

Présentation du module: Modélisation de donnée

Outils

- StarUML / Analyse SI / Draw.io ou autre outil : <https://launchpad.net/analysesi/> / <https://staruml.io/> / <https://app.diagrams.net/>
- MySQL : <https://www.mysql.com/fr/downloads/>

•

MySQL Installer 8.0.35

Note: MySQL 8.0 is the final series with MySQL Installer. As of MySQL 8.1, use a MySQL product's MSI or Zip archive for installation. MySQL Server 8.1 and higher also bundle MySQL Configurator, a tool that helps configure MySQL Server.

Select Version:
8.0.35

Select Operating System:
Microsoft Windows

Windows (x86, 32-bit), MSI Installer (mysql-installer-web-community-8.0.35.0.msi)	8.0.35	2.1M	Download
MD5: 214df2ccd83eb5edc6ca7c115792406 Signature			
Windows (x86, 32-bit), MSI Installer (mysql-installer-community-8.0.35.0.msi)	8.0.35	288.6M	Download
MD5: 2cfd4a48a2971b6b5323775ef9e8d012 Signature			

Note: We suggest that you use the [MD5 checksums](#) and [GnuPG signatures](#) to verify the integrity of the packages you download.

<https://www.mysql.com/fr/downloads/>

MySQL Installer 8.0.35



Note: MySQL 8.0 is the final series with MySQL Installer. As of MySQL 8.1, use a MySQL product's MSI or Zip archive for installation. MySQL Server 8.1 and higher also bundle MySQL Configurator, a tool that helps configure MySQL Server.

Select Version:

8.0.35

Select Operating System:

Microsoft Windows

Windows (x86, 32-bit), MSI Installer

8.0.35

2.1M

[Download](#)

(mysql-installer-web-community-8.0.35.0.msi)

MD5: 214df2ccd83eb5edc6ca7c115792406 | [Signature](#)

Windows (x86, 32-bit), MSI Installer

8.0.35

288.6M

[Download](#)

(mysql-installer-community-8.0.35.0.msi)

MD5: 2cfda448a2971b6b5323775ef9e8d012 | [Signature](#)



We suggest that you use the [MD5 checksums](#) and [GnuPG signatures](#) to verify the integrity of the packages you download.

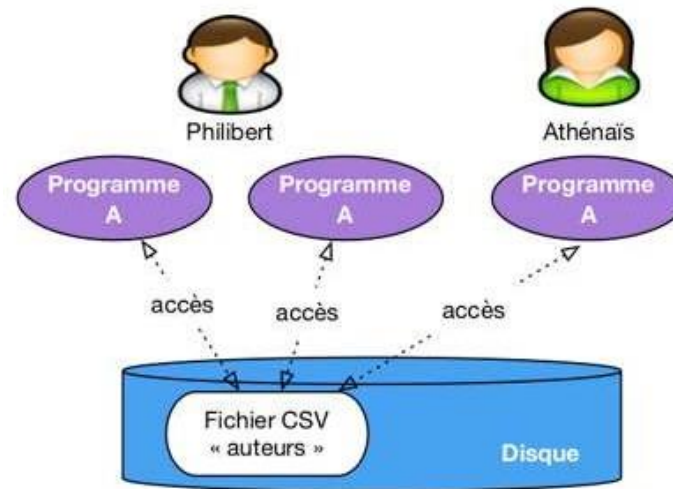
Base de données

Introduction

Qu'est ce qu'une BDD ?

Une base de données est un ensemble d'informations structurées mémorisées sur un support *persistant*.

Problèmes liés aux fichiers ?

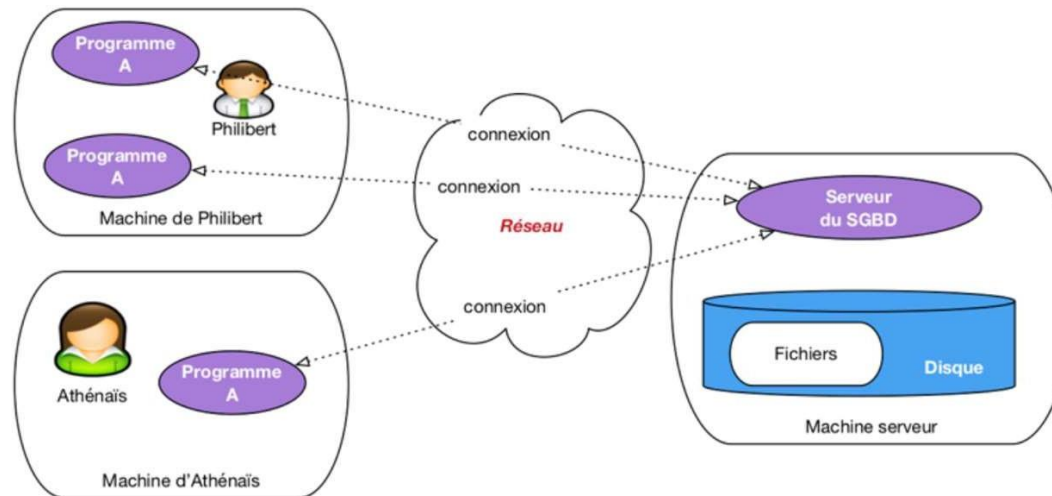


Pourquoi BDD Vs accès direct aux Fichiers?

- Comment être plusieurs à accéder aux mêmes données ?
- Comment stocker des dizaines de milliers d'informations ?
- Comment accéder simplement et rapidement à ces mêmes informations ?
Et comment faire des statistiques dessus ? *Par exemple, obtenir la liste de tous les clients que l'on a facturés cette année.*
- Comment faire des relations entre des « objets informatiques » ?
- Comment sauvegarder les données informatiques même si des personnes travaillent dessus ?
- Comment ne pas enregistrer 2 fois la même information ?
- Comment définir des droits aux utilisateurs ? *Par exemple dire que Julien a le droit de créer un contact mais n'a pas le droit de modifier l'entreprise cliente dont dépend le contact.*

Qu'est qu'un SGBD ?

Système de Gestion de Base de Données



Qu'est qu'un SGBD ?

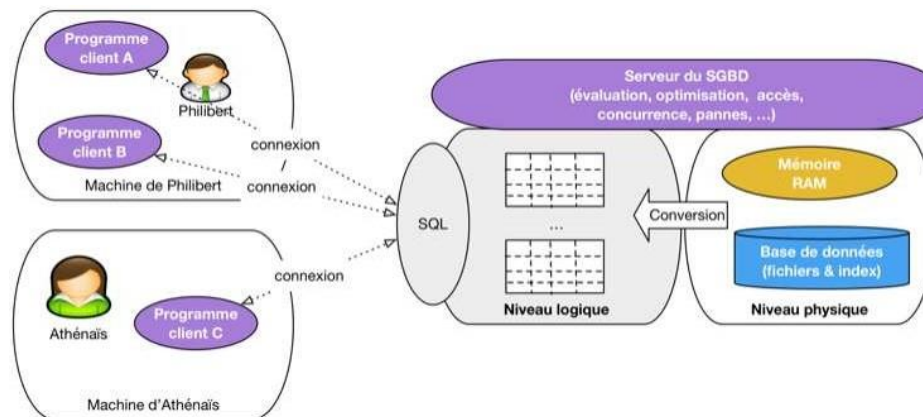
- La gestion et l'accès à une base de données sont assurés par un ensemble de programmes qui constituent le Système de gestion de base de données (SGBD). Un SGBD doit permettre l'ajout, la suppression, la modification et la recherche de données.
- Un système de gestion de bases de données héberge généralement plusieurs bases de données, qui sont destinées à des objectifs différents.
- Actuellement, la plupart des SGBD fonctionnent selon un mode client/serveur. Le serveur (sous-entendu la machine qui stocke les données) reçoit des requêtes de plusieurs clients et ceci de manière concurrente. Le serveur analyse la requête, la traite et retourne le résultat au client.

Les enjeux des SGBD

- **Indépendance logique :**
 - Un même ensemble de données peut être vu différemment par des utilisateurs différents. Toutes ces visions personnelles des données doivent être intégrées dans une vision globale.
- **Accès aux données :**
 - L'accès aux données se fait par l'intermédiaire d'un Langage de Manipulation de Données (**LMD**). Il est crucial que ce langage permette d'obtenir des réponses aux requêtes en un temps « raisonnable ». Le LMD doit donc être optimisé, minimiser le nombre d'accès disques, et tout cela de façon totalement transparente pour l'utilisateur.
- **Administration centralisée des données (intégration) :**
 - Toutes les données doivent être centralisées dans un réservoir unique commun à toutes les applications.
- **Non-redondance des données :**
 - Afin d'éviter les problèmes lors des mises à jour, chaque donnée ne doit être présente qu'une seule fois dans la base.
- **Cohérence des données :**
 - Les données sont soumises à un certain nombre de **contraintes d'intégrité** qui définissent un état cohérent de la base. Elles doivent pouvoir être exprimées simplement et vérifiées automatiquement à chaque insertion, modification ou suppression des données. Les contraintes d'intégrité sont décrites dans le Langage de Description de Données (**LDD**).
- **Partage des données :**
 - Il s'agit de permettre à plusieurs utilisateurs d'accéder aux mêmes données au même moment de manière transparente. Si ce problème est simple à résoudre quand il s'agit uniquement d'interrogations, cela ne l'est plus quand il s'agit de modifications dans un contexte multi-utilisateurs, car il faut : permettre à deux (ou plus) utilisateurs de modifier la même donnée « en même temps » et assurer un résultat d'interrogation cohérent pour un utilisateur consultant une table pendant qu'un autre la modifie.

Les enjeux des SGBD

- **Sécurité des données :**
 - Les données doivent pouvoir être protégées contre les accès non autorisés. Pour cela, il faut pouvoir associer à chaque utilisateur des droits d'accès aux données.
- **Résistance aux pannes :**
 - Que se passe-t-il si une panne survient au milieu d'une modification, si certains fichiers contenant les données deviennent illisibles ? Il faut pouvoir récupérer une base dans un état « sain ». Ainsi, après une panne intervenant au milieu d'une modification deux solutions sont possibles : soit récupérer les données dans l'état dans lequel elles étaient avant la modification, soit terminer l'opération interrompue.



Histoire

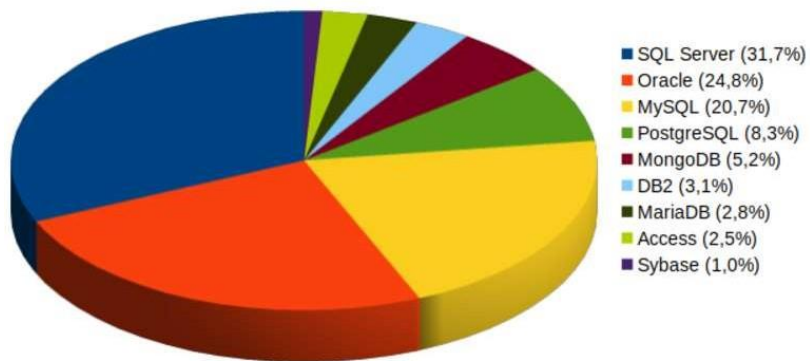
1. Modèle hiérarchique : Limite la description du monde réel
2. Modèle réseau : Résout les relations multiples mais les programmes de requêtes sont dépendants de l'implémentation
3. Modèle relationnel
 - *Le père des bases de données relationnelles est Edgar Frank Codd. Chercheur chez IBM à la fin des années 1960, mathématicien de formation, il travaille sur la théorie des ensembles et la logique des prédicats du premier ordre pour résoudre des difficultés telles que la redondance des données, l'intégrité des données ou l'indépendance de la structure de la base de données avec sa mise en œuvre physique.*
 - *En 1970, il publie un article où il propose de stocker des données hétérogènes dans des tables, permettant d'établir des relations entre elles.*
 - *Un premier prototype de Système de gestion de bases de données relationnelles (SGBDR) a été construit dans les laboratoires d'IBM. Depuis les années 80, cette technologie a mûri et a été adoptée par l'industrie.*
 - *En 1987, le langage SQL, qui étend l'algèbre relationnelle, a été standardisé.*

Aujourd'hui

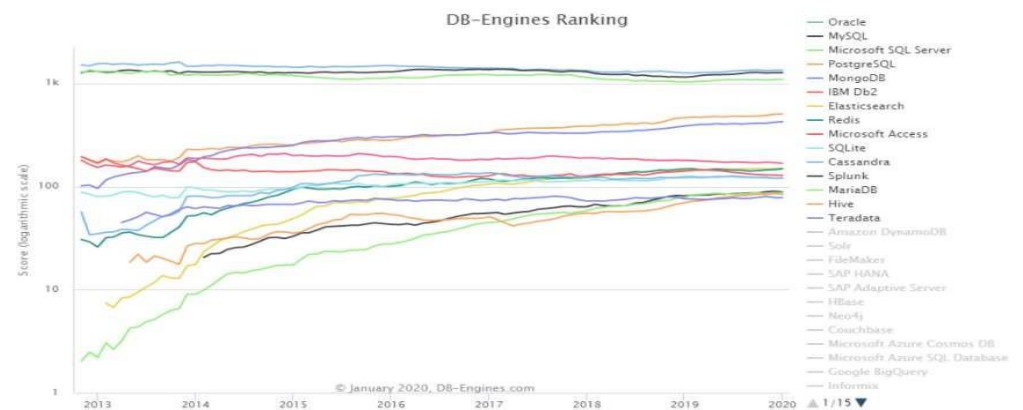
- SQL : relationnelle
 - Exemples
 - Oracle : Oracle 12c, MySQL
 - Microsoft : Access, Sql Server
 - MariaDB
 - PostgreSql (orienté objet)
- noSQL : orienté document depuis 2012
 - Exemples
 - MongoDB
 - Cassandra
 - Elasticsearch

Le marché et les SGBDs

Popularité des SGBD dans les offres d'emploi en 2019



Évolution de la demande dans les offres d'emplois des SGBD



<https://www.digora.com/fr/blog/top-10-des-bases-de-donnees-en-2020-par-popularite>

Quelques exemples ...

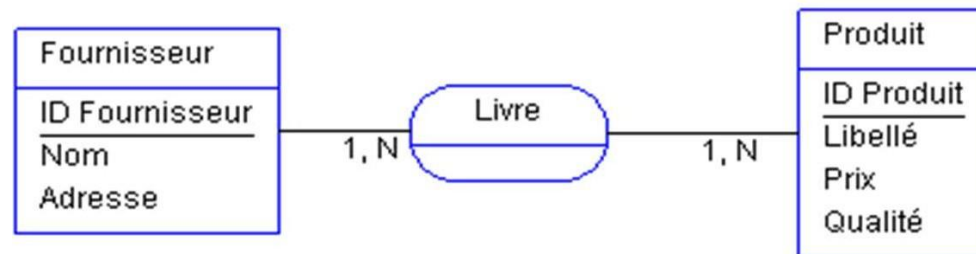
- Flickr : Partage de photos en ligne
 - Des centaines de milliers d'utilisateurs et des milliards de photos rangées proche des profils
 - 4 milliards de requêtes et 100000 nouvelles photos par jour
 - En 2007, 50 bases de données MySQL réparties sur une centaines de serveurs pour gérer les profils
- Spotify : Partage de musiques
 - Stocke plus de 2 milliards de liste de lecture.
 - Enregistre toutes les lectures / utilisateurs -> Permet en fin d'année d'analyser les données pour éditer la review
 - En 2016 c'est plus de 42 Po de données (1 Po = 1 000 000 de Go)
- Amazon : Pour attirer les entreprises vers les bases de données d'AWS, Amazon migre 7 500 bases Oracle utilisées par ses activités grand public vers ses services cloud, de DynamoDB à RedShift en passant par Aurora et RDS. (2019)

De la conception à la base de données

3 Niveaux de modèle de données

- **Le Modèle conceptuel de donnée (MCD)**

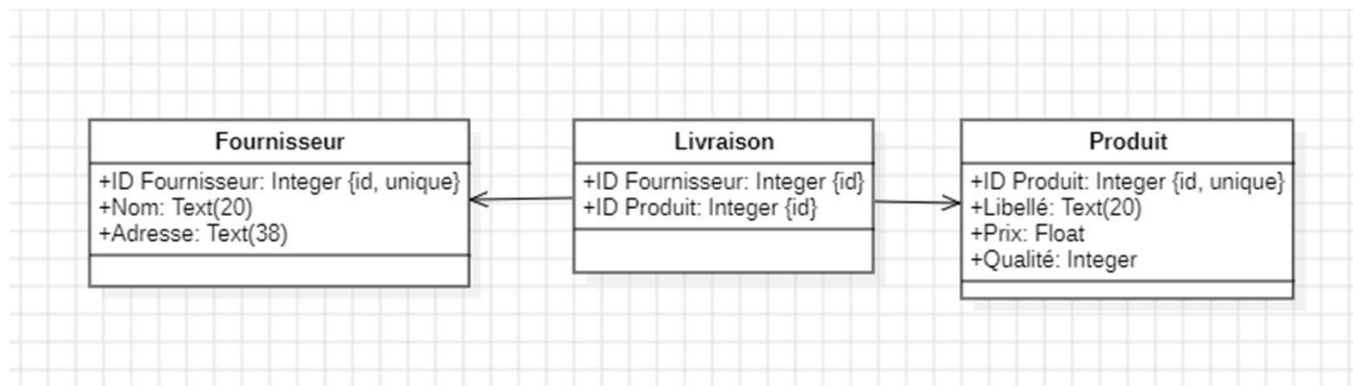
- Le modèle conceptuel de donnée présente une vue d'ensemble du projet. C'est un modèle qui doit rester compréhensible de tous et indépendant du système de gestion de base de données utilisée. L'information est définie à travers des entités, associations et cardinalités.



3 Niveaux de description de données

- **Le Modèle logique de donnée (MLD)**

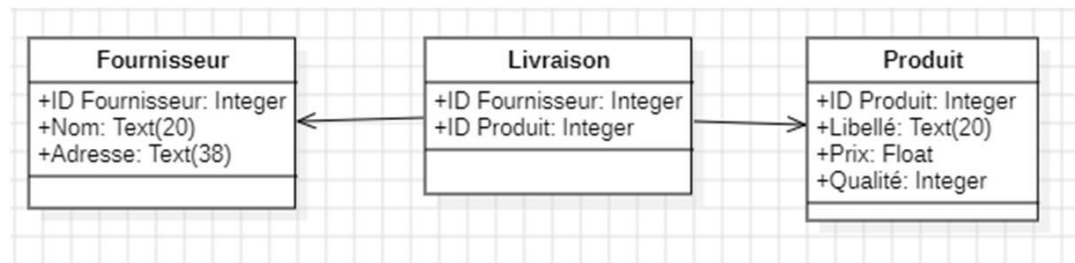
- Le modèle logique de donnée est un modèle plus technique que le précédent. Il utilise comme base le MCD mais y introduit plus de détails. L'information y est représentée à travers des tables, des relations et attributs. Les données y sont structurées et les contraintes définies.



3 Niveaux de description de données

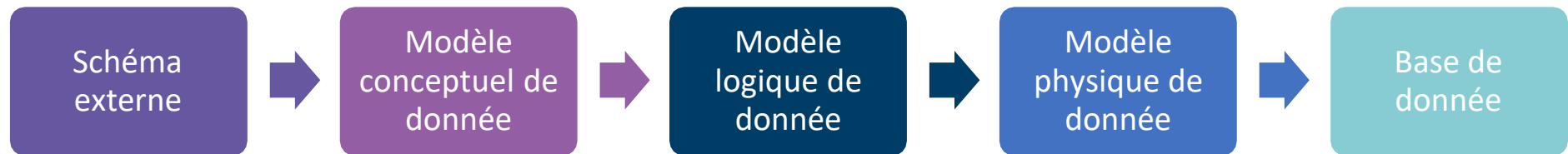
- **Le Modèle physique de donnée (MPD)**

- Le modèle physique de donnée est propre à un SGBD. Il dépend de la politique de stockage et de placement des données, de l'organisation physique des fichiers et méthodes et restrictions d'accès défini. Il permet de créer les scripts de création de la base de données.



Dans notre exemple, il n'y a pas de système de gestion existant, ni de contraintes d'accès spécifique, le MPD ne sera donc pas différent du MLD

De la conception à la base de données



Pourquoi modéliser ?

- Représenter les différents éléments constitutifs du système d'information
- Décrire des entités et leur dépendances
- La modélisation est un processus important et indispensable car elle conditionne la structure de la base de données. La structure sera déduite des différents éléments du schéma conceptuel
- Indépendant de la réalisation

UML / E/A

- Peu de différences si on veut modéliser la structure d'une base de données.

E/A historiquement pour BDD

/ UML pour POO

Entité pour EA

/ Classe pour UML

Cardinalités indiquées de façon opposée

Notion de clé primaire et étrangère non intégrées dans UML donc création d'un identifiant systématique

Notion de Visibilité, de Méthodes et Héritage UML inutiles pour modéliser une BDD

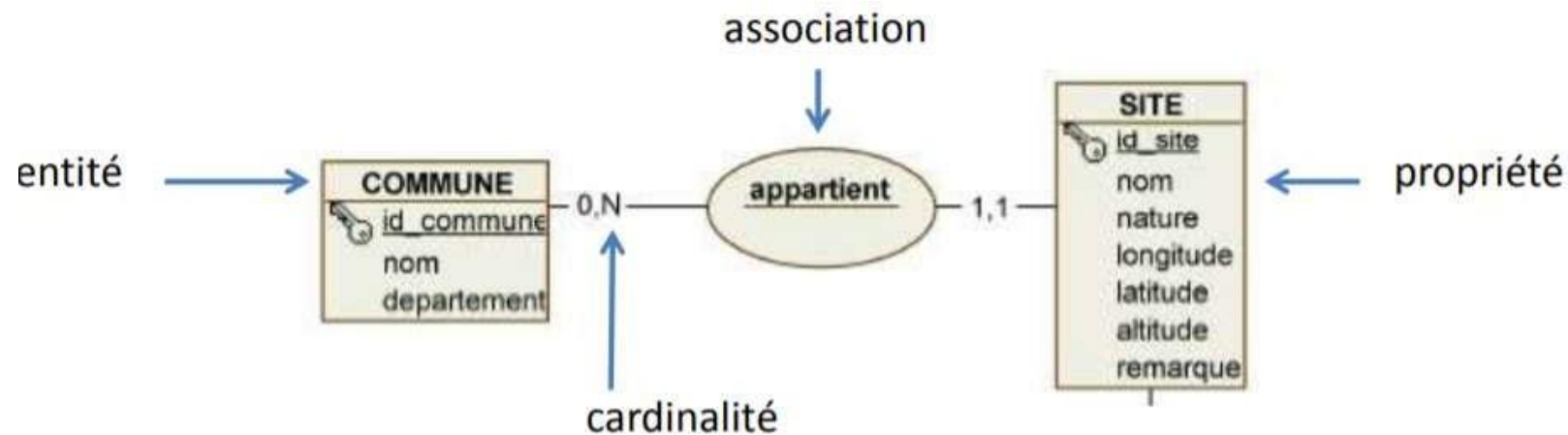
**Merise, Entite /
Association,
MCD**

Modèle Conceptuel : Entité Association



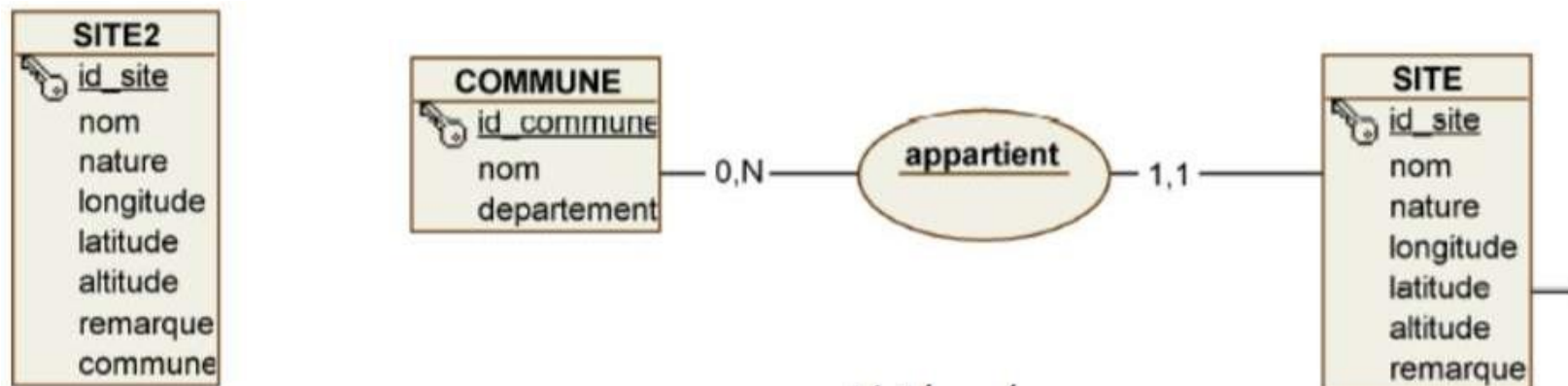
- S'inscrit MERISE (Méthode d'Étude et de Réalisation Informatique pour les Systèmes d'Entreprise)
- Introduit par Peter Chen en 1976 afin d'offrir un outil de conception

Représentation graphique et structurée des informations mémorisées dans un système d'information



Modèle Conceptuel : Entité

- Ensemble d'objets de même nature, concrets ou abstraits défini dans le cahier des charges
- Choix du concepteur en fonction de l'intérêt que présente cette entité dans son système d'information

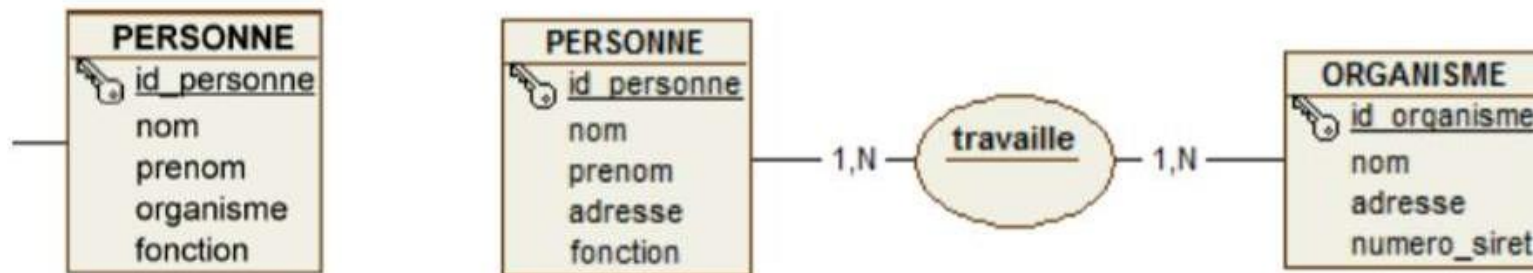


Modèle Conceptuel : Identifiant

- **Chaque entité doit être doté d'un identifiant (souligné dans la représentation)**
 - Propriété simple : nom
 - Propriété composée : nom + prénom
 - Propriété artificielle : id_personne
- **Avantages / inconvénients**
 - Une propriété artificielle est toujours unique vue du modèle mais elle ne garantit pas que la personne soit unique
 - Une propriété composée nécessite que chaque composant soit défini (c'est-à-dire non nulle)
- **Remarque : un identifiant doit être stable**

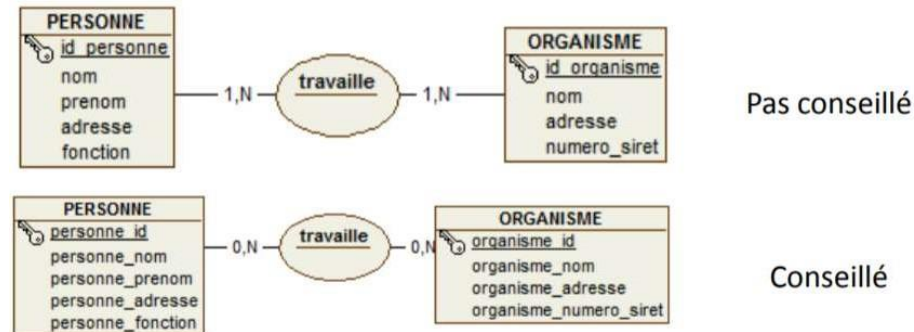
Modèle Conceptuel : Propriété

- Une entité est décrite par une liste de propriétés
- A toute occurrence de l'entité type, il ne peut y avoir, dans la mémoire du système d'information, au plus qu'une valeur de la propriété
- Ex : une personne travaille dans deux organismes



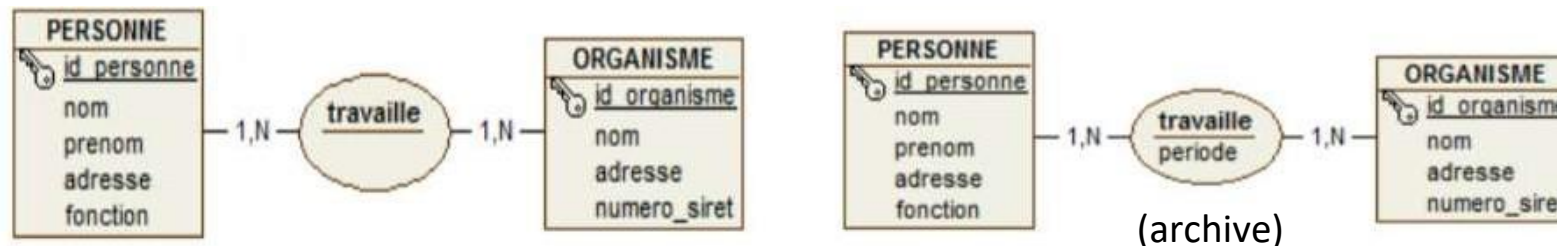
Modèle Conceptuel : Propriété

- Chaque propriété doit figurer une seule fois sur le modèle conceptuel (conseil)
- Les propriétés ne sont pas dérivées càd calculées
- Les mots réservés sont à proscrire



Modèle Conceptuel : Associations

- Liaison entre deux entités qui a une signification propre au système d'information
- Traduit une partie des règles de gestion qui n'ont pas été satisfaites par la simple définition des entités
- Certaines associations peuvent être porteuses de propriété



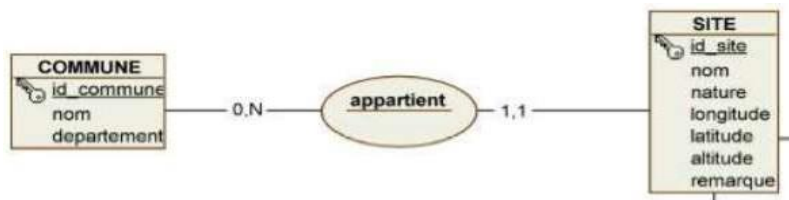
Modèle Conceptuel : Cardinalité

- Participation des occurrences d'une entité type aux occurrences d'une relation type
- Cardinalité minimum et cardinalité maximum
- Cardinalités les plus répandus : 0,n ; 1,n ; 0,1 ; 1,1
 - 0 exprime la participation optionnelle
 - 1 exprime la participation obligatoire
 - n exprime la multiplicité de participation

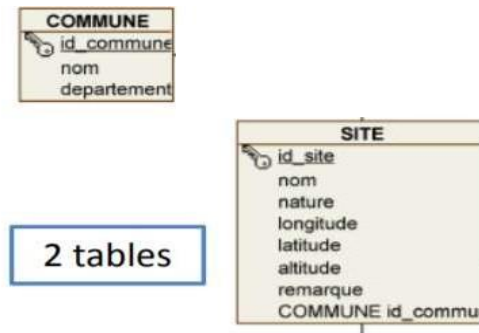


- 1 1 opération est sous la responsabilité d'1 ou plusieurs personnes
- 2 1 personne a la responsabilité d'1 ou plusieurs opérations

Modèle Conceptuel : Cardinalité



1 site est situé sur 1 seule commune
1 commune comprend 0 ou n sites



2 tables



1 site est situé sur 1 ou + de communes
1 commune comprend 0 ou n sites



3 tables

← analyse →

← implementation →

Modèle Conceptuel : 3èmes Formes normales

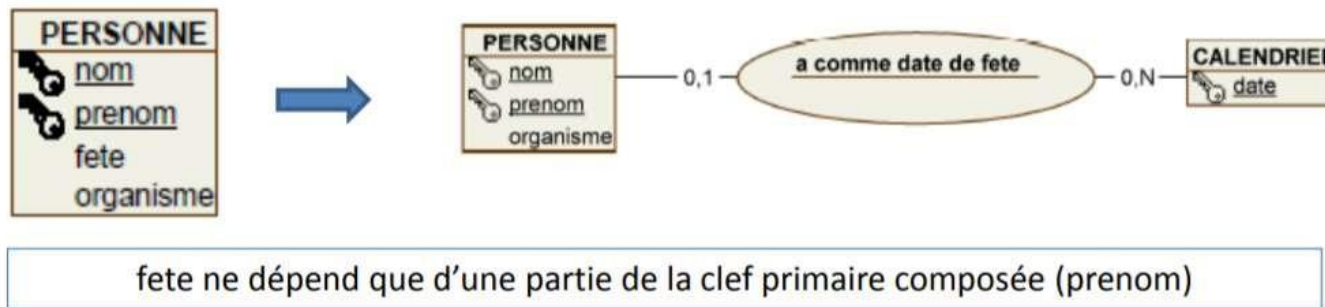
- Première forme normale

A un instant donné dans une entité, pour un individu, un attribut ne peut prendre qu'une valeur et non pas, un ensemble ou une liste de valeurs. Tout attribut est atomique

Si un attribut prend plusieurs valeurs, alors ces valeurs doivent faire l'objet d'une entité supplémentaire, en association avec la première.

- Deuxième forme normale

L'identifiant peut être composé de plusieurs attributs mais les autres attributs de l'entité doivent dépendre de l'identifiant en entier (et non pas une partie de cet identifiant).

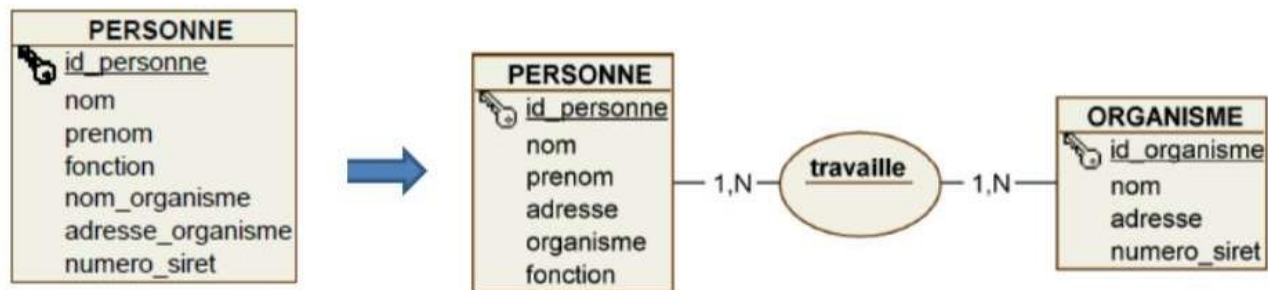


Modèle Conceptuel : 3èmes Formes normales

- Troisième forme normale

Tous les attributs d'une entité doivent dépendre directement de son identifiant et d'aucun autre attribut.

Si ce n'est pas le cas, il faut placer l'attribut pathologique dans une entité séparée, mais en association avec la première.



adresse_organisme et numero_siret ne dépendent pas directement de id_personne

Modèle Conceptuel : Compléments

- Entités faibles
 - Une entité ne pouvant exister qu'en association avec une autre entité et identifiée relativement à cette autre entité
 - ex : les salles d'un cinéma
 - Sa clé est composée de son identifiant propre (n°salle) et de l'identifiant de l'entité dont elle dépend (n°cinéma)
 - Elles sont identifiées sur les diagrammes E/A par un contour tracé avec un double trait
- Favoriser les relations binaires aux relations n-aires

.

Du MCD au Schéma Relationnel

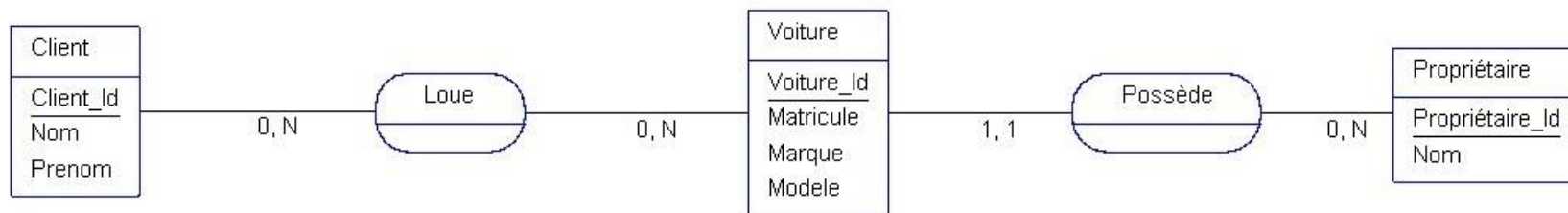
Cahier des charges

Supposons le cahier des charges suivant :

Un client (caractérisé par un nom, un prénom et une adresse) identifié par un numéro peut louer autant de voitures qu'il souhaite à des particuliers.

Chaque voiture est identifiée par sa plaque d'immatriculation, sa marque et un modèle.

Chaque voiture appartient à un propriétaire et un seul



Quel serait l'inconvénient de n'utiliser qu'une seule table ?

idc	nomC	prenomC	adresseC	idV	marque	modele	idP	nomP	adresseP
1	Durand	Karim	Paris	1	Renault	Clio	2	Dupond	Paris
1	Durand	Karim	Paris	2	Peugeot	208	2	Dupond	Paris
1	Durand	Karim	Paris	3	Citroën	DS	2	Dupond	Paris

idC	nomC	prenomC	adresseC
1	Durand	Karim	Paris
2	Fournier	Alain	Versailles
3	Oursel	Apolline	Vincennes

Clients

idC	idV	Date
1	1	10/10/2020
1	2	11/10/2020

Loue

idV	idP	marque	modele
1	1	Renault	Clio
2	1	Citroën	DS

Voiture

idP	nomP	adresseP
1	Comard	Rouen
2	Dupond	Paris

Propriétaire

Modèle relationnel : Domaine

- Le **domaine** d'une colonne ou attribut est un ensemble de valeurs

Exemples :

- Entiers
- Chaines de caractères
- Couleurs = {blanc, bleu, rouge}

Contrainte de domaine impose qu'une colonne d'une relation comporte des valeurs vérifiant une assertion unique (appartenant à une plage de valeur ou à une liste de valeurs)

Modèle relationnel : Relation

Produit cartésien

Le produit cartésien $D_1 \times D_2 \times \dots \times D_n$ des domaines $D_1, D_2, \dots, D_n$ est l'ensemble des N-uplets $\langle V_1, V_2, \dots, V_n \rangle$ tels que V_i appartient à D_i où l'ensemble des tuples possibles

Exemple

$D_1 = \text{COULEUR} \{\text{blanc, bleu, rouge}\}$

$D_2 = \text{BOOLEEN} \{\text{vrai, faux}\}$

$D_1 \times D_2 = \{\langle \text{blanc, vrai} \rangle, \langle \text{bleu, vrai} \rangle, \langle \text{rouge, vrai} \rangle, \langle \text{blanc, faux} \rangle, \langle \text{bleu, faux} \rangle, \langle \text{rouge, faux} \rangle\}$

Modèle relationnel : Relation

- Une relation est un sous-ensemble du produit cartésien d'une liste de domaines
- Une relation est caractérisée par un nom

Exemple

D1 = COULEUR {blanc, bleu, rouge}

D2 = BOOLEENS {vrai, faux}

Relation : R1 = Coulvins {<bleu,faux>,<blanc,vrai>,<rouge,vrai>}

CoulVins	Coul	Choix
	Bleu	Faux
	Blanc	Vrai
	Rouge	Vrai

Modèle relationnel : Clé primaire

- **Définition :**

Attribut ou groupe d'attributs qui détermine un tuple de manière unique dans une table (domaine ou relation)

- **Contraintes :**

- Contrainte d'entité : toute relation doit posséder au moins une clé primaire et tout attribut participant à cette clé doit être non nul.
- Choisir de préférence des clés primaires numériques artificielles.
- Unicité de clé : la valeur de la clé de chaque tuple de la relation est unique

Modèle relationnel : Clef étrangère

- Attribut ou groupe d'attributs qui apparaît comme clé primaire dans une autre relation
- Elle définit une **contrainte d'intégrité référentielle**
- Lors d'une insertion, la valeur des attributs doit exister dans la relation référencée
- Lors d'une suppression dans la relation référencée, les tuples référençant doivent disparaître (ou non...) notion de cascade

Modèle relationnel : Schéma relationnel

- 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
- Exemple d'un schéma relationnel qui représente l'ensemble des schémas des relations

Client (idC:entier, nomC:chaîne, prenomC:chaîne, adresseC:chaîne)

Voiture (idV:entier, marque:chaîne, modele:chaîne, #idP:entier)

Proprietaire (idP:entier, nomP:chaîne, adresseP:chaîne)

Loue (#idC:entier, #idV, dateLoc:Date)

Etude de cas

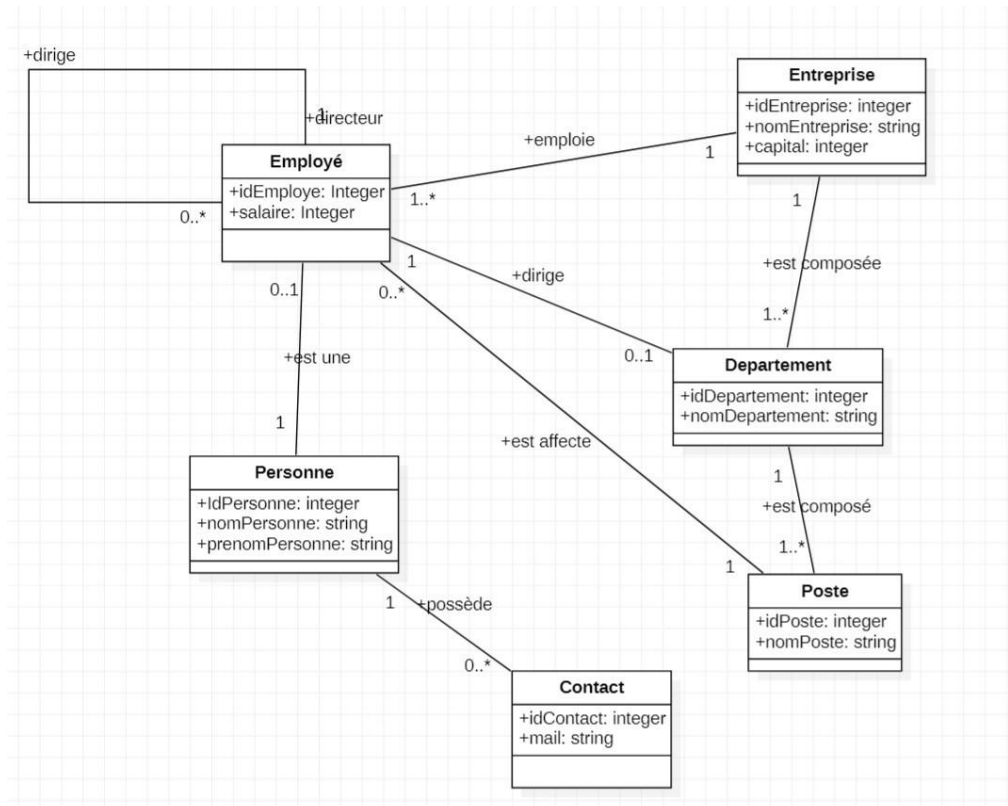
Etude de cas

Une entreprise souhaite mettre en place un Système d'Information pour gérer ses employés dans des départements en respectant les contraintes suivantes :

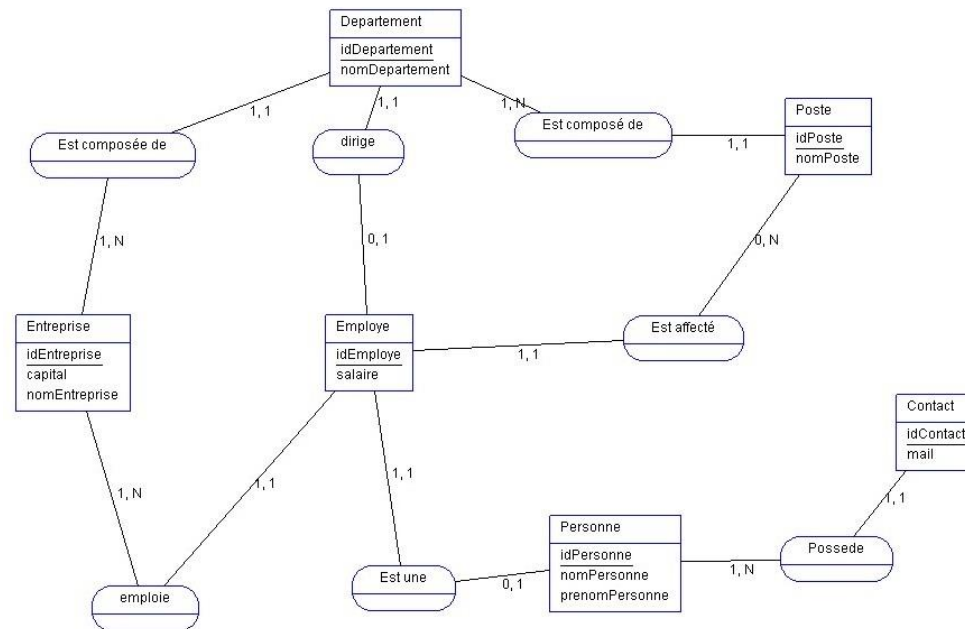
- **L'entreprise** a un **nom**, dispose d'un **capital** et **est constituée** de **départements**.
- Un **département** a un **nom** et **est composé** de plusieurs **postes** dont on connaît le **nom**
- L'entreprise **dispose** d'**employés** qui ont un **salaire**
- Un **employé** est une **personne** dont on connaît le **nom** et le **prénom**
- On **doit disposer** de **contacts** (**mail**, **téléphone** ...) pour chaque **personne** de l'**entreprise**
- Une **personne** **peut avoir** **plusieurs contacts**
- Un **employé** est affecté à un **poste**.
- Un **employé** **peut diriger** plusieurs autres **employés**
- Un **employé** **peut être responsable** d'un **département**

Etude de cas Modèle Conceptuel de Données MCD

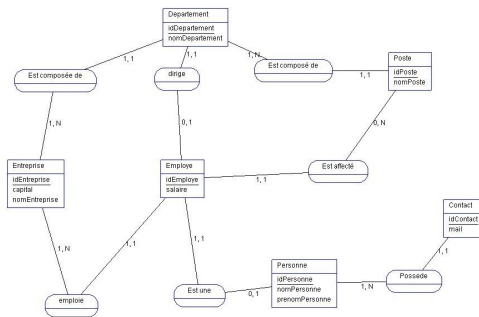
UML avec StarUML



Etude de cas : Modèle Conceptuel de Données MCD E/A avec AnalyseSI



Etude de cas : Modèle Logique de Données Relationnel MLDR



Entreprise (idEntreprise, nomEntreprise, capital)

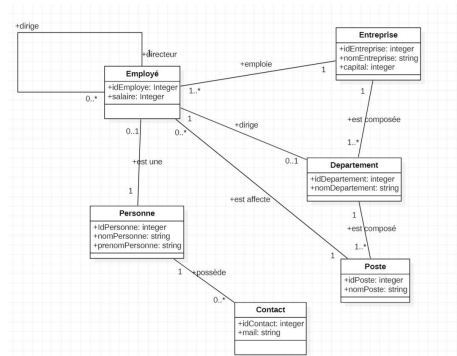
Poste(idPoste, nomPoste, #IdDepartement)

Departement(idDepartement, nomDepartement, #idEntreprise, #idEmploye)

Employe(idEmploye, salaire, #idPersonne, #idDepartement, #idPoste, #idEntreprise)

Personne(idPersonne, nomPersonne, prenomPersonne, #idEmploye)

Contact(idContact, mail, #idPersonne)



Etude de cas : Du modèle conceptuel au schéma relationnel



Client (id, nom, prenom)

Produit (idp, marque)

Achete (id, idp)

```
DROP TABLE IF EXISTS Client ;
CREATE TABLE Client (id INT AUTO_INCREMENT NOT NULL,
nom VARCHAR,
prenom VARCHAR,
PRIMARY KEY (id)) ENGINE=InnoDB;
```

```
DROP TABLE IF EXISTS Produit ;
CREATE TABLE Produit (idp INT AUTO_INCREMENT NOT NULL,
marque VARCHAR,
PRIMARY KEY (idp)) ENGINE=InnoDB;
```

```
DROP TABLE IF EXISTS Achete ;
CREATE TABLE Achete (id INT AUTO_INCREMENT NOT NULL,
idp INT NOT NULL,
PRIMARY KEY (id,
idp)) ENGINE=InnoDB;
```

```
ALTER TABLE Achete ADD CONSTRAINT FK_Achete_id FOREIGN KEY (id) REFERENCES Client (id);
```

```
ALTER TABLE Achete ADD CONSTRAINT FK_Achete_idp FOREIGN KEY (idp) REFERENCES Produit (idp);
```

Bibliographie

<http://sql.bdpedia.fr/conception.html>

<http://www-inf.int-evry.fr/COURS/BD/accueil-ext.html>

<https://stph.scenari-community.org/bdd/0/co/earUL040.html>

http://ressources.unisciel.fr/sillages/informatique/bdd/ch01_presentation/co/introduction_1.html

http://formations.imt-atlantique.fr/bd_ihm/fr/

<http://www-inf.it-sudparis.eu/COURS/bd/index.php?idr=10>