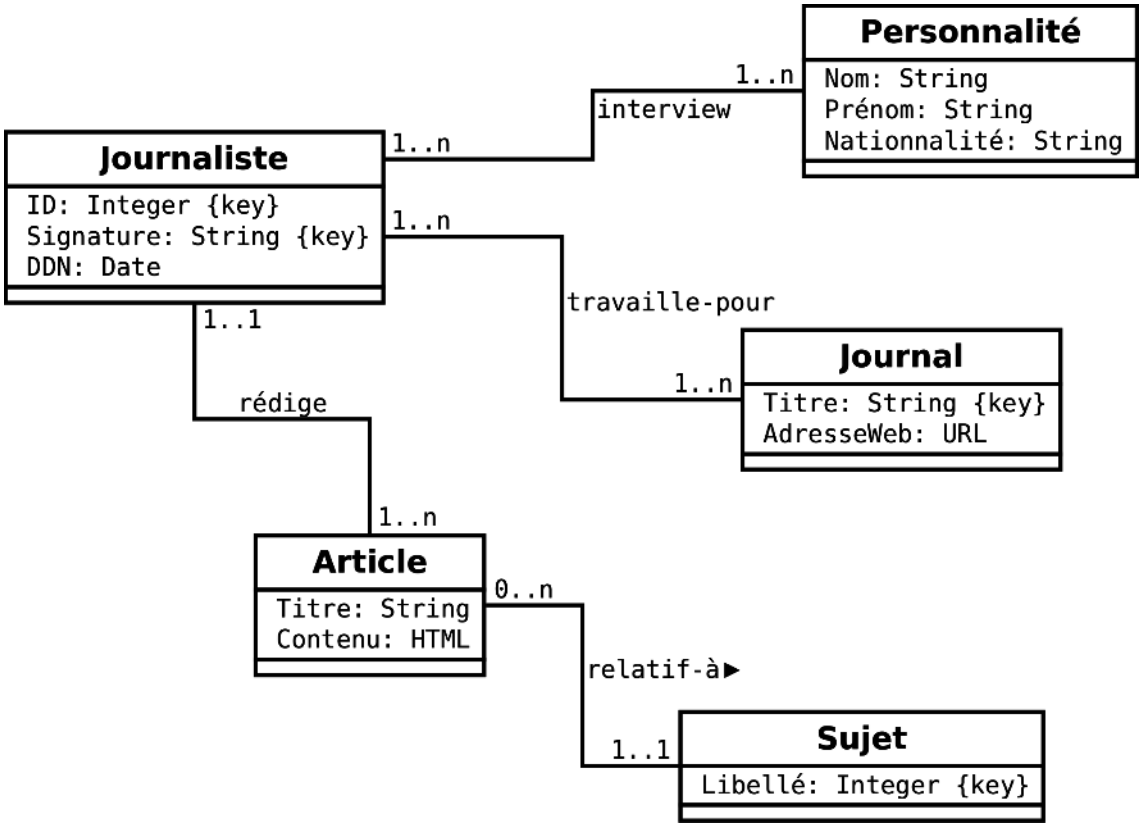


Image 11

☐	Un article peut être rédigé par plusieurs journalistes.
☐	Un article peut être publié plusieurs fois dans le même journal.
☐	Un article peut être publié dans un journal par un journaliste qui ne travaille pas pour ce journal.
☐	Il peut y avoir plusieurs articles sur le même sujet.
☐	Un journaliste peut interviewer une personnalité sans faire d'article à ce propos.

Logistique

Une société de transport routier veut installer un système d'information pour rendre plus efficace sa logistique. Embauché au service informatique de cette compagnie, vous êtes donc chargé de reprendre le travail déjà effectué (c'est à dire le schéma suivant).

Quelles sont les assertions vraies selon ce schéma ?

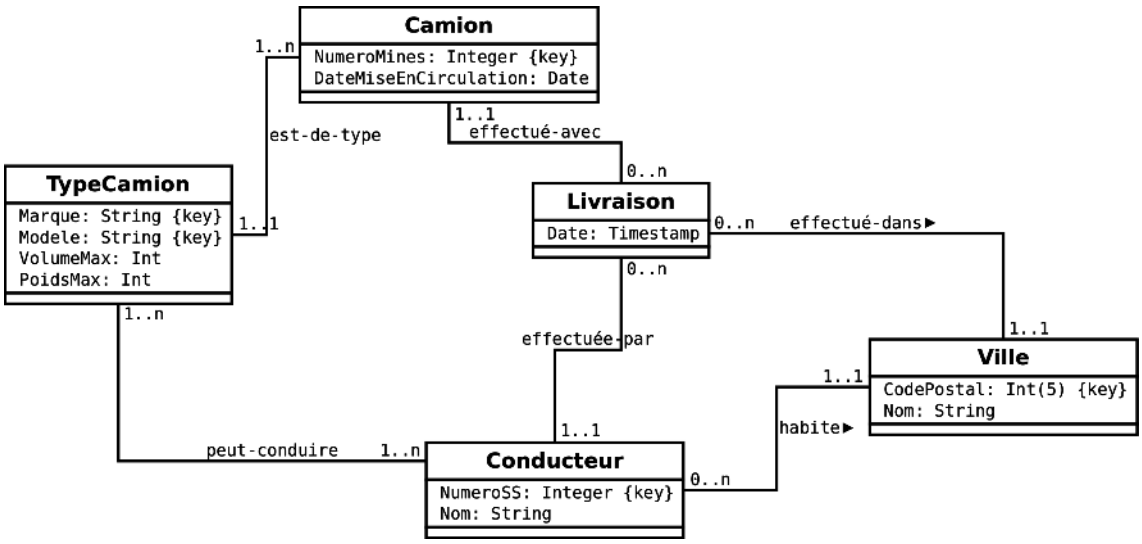


Image 12

☐	Un conducteur peut conduire plusieurs camions.
☐	Un conducteur peut conduire un camion sans y être autorisé.
☐	Il peut y avoir plusieurs conducteurs pour le même camion.
☐	Un conducteur peut livrer sa propre ville

2. Gestion d'une coopérative viticole

[20 minutes]

Cet exercice a été inspiré par Bases de données : objet et relationnel [Gardarin99].

On considère une base "Coopérative" qui possède les caractéristiques suivantes :

- Un vin est caractérisé par un numéro entier unique `nv`, un cru, une année de production et un degré.
- Un viticulteur est caractérisé par un numéro entier unique `nvt`, un nom et une ville.

- Un viticulteur produit un ou plusieurs vins et réciproquement, un vin est produit par un ou plusieurs producteurs (éventuellement aucun).
- Les buveurs sont caractérisés par un numéro de buveur `nb`, un nom, prénom et une adresse (limitée à la ville pour simplifier).
- Un buveur consomme des vins et peut passer des commandes pour acheter des vins.

Question 1

Lister tous les types d'objet à considérer, les attributs associés et les domaines de valeurs de ces attributs. Repérer les éventuelles clés.

Question 2

Lister toutes les associations à considérer et indiquer leurs cardinalités.

Question 3

Donner le diagramme UML de cette situation.

3. Cours et intervenants

[20 min]

On souhaite réaliser une base de données pour gérer les cours dispensés dans une école d'ingénieur, ainsi que les personnes qui interviennent dans ces cours.

Chaque cours est identifié par une année et un numéro. Chaque cours a donc un numéro unique localement à chaque année. Un cours possède un titre et un type ('C' pour Cours, 'TD' ou 'TP'). Un cours possède également une date de début, et une date de fin, qui est toujours de 5 jours après la date de début.

Chaque intervenant est identifié par son nom (deux intervenants ne peuvent pas avoir le même nom). Il a un prénom, un bureau, un ou plusieurs numéros de téléphones (jusqu'à trois numéros) et des spécialités. Un bureau est défini par un centre ('R' pour Royallieu, 'BF' pour Benjamin Franklin et 'PG' pour Pierre Guillaumat), un bâtiment (une lettre de A à Z), et un numéro (inférieur à 1000). Les spécialités sont des couples de chaînes de caractères désignant un domaine (par exemple 'BD') et une spécialité (par exemple 'SGBDRO').

Chaque cours est donné par un unique intervenant.

Voici un exemple : Le cours 'Machines universelles', n°21 de l'année 2014 est donné par Alan Turing entre le 05/01/2014 et le 10/01/2014. C'est un cours de type 'C'. Alan Turing a le bureau 666 au bâtiment X de PG. Il a les numéros de téléphone 0666666666 et 0766666666. Il possède les spécialités suivantes :

- Domaine : Mathématique ; Spécialité : Cryptographie
- Domaine : Informatique ; Spécialité : Algorithmie
- Domaine : Informatique ; Spécialité : Intelligence Artificielle

Question

Réaliser le modèle UML de la base de données. Préciser les clés et les types des attributs.

4. Gestion du personnel

[30 minutes]

Le service de gestion du personnel d'une entreprise désire s'équiper d'un outil lui permettant de gérer informatiquement ses employés, leurs salaires et leurs congés. Les spécifications suivantes ont pu être mises en exergue par une analyse des besoins réalisée auprès des utilisateurs du service du personnel.

Généralités :

- tout employé est identifié par un nom, un prénom et une date de naissance ;

- tout employé remplit une fonction et appartient à un service ;
- pour chaque employé on gère la date d'embauche et la quotité (c'est à dire le pourcentage de temps travaillé par rapport au temps plein, en cas de travail à temps partiel).

Gestion des salaires :

- pour chaque employé on gère l'historique de ses salaires dans l'entreprise, chaque salaire étant affecté à une période de temps ;
- un salaire est composé d'un salaire brut, de charges patronales et de charges salariales ;
- on cherchera à partir des ces données à obtenir également le salaire chargé (brut + charges patronales), et le salaire net (brut - charges salariales), et ce en particulier pour le salaire en cours (celui que touche actuellement le salarié).

Gestion des congés :

- pour chaque employé, on mémorise chaque congé pris (posant qu'un congé concerne toujours une ou plusieurs journées entières) ;
- chaque employé a le droit aux jours de congés suivants :
  - 25 jours (pour une quotité de 1) et 25 x quotité sinon,
  - chaque fonction ouvre les droits à un certain nombre de jours de RTT,
  - chaque service ouvre les droits à un certain nombre de jours de RTT,
  - chaque tranche de 5 ans passés dans l'entreprise donne droit à 1 jour supplémentaire,
  - les employés de plus de 40 ans ont un jour supplémentaire, et ceux de plus de 50 ans deux.
- pour chaque employé on cherchera à connaître le nombre total de jours de congés autorisés, le nombre de jours pris et le nombre de jours restants sur l'année en cours.

Question

Réaliser le diagramme de classes permettant de modéliser ce problème.

La modélisation logique relationnelle

III

A. Cours

Le modèle relationnel est aux fondements des SGBDR★. Il a été, et continue d'être, le modèle théorique dominant pour la représentation logique des base de données, même si le mouvement NoSQL propose des alternatives.

Le modèle relationnel permet de reformuler le modèle conceptuel dans un formalisme - le tableau - beaucoup plus proche de l'implémentation informatique, bien qu'encore indépendant d'une solution technologique particulière.

1. Introduction au modèle relationnel

Objectifs

Connaître les fondements théoriques du modèle relationnel.

a) Niveau logique

Le niveau logique est le lien entre le niveau conceptuel et l'implémentation effective de l'application. Le modèle conceptuel étant un modèle formel, le modèle logique a pour vocation d'être également un modèle formel, mais spécifiant non plus la réalité existante ou recherchée comme le modèle conceptuel, mais les données telles qu'elles vont exister dans l'application informatique.

Pour assumer cette fonction, le modèle relationnel [Codd70] s'est imposé en réaction aux insuffisances des modèles antérieurs, les modèles hiérarchique et réseau, et de part la puissance de ses fondements mathématiques.

Encore aujourd'hui dominant le modèle relationnel est un fondement indispensable à la conception de bases de données.

Rappel

Méthodologie générale de conception d'une base de données

b) Définition du modèle relationnel

Introduction

Le modèle relationnel a été introduit par Codd [Codd70], en 1970 au laboratoire de recherche d'IBM de San José. Il s'agit d'un modèle simple et puissant à la base de la majorité des bases de données, encore aujourd'hui.

Les objectifs du modèle relationnel, formulés par Codd, sont les suivants :

- Assurer l'indépendance des applications et de la représentation interne des données
- Gérer les problèmes de cohérence et de redondance des données
- Utiliser des langages de données basés sur des théories solides

Définition : Modèle relationnel

On appelle modèle relationnel un ensemble de concepts permettant de formaliser logiquement la description d'articles de fichiers plats, indépendamment de la façon dont ils sont physiquement stockés dans une mémoire numérique.

Le modèle relationnel inclut des concepts pour la **description** de données, ainsi que des concepts pour la **manipulation** de données.

Fondamental: Représenter le monde en tables

Le modèle relationnel permet de représenter les données que l'on va gérer à l'aide d'un très petit nombre de concepts très simples :

- **Les relations ou tables : des lignes et des colonnes**
- **Les domaines de valeurs : chaque case d'une table prend une unique valeur dans un domaine pré-défini**
- **Les clés : il existe des cases dont les valeurs doivent être uniques et non nulles**
- **Les clés étrangères : il existe des cases qui doivent prendre une valeur existante dans les cases d'une autre table**

Complément : Extension du modèle relationnel

Le modèle relationnel est un standard, normalisé par l'ISO à travers son langage, le SQL. Il se veut néanmoins dès l'origine extensible, pour permettre de gérer des données plus complexes que les données tabulaires. Le modèle relationnel-objet est né de cette extension.

2. Les concepts fondamentaux du modèle relationnel : attributs, enregistrement, domaine

Objectifs

Connaître les fondements théoriques du modèle relationnel.

a) Domaine

Définition : Domaine

Ensemble, caractérisé par un nom, dans lequel des données peuvent prendre leurs valeurs.

Remarque

Un domaine peut-être défini en intension (c'est à dire en définissant les propriétés caractéristiques des valeurs du domaine, on parle aussi de compréhension) ou en extension (c'est à dire en énumérant toutes les valeurs du domaine)

Exemple : Domaines définis en intension

- Tous les entiers
- Les réels inférieur à 5
- Les booléen (vrai ou faux)
- Toutes les chaînes de 1 à 255 caractères
- Les valeurs monétaires, définie comme des décimaux avec deux chiffres après la virgule
- Les dates, définies comme des chaînes de 10 caractères comprenant des chiffres et des tirets selon le patron "00-00-0000"
- Les salaires, définis comme des valeurs monétaires compris entre 15.000 et 100.000

Exemple : Domaines définis en extension

- Couleur : {Bleu, Vert, Rouge, Jaune, Blanc, Noir}
- SGBD : {Hiérarchique, Réseau, Relationnel, Objet, Relationnel-Objet}

b) Exercice

*Indiquez quelle définition et quel exemple correspondent respectivement aux mots **intension** et **extension**.*

1 - {bleu, rouge, vert}
2 - Énonciation exhaustive de l'ensemble des objets du domaine
3 - Le domaine des couleurs
4 - Explicitation d'un domaine par la description de ses caractéristiques (en vue de sa compréhension abstraite, générale).

Intension Extension

c) Attribut et enregistrement

Définition : Attribut

On appelle attribut d'une relation, une colonne de cette relation. Un attribut est caractérisé par un nom et un domaine dans lequel il prend ses valeurs.

Synonymes : Champs, Propriété, Colonne

Définition : Enregistrement

On appelle enregistrement d'une relation, une ligne de cette relation. Un enregistrement prend une valeur pour chaque attribut de la relation.

Synonymes : Tuple, N-uplet, Vecteur, Ligne

Exemple

A	B
1	1
1	2
2	2

Tableau 5 Relation R

La relation R comporte les deux attributs A et B et les trois enregistrements <1,1>, <1,2> et <2,2>

Remarque : Attribut, domaine, ordre

Un attribut se distingue d'un domaine car il peut ne comporter que certaines valeurs de ce domaine.

Les colonnes de la relation ne sont pas ordonnées et elles ne sont donc repérées que par le nom de l'attribut.

Remarque : Valeur nulle

Un enregistrement peut ne pas avoir de valeur pour certains attributs de la relation, parce que cette valeur est inconnue ou inapplicable, sa valeur est alors "null".

d) Exemple : La relation "Vol"

Exemple

Numero	Compagnie	Avion	Départ	Arrivée	Date
AF3245	Air France	747	Paris	Oulan Bator	01-08-2002
AF6767	Air France	A320	Paris	Toulouse	30-07-2002
KLM234	KML	727	Paris	Amsterdam	31-07-2002

Tableau 6 Relation Vol

3. Clés et clés étrangères dans le modèle relationnel :

Objectifs

Connaître les notions de clés candidates, naturelles, artificielles, primaire, étrangère

Aborder le principe d'éclatement des relations et de non-redondance.

a) Clé

Définition : Clé

Une clé est un groupe d'attributs minimum qui permet d'identifier de façon univoque un tuple dans une relation.

Fondamental

Toute relation doit comporter au moins une clé, ce qui implique qu'une relation ne peut pas contenir deux tuples identiques.

Attention : Attributs de clés unique et non null

Afin d'être déterminants pour l'identification d'un enregistrement, tous les attributs d'une clé doivent être valués, c'est-à-dire qu'aucun ne peut avoir de valeur *null*.

Dire qu'un groupe d'attribut est une clé équivaut à dire qu'il est unique et non *null*.

Exemple : Numéro d'étudiant

- Le numéro d'étudiant d'une relation *Etudiant* est une bonne clé car il y aura systématiquement une valeur non nulle.
- Le groupe d'attributs (nom, prénom) d'une relation *Etudiant* est en général une mauvaise clé, car les homonymes existent.

b) Déterminer les clés

Détermination d'une clé

Définir un groupe d'attributs comme étant une clé nécessite une réflexion sémantique sur les données composant ces attributs, afin de s'assurer de leur unicité.

Fondamental

La définition des clés est un acte de modélisation, elle ne renvoie pas donc pas à une vérité intangible, mais à la réalité telle qu'elle est représentée dans le modèle que l'on élabore.

Exemple

L'attribut numéro de sécurité sociale d'une relation personne peut paraître une bonne clé a priori car son unicité est assurée. Mais tout le monde n'en dispose pas forcément (les enfants, des étrangers), donc ce n'est une clé que si l'on considère des personnes affiliées à la sécurité sociale.

c) Clé primaire et clés candidates

Définition : Clé primaire

Si plusieurs clés existent dans une relation, on en choisit une parmi celles-ci. Cette clé est appelée **clé primaire**.

La clé primaire est généralement choisie de façon à ce qu'elle soit la plus simple, c'est à dire portant sur le moins d'attributs et sur les attributs de domaine les plus basiques (entiers ou chaînes courtes typiquement).

Définition : Clés candidates

On appelle **clés candidates** l'ensemble des clés d'une relation qui n'ont pas été choisies comme clé primaire (elles étaient candidates à cette fonction).

d) Clé artificielle

Définition : Clé artificielle

S'il est impossible de trouver une clé primaire, ou que les clés candidates sont trop complexes, il est possible de faire appel à une **clé artificielle**. Une clé artificielle est un attribut supplémentaire ajouté au schéma de la relation, qui n'est lié à aucune signification, et qui sert uniquement à identifier de façon unique les enregistrements et/ou à simplifier les références de clés étrangères.

Définition : Clé signifiante

Une clé est signifiante si elle n'est pas artificielle.

Synonyme : Clé naturelle

Attention : Clé artificielle et niveau logique

Au niveau du modèle logique, il faut éviter la simplicité consistant à identifier toutes les relations avec des clés artificielles, et ne réserver cet usage qu'aux cas particuliers.

Conseil

1. Si au moins une clé naturelle composée d'un seul attribut existe en choisir une parmi celles-ci comme clé primaire
2. Sinon, choisir une clé naturelle composée de plusieurs attributs si elle ne pose pas de problème identifié

3. Toujours justifier l'emploi d'une clé artificielle (au niveau logique uniquement pour des raisons de complexité du modèle, les questions de performance sont étudiées au niveau physique)

Remarque : Clé artificielle et niveau physique, évolutivité, maintenance et performance

Au niveau de l'implémentation physique par contre, il est courant que des clés artificielles soient utilisées de façon systématique.

- Du point de vue de **l'évolutivité** de la BD, il existe toujours un risque qu'une clé non-artificielle perde sa propriété d'unicité ou de non-nullité.
- Du point de vue de **la maintenance** de la BD, il existe toujours un risque qu'une clé non-artificielle voit sa valeur modifiée et dans ce cas, la répercution de ce changement pour mettre à jour toutes les références peut poser problème.
- Du point de vue de **la performance** de la BD, les clés non-artificielles ne sont pas en général optimisées en terme de type et de taille, et donc peuvent limiter les performances dans le cadre des jointures. Précisons néanmoins qu'inversement les clés artificielles ont pour conséquence de systématiser des jointures qui auraient pu être évitées avec des clés primaires signifiantes.

Exemple : Problème d'évolutivité posé par une clé signifiante

Soit le numéro de sécurité sociale la clé primaire d'une table d'une BD française, elle ne permettra pas d'entrer un individu non-français issu d'un pays ne disposant pas d'un tel numéro.

Exemple : Problème de maintenance posé par une clé signifiante

Soit le numéro de sécurité sociale la clé primaire d'une table d'une BD centrale dont les données sont exploitées par d'autres tables d'autres BD qui viennent "piocher" dans cette BD pour leurs propres usages, sans que la BD centrale ne connaisse ses "clients". Soit une erreur dans la saisie d'un numéro de sécurité sociale dans la BD centrale, si ce numéro est corrigé, il faudrait (ce qui n'est pas possible dans notre cas) impérativement en avertir toutes les bases utilisatrices pour qu'elles mettent à jour leurs références.

Exemple : Problème de performance posé par une clé signifiante

Soit le numéro de sécurité sociale la clé primaire d'une table comptant un million d'enregistrements, ce numéro est généralement un nombre à 13 chiffres ou une chaîne à 13 caractères, ce qui dans les deux cas est supérieur au nombre à 7 chiffres suffisant pour identifier tous les individus de la BD. Les performances seront donc toujours moins bonnes, lors des jointures, si une clé prend deux fois plus de place en mémoire que son optimum. Mais ajoutons que cette perte de performance n'a pas toujours de conséquence sur la réalité perceptible par les utilisateurs de la BD.

Inversement, soit une clé artificielle la clé primaire d'une table T1, par ailleurs référencée par une autre table T2. Soit le numéro de sécurité sociale un attribut clé de T1. Si l'on veut par requête disposer des informations de T2 ainsi que du numéro de sécurité sociale de T1, alors il faudra faire une jointure, tandis que si ce numéro signifiant avait été choisi comme clé primaire, cela n'aurait pas été nécessaire.

e) Clé étrangère

Définition : Clé étrangère

Une clé étrangère est un attribut ou un groupe d'attributs d'une relation R1 devant apparaître comme clé primaire dans une relation R2 afin de matérialiser une référence entre les tuples de R1 et les tuples de R2.

Une clé étrangère d'un tuple référence une clé primaire d'un autre tuple.

Attention

Seule une clé primaire peut être référencée par une clé étrangère, c'est même le seule fonction de la clé primaire : être la clé qui peut être référencée par les clés étrangères.

Définition : Contrainte d'intégrité référentielle

Une clé étrangère respecte la contrainte d'intégrité référentielle si sa valeur est effectivement existante dans la clé primaire d'un tuple de la relation référencée, ou si sa valeur est *null*.

Une clé étrangère qui ne respecte pas la contrainte d'intégrité référentielle exprime un lien vers un tuple qui n'existe pas et donc n'est pas cohérente.

f) Référence entre relations

Le modèle relationnel a pour objectif la structuration de données selon des relations. L'enjeu est de parvenir à traduire un modèle conceptuel en modèle logique relationnel. Or, il n'y a pas de notion d'association en relationnel, donc il faudra pouvoir traduire les associations avec les concepts dont on dispose : relation, clé, clé étrangère.

Afin de représenter des références entre relations dans un modèle relationnel, la seule solution est de stocker l'information dans une relation, et donc que certains attributs d'une relation servent à pointer sur d'autres relations.

Attention

Il n'y a pas vraiment de référence ou de lien en relationnel, puisque nous ne disposons que de tables, de clés, de clés étrangère et de valeurs.

On va donc devoir se servir de ces outils pour matérialiser une notion de référence.

Méthode : Référence

La référence entre deux tuples T1 et T2 de deux relations différentes est exprimable par une valeur identique entre une clé étrangère du tuple T1 et la clé primaire de l'autre tuple T2.

Synonyme : Lien

Exemple

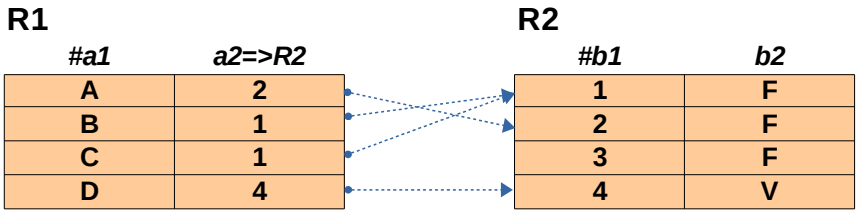


Image 13

L'attribut *a2* de la relation *R1* référence l'attribut *b1* de la relation *R2* car *a2* est une clé étrangère de R1 vers R2 (*b1* est la clé primaire de *R2*).

Ici on a donc par exemple les tuples identifiés par *B* et *C* de *R1* qui référencent le tuple identifié par *1* dans *R2*.

g) Schéma relationnel

Définition : Schéma d'une relation

Le schéma d'une relation définit cette relation en intension. Il est composé :

- du nom de la relation,
- de la liste de ses attributs avec les domaines respectifs dans lesquels ils prennent leurs valeurs,
- de la clé primaire,

- des clés étrangères,
- des clés candidates.

Définition : Schéma relationnel d'une base de donnée

Le schéma relationnel d'une BD✯ est la définition en intension de cette BD (par opposition à l'instance de la BD qui est une extension de la BD). Il est composé de l'ensemble des schémas de chaque relation de la BD.

Syntaxe : Relation

```
1 Relation (Attribut1:Domaine1, Attribut2:Domaine2, ... , AttributN:DomaineN)
```

La relation "Relation" contient N attributs chacun défini sur son domaine.

Syntaxe : Clé primaire

```
1 Relation (#Attribut1:Domaine1, ... , #AttributM:DomaineM, ... ,
  AttributN:DomaineN)
```

La clé de la relation "Relation" est composée des attributs "Attribut1" à "AttributM" (attribut précédés de # ou bien soulignés)

En général on note la clé primaire en premier dans la relation.

Syntaxe : Clé étrangère

```
1 Relation1 (..., AttributM=>Relation2, ... , AttributN=>Relation2)
```

La relation "Relation1" comporte une clé étrangère (composée des attributs "AttributM" à "AttributN") référençant la clé primaire de "Relation2". Bien sûr il peut exister plusieurs clés étrangères vers plusieurs relations distinctes. Une clé étrangère et sa clé primaire référencée sont toujours composées du même nombre d'attributs. Il n'est pas nécessaire de préciser les domaines des attributs appartenant à la clé étrangère car ce sont forcément les mêmes que ceux de la clé primaire référencée. Il n'est pas non plus en général nécessaire de préciser dans le schéma relationnel quels attributs de la clé étrangère référencent quels attributs de la clé primaire (cela est généralement évident) mais il est possible de la faire on notant "Attribut=>Relation.Attribut".

En général on note les clés étrangères en dernier dans la relation, sauf pour les clés étrangères qui font partie de la clé primaire (clés identifiantes).

Syntaxe : Clé candidates

```
1 Relation1 (...AttributM:DomaineM, ......) avec AttributM clé
```

Les clés candidates doivent être notées sur le schéma relationnel :

- S'il n'y a qu'une ou deux clés candidates, les noter directement après la définition de la relation
- S'il y a beaucoup de clés, pour ne pas trop alourdir la notation, renvoyer à un tableau à part

Attention : Clés candidates et clé primaire

La notation `R(#a,#b)` signifie toujours que `R` a comme clé primaire `(a,b)`, et non que `R` aurait deux clés `a` et `b` (dont on ne saurait pas laquelle est primaire).

La notation `R(#a,b) avec b clé` signifie bien que `a` et `b` sont deux clés de `R`, et que `a` est primaire.

Il ne faut pas confondre une clé composée de deux attributs avec deux clés.

h) Exemple de schéma relationnel pour la géographie

Exemple

```
1  Personne (#Numero:Entier, Nom:Chaîne, Prénom:Chaîne, LieuNaissance=>Ville)
2  Pays (#Nom:Chaîne, Population:Entier, Superficie:Réel, Dirigeant=>Personne)
3  Région (#Pays=>Pays, #Nom:Chaîne, Superficie, Dirigeant=>Personne)
4  Ville (#CodePostal:CP, Nom:Chaîne, Pays=>Région.Pays, Région=>Région.Nom,
   Dirigeant=>Personne)
```

Exemple : Exemple d'instance de la base de données

Personne

#Numero	Nom	Prenom	LieuNaissance
1	Durand	Pierre	60200
2	Dupont	Marie	60200

Pays

#Nom	Population	Superficie	Dirigeant
France	60	500001,01	2
Allemagne	80	600000,23564	2
Espagne	40	350000,1	1

Région

#Pays	#Nom	Superficie	Dirigeant
France	Picardie	50	1
Espagne	Picardie	40	1
France	Normandie	30	2

Ville

#CodePostal	Nom	Pays	Région	Dirigeant
F60200	Compiègne	France	Picardie	1
F60300	Senlis	France	Picardie	2
F60301	Senlis	France	Picardie	2
E8000	Senlis	Espagne	Picardie	2

Le schéma relationnel précédent décrit :

- **Des personnes**

  Elles sont identifiées par un numéro qui est en fait une clé artificielle. En effet, même une clé composée de tous les attributs (Nom, Prénom, LieuNaissance) laisse une possibilité de doublons (homonymes nés dans la même ville).

  La clé étrangère LieuNaissance fait référence à la relation Ville, et plus précisément à sa clé primaire CodePostal, ce qui est est laissé implicite car évident.

- **Des pays**

  Ils sont identifiés par leur nom, puisque deux pays ne peuvent avoir le même nom.

  Les pays sont dirigés par des personnes, et ce lien est matérialisé par la clé étrangère Dirigeant.

- **Des régions**

  Elles font partie d'un pays et ont un nom. Deux régions de pays différents pouvant avoir le même nom, il faut utiliser une clé primaire composée à la fois du nom de la région et du nom du pays, qui est une clé étrangère (le nom est appelé clé locale car il n'est pas suffisant pour identifier un tuple de la relation Région, et la clé étrangère vers la relation Pays est appelée clé identifiante).

- **Des villes**

  Elles sont identifié par un code postal qui est unique dans le monde (en utilisant le préfixe de pays de type "F-60200"). Ce code postal a pour domaine CP qui est une chaîne composée d'une ou deux lettres, d'un tiret, puis d'une série de chiffres.

  Le lien d'appartenance entre une ville et une région est matérialisé par la clé étrangère composée des deux attributs Pays et Région. Cette clé référence la clé primaire de la relation Région, également composée de deux attributs. Pour clairement expliciter les références (bien que sémantiquement la dénomination des attributs ne laisse pas de place au doute) on utilise la syntaxe Région.Pays et Région.Nom.

4. Synthèse

a) Synthèse : Schéma relationnel

Schéma relationnel

Un schéma relationnel permet une formalisation d'un modèle logique.

- Relation ou table

  Sous-ensemble d'un produit cartésien
  - Attribut ou colonne
    Prend ses valeurs dans un domaine
  - Enregistrement ou ligne
    Pose une valeur (y compris la valeur "null") pour chaque attribut
- Clé

  Groupe d'attributs ayant un rôle d'identification au sein d'un enregistrement
  - Clé candidate
    Identifie de façon unique un enregistrement
  - Clé primaire
    Clé candidate choisie pour représenter un enregistrement pour sa facilité d'usage
  - Clé étrangère
    Référence la clé primaire d'un tuple d'une autre relation pour exprimer un lien

b) Bibliographie commentée sur le modèle relationnel

Complément : Synthèses

SQL2 SQL3, applications à Oracle [Delmal01]

Une définition synthétique et efficace du domaine relationnel : relation, domaine, attribut, clé, intégrité, opérateurs (Premier chapitre).

5. Définition formelle d'une relation

a) Produit cartésien

Définition : Produit cartésien

Le produit cartésien, noté "X", des domaines D1, D2, ... , Dn, noté "D1 X D2 X ... X Dn" est l'ensemble des tuples (ou n-uplets ou vecteurs) <V1,V2,...,Vn> tel que Vi est une valeur de Di et tel que toutes les combinaisons de valeurs possibles sont exprimées.

Exemple

```
1  D1 = {A, B, C}
```

```
2  D2 = {1, 2, 3}
3  D1 X D2 = {<A,1>, <A,2>, <A,3>, <B,1>, <B,2>, <B,3>, <C,1>, <C,2>, <C,3>,}
```

b) Relation

Définition : Relation

Une relation sur les domaines D1, D2, ... , Dn est un sous-ensemble du produit cartésien "D1 X D2 X ... X Dn". Une relation est caractérisée par un nom.

Synonymes : Table, tableau

Syntaxe

On peut représenter la relation R sur les domaine D1, ... , Dn par une table comportant une colonne pour chaque domaine et une ligne pour chaque tuple de la relation.

D1	...	Dn
V1	...	Vn
...	...	...
V1	...	Vn

Tableau 7 Relation R

Remarque

Une relation est définie en extension, par l'énumération des tuples la composant.

B. Exercices

1. Lab I-

Description du problème

[20 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).
- À chaque médicament on associe une liste de contre-indications, généralement plusieurs, parfois aucune. Une contre-indication comporte un code unique qui l'identifie, ainsi qu'une description. Une contre-indication est toujours associée à un et un seul médicament.

Exemple de données

Afin de matérialiser notre base de données, nous obtenons les descriptions suivantes :

- Le **Chourix** a pour description courte *« Médicament contre la chute des choux »* et pour description longue *« Vivamus fermentum semper porta. Nunc diam velit, adipiscing ut tristique vitae, sagittis vel odio. Maecenas convallis ullamcorper ultricies. Curabitur ornare. »*. Il est conditionné en boîte de 13.

Ses contre-indications sont :
- CI1 : Ne jamais prendre après minuit.
- CI2 : Ne jamais mettre en contact avec de l'eau.

- Le **Tropas** a pour description courte *« Médicament contre les dysfonctionnements intellectuels »* et pour description longue *« Suspendisse lectus leo, consectetur in tempor sit amet, placerat quis neque. Etiam luctus porttitor lorem, sed suscipit est rutrum non. »*. Il est conditionné en boîte de 42.

  Ses contre-indications sont :
  - CI3 : Garder à l'abri de la lumière du soleil

Question 1

Dessiner des relations instanciées (en extension donc) remplies avec les données fournies en exemple.

Question 2

Écrivez le schéma relationnel (définition en intension donc) permettant de représenter une base de données relationnelle pour le laboratoire.

Introduction au passage UML-Relationnel : classes et associations

IV

A. Cours

Afin de pouvoir implémenter une base de données, il faut pouvoir traduire le modèle conceptuel en modèle logique. Cela signifie qu'il faut pouvoir convertir un modèle UML en modèle relationnel. Les modèles conceptuels sont suffisamment formels pour que ce passage soit systématisé dans la plupart des cas.

1. Transformation des classes et attributs

Objectifs

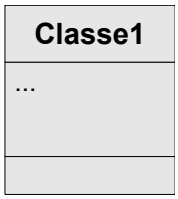
Savoir faire le passage d'un schéma conceptuel UML à un schéma relationnel pour les cas simples.

a) Transformation des classes

Méthode : Classe

Pour chaque classe non abstraite,

- on crée une relation dont le schéma est celui de la classe ;
- la clé primaire de cette relation est une des clés de la classe.



Graphique 2 Classe

```
Classe1(...)
```

Remarque

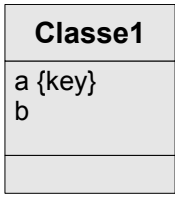
Les classes abstraites sont ignorées à ce stade, et n'étant pas instanciables, ne donnent généralement pas lieu à la création de relation.

b) Transformation des attributs

Méthode : Attributs simples

Pour chaque attribut élémentaire et monovalué d'une classe,

- on crée un attribut correspondant.



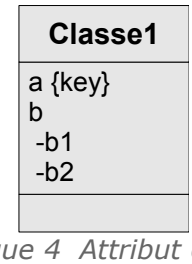
Graphique 3 Attribut

```
Classe1(#a,b)
```

Méthode : Attributs composites

Pour chaque attribut composite comprenant N sous-attributs d'une classe,

- on crée N attributs correspondants,
- dont les noms sont la concaténation du nom de l'attribut composite avec celui du sous-attribut.



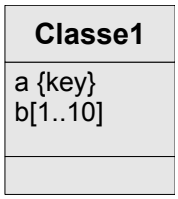
Graphique 4 Attribut composé

```
Classe1(#a,b_b1,b_b2)
```

Méthode : Attributs multivalués

Pour chaque attribut multivalué b d'une classe C,

- on crée une nouvelle relation RB,
- qui comprend un attribut monovalué correspondant à b,
- plus la clé de la relation représentant C ;
- la clé de RB est la concaténation des deux attributs.



Graphique 5 Attribut multivalué

```
Classe1(#a)
RB(#b,#a=>Classe1)
```

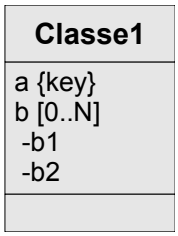
Méthode : Attributs multivalués (méthode alternative)

Dans le cas où le nombre maximum de `b` est fini, et petit, on peut également adopter la transformation suivante : `Classe1(#a,b1,b2,b3,b4,b5,b6,b7,b8,b9,b10)`.

Si le nombre d'attributs est infini (`b[1..*]`) c'est impossible, s'il est trop grand ce n'est pas souhaitable.

Méthode : Attributs composés multivalués

On combine les règles énoncées pour les attributs composés et pour les attributs multivalués.



Graphique 6 Attribut composé multivalué

```
Classe1(#a)
RB(#b_b1,#b_b2,#a=>Classe1)
```

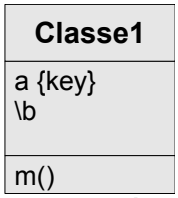
Rappel : Voir aussi

Transformation des compositions

c) Transformation des attributs dérivés et méthodes

Méthode : Attributs dérivés et méthodes

On ne représente pas en général les attributs dérivés ni les méthodes dans le modèle relationnel, ils seront calculés dynamiquement soit par des procédures internes à la BD (procédures stockées), soit par des procédures au niveau applicatif.



Graphique 7 Attribut dérivé et méthodes

```
Classe1(#a)
```

Complément : Attribut dérivé stockés

On peut décider (pour des raisons de performance essentiellement) de représenter l'attribut dérivé ou la méthode comme s'il s'agissait d'un attribut simple, mais il sera nécessaire dans ce cas d'ajouter des mécanismes de validation de contraintes dynamiques (avec des *triggers* par

exemple) pour assurer que la valeur stockée évolue en même temps que les attributs sur lesquels le calcul dérivé porte.

Notons qu'introduire un attribut dérivé ou un résultat de méthode dans le modèle relationnel équivaut à introduire de la redondance, ce qui est en général déconseillé, et ce qui doit être dans tous les cas contrôlé.

2. Transformation des associations

Objectifs

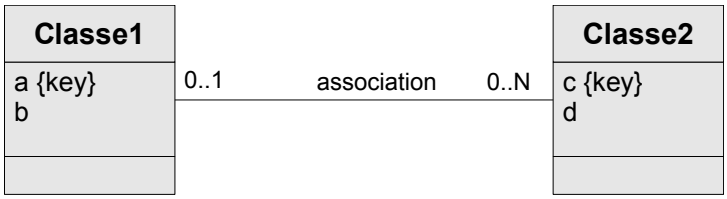
Savoir faire le passage d'un schéma conceptuel UML à un schéma relationnel pour les cas simples.

a) Transformation des associations 1:N

Méthode

Pour chaque association binaire de type 1:N :

- on ajoute à la relation côté N une clé étrangère vers la relation côté 1.



Graphique 8 Association 1:N

```
Classe1(#a,b)
Classe2(#c,d,a=>Classe1)
```

Complément

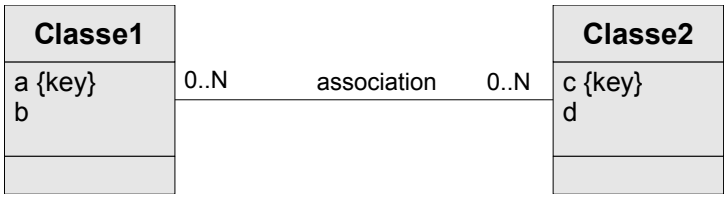
Contrainte de cardinalité minimale 1 dans les associations 1:N

b) Transformation des associations N:M

Méthode

Pour chaque association binaire de type M:N :

- on crée une nouvelle relation,
- composée de clés étrangères vers chaque relation associée,
- et dont la clé primaire est la concaténation de ces clés étrangères.



Graphique 9 Association N:M

```
Classe1(#a,b)
Classe2(#c,d)
Assoc(#a=>Classe1,#c=>Classe2)
```

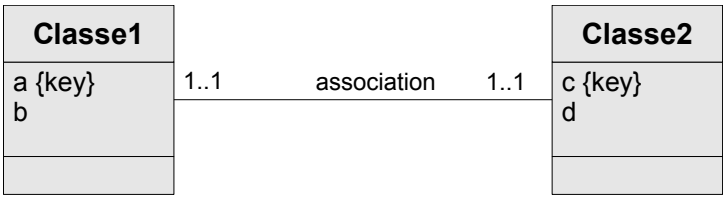
Complément

Contrainte de cardinalité minimale 1 dans les associations N:M

c) Transformation des associations 1:1 (approche simplifiée)

Méthode

La solution la plus simple et la plus générale pour transformer une association 1:1 consiste à traiter cette association 1:1 comme une association 1:N, puis à ajouter une contrainte UNIQUE sur la clé étrangère pour limiter la cardinalité maximale à 1.



Graphique 10 Association 1:1

```
Classe1(#a,b,c=>Classe2) avec c UNIQUE
Classe2(#c,d)
```

ou

```
Classe1(#a,b)
Classe2(#c,d,a=>Classe1) avec a UNIQUE
```

Remarque

Il existe toujours deux solutions selon que l'on choisit une ou l'autre relation pour accueillir la clé étrangère. Selon la cardinalité minimale, un des deux choix peut être plus pertinent.

Complément

Il est parfois possible de choisir de fusionner les deux classes au sein d'une seule relation plutôt que d'opter pour une clé étrangère.

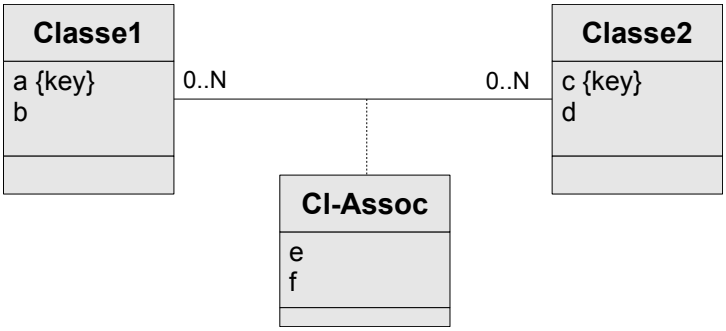
Complément

Transformation des associations 1:1 (approche générale)

d) Transformation des classes d'association

Méthode : Classe d'association N:M

Les attributs de la classe d'association sont ajoutés à la relation issue de l'association N:M.



Graphique 11 Classe assocation (N:M)

```
Classe1(#a,b)
Classe2(#c,d)
Assoc(#a=>Classe1,#c=>Classe2,e,f)
```

Complément : Classe d'association 1:N

Les attributs de la classe d'association sont ajoutés à la relation issue de la classe côté N.

Complément : Classe d'association 1:1

Les attributs de la classe d'association sont ajoutés à la relation qui a été choisie pour recevoir la clé étrangère. Si les deux classes ont été fusionnées en une seule relation, les attributs sont ajoutés à celle-ci.

B. Exercices

1. Lab I+

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).
- À chaque médicament on associe une liste de contre-indications, généralement plusieurs, parfois aucune. Une contre-indication comporte un code unique qui l'identifie, ainsi qu'une description. Une contre-indication est toujours associée à un et un seul médicament.

Exemple de données

Afin de matérialiser notre base de données, nous obtenons les descriptions suivantes :

- Le **Chourix** a pour description courte *« Médicament contre la chute des choux »* et pour description longue *« Vivamus fermentum semper porta. Nunc diam velit, adipiscing ut tristique vitae, sagittis vel odio. Maecenas convallis ullamcorper ultricies. Curabitur ornare. »*. Il est conditionné en boîte de 13.

  Ses contre-indications sont :
  - CI1 : Ne jamais prendre après minuit.
  - CI2 : Ne jamais mettre en contact avec de l'eau.
- Le **Tropas** a pour description courte *« Médicament contre les dysfonctionnements intellectuels »* et pour description longue *« Suspendisse lectus leo, consectetur in tempor sit amet, placerat quis neque. Etiam luctus porttitor lorem, sed suscipit est rutrum non. »*. Il est conditionné en boîte de 42.

  Ses contre-indications sont :
  - CI3 : Garder à l'abri de la lumière du soleil

Question 1

Réaliser le modèle conceptuel de données en UML du problème.

Question 2

Étendre le modèle conceptuel UML afin d'ajouter la gestion des composants. Un composant est identifié par un code unique et possède un intitulé. Tout médicament possède au moins un composant, souvent plusieurs. Tout composant peut intervenir dans la fabrication de plusieurs médicaments. Il existe des composants qui ne sont pas utilisés pour fabriquer des médicaments et que l'on veut quand même gérer.

Question 3

En mobilisant les règles adéquates, proposer un modèle logique de données correspondant en relationnel. Le repérage des domaines et des clés est obligatoire.

Question 4

Dessiner des tableaux remplis avec les données fournies en exemple, afin de montrer que le modèle fonctionne selon le besoin exprimé initialement. On pourra mettre le premier mot seulement des descriptions pour gagner du temps.

2. Usine de production

[30 minutes]

Une usine cherche à modéliser sa production de véhicules et de moteurs :

- Les véhicules sont identifiés par un numéro d'immatriculation alphanumérique et caractérisés par une couleur, dont la dénomination est une chaîne de caractères. Chaque véhicule peut comporter un unique moteur et/ou un nombre quelconque de pneus.
- Chaque moteur est monté sur un et un seul véhicule et est identifié par un numéro de série. Un moteur est caractérisé par une puissance, en chevaux.
- Tout pneu est monté sur un unique véhicule et est identifié par un numéro de série. Sa position est définie localement sur ce véhicule et par rapport à l'essieu : *Dn* pour les pneus situé sur la droite de l'essieu et *Gn* pour les pneus situés à gauche ; *n* représentant le numéro de l'essieu (1 pour celui situé devant, 2 pour la deuxième rangée, etc.). Un pneu est caractérisé par un diamètre et une largeur en pouces.
- Les moteurs, les pneus et les véhicules sont fabriqués sous une marque. Les mêmes marques peuvent fabriquer indifféremment des moteurs, des pneus et/ou des véhicules, et un véhicule d'une certaine marque peut comporter un moteur et/ou des pneus de marque différente.

Question 1

Réaliser le modèle UML de ce problème en faisant apparaître les domaines et les clés.

Question 2

Réaliser le passage au modèle relationnel, en faisant apparaître les clés primaires, candidates et étrangères.

Question 3

Dessiner les tableaux correspondant aux relations du modèle. Instancier au minimum deux véhicules et quatre marques.

Question 4

Donner quatre exemples d'enregistrements qui seront refusés - étant données les données déjà insérées - pour quatre raisons différentes :

- contrainte de clé sur une clé primaire
- contrainte de clé sur une clé candidate
- contrainte d'intégrité référentielle
- contrainte de non nullité

Création et alimentation de bases de données SQL

V

A. Cours

SQL✯ est un langage standardisé, implémenté par tous les SGBDR✯, qui permet, indépendamment de la plate-forme technologique et de façon déclarative, de définir le modèle de données, de le contrôler et enfin de le manipuler.

1. Le langage SQL

Définition : SQL

SQL✯ (pour langage de requêtes structuré) est un langage déclaratif destiné à la manipulation de bases de données au sein des SGBD✯ et plus particulièrement des SGBDR✯.

SQL : LDD, LCD, LMD, LCT

SQL est un langage déclaratif, il n'est donc pas a proprement parlé un langage de programmation, mais plutôt une interface standard pour accéder aux bases de données.

Il est composé de quatre sous ensembles :

- Le Langage de Définition de Données (LDD✯, ou en anglais DDL, *Data Definition Language*) pour créer et supprimer des objets dans la base de données (tables, contraintes d'intégrité, vues, etc.).

  Exemple de commandes : `CREATE DROP ALTER`
- Le Langage de Contrôle de Données (LCD✯, ou en anglais DCL, *Data Control Language*) pour gérer les droits sur les objets de la base (création des utilisateurs et affectation de leurs droits).

  Exemple de commandes : `GRANT REVOKE`
- Le Langage de Manipulation de Données (LMD✯, ou en anglais DML, *Data Manipulation Language*) pour la recherche, l'insertion, la mise à jour et la suppression de données. Le LMD est basé sur les opérateurs relationnels, auxquels sont ajoutés des fonctions de calcul d'agrégats et des instructions pour réaliser les opérations d'insertion, mise à jour et suppression.

Exemple de commandes : `INSERT UPDATE DELETE SELECT`

- Le Langage de Contrôle de Transaction (LCT, ou en anglais TCL, *Transaction Control Language*) pour la gestion des transactions (validation ou annulation de modifications de données dans la BD)

  Exemple de commandes : `COMMIT ROLLBACK`

Fondamental : Référence SQL : SQL-99 complete, really

Gulutzan and Pelzer, 1999 [Gulutzan and Pelzer, 1999]

https://mariadb.com/kb/en/sql-99)[2]

Complément : Origine du SQL

Le modèle relationnel a été inventé par E.F. Codd (Directeur de recherche du centre IBM de San José) en 1970, suite à quoi de nombreux langages ont fait leur apparition :

- IBM Sequel (Structured English Query Language) en 1977
- IBM Sequel/2
- IBM System/R
- IBM DB2

Ce sont ces langages qui ont donné naissance au standard SQL, normalisé en 1986 au États-Unis par l'ANSI★ pour donner SQL/86 (puis au niveau international par l'ISO★ en 1987).

Complément : Versions de SQL

- SQL-86 (ou SQL-87) : Version d'origine
- SQL-89 (ou SQL-1) : Améliorations mineures
- SQL-92 (ou SQL-2) : Extensions fonctionnelles majeures (types de données, opérations relationnelles, instruction LDD, transactions, etc.
- SQL-99 (ou SQL-3) : Introduction du PSM★ (couche procédurale sous forme de procédure stockées) et du RO★
- SQL-2003 : Extensions XML★
- SQL-2006 : Améliorations mineures (pour XML notamment)
- SQL-2008 : Améliorations mineures (pour le RO notamment)

Remarque : Version SQL et implémentations SGBD

Selon leur niveau d'implémentation de SQL, les SGBD acceptent ou non certaines fonctions.

Certains SGBD ayant entamé certaines implémentations avant leur standardisation définitive, ces implémentations peuvent différer de la norme.

2. Créer des tables en SQL (Langage de Définition de Données)

Objectifs

Maîtriser les bases du SQL pour créer et modifier des tables et des vues.

Le LDD★ est la partie du langage SQL qui permet de créer de façon déclarative les objets composant une BD★. Il permet notamment la définition des schémas, des relations, des contraintes d'intégrité, des vues.

2 - https://mariadb.com/kb/en/sql-99/

Rappel : Le code SQL peut être testé avec Db Disco[dbdisco.crzt.fr]

a) Lab I++

Description du problème

[20 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).
- À chaque médicament on associe une liste de contre-indications, généralement plusieurs, parfois aucune. Une contre-indication comporte un code unique qui l'identifie, ainsi qu'une description. Une contre-indication est toujours associée à un et un seul médicament.

Exemple de données

Afin de matérialiser notre base de données, nous obtenons les descriptions suivantes :

- Le **Chourix** a pour description courte *« Médicament contre la chute des choux »* et pour description longue *« Vivamus fermentum semper porta. Nunc diam velit, adipiscing ut tristique vitae, sagittis vel odio. Maecenas convallis ullamcorper ultricies. Curabitur ornare. »*. Il est conditionné en boîte de 13.

  Ses contre-indications sont :
  - CI1 : Ne jamais prendre après minuit.
  - CI2 : Ne jamais mettre en contact avec de l'eau.
- Le **Tropas** a pour description courte *« Médicament contre les dysfonctionnements intellectuels »* et pour description longue *« Suspendisse lectus leo, consectetur in tempor sit amet, placerat quis neque. Etiam luctus porttitor lorem, sed suscipit est rutrum non. »*. Il est conditionné en boîte de 42.

  Ses contre-indications sont :
  - CI3 : Garder à l'abri de la lumière du soleil

Question 1

Réaliser le modèle conceptuel de données en UML du problème.

Question 2

En mobilisant les règles adéquates, proposer un modèle logique de données correspondant en relationnel. Le repérage des domaines et des clés est obligatoire.

Question 3

Créer une base de données en SQL correspondant au modèle relationnel.

Question 4

Insérer les données fournies en exemple dans la base de données.

b) Création de tables

Rappel

Qu'est ce que le SQL ?

Introduction

La création de table est le fondement de la création d'une base de données en SQL.

Définition : Création de table

La création de table est la définition d'un schéma de relation en intension, par la spécification de tous les attributs le composant avec leurs domaines respectifs.

Syntaxe

```
1  CREATE TABLE nom_table (
2    nom_colonne1 domaine1,
3    nom_colonne2 domaine2,
4    ...
5    nom_colonneN domaineN
6  );
```

Exemple

```
1  CREATE TABLE Personne (
2    Nom VARCHAR(25),
3    Prenom VARCHAR(25),
4    Age NUMERIC(3)
5  );
```

Attention : Contrainte d'intégrité

La définition des types n'est pas suffisante pour définir un schéma relationnel, il faut lui adjoindre la définition de contraintes d'intégrité, qui permette de poser les notions de clé, d'intégrité référentielle, de restriction de domaines, etc.

c) Domaines de données

Introduction

Un attribut d'une relation est défini pour un certain domaine ou type. Les types de données disponibles en SQL varient d'un SGBD à l'autre, on peut néanmoins citer un certain nombre de types standards que l'on retrouve dans tous les SGBD.

Fondamental: Les types standard

- **INTEGER ou INT, SMALLINT**
- **NUMERIC(X)**
- **DECIMAL(X,Y) ou NUMERIC(X,Y)**
- **FLOAT(X), REAL**
- **CHAR(X)**
- **VARCHAR(X)**
- **DATE (AAAA-MM-JJ)**
- **DATETIME (AAAA-MM-JJ HH:MM:SS)**

Les types numériques standard

- **Les nombres entiers**

  INTEGER (ou INT) et SMALLINT, permettent de coder des entiers sur 4 octets (-2.147.483.648 à 2.147.483.647) ou 2 octets (-32.768 à 32.767).
- **Les nombres entiers**

  NUMERIC(X) désigne un entier de X chiffres au maximum.
- **Les nombres décimaux**

  DECIMAL(X,Y), où X et Y sont optionnels et désignent respectivement le nombre de chiffres maximum pouvant composer le nombre, et le nombre de chiffres après la virgule.

NUMERIC(X,Y) est un synonyme standard.

- **Les nombres à virgule flottante**

  FLOAT(X), avec X définissant la précision (nombre de bits de codage de la mantisse).

  REAL est un synonyme standard de FLOAT(24).

Conseil : FLOAT versus DECIMAL

Il est conseillé d'utiliser DECIMAL qui est un nombre exact, plutôt que FLOAT qui est un nombre approximatif, si la précision requise est suffisante. FLOAT sera réservé typiquement à des calculs scientifiques nécessitant un degré de précision supérieur.

Les types chaîne de caractères standard

On distingue principalement les types CHAR(X) et VARCHAR(X), où X est obligatoire et désigne la longueur de la chaîne.

- CHAR définit des chaînes de longueur fixe (complétée à droites par des espaces, si la longueur est inférieure à X) ;
- et VARCHAR des chaînes de longueurs variables.

CHAR et VARCHAR sont généralement limités à 255 caractères. La plupart des SGBD proposent des types, tels que TEXT ou CLOB (Character Long Object), pour représenter des chaînes de caractères longues, jusqu'à 65000 caractères par exemple.

Les types date standard

Les types date dont introduits avec la norme SQL2. On distingue :

- DATE qui représente une date selon un format de type "AAAA-MM-JJ" ;
- et DATETIME qui représente une date plus une heure, dans un format tel que "AAAA-MM-JJ HH:MM:SS".

Complément : Les autres types

En fonction du SGBD, il peut exister de nombreux autres types. On peut citer par exemple :

- MONEY pour représenter des décimaux associés à une monnaie,
- BOOLEAN pour représenter des booléens,
- BLOB (pour Binary Long Oject) pour représenter des données binaires tels que des documents multimédia (images bitmap, vidéo, etc.)
- ...

d) La valeur NULL

L'absence de valeur, représentée par la valeur `NULL`, est une information fondamentale en SQL, qu'il ne faut pas confondre avec la chaîne espace de caractère où bien la valeur 0. Il ne s'agit pas d'un type à proprement parler, mais d'une valeur possible dans tous les types.

Fondamental

Par défaut en SQL `NULL` fait partie du domaine, il faut l'exclure explicitement par la clause `NOT NULL` après la définition de type, si on ne le souhaite pas.

Syntaxe

```
CREATE TABLE nom de table (
CREATE TABLE nom_table (
  nom_colonne1 domaine1 NOT NULL,
  nom_colonne2 domaine2,
  ...
  nom_colonneN domaineN NOT NULL
);
```

e) Contraintes d'intégrité

Fondamental

- **PRIMARY KEY (<liste d'attibuts>)**
- **UNIQUE (<liste d'attibuts>)**
- **FOREIGN KEY (<liste d'attibuts>) REFERENCES <nom table>(<nom colonnes>)**
- **CHECK (<condition>)**

Une contrainte d'intégrité est une règle qui définit la cohérence d'une donnée ou d'un ensemble de données de la BD✯.

Il existe deux types de contraintes :

- sur une colonne unique,
- ou sur une table lorsque la contrainte porte sur une ou plusieurs colonnes.

Les contraintes sont définies au moment de la création des tables.

Définition : Contraintes d'intégrité sur une colonne

Les contraintes d'intégrité sur une colonne sont :

- PRIMARY KEY : définit l'attribut comme la clé primaire
- UNIQUE : interdit que deux tuples de la relation aient la même valeur pour l'attribut.
- REFERENCES <nom table> (<nom colonnes>) : contrôle l'intégrité référentielle entre l'attribut et la table et ses colonnes spécifiées
- CHECK (<condition>) : contrôle la validité de la valeur de l'attribut spécifié dans la condition dans le cadre d'une restriction de domaine

Définition : Contraintes d'intégrité sur une table

Les contraintes d'intégrité sur une table sont :

- PRIMARY KEY (<liste d'attibuts>) : définit les attributs de la liste comme la clé primaire
- UNIQUE (<liste d'attibuts>) : interdit que deux tuples de la relation aient les mêmes valeurs pour l'ensemble des attributs de la liste.
- FOREIGN KEY (<liste d'attibuts>) REFERENCES <nom table>(<nom colonnes>) : contrôle l'intégrité référentielle entre les attributs de la liste et la table et ses colonnes spécifiées
- CHECK (<condition>) : contrôle la validité de la valeur des attributs spécifiés dans la condition dans le cadre d'une restriction de domaine

Syntaxe

```
1  CREATE TABLE nom de table (
2  CREATE TABLE nom_table (
3    nom_colonne1 domaine1 <contraintes colonne1>,
4    nom_colonne2 domaine2 <contraintes colonne2>,
5    ...
6    nom_colonneN domaineN <contraintes colonneN>,
7   <contraintes de table>
8  );
```

Exemple

```
1  CREATE TABLE Personne (
2    N°SS CHAR(13) PRIMARY KEY,
3    Nom VARCHAR(25) NOT NULL,
4    Prenom VARCHAR(25) NOT NULL,
```

```
  Age INTEGER(3) CHECK (Age BETWEEN 18 AND 65),
  Mariage CHAR(13) REFERENCES Personne(N°SS),
  UNIQUE (Nom, Prenom)
);
```

Remarque : Clé candidate

La clause UNIQUE NOT NULL sur un attribut ou un groupe d'attributs définit une clé candidate non primaire.

Remarque

Les contraintes sur une colonne et sur une table peuvent être combinées dans la définition d'un même schéma de relation.

Remarque

Une contrainte sur une colonne peut toujours être remplacée par une contrainte sur une table.

f) Exemple de contraintes d'intégrité

Exemple

```
CREATE TABLE Personne (
  N°SS CHAR(13) PRIMARY KEY,
  Nom VARCHAR(25) NOT NULL,
  Prenom VARCHAR(25) NOT NULL,
  Age INTEGER(3) CHECK (Age BETWEEN 18 AND 65),
  Mariage CHAR(13) REFERENCES Personne(N°SS),
  Codepostal INTEGER(5),
  Pays VARCHAR(50),
  UNIQUE (Nom, Prenom),
  FOREIGN KEY (Codepostal, Pays) REFERENCES Adresse (CP, Pays)
);

CREATE TABLE Adresse (
  CP INTEGER(5) NOT NULL,
  Pays VARCHAR(50) NOT NULL,
  Initiale CHAR(1) CHECK (Initiale = LEFT(Pays, 1)),
  PRIMARY KEY (CP, Pays)
);
```

Dans la définition de schéma précédente on a posé les contraintes suivantes :

- La clé primaire de Personne est N°SS et la clé primaire de Adresse est (CP, Pays).
- Nom, Prénom ne peuvent pas être null et (Nom, Prénom) est une clé.
- Age doit être compris entre 18 et 65 et Initiale doit être la première lettre de Pays (avec la fonction LEFT qui renvoie la sous chaîne à gauche de la chaîne passée en premier argument, sur le nombre de caractères passés en second argument)
- Mariage est clé étrangère vers Personne et (Codepostal, Pays) est une clé étrangère vers Adresse.

Exemple : Réécriture avec uniquement des contraintes de table

```
CREATE TABLE Personne (
  N°SS CHAR(13) ,
  Nom VARCHAR(25) NOT NULL,
  Prenom VARCHAR(25) NOT NULL,
  Age INTEGER(3) ,
  Mariage CHAR(13),
  Codepostal INTEGER(5),
  Pays VARCHAR(50),
```

```
  PRIMARY KEY (N°SS),
  UNIQUE (Nom, Prenom),
  CHECK (Age BETWEEN 18 AND 65),
  FOREIGN KEY (Mariage) REFERENCES Personne(N°SS),
  FOREIGN KEY (Codepostal, Pays) REFERENCES Adresse (CP, Pays)
);

CREATE TABLE Adresse (
  CP INTEGER(5) NOT NULL,
  Pays VARCHAR(50) NOT NULL,
  Initiale CHAR(1),
  PRIMARY KEY (CP, Pays),
  CHECK (Initiale = LEFT(Pays, 1))
);
```

Ce schéma est strictement le même que le précédent, simplement les contraintes ont toutes été réécrites comme des contraintes de table.

g) Exercice

Les instructions SQL ci-après ont-elles pu permettre de créer le schéma des trois relations instanciées ci-dessous ?

```
CREATE TABLE A (
A1 CHAR(255),
A2 CHAR(255),
A3 CHAR(255),
A4 CHAR(255)
);
CREATE TABLE B (
B1 CHAR(255),
B2 CHAR(255),
B3 CHAR(255),
B4 CHAR(255),
B5 CHAR(255),
B6 CHAR(255),
B7 CHAR(255)
);
CREATE TABLE C (
C1 CHAR(255),
C2 CHAR(255)
);
```

- ○ Oui
- ○ Non

3. Insérer, modifier et supprimer des données en SQL (Langage de Manipulation de Données)

Objectifs

Maîtriser les bases du SQL pour entrer, modifier et effacer des données dans les tables.

a) Exercice

Quelle valeur renvoie la dernière instruction SQL de la liste ci-dessous :

```
CREATE TABLE t (
```

```
2    a integer,
3    b integer,
4    c integer);
5
6    INSERT INTO t
7    VALUES (0, 0, 0);
8
9    INSERT INTO t
10   VALUES (1, 0, 0);
11
12   INSERT INTO t
13   SELECT * FROM t;
14
15   SELECT sum(a) + count(b)
16   FROM t;
```

b) Insertion de données

Le langage SQL fournit des instructions pour ajouter des nouveaux tuples à une relation. Il offre ainsi une interface standard pour ajouter des information dans une base de données.

Il existe deux moyens d'ajouter des données, soit par fourniture directe des valeurs des propriétés du tuple à ajouter, soit par sélection des tuples à ajouter dans une autre relation.

Syntaxe : Insertion directe de valeurs

```
1    INSERT INTO <Nom de la relation> (<Liste ordonnée des propriétés à valoriser>)
2    VALUES (<Liste ordonnée des valeurs à affecter aux propriétés spécifiées ci-
     dessus>)
```

Exemple : Insertion directe de valeurs

```
1    INSERT INTO Virement (Date, Montant, Objet)
2    VALUES (14-07-1975, 1000, 'Prime de naissance');
```

Syntaxe : Insertion de valeurs par l'intermédiaire d'une sélection

```
1    INSERT INTO <Nom de la relation> (<Liste ordonnée des propriétés à valoriser>)
2    SELECT ...
```

L'instruction SELECT projetant un nombre de propriétés identiques aux propriétés à valoriser.

Exemple : Insertion de valeurs par l'intermédiaire d'une sélection

```
1    INSERT INTO Credit (Date, Montant, Objet)
2    SELECT Date, Montant, 'Annulation de débit'
3    FROM Debit
4    WHERE Debit.Date = 25-12-2001;
```

Dans cet exemple tous les débits effectués le 25 décembre 2001, sont re-crédités pour le même montant (et à la même date), avec la mention annulation dans l'objet du crédit. Ceci pourrait typiquement réalisé en cas de débits erronés ce jour là.

Remarque

- Les propriétés non valorisées sont affectées à la valeur NULL.
- Il est possible de ne pas spécifier les propriétés à valoriser, dans ce cas, toutes les propriétés de la relation seront considérées, dans leur ordre de définition dans la relation (à n'utiliser que dans les cas les plus simples).

c) Mise à jour de données

Le langage SQL fournit une instruction pour modifier des tuples existants dans une relation.

Syntaxe : Mise à jour directe de valeurs

```
UPDATE <Nom de la relation>
SET <Liste d'affectations Propriété=Valeur, Propriété=Valeur>
WHERE <Condition pour filtrer les tuples à mettre à jour>
```

```
UPDATE r
SET a=1, b='x'
WHERE c=0
```

Exemple : Mise à jour directe de valeurs

```
UPDATE Compte
SET Monnaie='Euro'
WHERE Monnaie='Franc'
```

Exemple : Mise à jour par calcul sur l'ancienne valeur

```
UPDATE Compte
SET Total=Total * 6,55957
WHERE Monnaie='Euro'
```

d) Suppression de données

Le langage SQL fournit une instruction pour supprimer des tuples existants dans une relation.

Syntaxe

```
DELETE FROM <Nom de la relation>
WHERE <Condition pour filtrer les tuples à supprimer>
```

Exemple : Suppression de tous les tuples d'une relation

```
DELETE FROM FaussesFactures
```

Exemple : Suppression sélective

```
DELETE FROM FaussesFactures
WHERE Auteur='Moi'
```

4. Supprimer et modifier des tables en SQL (Langage de Définition de Données)

Objectifs

Maîtriser les bases du SQL pour créer et modifier des tables et des vues.

Le LDD✯ permet de créer les objets composant une BD✯ de façon déclarative. Il permet notamment la définition des schémas des relations, la définition des contraintes d'intégrité, la définition de vues relationnelles.

a) Suppression d'objets

Il est possible de supprimer des objets de la BD✯, tels que les tables ou les vues.

Syntaxe

```
1  DROP <type objet> <nom objet>
```

Exemple

```
1  DROP TABLE Personne;
2  DROP VIEW Employe;
```

b) Modification de tables

Introduction

L'instruction ALTER TABLE permet de modifier la définition d'une table (colonnes ou contraintes) préalablement créée.

Cette commande absente de SQL-89 est normalisée dans SQL-92

Syntaxe : Ajout de colonne

```
1  ALTER TABLE <nom de table>
2  ADD <définition de colonne>
```

Syntaxe : Suppression de colonne

```
1  ALTER TABLE <nom de table>
2  DROP <nom de colonne>
```

Syntaxe : Ajout de contrainte

```
1  ALTER TABLE <nom de table>
2  ADD <définition de contrainte de table>
```

Remarque : Modification de table sans donnée sans la commande ALTER

Pour modifier une table ne contenant pas encore de donnée, la commande ALTER n'est pas indispensable, l'on peut supprimer la table à modifier (DROP) et la recréer telle qu'on la souhaite. Notons néanmoins que si la table est référencée par des clauses FOREIGN KEY, cette suppression sera plus compliquée, car il faudra également supprimer et recréer les tables référençantes (ce qui ce complique encore si ces dernières contiennent des données).

Remarque : Modification de table avec données sans la commande ALTER

Pour modifier une table contenant des données, la commande ALTER n'est pas indispensable. On peut en effet :

1. Copier les données dans une table temporaire de même schéma que la table à modifier
2. Supprimer et recréer la table à modifier avec le nouveau schéma
3. Copier les données depuis la table temporaire vers la table modifiée

c) Exemple de modifications de tables

Table initiale

Soit une table initiale telle que définie ci-après.

```
CREATE TABLE Personne (
 pk_n NUMERIC(4),
 nom VARCHAR(50),
 prenom VARCHAR(50),
 PRIMARY KEY (pk_n)
);
```

Modifications

On décide d'apporter les aménagements suivants à la table : on définit "nom" comme UNIQUE et on supprime le champ "prenom".

```
ALTER TABLE Personne
ADD UNIQUE (nom);

ALTER TABLE Personne
DROP prenom;
```

Table finale

La table obtenue après modification est identique à la table qui aurait été définie directement telle que ci-après.

```
CREATE TABLE Personne (
 pk_n NUMERIC(4),
 nom VARCHAR(50),
 PRIMARY KEY (pk_n),
  UNIQUE (nom)
);
```

B. Exercices

1. The show

[30 minutes]

Soit le schéma relationnel suivant décrivant un système de réservations de places de spectacles :

```
SPECTACLE (#nospectacle:int, nom:str, durée:int, type:{théâtre|danse|concert})
SALLE (#nosalle:int, nbplaces:int)
REPRESENTATION (#date:date, #nospectacle=>SPECTACLE, #nosalle=>SALLE,
prix:decimal)
```

En faisant les suppositions suivantes :

- On gère un espace de spectacles ayant un ensemble de salles (décrit par la relation SALLE).
- On suppose que pour un jour donné et une salle donnée, il n'y a qu'un seul spectacle représenté.

Question 1

Retro-concevoir le MCD en UML.

Question 2

Proposer des contraintes d'intégrité réalistes pour ce schéma (en français).

Question 3

Proposer une définition du schéma en SQL qui prenne en compte certaines de ces contraintes.

Question 4

Insérer des données réalistes dans votre schéma afin de vérifier son bon fonctionnement.

2. Du producteur au consommateur

[30 min]

Soit le modèle relationnel suivant :

```
1 Producteur(#raison_sociale:chaîne(25), ville:chaîne(255))
2 Consommateur(#login:chaîne(10), #email:chaîne(50),
  nom:chaîne(50),prenom:chaîne(50), ville:chaîne(255))
3 Produit(#id:entier, description:chaîne(100), produit-par=>Producteur, consomme-
  par-login=>Consommateur, consomme-par-email=>Consommateur)
```

On ajoute que :

- `(nom,prenom,ville)` est une clé candidate de `Consommateur`
- Tous les produits sont produits
- Tous les produits ne sont pas consommés

Question 1

Rétro-concevez le modèle conceptuel sous-jacent à ce modèle relationnel.

Question 2

Établissez le code LDD standard permettant d'implémenter ce modèle en SQL.

Question 3

Insérez les données dans votre base de données correspondant aux assertions suivantes :

- L'entreprise de Compiègne "Pommes Picardes SARL" a produit 4 lots de pommes, et 2 lots de cidre.
- Il existe trois utilisateurs consommateurs dans la base, donc les adresses mails sont :

  Al.Un@compiegne.fr - Bob.Deux@compiegne.fr - Charlie.Trois@compiegne.fr

  Ce sont des employés de la ville de Compiègne qui habitent cette ville. Leur mail est construit sur le modèle Prenom.Nom@compiegne.fr. Leur login est leur prénom.

Question 4

Modifiez les données de votre base de données pour intégrer les assertions suivantes :

- 1 lots de pommes a été consommés par Al Un.
- 2 lots de pomme ont été consommé par Bob Deux.
- Tous les lots de cidre ont été consommés par Al Un.

Question 5

Charlie Trois n'ayant rien consommé, modifiez votre base de données afin de le supprimer de la base.

Implémentation de bases de données relationnelles avec PostgreSQL

VI

A. Cours

1. Introduction à PostgreSQL : présentation, installation, manipulations de base

a) Présentation

PostgreSQL est :

- un SGBDR
- libre (licence BSD)
- multi-plate-formes (Unix, Linux, Windows, MacOS, ...)
- puissant
- très respectueux du standard
- très bien documenté

Fondamental : Documentation de PostgreSQL

https://www.postgresql.org/docs/[3]

(en français : *http://docs.postgresqlfr.org/*[4])

Fonctionnement général

PostgreSQL comme la grande majorité des SGBD est un logiciel client-serveur. Il faut donc pour l'utiliser un serveur (programme *postgresql* sous Linux) et un client.

Il existe plusieurs clients, les deux plus utilisés sont le client textuel *psql* et le client graphique *PgAdmin III*.

3 - https://www.postgresql.org/docs/
4 - http://docs.postgresqlfr.org/

Complément

- *http://www.postgresql.org/*[5]
- *http://www.postgresql.fr/*[6]

b) Installation

Attention

Nous présentons ici une installation *a minima* de PostgreSQL uniquement à des fins de test et d'apprentissage, et pour un usage local. Pour mettre en production une base de données PostgreSQL sur le réseau, il faut suivre des directives supplémentaires, notamment pour assurer la sécurité du système, sa sauvegarde...

Ces thèmes ne sont pas abordés dans le cadre de ce cours.

PostgreSQL est à l'origine une base de données conçue pour Unix, son installation et son fonctionnement sont possibles aujourd'hui sur les trois OS, mais Linux reste son environnement de prédilection. C'est l'architecture PostgreSQL sur Linux qui est étudiée ici.

https://www.postgresql.org/download/[7]

Exemple : Installer PostgreSQL sur Ubuntu

Une installation sur Ubuntu est très facile :

1. Installer les paquets : `sudo apt-get install postgresql`
2. Activer l'utilisateur *postgres* : `sudo passwd postgres`
3. Se connecter en tant que *postgres* : `su postgres`
4. Lancer le client *psql* : `psql -h localhost -d postgres -U postgres` (ou `psql`)

Complément

https://doc.ubuntu-fr.org/postgresql[8]

Complément : Base de données par défaut

L'installation crée une base de données par défaut qui s'appelle *postgres* et un utilisateur *postgres* qui possède cette base (`OWNER`).

Complément : Informations réseau

- Le port standard de communication de PosgreSQL est *5432*.
- Si le client et le serveur sont sur la même machine, le client peut bien entendu se connecter au serveur *localhost* sur le port *5432*.

Complément : Installer PostgreSQL sous Windows

1. Télécharger un *installer* depuis *http://www.enterprisedb.com/products-services-training/pgdownload#windows*[9]
2. Exécuter l'installation en validant les propositions par défaut (et sans installer les éventuels programmes complémentaires proposés à l'issue de l'installation)
3. Exécuter le client *psql* (également appelé *Shell SQL*)

c) Le client textuel "psql"

Définition : psql

`psql` est le client textuel de PostgreSQL.

5 - http://www.postgresql.org/
6 - http://www.postgresql.fr/
7 - https://www.postgresql.org/download/
8 - https://doc.ubuntu-fr.org/postgresql
9 - http://www.enterprisedb.com/products-services-training/pgdownload#windows

Syntaxe : Connexion à un serveur PostgreSQL avec le client psql

```
1 psql -h server.adress.or.ip -d database -U user
```

Complément

La commande *psql* utilisée sans paramètre se connecte au serveur *localhost* avec pour nom de *database* **et** pour nom de *user* le nom de l'utilisateur du système qui invoque la commande (utilisateur Linux typiquement).

`me@mypc:~$ psql` équivaut à : `me@mypc:~$ psql -h localhost -d me -U me`

Syntaxe : Écrire une instruction SQL

```
1 dbnf17p015=> SELECT * FROM maTable ;
```

Syntaxe : Écrire une instruction SQL sur plusieurs lignes

Une instruction SQL peut s'écrire sur une ou plusieurs lignes, le `retour chariot` n'a pas d'incidence sur la requête, c'est le `;` qui marque la fin de l'instruction SQL et provoque son exécution.

```
1 dbnf17p015=> SELECT *
2 dbnf17p015-> FROM maTable
3 dbnf17p015-> ;
```

On notera dans `psql` la différence entre les caractères `=>` et `->` selon que l'on a ou pas effectué un retour chariot.

Fondamental : Commandes de base : aide

`\?` : Liste des commandes `psql`

`\h` : Liste des instructions SQL

`\h CREATE TABLE` : Description de l'instruction SQL `CREATE TABLE`

Fondamental : Commandes de base : catalogue

`\d` : Liste des relations (catalogue de données)

`\d maTable` : Description de la relation `maTable`

Fondamental : Commandes de base : quitter

`\q` : Quitter `psql`

Complément

http://www.postgresql.org/docs/current/static/app-psql.html[10]

d) Types de données

Types standards

- numériques : integer (int2, int4, int8), real (float4, float8)
- dates : date (time, timestamp)
- chaînes : char, varchar, text
- autres : boolean

10 - http://www.postgresql.org/docs/current/static/app-psql.html

Complément : Documentation

http://docs.postgresqlfr.org/8.1/datatype.html[11]

2. Éléments complémentaires indispensables à l'utilisation de PostgreSQL

a) Exécuter des instructions SQL depuis un fichier

Il est souvent intéressant d'exécuter un fichier contenant une liste de commandes SQL, plutôt que de les entrer une par une dans le terminal. Cela permet en particulier de recréer une base de données à partir du script de création des tables.

Syntaxe

Pour exécuter un fichier contenant du code SQL utiliser la commande PostgreSQL `\i chemin/fichier.sql`

- `chemin` désigne le répertoire dans lequel est le fichier `fichier.sql`
- le dossier de travail de `psql` est le dossier dans lequel il a été lancé, le script peut être lancé à partir de son dossier `home` pour en être indépendant (`~/.../fichier.sql`)
- chaque commande doit être terminée par un `;`

```
1  dbnf17p015=> \i /home/me/bdd.sql
```

b) Importer un fichier CSV

Syntaxe

```
1  \copy nom_table (att1, att2, ...) FROM 'fichier.csv' WITH CSV DELIMITER ';'
   QUOTE '"'
```

- `WITH` introduit les options de l'import
- `CSV` indique qu'il s'agit d'un fichier CSV
- `DELIMITER 'c'` indique que le caractère **c** est utilisé comme délimiteur de champ (en général **;** ou **,**)
- `QUOTE 'c'` indique que le caractère **c** est utilisé comme délimiteur de chaîne (en général **"**)

Remarque

- La table `nom_table` doit déjà exister
- Le nombre de colonnes spécifié doit correspondre au nombre de colonnes du fichier CSV
- Les types doivent être compatibles

Remarque

Ajouter l'option `HEADER` après `WITH CSV` si le fichier CSV contient une ligne s'entête.

```
1  \copy nom_table (att1, att2, ...) FROM 'fichier.csv' WITH CSV HEADER DELIMITER
   ';' QUOTE '"'
```

Méthode : Localisation du fichier CSV depuis psql

Par défaut, la commande `\copy` prendra le chemin du répertoire courant au moment où la commande `psql` a été lancée.

11 - http://docs.postgresqlfr.org/8.1/datatype.html

Sous `psql`, vous pouvez utiliser les commandes :

- `dbnf17p007=> \! pwd`
  Pour exécuter la commande *shell* `pwd` et obtenir le répertoire courant
- `dbnf17p007=> \cd directory`
  Pour changer le répertoire courant

Complément

PostgreSQL sous Linux
Fichier CSV

c) Notion de schéma

Définition

A PostgreSQL database cluster contains one or more named databases. Users and groups of users are shared across the entire cluster, but no other data is shared across databases. Any given client connection to the server can access only the data in a single database, the one specified in the connection request.

A database contains one or more named schemas, which in turn contain tables. Schemas also contain other kinds of named objects, including data types, functions, and operators. The same object name can be used in different schemas without conflict; for example, both schema1 and myschema can contain tables named mytable. Unlike databases, schemas are not rigidly separated: a user can access objects in any of the schemas in the database he is connected to, if he has privileges to do so.

http://www.postgresql.org/docs/8.4/static/ddl-schemas.html[12]

12 - http://www.postgresql.org/docs/8.4/static/ddl-schemas.html

Graphique 12 Organisation en cluster, base, schéma, table dans PostgreSQL

Syntaxe : Créer un schéma

```
1 CREATE SCHEMA myschema;
```

Syntaxe : Créer une table dans un schéma

```
1 CREATE TABLE myschema.mytable (
2 ...
3 );
```

Syntaxe : Requêter dans un schéma

```
1 SELECT ...
2 FROM myschema.mytable
```

Exemple

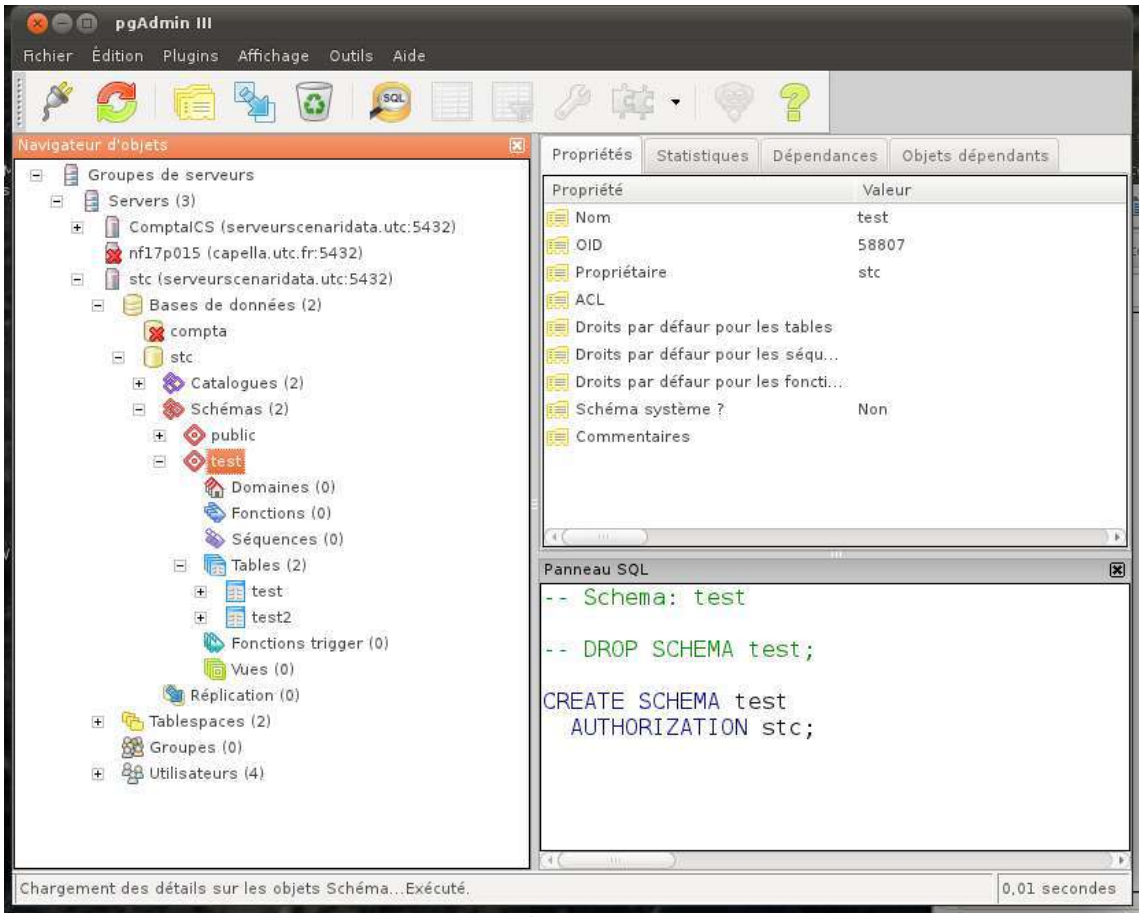


Image 14 Schéma sous PostgreSQL

Syntaxe : Catalogue : schémas

`\dn` : Liste des schémas de ma base de données

Complément : Schéma par défaut : search_path

Afin d'alléger la syntaxe il est possible de définir un schéma par défaut, dans lequel seront créer les tables non-préfixées et un ou plusieurs schémas par défaut dans lesquels seront requêtées les tables non-préfixées.

```
1  SET search_path TO myschema,public;
```

Cette instruction définit le schéma `myschema` comme schéma par défaut pour la création de table et le requêtage, puis `public` pour le requêtage, le premier étant prioritaire sur le second :

- `CREATE mytable` créera ma `mytable` dans le schéma `mychema`.
- `SELECT FROM mytable` cherchera la table dans la schéma `mychema`, puis dans le schéma `public` si la table n'existe pas dans le premier schéma.

Remarque : Schéma "public"

Le schéma `public` est un schéma créé par défaut à l'initialisation de la base, et qui sert de schéma par défaut en l'absence de toute autre spécification.

Remarque : Catalogue : \d

La liste des tables retournée par `\d` dépend du *search_path*. Pour que `\d` retourne les tables de `schema1`, il faut donc exécuter l'instruction :

```
SET search_path TO public, schema1
```

Complément : Pour aller plus loin

http://www.postgresql.org/docs/8.4/static/ddl-schemas.html[13]

d) Créer des bases de données et des utilisateurs

Syntaxe : Créer un utilisateur

```
1  CREATE USER user1 WITH ENCRYPTED PASSWORD 'password';
```

Syntaxe : Créer une base de données

```
1  CREATE DATABASE mydb WITH OWNER user1;
2
```

Complément : Supprimer des bases de données et des utilisateurs

```
1  DROP DATABASE mydb;
2  DROP USER user1;
```

Syntaxe : Catalogue : utilisateurs et bases de données

- `\du` : liste des utilisateurs
- `\l` : liste des bases de données

Syntaxe : psql : changer d'utilisateur

`\c db user` : se connecter à la base *db* avec le compte *user*.

e) PostgreSQL sous Linux

Syntaxe : Exécuter une instruction PostgreSQL depuis Linux

```
1  psql -c "instruction psql"
```

Exemple : Exécuter un script PostgreSQL depuis Linux

```
1  #!/bin/sh
2  psql -c "DROP DATABASE mydb"
3  psql -c "CREATE DATABASE mydb"
4  psql -c "GRANT ALL PRIVILEGES ON DATABASE mydb TO user1"
```

Complément : psql : exécuter une commande Linux

- `\!` : permet d'exécuter certaines commandes du *shell* Linux depuis le client *psql*.

13 - http://www.postgresql.org/docs/8.4/static/ddl-schemas.html

f) Les clients graphiques pgAdminIII et phpPgAdmin

pgAdminIII

Un client graphique une interface graphique permettant d'effectuer les mêmes opérations qu'avec le client `psql`.

- Le client graphique pgAdminIII est un client lourd qui fonctionne très bien sous Linux et sous Windows.
- Le client graphique phpPgAdmin est un client léger (qui tourne dans un navigateur Web donc).

Déclarer une connexion dans pgAdminIII

1. Sélectionner `Fichier > Ajouter un serveur`
2. Saisissez les informations de connexion à un serveur PostgreSQL existant :
   - Hôte : tuxa.sme.utc
   - Port : 5432 (port standard de PostgreSQL)
   - Base : dbnf17p...
   - Nom : nf17p...
   - Mot de passe : ...

Image 15 Fenêtre de connexion pgAdmin III

Ouvrir un terminal SQL dans pgAdminIII

1. Sélectionner sa base de données dans la liste `Bases de données`
2. Sélectionner `Outils > Éditeur de requêtes` (ou `CTRL+E`)

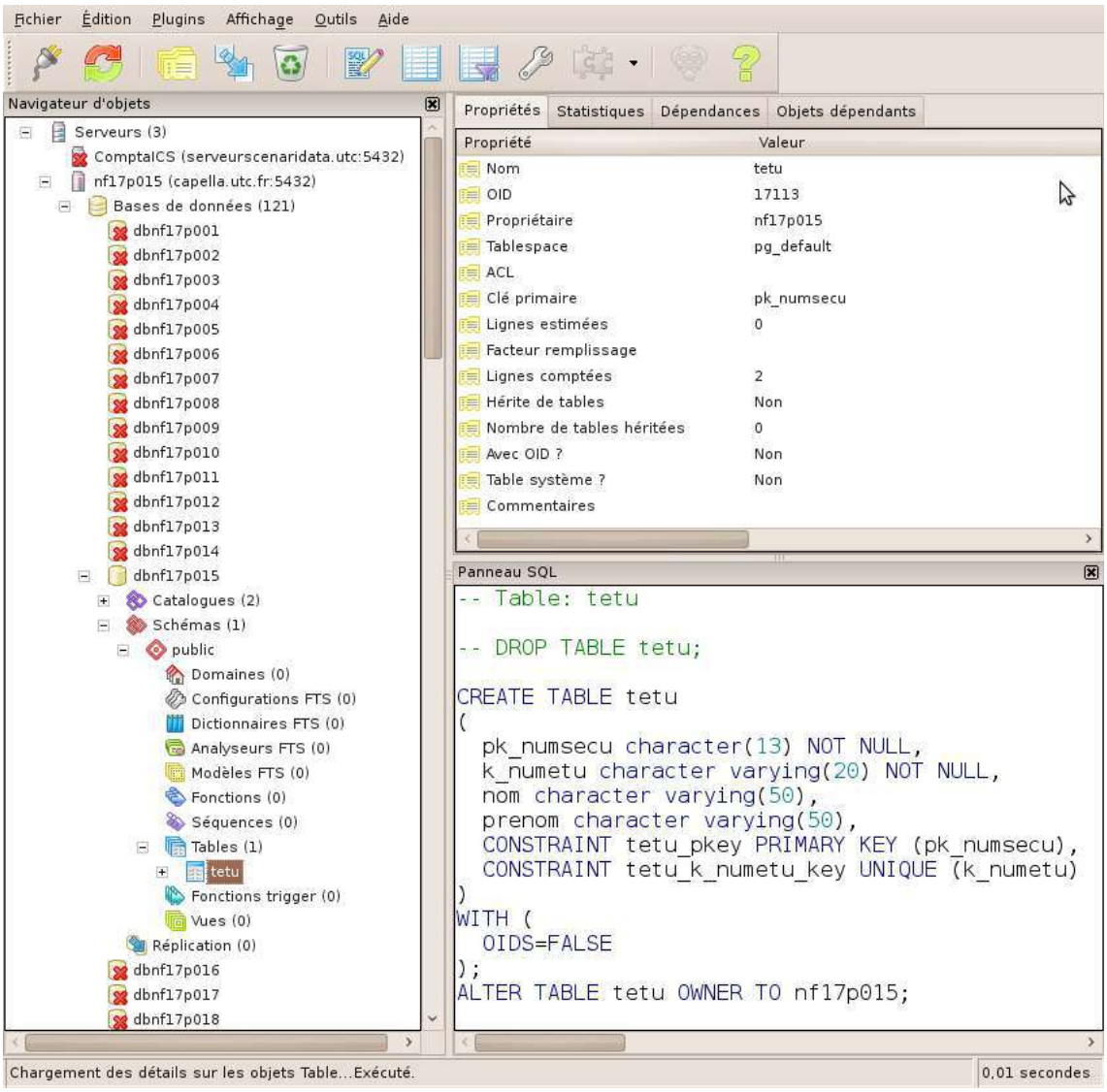


Image 16 pgAdminIII

Complément : phpPgAdmin

http://phppgadmin.sourceforge.net[14]

g) Compléments

Complément : Héritage (clause INHERITS)

http://www.postgresql.org/docs/current/static/sql-createtable.html[15]

Complément : PL/pgSQL

http://www.postgresql.org/docs/current/static/plpgsql.html[16]

Complément : Autres langages procéduraux (PL)

- PL/Tcl
- PL/Perl

14 - http://phppgadmin.sourceforge.net
15 - http://www.postgresql.org/docs/current/static/sql-createtable.html
16 - http://www.postgresql.org/docs/current/static/plpgsql.html

- PL/Python
- ...

http://www.postgresql.org/docs/current/static/xplang.html[17]
http://www.postgresql.org/docs/current/static/server-programming.html[18]

Complément : Triggers

http://www.postgresql.org/docs/current/static/sql-createtrigger.html[19]
(*http://www.postgresql.org/docs/current/static/triggers.html*[20])

B. Exercices

1. Découverte d'un SGBDR avec PostgreSQL

a) Configuration technique pour réaliser les exercices

Fondamental: Serveur Linux + Client Linux

La meilleure configuration pour réaliser ces exercices est de disposer :

1. **d'un serveur Linux hébergeant un serveur PostgreSQL ;**
2. **d'une base de données déjà créée (cela peut être la base par défaut *postgres*) ;**
3. **d'un utilisateur ayant les pleins droits sur cette base de données (cela peut être l'utilisateur par défaut *postgres*) ;**
4. **d'un client *psql* sous Linux connecté à la base de données.**

Rappel

Présentation
Le client textuel "psql"
Installation (du serveur PostgreSQL et du client psql)

Serveur Linux + Client Windows

Si un serveur est disponible sous Linux mais que le client est sous Windows, il y a deux solutions :

- **Se connecter en SSH au serveur Linux avec Putty[Putty] (on est ramené au cas précédent).**
- Installer le client *psql* sous Windows, mais cette solution est déconseillée :
  - le terminal est bien moins confortable que sous Linux,
  - une partie des questions sont relatives aux systèmes Linux et ne pourra être réalisée.

Complément : Serveur Windows + Client Windows

Cette configuration permettra de faire une partie des exercices, avec des adaptations.

b) Connexion à une base de données PostgreSQL

Cet exercice commence alors que vous êtes connecté à une base Oracle avec le client *psql*.

17 - http://www.postgresql.org/docs/current/static/xplang.html
18 - http://www.postgresql.org/docs/current/static/server-programming.html
19 - http://www.postgresql.org/docs/current/static/sql-createtrigger.html
20 - http://www.postgresql.org/docs/current/static/triggers.html

```
psql (9.x.x)
Saisissez « help » pour l'aide.

mydb=>
```

Demander de l'aide

Demander de l'aide en testant :

- help
- \?
- \h
- \h CREATE TABLE
- \h SELECT

c) Créer une base de données avec PostgreSQL

Créer une table

Exécuter les instructions suivantes.

```
CREATE TABLE etu (
pknumsecu CHAR(13) PRIMARY KEY,
knumetu VARCHAR(20) UNIQUE NOT NULL,
nom VARCHAR(50),
prenom VARCHAR(50));
```

```
INSERT INTO etu (pknumsecu, knumetu, nom, prenom)
VALUES ('1800675001066', 'AB3937098X', 'Dupont', 'Pierre');
INSERT INTO etu (pknumsecu, knumetu, nom, prenom)
VALUES ('2820475001124', 'XGB67668', 'Durand', 'Anne');
```

```
CREATE TABLE uv (
pkcode CHAR(4) NOT NULL,
fketu CHAR(13) NOT NULL,
PRIMARY KEY (pkcode, fketu),
FOREIGN KEY (fketu) REFERENCES etu(pknumsecu));
```

```
INSERT INTO uv (pkcode, fketu)
VALUES ('NF17', '1800675001066');
INSERT INTO uv (pkcode, fketu)
VALUES ('NF26', '1800675001066');
INSERT INTO uv (pkcode, fketu)
VALUES ('NF29', '1800675001066');
```

Question 1

Utiliser le catalogue pour vérifier la création de la table.

Question 2

Utiliser deux instructions SELECT pour vérifier le contenu de la table.

d) Import de données depuis un fichier CSV

Nous allons à présent réinitialiser la base avec des données contenues dans un fichier.

Question 1

Exécuter les instructions nécessaires afin de **supprimer** les données existantes dans les tables (instruction DELETE du SQL LMD). Vérifier que les tables sont vides.

Question 2

Créer les deux fichiers de données suivants, respectivement *etus.csv* et *uvs.csv*. Regarder le contenu des fichiers. Pourquoi les appelle-t-on des fichiers CSV ?

```
1;A;Dupont;Pierre
2;B;Durand;Georges
3;C;Duchemin;Paul
4;D;Dugenou;Alain
5;E;Dupied;Albert
```

```
1;NF17
1;NF18
1;NF19
1;NF20
1;LA13
1;PH01
2;NF17
2;NF18
2;NF19
2;TN01
2;LA14
2;PH01
3;NF17
3;NF18
3;NF19
3;NF21
3;LA14
3;PH01
4;NF17
4;NF20
4;NF21
4;GE10
4;LA14
4;PH01
5;NF17
5;NF18
5;NF20
5;GE10
5;PH01
5;PH02
```

Indice :

Fichier CSV

Question 3

Insérer les données en complétant les instructions suivantes.

```
\copy ... (...) FROM '...' WITH CSV DELIMITER '...'
\copy ... (...) FROM '...' WITH CSV DELIMITER '...'
```

Indice :

Importer un fichier CSV

Question 4

Écrivez les requêtes permettant d'obtenir le nombre d'UV suivies par un étudiant et le nombre d'étudiants inscrits par UV.

e) Manipulation de schémas

Question 1

Visualiser la liste des schémas existants dans votre base de données.

Indice :

Notion de schéma

Question 2

Créer un second schéma *loisir*.

Question 3

Créer une table *sport* dans le schéma *loisir* ; l'objectif est d'indiquer pour chaque étudiant, la liste des sports qu'il pratique, par exemple : Tennis, Karaté, Aviron...

Vérifier votre création dans le dictionnaire.

f) Exécution de fichiers SQL et de scripts Linux

Reporter dans un fichier SQL l'ensemble des instruction permettant de supprimer, de créer et d'instancier les tables de la base de données.

Question 1

Exécuter ce fichier depuis *psql*.

Indice :

Exécuter des instructions SQL depuis un fichier

Question 2

Exécuter ce fichier depuis un script Linux.

Indice :

PostgreSQL sous Linux

g) Test de pgAdminIII

Connexion depuis un PC avec pgAdmin III

Installer si nécessaire, puis lancer et tester le programme `pgAdminIII`.

Exécuter une ou deux requêtes permettant de se familiariser avec son fonctionnement.

Algèbre relationnelle

VII

A. Cours

Le modèle relationnel, et en particulier l'algèbre relationnel qui lui est associée, est aussi le fondement théorique du langage standard SQL, qui est utilisé pour manipuler les données stockées dans une BD.

1. Opérateurs fondamentaux : projection, restriction et jointure

Objectifs

Connaître et savoir utiliser les opérateurs relationnels de projection, restriction, produit et jointure.

a) Introduction

La représentation d'information sous forme relationnelle est intéressante car les fondements mathématiques du relationnel, outre qu'ils permettent une modélisation logique simple et puissante, fournissent également un ensemble de concepts pour manipuler formellement l'information ainsi modélisée.

Ainsi une algèbre relationnelle, sous forme d'un ensemble d'opérations formelles, permet d'exprimer des questions, ou requêtes, posées à une représentation relationnelle, sous forme d'expressions algébriques.

L'algèbre relationnelle est composée par les cinq opérateurs de base et les trois opérateurs additionnels suivants :

- **Opérateurs de base**
  - Union
  - Différence
  - Projection
  - Restriction
  - Produit cartésien
- **Opérateurs additionels**
  - Intersection
  - Jointure
  - Division

Fondamental : Algèbre relationnelle et SQL

Les questions formulées en algèbre relationnelle sont la base des questions formulées en SQL pour interroger une base de données relationnelle.

b) Employés et départements

[30 minutes]

Soit les deux relations EMP et DEPT ci-après.

```
1  EMP (#ENO, ENOM, PROF, SAL, COMM, DNO=>DEPT(DNO))
2  DEPT (#DNO, DNOM, DIR=>EMP(ENO), VILLE)
```

- ENO : numéro d'employé, clé
- ENOM : nom de l'employé
- PROF : profession (directeur n'est pas une profession)
- SAL : salaire
- COMM : commission (un employé peut ne pas avoir de commission)
- DNO : numéro de département auquel appartient l'employé
- DNO : numéro de département, clé
- DNOM : nom du département
- DIR : numéro d'employé du directeur du département
- VILLE : lieu du département (ville)

Écrire en algèbre relationnelle les requêtes permettant d'obtenir les informations suivantes.

Question 1

Lister les employés ayant des revenus supérieurs à 10.000 euros.

Question 2

Trouver le nom et la profession de l'employé numéro 10.

Question 3

Lister les noms des employés qui travaillent à Paris.

Question 4

Trouver le nom du directeur du département *Commercial*.

Question 5

Trouver les professions des directeurs des départements.

Question 6

Trouver le nom des directeurs de département ayant comme profession *Ingénieur*.

c) Projection

Définition : Projection

La projection est une opération unaire (c'est à dire portant sur une seule relation). La projection de R1 sur une partie de ses attributs {A1, A2, ...} produit une relation R2 dont le schéma est restreint aux attributs mentionnés en opérande, comportant les mêmes tuples que R1, et dont les doublons sont éliminés.

Remarque : Élimination des doublons

Après suppression d'une partie des attributs du schéma, la relation peut comporter des doublons. Étant donné que l'on ne pourrait plus identifier ces doublons les uns par rapport aux

autres, la seule solution sensée est donc de considérer que deux doublons sont équivalents, et donc de n'en garder qu'un seul dans la relation résultante.

Exemple

Soit la relation suivante :

```
1 Personne (#Nom, Prénom, Age)
```

Dupont	Pierre	20
Durand	Jean	30

Tableau 8 Personne

Soit l'opération suivante :

```
1 R = Projection (Personne, Nom, Age)
```

On obtient alors la relation R composée des tuples suivants :

Dupont	20
Durand	30

Tableau 9 R

d) Restriction

Définition : Restriction

La restriction est une opération unaire (c'est à dire portant sur une seule relation). La restriction de R1, étant donnée une condition C, produit une relation R2 de même schéma que R1 et dont les tuples sont les tuples de R1 vérifiant la condition C.

Exemple

Soit la relation suivante :

```
1 Personne (#Nom, Prénom, Age)
```

Soit les tuples suivants :

Dupont	Pierre	20
Durand	Jean	30

Tableau 10 Personne

Soit l'opération suivante :

```
1 R = Restriction (Personne, Age>25)
```

On obtient alors la relation R composée de l'unique tuple restant suivant :

Durand	Jean	30

Tableau 11 R

e) Produit

Définition : Produit cartésien

Le produit cartésien est une opération binaire (c'est à dire portant sur deux relations). Le produit de R1 par R2 (équivalent au produit de R2 par R1) produit une relation R3 ayant pour

schéma la juxtaposition de ceux des relations R1 et R2 et pour tuples l'ensemble des combinaisons possibles entre les tuples de R1 et ceux de R2.

Synonymes : Produit

Remarque

Le nombre de tuples résultant du produit de R1 par R2 est égal au nombre de tuples de R1 **fois** le nombre de tuples de R2.

Remarque

Le nombre de colonne du produit de R1 par R2 est égal au nombre de colonne de R1 **plus** le nombre de colonnes de R2.

Exemple

Soit les deux relations suivantes :

```
1  Homme (#Nom, Prénom, Age)
2  Voiture (#Type, #Marque)
```

Soit les tuples suivants pour ces deux relations respectivement :

Dupont	Pierre	20
Durand	Jean	30

Tableau 12 Homme

Tesla	Model X
Citroën	2 CV

Tableau 13 Voiture

Soit l'opération suivante :

```
1  R = Produit (Homme, Voiture)
```

On obtient alors la relation R composée des tuples suivants :

Dupont	Pierre	20	Tesla	Model X
Dupont	Pierre	20	Citroën	2 CV
Durand	Jean	30	Tesla	Model X
Durand	Jean	30	Citroën	2 CV

Tableau 14 R

Remarque

Le produit cartésien est rarement utilisé seul, mais il est à la base de la jointure.

f) Jointure

Définition : Jointure

La jointure est une opération binaire (c'est à dire portant sur deux relations). La jointure de R1 et R2, étant donné une condition C portant sur des attributs de R1 et de R2, **de même domaine**, produit une relation R3 ayant pour schéma la juxtaposition de ceux des relations R1 et R2 et pour tuples l'ensemble de ceux obtenus par concaténation des tuples de R1 et de R2, et qui vérifient la condition C.

Exemple

Soit les deux relations suivantes :

```
Homme (#Nom, Prénom, Age)
Voiture (#Type, #Marque, Propriétaire)
```

Soit les tuples suivants pour ces deux relations respectivement :

Dupont	Pierre	20
Durand	Jean	30

Tableau 15 Homme

Tesla	Model X	Dupont
Citroën	2 CV	Durand
Citroën	3 CV	Dupont

Tableau 16 Voiture

Soit l'opération suivante :

```
R = Jointure (Homme, Voiture, Homme.Nom=Voiture.Propriétaire)
```

On obtient alors la relation R composée des tuples suivants :

Dupont	Pierre	20	Tesla	Model X	Dupont
Dupont	Pierre	20	Citroën	3 CV	Dupont
Durand	Jean	30	Citroën	2 CV	Durand

Tableau 17 R

Fondamental

La jointure est l'opération qui permet de rassembler les informations séparées entre plusieurs tables et référencées par des clés étrangères.

Remarque : Opération additionnelle

La jointure n'est pas une opération de base, elle peut être réécrite en combinant le produit et la restriction.

g) Exercice

Quelles sont les expressions relationnelles équivalentes à :

```
Projection ( Jointure (R1, R2, R1.A1=R2.A1), R1.A1, R2.A2)
```

- ☐ Jointure (Projection(R1, A1), Projection(R2, A2), R1.A1=R2.A1)
- ☐ Projection (Jointure (R2, R1, R2.A1=R1.A1), R1.A1, R2.A2)
- ☐ Projection (Restriction (Produit(R1, R2), R1.A1=R2.A1), R1.A1, R2.A2)
- ☐ Produit (R1, R2, R1.A1=R2.A1, R1.A1, R2.A2)

2. Opérateurs complémentaires

Objectifs

Maîtriser l'algèbre relationnelle.

a) Jointure naturelle

Définition : Jointure naturelle

La jointure naturelle entre R1 et R2 est une jointure pour laquelle la condition est l'égalité entre les attributs de même nom de R1 et de R2. Il est donc inutile de spécifier la condition dans une jointure naturelle, elle reste toujours implicite.

Exemple

Soit deux relations R1 (A, B, C) et R2 (A, D), l'opération Jointure(R1,R2,R1.A=R2.A) est équivalente à l'opération JointureNaturelle(R1,R2).

Remarque

Pour appliquer une jointure naturelle, il faut que les deux relations opérandes aient au moins un attribut ayant le même nom en commun.

b) Jointure externe

Introduction

La jointure est une opération qui entraîne la perte de certains tuples : ceux qui appartiennent à une des deux relations opérandes et qui n'ont pas de correspondance dans l'autre relation. Il est nécessaire dans certains cas de palier cette lacune, et l'on introduit pour cela la notion de jointure externe.

Définition : Jointure externe

La jointure externe entre R1 et R2 est une jointure qui produit une relation R3 à laquelle on ajoute les tuples de R1 et de R2 exclus par la jointure, en complétant avec des valeurs nulles pour les attributs de l'autre relation.

Définition : Jointure externe gauche

La jointure externe gauche entre R1 et R2 est une jointure externe pour laquelle on ajoute seulement les tuples de R1 (c'est à dire la relation de gauche) ayant été exclus.

Synonymes : Jointure gauche

Définition : Jointure externe droite

La jointure externe droite entre R1 et R2 est une jointure externe pour laquelle on ajoute seulement les tuples de R2 (c'est à dire la relation de droite) ayant été exclus.

Bien entendu une jointure externe droite peut être réécrite par une jointure externe gauche (et réciproquement) en substituant les relations opérandes R1 et R2.

Synonymes : Jointure droite

Exemple

Soit les deux relations suivantes :

```
1 Homme (#Nom, Prénom, Age)
2 Voiture (#Type, #Marque, Propriétaire)
```

Soit les tuples suivants pour ces deux relations respectivement :

Dupont	Pierre	20
Durand	Jean	30
Martin	Georges	40

Tableau 18 Homme

Tesla	Model X	Dupont
Citroën	2 CV	Durand
Citroën	3 CV	NULL

Tableau 19 Voiture

Soit l'opération suivante :

```
1 R = JointureExterne (Homme, Voiture, Homme.Nom=Voiture.Propriétaire)
```

On obtient alors la relation R composée des tuples suivants :

Dupont	Pierre	20	Tesla	Model X	Dupont
Durand	Jean	30	Citroën	2 CV	Durand
Martin	Georges	40	NULL	NULL	NULL
NULL	NULL	NULL	Citroën	3 CV	NULL

Tableau 20 R

Une jointure externe gauche n'aurait renvoyé que les trois premiers tuples et une jointure externe droite n'aurait renvoyée que les deux premiers et le dernier tuple.

c) Opérateurs ensemblistes

Attention

Les opérateurs ensemblistes sont des relations binaires (c'est à dire entre deux relations) portant sur des relations de même schéma.

Définition : Union

L'union de deux relations R1 et R2 de même schéma produit une relation R3 de même schéma constituée de l'ensemble des tuples appartenant à R1 et/ou à R2.

Définition : Différence

La différence entre deux relations R1 et R2 de même schéma produit une relation R3 de même schéma constituée de l'ensemble des tuples de R1 n'appartenant pas à R2. Notons que la différence entre R1 et R2 n'est pas égale à la différence entre R2 et R1.

Définition : Intersection

L'intersection de deux relations R1 et R2 de même schéma produit une relation R3 de même schéma constituée de l'ensemble des tuples appartenant à la fois à R1 et à R2. Notons que l'intersection n'est pas une opération de base, car elle est équivalent à deux opérations de différence successives.

Exemple

Soit les deux relations suivantes :

```
1 Homme (#Nom, Prénom, Age)
```

```
2 Femme (#Nom, Prénom, Age)
```

Soit les tuples suivants pour ces deux relations respectivement :

Dupont	Pierre	20
Durand	Jean	30

Tableau 21 Homme

Martin	Isabelle	24
Blanc	Hélène	25

Tableau 22 Femme

Soit l'opération suivante :

```
1 R = Union (Homme, Femme)
```

On obtient alors la relation R composée des tuples suivants :

Dupont	Pierre	20
Durand	Jean	30
Martin	Isabelle	24
Blanc	Hélène	25

Tableau 23 R

La différence entre Homme et Femme (respectivement entre Femme et Homme) renvoie la relation Homme (respectivement Femme), car aucun tuple n'est commun aux deux relations. L'intersection entre Homme est Femme est vide, pour la même raison.

Remarque : Union externe

Il est possible de définir une opération d'union externe, qui permet de réaliser l'union de deux relations de schéma différent, en ramenant les relations aux mêmes schémas, et en les complétant avec des valeurs nulles.

d) Division

Définition : Division

La division est une opération binaire (c'est à dire portant sur deux relations). La division de R1 par R2, sachant que R1 et R2 ont au moins un attribut commun (c'est à dire de même nom et de même domaine), produit une relation R3 qui comporte les attributs appartenant à R1 mais n'appartenant pas à R2 et l'ensemble des tuples qui concaténés à ceux de R2 donnent toujours un tuple de R1.

Exemple

Soit les deux relations suivantes :

```
1 Pratique (#Homme, #Métier, Salaire)
2 Métier (#Metier)
```

Soit les tuples suivants pour ces deux relations respectivement :

Dupont	Ingénieur	35
Durand	Professeur	40
Dupont	Ingénieur	45

Martin	Ingénieur	50

Tableau 24 Pratique

Ingénieur
Professeur

Tableau 25 Métier

Soit l'opération suivante :

```
1 R = Division (Homme, Métier)
```

On obtient alors la relation R composée des tuples suivants :

Dupont	35

Tableau 26 R

Méthode : Réponse aux questions : Pour tous les ...

La division permet de répondre aux questions du type : "Donnez toutes les personnes qui pratiquent tous les métiers de la relation métier".

Remarque : Opération additionnelle

La division n'est pas une opération de base, elle peut être réécrite en combinant le produit, la restriction et la différence.

e) Proposition de notations

Introduction

Il existe plusieurs syntaxes pour écrire des opérations d'algèbre relationnelle, certaines inspirées de l'algèbre classiques, d'autres reposant sur des notations graphiques. Nous proposons une notation fonctionnelle qui a le mérite d'être facile à écrire et d'être lisible. Si cette notation peut parfois perdre en simplicité, lorsqu'elle concerne un nombre élevé d'opérateurs, il est possible de décomposer une opération compliquée afin de l'alléger.

Syntaxe

```
1  R = Union (R1, R2)
2  R = Différence (R1, R2)
3  R = Intersection (R1, R2)
4  R = Projection (R1, A1, A2, ...)
5  R = Restriction (R1, condition)
6  R = Produit (R1, R2)
7  R = Jointure (R1, R2, condition)
8  R = JointureNaturelle (R1, R2)
9  R = JointureExterne (R1, R2, condition)
10 R = JointureGauche (R1, R2, condition)
11 R = JointureDroite (R1, R2, condition)
12 R = Division (R1, R2)
```

Exemple : Notation synthétique

```
1 R = Projection(Restriction(R1, A1=1 AND A2=2), A3)
```

Exemple : Notation décomposée

```
1 R' = Restriction(R1, A1=1 AND A2=2)
2 R = Projection (R', A3)
```

f) Exercice

Soit les deux relations R1 et R2 suivantes, définies en extension :

A	B
1	A
2	B
3	C

Tableau 27 R1

A
1
2

Tableau 28 R2

Combien de tuples renvoie l'opération relationnelle suivante ?

```
1 R3 = JointureNaturelle (Intersection (Projection(R1,A), R2), R2)
```

g) Opérateurs de base et additionnels

Réécrivez les opérateurs additionnels suivants, à partir d'opérateurs de base :

Question 1

Réécrivez Intersection à partir de Différence

Question 2

Réécrivez Jointure à partir de Produit et Restriction

B. Exercices

1. Faire du Cinéma

[30 minutes]

On considère les deux relations suivantes :

```
1  FILMS (titre, pays, année, réalisateur, durée)
2  ACTEURS (titre, acteur)
```

où les attributs ont les significations et les types suivants :

- **titre** : titre d'un film (chaîne 50 caractères)
- **pays** : pays d'où un film est originaire (chaîne 10 caractères)
- **annee** : année de sortie du film (entier 4 chiffres)
- **realisateur** : nom du réalisateur du film (chaîne 20 caractères)
- **duree** : durée du film en minutes (entier 3 chiffres)
- **acteur** : nom d'acteur (chaîne 20 caractères)

La relation **FILMS** donne pour chaque film, identifié par son titre, le pays, l'année de sortie, réalisateur et la durée.

La relation **ACTEURS** donne pour chaque film l'ensemble des principaux acteurs.

À l'aide de l'algèbre relationnelle, exprimer les requêtes suivantes :

Question 1

Lister les films français (titre, année, réalisateur).

Question 2

Donnez les années de sortie des films dans lesquels l'acteur Jean GABIN a joué.

Question 3

Trouver les acteurs qui ont tourné avec François Truffaut comme réalisateur.

Question 4

Trouver tous les acteurs qui ont été partenaires de l'actrice Catherine Deneuve.

Question 5

Lister les films dans lesquels le réalisateur est aussi acteur.

Question 6

Lister les réalisateurs n'ayant joué comme acteurs que dans des films qu'ils ne réalisaient pas eux-mêmes.

Question 7

Lister les réalisateurs ayant joué comme acteurs dans des films qu'ils ne réalisaient pas eux-mêmes.

Question 8

Donnez les acteurs qui jouent dans tous les films de François TRUFFAUT.

2. Le retour des écoliers

[45 minutes]

Soit le schéma relationnel suivant :

```
1  IMMEUBLE (#ADI, NBETAGES, DATEC, PROP)
2  APPIM (#ADI, #NAPR, OCCUP, TYPE, SUPER, ETAGE)
3  PERSONNE (#NOM, AGE, PROF, ADR, NAPR)
4  ÉCOLE (#NOMEC, ADEC, DIR)
5  CLASSE (#NOMEC, #NCL, MAITRE)
6  ENFANT (#NOMP, #PRENOM, AN, NOMEC, NCL)
```

Avec la signification suivante :

- **Relation IMMEUBLE**

  ADI : adresse d'immeuble, clé ; on fait l'hypothèse pour simplifier, que l'adresse identifie de manière unique un immeuble

  NBETAGES : nombre d'étages d'un immeuble

  DATEC : date de construction (année)

  PROP : nom du propriétaire de l'immeuble qui est une personne

- **Relation APPIM (Appartement)**

  ADI : adresse d'immeuble

  NAPR : numéro d'appartement

  OCCUP : occupant de l'appartement (nom de la personne ayant signé le contrat de location, éventuellement aucun)

  TYPE : type de l'appartement (Studio, F2, ...)

  SUPER : superficie de l'appartement

  ETAGE : étage où se situe l'appartement

- **Relation PERSONNE**

  NOM : nom de personne, clé ; on fait l'hypothèse pour simplifier, que ce nom est unique sur l'ensemble des personnes que l'on considère dans la base

  AGE : âge de la personne

  PROF : profession de la personne

  ADR : adresse de la résidence d'une personne, il s'agit d'un immeuble

  NAPR : numéro d'appartement

- **Relation ÉCOLE**

  NOMEC : nom d'une école, clé

  ADEC : adresse d'une école

  DIR : nom du directeur

- **Relation CLASSE**

  NOMEC : nom d'une école

  NCL : nom de la classe, e.g., CP1, CE2, CE3, etc...

  MAITRE : nom de l'instituteur

- **Relation ENFANT**

  NOMP : nom de la personne responsable de l'enfant, clé e.g., père, mère etc...

  PRENOM : prénom de l'enfant

  AN : année de naissance

  NOMEC : nom d'une école

  NCL : nom de la classe

La relation IMMEUBLE décrit un ensemble d'immeubles. Chaque immeuble a un propriétaire. La relation APPIM décrit pour chaque immeuble l'ensemble des appartements qui le compose (il y a au mimimum un appartement par immeuble). Chaque appartement peut héberger plusieurs personnes mais il y en a une qui est responsable (par exemple la personne qui a signé le contrat de location) et qui est désignée par l'attribut OCCUP. Si l'appartement est inoccupé, il prend la valeur NULL. La relation PERSONNE décrit un ensemble de personnes. ADR et NAPR représentent l'adresse où réside une personne. Une personne peut avoir plusieurs enfants décrits par la relation ENFANT. Pour simplifier, on ne considère que les enfants allant à l'école primaire. Les écoles et les classes sont décrites dans les relations ÉCOLE et CLASSE, chaque école est composée au minimum d'une classe et chaque classe est au moins fréquentée par un enfant.

Question 1

Donner l'adresse des immeubles ayant plus de 10 étages et construits avant 1970.

Question 2

Donner les noms des personnes qui habitent dans un immeuble dont ils sont propriétaires.

Question 3

Donner les noms des personnes qui ne sont pas propriétaires.

Question 4

Donner les adresses des immeubles possédés par des informaticiens dont l'âge est inférieur à 40 ans .

Question 5

Donner la liste des occupants (nom, âge, profession) des immeubles possédés par DUPONT.

Question 6

Donner le nom et la profession des propriétaires d'immeubles dans lesquels il y a des appartements vides.

Question 7

Donner les noms des maîtres qui habitent dans le même immeuble (à la même adresse) qu'au moins un de leurs élèves (on suppose que les enfants vivent sous le même toit que leur parents).

Question 8

Donner l'adresse de l'immeuble, la date de construction, le type d'appartement et l'étage où habitent chacun des maîtres des enfants de DUPONT.

3. Quiz : Algèbre relationnelle

Exercice 1

Quelles sont les opérations relationnelles, qui appliquées sur les relations instanciées ci-dessous, renvoient un ensemble non nul de tuples ?

Num	Nom	Famille
1	Ours	Mammifère
2	Truite	Poisson
3	Homme	Mammifère
4	Martin-pêcheur	Oiseau

Tableau 29 Animal

Num	Nom
1	Forêt
2	Montagne
3	Ciel
4	Rivière
5	Mer

Tableau 30 Environnement

Animal	Environnement
1	1
1	2
1	4
2	4
4	3

Tableau 31 Habiter

Mangeur	Mangé	Fréquence
1	2	Souvent
1	3	Rarement
1	4	Rarement
4	2	Souvent
3	1	Rarement
3	2	Souvent

Tableau 32 Manger

- ☐ Restriction(Projection (Animal, Nom, Famille), Famille='Mammifère')
- ☐ Restriction(Jointure(Jointure (Animal, Habiter, Animal.Num=Habiter.Animal), Environnement, Environnement.Num=Habiter.Environnement), Animal.Num='3')
- ☐ Restriction(JointureExterneGauche(Animal, Habiter, Animal.Num=Habiter.Animal), Animal.Nom='Homme')
- ☐ Jointure(Animal, Manger, Animal.Num=Manger.Mangeur AND Animal.Num=Manger.Mangé)

Exercice 2

Soit le schéma relationnel :

```
1 R1(X, Y)
2 R2(X, Y)
```

Quelles sont les opérations relationnelles équivalentes à l'opération :

```
1 Projection( JointureNaturelle(R1,R2), R1.X)
```

- [ ] JointureNaturelle(Projection(R1, X), Projection(R2, X))
- [ ] Projection(Selection (Produit(R1, R2), R1.X=R2.X AND R1.Y=R2.Y), R1.X)
- [ ] Projection(Union(R1, R2), R1.X)
- [ ] Projection(JointureExterne (R1, R2, R1.X=R2.X AND R1.Y=R2.Y), R1.X)
- [ ] Projection(Jointure (R1, R2, R1.X=R2.X AND R1.Y=R2.Y), R1.X)

Exercice 3

Soit la relation instanciée suivante :

A	B	C
1	1	0
1	0	1
0	1	1

Tableau 33 Relation R1

Quelles relations sont retournées par l'opération relationnelle suivante :

```
1 R2 = JointureNaturelle(R1, R1)
```

1	1	0
1	0	1
0	1	1

Tableau 34 R2a

1	1	0
1	0	1
0	1	1
1	1	0
1	0	1
0	1	1

Tableau 35 R2b

1	1	1
0	0	0

Tableau 36 R2c

1	1	0	1	1	0
1	0	1	1	0	1
0	1	1	0	1	1

Tableau 37 R2d

☐	R2a
☐	R2b
☐	R2c
☐	R2d
☐	Une relation vide (aucun tuple)

Interrogation de bases de données SQL

VIII

A. Cours

SQL✯ est un langage standardisé, implémenté par tous les SGBDR✯, qui permet, indépendamment de la plate-forme technologique et de façon déclarative, de définir le modèle de données, de le contrôler et enfin de le manipuler.

1. Questions simples avec le Langage de Manipulation de Données (SELECT)

a) Représentation de représentants

[30 minutes]

Soit le schéma relationnel et le code SQL suivants :

```
1  REPRESENTANTS (#NR, NOMR, VILLE)
2  PRODUITS (#NP, NOMP, COUL, PDS)
3  CLIENTS (#NC, NOMC, VILLE)
4  VENTES (#NR=>REPRESENTANTS(NR), #NP=>PRODUITS(NP), #NC=>CLIENTS(NC), QT)
```

```
1  /* Les requêtes peuvent être testées dans un SGBDR, en créant une base de données
     avec le script SQL suivant */
2
3  /*
4  DROP TABLE VENTES ;
5  DROP TABLE CLIENTS ;
6  DROP TABLE PRODUITS ;
7  DROP TABLE REPRESENTANTS ;
8  */
9
10 CREATE TABLE REPRESENTANTS (
11   NR INTEGER PRIMARY KEY,
12   NOMR VARCHAR,
13   VILLE VARCHAR
14 );
15
16 CREATE TABLE PRODUITS (
```

```
 NP INTEGER PRIMARY KEY,
 NOMP VARCHAR,
 COUL VARCHAR,
 PDS INTEGER
);

CREATE TABLE CLIENTS (
 NC INTEGER PRIMARY KEY,
 NOMC VARCHAR,
 VILLE VARCHAR
);

CREATE TABLE VENTES (
 NR INTEGER REFERENCES REPRESENTANTS(NR),
 NP INTEGER REFERENCES PRODUITS(NP),
 NC INTEGER REFERENCES CLIENTS(NC),
 QT INTEGER,
 PRIMARY KEY (NR, NP, NC)
);

INSERT INTO REPRESENTANTS (NR, NOMR, VILLE) VALUES (1, 'Stephane', 'Lyon');
INSERT INTO REPRESENTANTS (NR, NOMR, VILLE) VALUES (2, 'Benjamin', 'Paris');
INSERT INTO REPRESENTANTS (NR, NOMR, VILLE) VALUES (3, 'Leonard', 'Lyon');
INSERT INTO REPRESENTANTS (NR, NOMR, VILLE) VALUES (4, 'Antoine', 'Brest');
INSERT INTO REPRESENTANTS (NR, NOMR, VILLE) VALUES (5, 'Bruno', 'Bayonne');

INSERT INTO PRODUITS (NP, NOMP, COUL, PDS) VALUES (1, 'Aspirateur', 'Rouge',
3546);
INSERT INTO PRODUITS (NP, NOMP, COUL, PDS) VALUES (2, 'Trottinette', 'Bleu',
1423);
INSERT INTO PRODUITS (NP, NOMP, COUL, PDS) VALUES (3, 'Chaise', 'Blanc', 3827);
INSERT INTO PRODUITS (NP, NOMP, COUL, PDS) VALUES (4, 'Tapis', 'Rouge', 1423);

INSERT INTO CLIENTS (NC, NOMC, VILLE) VALUES (1, 'Alice', 'Lyon');
INSERT INTO CLIENTS (NC, NOMC, VILLE) VALUES (2, 'Bruno', 'Lyon');
INSERT INTO CLIENTS (NC, NOMC, VILLE) VALUES (3, 'Charles', 'Compiègne');
INSERT INTO CLIENTS (NC, NOMC, VILLE) VALUES (4, 'Denis', 'Montpellier');
INSERT INTO CLIENTS (NC, NOMC, VILLE) VALUES (5, 'Emile', 'Strasbourg');

INSERT INTO VENTES (NR, NP, NC, QT) VALUES (1, 1, 1, 1);
INSERT INTO VENTES (NR, NP, NC, QT) VALUES (1, 1, 2, 1);
INSERT INTO VENTES (NR, NP, NC, QT) VALUES (2, 2, 3, 1);
INSERT INTO VENTES (NR, NP, NC, QT) VALUES (4, 3, 3, 200);
INSERT INTO VENTES (NR, NP, NC, QT) VALUES (3, 4, 2, 190);
INSERT INTO VENTES (NR, NP, NC, QT) VALUES (1, 3, 2, 300);
INSERT INTO VENTES (NR, NP, NC, QT) VALUES (3, 1, 2, 120);
INSERT INTO VENTES (NR, NP, NC, QT) VALUES (3, 1, 4, 120);
INSERT INTO VENTES (NR, NP, NC, QT) VALUES (3, 4, 4, 2);
INSERT INTO VENTES (NR, NP, NC, QT) VALUES (3, 1, 1, 3);
INSERT INTO VENTES (NR, NP, NC, QT) VALUES (3, 4, 1, 5);
INSERT INTO VENTES (NR, NP, NC, QT) VALUES (3, 1, 3, 1);
```

Écrire en SQL les requêtes permettant d'obtenir les informations ci-après.

Question 1

Tous les détails de tous les clients.

Question 2

Les numéros et les noms des produits de couleur rouge et de poids supérieur à 2000.

Question 3

Les représentants ayant vendu au moins un produit.

Question 4

Les noms des clients de Lyon ayant acheté un produit pour une quantité supérieure à 180.

Question 5

Les noms des représentants et des clients à qui ces représentants ont vendu un produit de couleur rouge pour une quantité supérieure à 100.

b) Question (SELECT)

Fondamental : Question

La requête de sélection ou question est la base de la recherche de données en SQL.

Définition : Sélection

La sélection est la composition d'un **produit cartésien**, d'une **restriction** et d'une **projection** (ou encore la composition d'une jointure et d'une projection).

Syntaxe

```
1  SELECT liste d'attributs projetés
2  FROM liste de relations du produit cartésien
3  WHERE condition de la restriction
```

- La partie SELECT indique le sous-ensemble des attributs qui doivent apparaître dans la réponse (c'est le schéma de la relation résultat).
- La partie FROM décrit les relations qui sont utilisables dans la requête (c'est à dire l'ensemble des attributs que l'on peut utiliser).
- La partie WHERE exprime les conditions que doivent respecter les attributs d'un tuple pour pouvoir être dans la réponse. Une condition est un prédicat et par conséquent renvoie un booléen. Cette partie est optionnelle.

Exemple

```
1  SELECT Nom, Prenom
2  FROM Personne
3  WHERE Age>18
```

Cette requête sélectionne les attributs Nom et Prénom des tuples de la relation Personne, ayant un attribut Age supérieur à 18.

Syntaxe : Notation préfixée

Afin de décrire un attribut d'une relation en particulier (dans le cas d'une requête portant sur plusieurs relations notamment), on utilise la notation `relation.attribut`.

Exemple

```
1  SELECT Personne.Nom, Personne.Prenom, Vol.Depart
2  FROM Personne, Vol
3  WHERE Personne.Vol=Vol.Numero
```

*Syntaxe : SELECT **

Pour projeter l'ensemble des attributs d'une relation, on peut utiliser le caractère `*` à la place de la liste des attributs à projeter.

Exemple

```
1  SELECT *
2  FROM Avion
```

Cette requête sélectionne tous les attributs de la relation Avion.

Notons que dans cet exemple, la relation résultat est exactement la relation Avion

c) Opérateurs de comparaisons et opérateurs logiques

Introduction

La clause WHERE d'une instruction de sélection est définie par une condition. Une telle condition s'exprime à l'aide d'opérateurs de comparaison et d'opérateurs logiques. Le résultat d'une expression de condition est toujours un booléen.

Définition : Condition

```
1 Condition Elémentaire ::= Propriété <Opérateur de comparaison> Constante
2 Condition ::= Condition <Opérateur logique> Condition | Condition Elémentaire
```

Les opérateurs de comparaison sont :

- P = C
- P <> C
- P < C
- P > C
- P <= C
- P >= C
- P BETWEEN C1 AND C2
- P IN (C1, C2, ...)
- P LIKE 'chaîne'
- P IS NULL

Les opérateur logique sont :

- OR
- AND
- NOT

Remarque : Opérateur LIKE

L'opérateur `LIKE` 'chaîne' permet d'insérer des jokers dans l'opération de comparaison (alors que l'opérateur `=` teste une égalité stricte) :

- Le joker `%` désigne 0 ou plusieurs caractères quelconques
- Le joker _ désigne 1 et 1 seul caractère

On préférera l'opérateur `=` à l'opérateur `LIKE` lorsque la comparaison n'utilise pas de joker.

d) Renommage de colonnes et de tables avec les alias

Syntaxe : Alias de table

Il est possible de redéfinir le nom des relations au sein de la requête afin d'en simplifier la syntaxe.

```
1 SELECT t1.attribut1, t2.attribut2
2 FROM table1 t1, table2 t2
```

Exemple

```
1 SELECT Parent.Prenom, Enfant.Prenom
2 FROM Parent, Enfant
3 WHERE Enfant.Nom=Parent.Nom
```

Cette requête sélectionne les prénoms des enfants et des parents ayant le même nom. On remarque la notation Parent.Nom et Enfant.Nom pour distinguer les attributs Prenom des relations Parent et Enfant.

On notera que cette sélection effectue une jointure sur les propriétés Nom des relations Parent et Enfant.

Remarque : Alias d'attribut (AS)

Il est possible de redéfinir le nom des propriétés de la relation résultat.

```
1  SELECT attribut1 AS a1, attribut2 AS a2
2  FROM table
```

e) Dédoublonnage (SELECT DISTINCT)

Attention : SELECT DISTINCT

L'opérateur SELECT n'élimine pas les doublons (i.e. les tuples identiques dans la relation résultat) par défaut. Il faut pour cela utiliser l'opérateur SELECT DISTINCT.

Exemple

```
1  SELECT DISTINCT Avion
2  FROM Vol
3  WHERE Date=31-12-2000
```

Cette requête sélectionne l'attribut Avion de la relation Vol, concernant uniquement les vols du 31 décembre 2000 et renvoie les tuples sans doublons.

f) Tri (ORDER BY)

Introduction

On veut souvent que le résultat d'une requête soit trié en fonction des valeurs des propriétés des tuples de ce résultat.

Syntaxe : ORDER BY

```
1  SELECT liste d'attributs projetés
2  FROM liste de relations
3  WHERE condition
4  ORDER BY liste ordonnée d'attributs
```

Les tuples sont triés d'abord par le premier attribut spécifié dans la clause ORDER BY, puis en cas de doublons par le second, etc.

Remarque : Tri décroissant

Pour effectuer un tri décroissant on fait suivre l'attribut du mot clé "DESC".

Exemple

```
1  SELECT *
2  FROM Personne
3  ORDER BY Nom, Age DESC
```

g) Projection de constante

Syntaxe : Projection de constante

Il est possible de projeter directement des constantes (on utilisera généralement un alias d'attribut pour nommer la colonne).

```
1  SELECT constante AS nom
```

Cette requête renverra une table avec une seule ligne et une seule colonne à la valeur de *constante*.

Exemple

```
1  SELECT '1' AS num
```

```
1   num
2  -----
3   1
```

Exemple

```
1  SELECT 'Hello world' AS hw
```

```
1        hw
2  -------------
3   Hello world
```

Exemple

```
1  SELECT CURRENT_DATE AS now
```

```
1       now
2  ------------
3   2016-10-21
```

h) Commentaires en SQL

L'introduction de commentaires au sein d'une requête SQL se fait :

- En précédant les commentaires de `--` pour les commentaires mono-ligne
- En encadrant les commentaires entre `/*` et `*/` pour les commentaires multi-lignes.

Exemple

```
1  /*
2  * Creation of table Student
3  * Purpose : managing information about students in the system
4  */
5  CREATE TABLE person (
6  pknum VARCHAR(13) PRIMARY KEY, --Student national number
7  name VARCHAR(25)
8  );
9
10
```

2. Opérations d'algèbre relationnelle en SQL

a) Expression d'une restriction

Syntaxe

```
1  SELECT *
2  FROM R
3  WHERE <condition>
```

Exemple

R1	#A	B	C=>R2
	1	Alpha	10
	2	Bravo	10
	3	Charlie	20
	4	Delta	

R	A	B	C
	1	Alpha	10
	2	Bravo	10

```
Restriction(R1,C<20)

SELECT *
FROM R1
WHERE C<20
```

Tableau 38 Exemple de restriction (SQL et Algèbre)

b) Expression d'une projection

Syntaxe

```
1  SELECT P1, P2, Pi
2  FROM R
```

Exemple

R1

#A	B	C=>R2
1	Alpha	10
2	Bravo	10
3	Charlie	20
4	Delta	

R

A	C
1	10
2	10
3	20
4	

```
Projection(R1,A,C)

SELECT A, C
FROM R1
```

Tableau 39 Exemple de projection (SQL et Algèbre)

c) Expression du produit cartésien

Syntaxe

```
1  SELECT *
2  FROM R1, R2, Ri
```

Exemple

R1

#A	B	C=>R2
1	Alpha	10
2	Bravo	10
3	Charlie	20
4	Delta	

R2

#X	Y
10	Echo
20	Fox
30	Golf

A	B	C	X	Y
1	Alpha	10	10	Echo
1	Alpha	10	20	Fox
1	Alpha	10	30	Golf
2	Bravo	10	10	Echo
2	Bravo	10	20	Fox
2	Bravo	10	30	Golf
3	Charlie	20	10	Echo
3	Charlie	20	20	Fox
3	Charlie	20	30	Golf
4	Delta		10	Echo
4	Delta		20	Fox
4	Delta		30	Golf

```
Produit(R1,R2)

SELECT *
FROM R1,R2
```

Tableau 40 Exemple de produit (SQL et Algèbre)

d) Expression d'une jointure

Syntaxe : Jointure par la clause WHERE

En tant que composition d'un produit cartésien et d'une restriction la jointure s'écrit :

```
1  SELECT *
2  FROM R1, R2, Ri
3  WHERE <condition>
```

Avec `Condition` permettant de joindre des attributs des `Ri`

Syntaxe : Jointure par la clause ON

On peut également utiliser la syntaxe dédiée suivante :

```
1  SELECT *
2  FROM R1 INNER JOIN R2
3  ON <condition>
```

Et pour plusieurs relations :

```
1  SELECT *
2  FROM (R1 INNER JOIN R2 ON <condition>) INNER JOIN Ri ON <condition>
3
```

Exemple

R1

#A	B	C=>R2
1	Alpha	10
2	Bravo	10
3	Charlie	20
4	Delta	

R2

#X	Y
10	Echo
20	Fox
30	Golf

R

A	B	C	X	Y
1	Alpha	10	10	Echo
2	Bravo	10	10	Echo
3	Charlie	20	20	Fox

```
Jointure(R1,R2,R1.C=R2.X)

SELECT *
FROM R1 INNER JOIN R2
ON R1.C=R2.X
```

Tableau 41 Exemple de jointure (SQL et Algèbre)

Exemple : Une jointure naturelle

```
1  SELECT *
2  FROM R1, R2
3  WHERE R2.NUM = R1.NUM
```

Remarque : Auto-jointure

Pour réaliser une auto-jointure, c'est à dire la jointure d'une relation avec elle-même, on doit utiliser le renommage des relations. Pour renommer une relation, on note dans la clause FROM le nom de renommage après le nom de la relation : "FROM NOM_ORIGINAL NOUVEAU_NOM".

Exemple : Auto-jointure

```sql
SELECT E1.Nom
FROM Employe E1, Employe E2
WHERE E1.Nom= E2.Nom
```

e) Exercice : Tableau final

Que renvoie la dernière instruction SQL de la séquence ci-dessous ?

```sql
CREATE TABLE R (a INTEGER, b INTEGER);
INSERT INTO R VALUES (1,1);
INSERT INTO R VALUES (1,2);
INSERT INTO R VALUES (3,3);
SELECT * FROM R R1, R R2 WHERE R1.a=R2.a;
```

☐

1	1
1	2
3	3

☐

1	1
1	1
1	2
1	2
3	3
3	3

☐

1	1	1	1
1	1	1	2
1	1	3	3
1	2	1	1
1	2	1	2
1	2	3	3
3	3	1	1
3	3	1	2
3	3	3	3

☐

1	1	1	1
1	1	1	2
1	2	1	1
1	2	1	2
3	3	3	3

☐

1	1	1	2
1	1	3	3
1	2	1	1
1	2	3	3
3	3	1	1
3	3	1	2

f) Expression d'une jointure externe

Syntaxe : Jointure externe, gauche ou droite

Pour exprimer une jointure externe on se base sur la syntaxe INNER JOIN en utilisant à la place OUTER JOIN, LEFT OUTER JOIN ou RIGHT OUTER JOIN.

Exemple : Jointure externe gauche

```
1  SELECT Num
2  FROM Avion LEFT OUTER JOIN Vol
3  ON Avion.Num=Vol.Num
```

Cette requête permet de sélectionner tous les avions, y compris ceux non affectés à un vol.

Remarque

Remarquons que "Avion LEFT OUTER JOIN Vol" est équivalent à "Vol RIGHT OUTER JOIN Avion" en terme de résultat.

Intuitivement, on préfère utiliser la jointure gauche pour sélectionner tous les tuple du côté N d'une relation 1:N, même si il ne sont pas référencés ; et la jointure droite pour pour sélectionner tous les tuples d'une relation 0:N, y compris ceux qui ne font pas de référence. Cette approche revient à toujours garder à gauche de l'expression "JOIN" la relation "principale", i.e. celle dont on veut tous les tuples, même s'ils ne référencent pas (ou ne sont pas référencés par) la relation "secondaire".

Exemple

R1

#A	B	C=>R2
1	Alpha	10
2	Bravo	10
3	Charlie	20
4	**Delta**	

R2

#X	Y
10	Echo
20	Fox
30	**Golf**

R

A	B	C	X	Y
1	Alpha	10	10	Echo
2	Bravo	10	10	Echo
3	Charlie	20	20	Fox
4	**Delta**			
			30	**Golf**

```
JointureExterne(R1,R2,R1.C=R2.X)

SELECT *
FROM R1 OUTER JOIN R2
ON R1.C=R2.X
```

Tableau 42 Exemple de jointure externe (SQL et Algèbre)

Exemple

R1

#A	B	C=>R2
1	Alpha	10
2	Bravo	10
3	Charlie	20
4	**Delta**	

R2

#X	Y
10	Echo
20	Fox
30	Golf

R

A	B	C	X	Y
1	Alpha	10	10	Echo
2	Bravo	10	10	Echo
3	Charlie	20	20	Fox
4	**Delta**			

```
JointureExterneGauche(R1,R2,R1.C=R2.X)

SELECT *
FROM R1 LEFT OUTER JOIN R2
ON R1.C=R2.X
```

Tableau 43 Exemple de jointure externe gauche (SQL et Algèbre)

Exemple

R1

#A	B	C=>R2
1	Alpha	10
2	Bravo	10
3	Charlie	20
4	Delta	

R2

#X	Y
10	Echo
20	Fox
30	**Golf**

R

A	B	C	X	Y
1	Alpha	10	10	Echo
2	Bravo	10	10	Echo
3	Charlie	20	20	Fox
			30	**Golf**

```
JointureExterneDroite(R1,R2,R1.C=R2.X)

SELECT *
FROM R1 RIGHT OUTER JOIN R2
ON R1.C=R2.X
```

Tableau 44 Exemple de jointure externe droite (SQL et Algèbre)

Méthode : Trouver les enregistrements non joints

La jointure externe sert en particulier pour trouver les enregistrements d'une table qui ne sont pas référencés par une clé étrangère. Il suffit de sélectionner, après la jointure externe, tous les enregistrements pour lesquels la clé de la relation référençante est nulle, on obtient alors ceux de la relation référencée qui ne sont pas référencés.

R1

#A	B	C=>R2
1	Alpha	10
2	Bravo	10
3	Charlie	20
4	Delta	

R2

#X	Y
10	Echo
20	Fox
30	Golf

R

A	B	C	X	Y
			30	Golf

```
Restriction(
        JointureExterneDroite(R1,R2,R1.C=R2.X),
        R1.A IS NULL)

SELECT *
FROM R1 RIGHT OUTER JOIN R2
ON R1.C=R2.X
WHERE R1.A IS NULL
```

Tableau 45 Exemple de sélection d'enregistrements non référencés (SQL et Algèbre)

g) Exercice : Photos à gauche

Soit les trois relations instanciées suivantes :

Numero	Nom	Prenom
1	Jacques	Brel
2	Georges	Brassens
3	Léo	Ferré

Tableau 46 Relation Personne

Personne	Photo
1	1
2	1
3	1

Tableau 47 Relation PersonnesPhotos

Numero	Date	Lieu
1	10/12/1965	Paris
2	18/03/2002	Lille

Tableau 48 Relation Photo

Combien de tuples renvoie l'instruction SQL suivante :

```
1  SELECT *
2  FROM Photo ph LEFT JOIN (
3    PersonnesPhotos pp INNER JOIN Personne pe
4      ON pp.Personne=pe.Numero
5    ) ON ph.Numero=pp.Photo;
```

h) Opérateurs ensemblistes

Introduction

Les opérateurs ensemblistes ne peuvent être exprimés à l'aide de l'instruction de sélection seule.

Syntaxe : Union

```
1 SELECT * FROM R1
2 UNION
3 SELECT * FROM R2
```

Syntaxe : Intersection

```
1 SELECT * FROM R1
2 INTERSECT
3 SELECT * FROM R2
```

Syntaxe : Différence

```
1 SELECT * FROM R1
2 EXCEPT
3 SELECT * FROM R2
```

Remarque

Les opérations INTERSECT et EXCEPT n'existe que dans la norme SQL2, et non dans la norme SQL1. Certains SGBD sont susceptibles de ne pas les implémenter.

B. Exercices

1. Location d'appartements

[20 min]

Soit le schéma relationnel suivant gérant le fonctionnement d'une agence de location d'appartements.

```
1 APPARTEMENT(#code_appt:String, adresse:String, type:{studio,F1,F2,F3,F4,F5+},
  prix_loyer:Real)
2 LOCATAIRE(#code_loc:String, nom:String, prenom:String)
3 LOCATION(#code_loc=>Locataire, #code_appt=>Appartement)
4 PAIEMENT_LOYER(#code_loc=>Locataire, #code_appt=>Appartement,
  #date_payement:Date, prix_paye:Real)
```

Question 1

En algèbre relationnelle et en SQL afficher tous les paiements effectués par un locataire avec le code X.

Question 2

En algèbre relationnelle et en SQL afficher l'adresse de l'appartement loué par un locataire avec le code X.

Question 3

En algèbre relationnelle et en SQL proposer une requête qui affiche tous les appartements libres.

2. Employés et salaires

[20 minutes]

Soit le schéma relationnel :

```
1 Employe (#Num, Nom, Prenom, Age, Salaire, Fonction=>Fonction, Societe=>Societe)
2 Fonction (#Intitule, SalaireMin, SalaireMax, NbHeures)
3 Societe (#Nom, Pays, Activite)
```

Question 1

Écrivez une requête SQL permettant de sélectionner les noms de tous les directeurs de France.

Question 2

Écrivez une requête SQL permettant d'afficher le salaire de tous les employés en francs (sachant que le salaire dans la table est en euros et que un euro vaut 6.55957 franc).

Question 3

Écrivez une requête SQL permettant de vérifier que les salaires de chaque employé correspondent bien à ce qui est autorisé par leur fonction.

Question 4

Écrivez une requête SQL permettant le passage aux 35 heures :

- en modifiant le nombre d'heures travaillées pour chaque fonction (ramené à 35 s'il est supérieur à 35),
- et en réajustant les échelles de salaire au pro-rata du nombre d'heures travaillées avant le passage aux 35 heures.

Glossaire

Clé artificielle

Une clé artificielle est un attribut ajouté à une relation afin d'identifier ses enregistrements. On fait appel à une clé artificielle lorsque la relation ne comporte aucune clé naturelle ou que ses clés naturelles sont jugées inadaptées à l'identification au sein de la base de données.

Client

Un client est un programme informatique qui a pour fonction d'envoyer des requêtes à un autre programme informatique, appelé serveur, d'attendre le résultat de cette requête et de traiter le résultat de la requête. Notons qu'un programme peut-être client vis à vis d'un programme et serveur vis à vis d'un autre. On ne prend pas ici le terme client dans son acception matérielle, qui signifie alors un ordinateur qui a pour fonction d'héberger des programmes clients.

Extension

L'extension est l'explicitation d'un domaine par l'énonciation exhaustive de l'ensemble des objets du domaine.

Elle s'oppose à l'instension qui est une description abstraite des caractéristiques du domaine.

- Exemple : {bleu, rouge, vert}
- Contre-exemple : Le domaine des couleurs

Intension

L'intension est l'explicitation d'un domaine par la description de ses caractéristiques (en vue de sa compréhension abstraite, générale).

Elle s'oppose à l'extension qui est l'énonciation exhaustive de l'ensemble des objets du domaine.

- Exemple : Le domaine des couleurs
- Contre-exemple : {bleu, rouge, vert}

Serveur

Un serveur est un programme informatique qui a pour fonction de recevoir des requêtes d'un autre programme, appelé client, de traiter ces requêtes et de renvoyer en retour une réponse. Notons qu'un programme peut-être serveur vis à vis d'un programme et client vis à vis d'un autre. On ne prend pas ici le terme serveur dans son acception matérielle, qui signifie alors un ordinateur qui a pour fonction d'héberger des programmes serveurs.

Signification des abréviations

- ANSI	American National Standards Institute
- BD	Base de Données
- CSV	Comma Separated Values
- E-A	Entité-Association
- ISO	International Standardization Organization
- LCD	Langage de Contrôle de Données
- LDD	Langage de Définition de Données
- LMD	Langage de Manipulation de Données
- OMG	Object Management Group
- OS	Operating Système (Système d'Exploitation)
- PSM	Persistent Stored Modules
- RO	Relationnel-Objet
- SGBD	Système de Gestion de Bases de Données
- SGBDR	Système de Gestion de Bases de Données Relationnelles
- SQL	Structured Query Language
- UML	Unified Modeling Language
- XML	eXtensible Markup Language

Références

[*dbdisco.crzt.fr*] *http://dbdisco.crzt.fr*[21]

[*Putty*] Putty est un logiciel qui propose un terminal sous Windows plus confortable que le terminal par défaut. De plus c'est un client *ssh*, plus sécurisé que *telnet*. C'est un petit exécutable qui ne nécessite pas d'installation, il peut donc être téléchargé sur le disque Z et lancé directement.

http://www.chiark.greenend.org.uk/~sgtatham/putty/[22]

21 - http://dbdisco.crzt.fr
22 - http://www.chiark.greenend.org.uk/~sgtatham/putty/

Bibliographie

[Arribe, 2014] Arribe Thibaut. 2014. *Conception des chaînes éditoriales : documentariser l'activité et structurer le graphe documentaire pour améliorer la maîtrise de la rééditorialisation*. Université de Technologie de Compiègne, Mémoire de Doctorat. *http://ics.utc.fr/~tha* .

[Codd70] Codd EF, *A relational model for large shared data banks*, Communications de l'ACM, juin 1970.

[Delmal01] Delmal Pierre. *SQL2 SQL3, applications à Oracle*. De Boeck Université, 2001.

[Gardarin99] Gardarin Georges. *Bases de données : objet et relationnel*. Eyrolles, 1999.

[Gulutzan and Pelzer, 1999] Gulutzan Peter, Pelzer Trudy. 1999. *SQL-99 complete, really*. CMP books.

[Muller98] Muller P.A., *Modélisation objet avec UML*, Eyrolles, 1998.

[Roques04] Roques Pascal, Vallée Franck. *UML 2 en action : De l'analyse des besoins à la conception J2EE*. ISBN 2-212-11462-1 (3ème édition). Paris : Eyrolles, 2004. 385 p. architecte logiciel.

[Rothenberg et al., 1989] Rothenberg Jeff, Widman Lawrence E, Loparo Kenneth A, Nielsen Norman R. 1989. *The nature of modeling*. Rand. vol.3027.

[Soutou02] Soutou Christian. *De UML à SQL : Conception de bases de données*. Eyrolles, 2002.

[Tardieu83] Tardieu H., Rochfeld A., Colleti R., *Méthode MERISE Tome 1 : Principes et outils*, Les Editions d'Organisation, 1983.

Webographie

[w_journaldunet.com(1)] MORLON JÉRÔME, *UML en 5 étapes*, *http://developpeur.journaldunet.com/dossiers/alg_uml.shtml* , 2004.

[w_journaldunet.com(2)] BORDERIE XAVIER, *Cinq petits conseils pour un schéma UML efficace*, *http://developpeur.journaldunet.com/tutoriel/cpt/031013cpt_uml5conseils.shtml* , 2004.

[w_objecteering] *Objecteering software. http://www.objecteering.com* . [2002-septembre].

[w_uml.free.fr] *UML en Français*, *http://uml.free.fr* , consulté en 2002.

Index