

Introduction aux Bases de Données (BDD)

1. Qu'est-ce qu'une Base de Données (BDD) ?

Une base de données (BDD) est un système organisé de collecte, de stockage et de gestion d'informations. Elle permet de structurer les données pour qu'elles soient facilement accessibles, manipulables et sécurisées.

Exemple :

- Une base de données relationnelle comme MySQL peut contenir une table "Clients" avec des colonnes telles que **Nom** , **Prénom** , **Email** , et **Adresse** .
- Une base non relationnelle, comme MongoDB, stocke les mêmes données sous forme de documents JSON.

Les bases relationnelles (MySQL, PostgreSQL) utilisent un modèle structuré avec des tables, tandis que les bases non relationnelles (MongoDB, Cassandra) sont plus flexibles, adaptées aux données non structurées comme les réseaux sociaux.

2. Pourquoi avons-nous besoin des Bases de Données ?

Les bases de données jouent un rôle central dans la gestion des informations, et leur utilité s'articule autour de plusieurs points :

- **Centralisation des données** : Stocker les données au même endroit évite la duplication.
Exemple : Une entreprise utilise une BDD pour centraliser les informations de ses clients et éviter d'avoir des fichiers Excel éparpillés.
- **Accès rapide et efficace** : Un SGBD permet d'effectuer des recherches et des traitements rapides.
Exemple : Dans une boutique en ligne, la recherche d'un produit spécifique dans une base contenant des milliers d'articles est instantanée grâce à un index.
- **Intégrité et cohérence** : Les règles de la base garantissent que les données restent exactes.
Exemple : Dans une BDD, il est impossible d'enregistrer une commande sans associer un client existant grâce à une contrainte de clé étrangère.
- **Sécurité** : Les accès aux données sont contrôlés via des systèmes d'authentification et des permissions.
Exemple : Un administrateur peut limiter l'accès des employés au strict nécessaire, protégeant ainsi les informations sensibles.

- **Partage des données** : Plusieurs utilisateurs ou applications peuvent accéder simultanément aux informations.
Exemple : Dans un CRM, les équipes de vente et de support client accèdent en temps réel aux mêmes données client.

3. Principe des Systèmes de Gestion de Bases de Données Relationnelles (SGBDR)

Un SGBDR (Système de Gestion de Bases de Données Relationnelles) organise les données sous forme de tables (relations). Chaque table est composée :

- de **colonnes** (attributs) représentant les propriétés des données,
- et de **lignes** (enregistrements) représentant les données elles-mêmes.

Les tables sont reliées par des **clés primaires** et **clés étrangères**.

Exemple :

Deux tables dans une base de données :

1. **Clients**

ID_Client	Nom	Email
1	Martin	martin@test.com
2	Dubois	dubois@test.com

2. **Commandes**

ID_Commande	Date	ID_Client
101	2025-01-01	1
102	2025-01-02	2

Dans cet exemple, la colonne `ID_Client` dans la table **Commandes** est une clé étrangère qui fait référence à la colonne `ID_Client` dans **Clients**, établissant ainsi une relation entre les deux.

Les SGBDR utilisent le **SQL (Structured Query Language)** pour gérer les données.

Exemple de requête SQL :

- Trouver toutes les commandes passées par le client "Martin" :



```
SELECT Commandes.ID_Commande, Commandes.Date
FROM Commandes
INNER JOIN Clients ON Commandes.ID_Client = Clients.ID_Client
WHERE Clients.Nom = 'Martin';
```

4. Principe ACID des SGBDR

Le principe ACID garantit la fiabilité des transactions dans un SGBDR. Voici ses propriétés, accompagnées d'exemples concrets :

- **Atomicité** : Une transaction est tout ou rien.
Exemple : Si une banque transfère 100 € du compte A au compte B, mais qu'une panne survient avant la fin, la transaction est annulée pour que le montant reste cohérent.
- **Cohérence** : Après une transaction, la base respecte les règles définies.
Exemple : Lors de l'ajout d'une commande, si le client n'existe pas (violation de la contrainte de clé étrangère), l'opération échoue.
- **Isolation** : Les transactions parallèles n'interfèrent pas entre elles.
Exemple : Deux employés enregistrent des commandes en même temps. Les transactions sont gérées de manière à ne pas provoquer de conflits.
- **Durabilité** : Une fois validée, une transaction est enregistrée définitivement, même en cas de panne.
Exemple : Après la validation d'un paiement en ligne, le système enregistre la transaction et la commande reste accessible même si le serveur tombe en panne.

Ces principes assurent la fiabilité et la robustesse des bases de données, indispensables dans des systèmes critiques tels que les banques ou les hôpitaux.

Module 1 : Introduction à MERISE

Chapitre 1 : Introduction aux systèmes d'information

1.1 Définition et composantes d'un SI

Un Système d'Information (SI) est un ensemble organisé de ressources (matérielles, logicielles, humaines, données et procédures) permettant de collecter, stocker, traiter et diffuser de l'information au sein d'une organisation.

Composantes d'un SI :

1. **Données** : Informations brutes ou traitées.
2. **Matériel (Hardware)** : Équipements physiques (ordinateurs, serveurs, réseaux).
3. **Logiciel (Software)** : Programmes et applications.
4. **Procédures** : Règles et processus pour l'utilisation et la gestion du SI.
5. **Ressources humaines** : Utilisateurs, administrateurs, développeurs.

Exemple : Dans une bibliothèque, le SI comprend :

- Données : Liste des livres, informations sur les membres
- Matériel : Ordinateurs, scanners pour les codes-barres
- Logiciel : Système de gestion de bibliothèque
- Procédures : Processus d'emprunt et de retour de livres
- Ressources humaines : Bibliothécaires, administrateurs du système

1.2 Cycle de vie d'un système d'information

Le cycle de vie d'un SI comprend plusieurs phases :

1. **Planification** : Définition des objectifs et de la portée du projet.
2. **Analyse** : Étude des besoins et des exigences.
3. **Conception** : Élaboration de la structure et de l'architecture du système.
4. **Développement** : Création du système (programmation, configuration).
5. **Tests** : Vérification du bon fonctionnement.
6. **Déploiement** : Mise en production du système.
7. **Maintenance** : Corrections, mises à jour et améliorations continues.

1.3 Enjeux de la conception

Les principaux enjeux de la conception d'un SI sont :

1. **Alignement stratégique** : Le SI doit répondre aux objectifs de l'organisation.
2. **Efficacité opérationnelle** : Amélioration des processus et de la productivité.
3. **Sécurité et confidentialité** : Protection des données et respect des réglementations.
4. **Évolutivité** : Capacité à s'adapter aux changements futurs.
5. **Interopérabilité** : Capacité à communiquer avec d'autres systèmes.
6. **Ergonomie** : Facilité d'utilisation pour les utilisateurs finaux.

Exercice 1 : Identifiez les composantes du SI d'une banque en ligne.

Correction :

- Données : Informations des clients, transactions, soldes des comptes
- Matériel : Serveurs, ordinateurs, dispositifs de sécurité
- Logiciel : Application bancaire en ligne, système de gestion de base de données
- Procédures : Processus de vérification d'identité, règles de sécurité
- Ressources humaines : Développeurs, agents du service client, administrateurs système

Chapitre 2 : Fondamentaux de la méthode MERISE

2.1 Origines et objectifs

MERISE (Méthode d'Étude et de Réalisation Informatique pour les Systèmes d'Entreprise) est une méthode d'analyse, de conception et de gestion de projet informatique développée en France dans les années 1970.

Objectifs principaux :

1. Fournir une approche structurée pour la conception de SI.
2. Améliorer la communication entre les différents acteurs du projet.
3. Garantir la cohérence et la qualité des systèmes développés.
4. Faciliter la maintenance et l'évolution des SI.

2.2 Pourquoi modéliser un système d'information ?

La modélisation d'un SI est cruciale pour plusieurs raisons :

1. **Compréhension** : Permet de visualiser et de comprendre la structure et le fonctionnement du système.
2. **Communication** : Facilite les échanges entre les différentes parties prenantes (développeurs, utilisateurs, managers).
3. **Analyse** : Aide à identifier les problèmes potentiels et les opportunités d'amélioration.
4. **Conception** : Sert de base pour la création et l'implémentation du système.
5. **Documentation** : Fournit une référence pour la maintenance et les évolutions futures.

2.3 Les principes clés de MERISE

2.3.1 Séparation données/traitements

MERISE distingue clairement :

- **Les données** : Ce qui est manipulé (informations stockées et gérées).
- **Les traitements** : Comment les données sont manipulées (processus, actions).

Cette séparation permet :

- Une meilleure compréhension du système
- Une plus grande flexibilité dans la conception
- Une facilité de maintenance et d'évolution

Exemple : Dans un système de gestion de bibliothèque

- Données : Livres, membres, emprunts
- Traitements : Enregistrer un emprunt, retourner un livre, ajouter un nouveau membre

2.3.2 Niveaux d'abstraction

MERISE utilise différents niveaux d'abstraction pour décrire un système :

1. **Niveau conceptuel** : Vision globale et abstraite du système (quoi ?).
2. **Niveau organisationnel** : Description des processus et de l'organisation (qui, où, quand ?).
3. **Niveau logique** : Spécifications techniques indépendantes des outils (comment ?).
4. **Niveau physique** : Implémentation concrète avec des outils spécifiques.

Cette approche permet une progression du général au particulier, facilitant la compréhension et la conception du système.

Exercice 2 : Pour un système de gestion des commandes en ligne, donnez un exemple pour chaque niveau d'abstraction.

Correction :

1. Niveau conceptuel : Modèle conceptuel des données (entités : Client, Produit, Commande)
2. Niveau organisationnel : Processus de traitement d'une commande (réception, validation, préparation, expédition)
3. Niveau logique : Structure de la base de données relationnelle (tables, relations)
4. Niveau physique : Implémentation dans un SGBD spécifique (ex: MySQL) avec des scripts SQL

Chapitre 3 : Vue d'ensemble de MERISE

3.1 Les niveaux de modélisation

MERISE propose une approche structurée avec différents niveaux de modélisation :

3.1.1 Niveau conceptuel

- **Objectif** : Décrire le "QUOI" du système, indépendamment de toute contrainte technique ou organisationnelle.
- **Modèles** :
 - MCD (Modèle Conceptuel de Données)
 - MCT (Modèle Conceptuel des Traitements)

Exemple MCD simplifié pour une bibliothèque :

```
[LIVRE] (0,n) ——— EMPRUNTER ——— (0,n) [MEMBRE]
```

3.1.2 Niveau organisationnel

- **Objectif** : Décrire le "QUI, OÙ, QUAND" du système, en intégrant les contraintes de l'organisation.
- **Modèles** :
 - MOD (Modèle Organisationnel des Données)
 - MOT (Modèle Organisationnel des Traitements)

Exemple MOT simplifié pour l'emprunt d'un livre :

```
1. Membre : Demande d'emprunt
2. Bibliothécaire : Vérification disponibilité
3. Système : Enregistrement de l'emprunt
4. Bibliothécaire : Remise du livre au membre
```

3.1.3 Niveau logique

- **Objectif** : Décrire le "COMMENT" du système, indépendamment des outils techniques spécifiques.
- **Modèles** :
 - MLD (Modèle Logique de Données)

- MLT (Modèle Logique des Traitements)

Exemple MLD simplifié :

```
LIVRE (ISBN, Titre, Auteur)
MEMBRE (ID_Membre, Nom, Prénom)
EMPRUNT (ISBN#, ID_Membre#, Date_Emprunt, Date_Retour)
```

3.1.4 Niveau physique

- **Objectif** : Décrire l'implémentation concrète du système avec des outils spécifiques.
- **Modèles** :
 - MPD (Modèle Physique de Données)
 - MPT (Modèle Physique des Traitements)

Exemple MPD (extrait de code SQL) :

```
CREATE TABLE LIVRE (
  ISBN VARCHAR(13) PRIMARY KEY,
  Titre VARCHAR(100) NOT NULL,
  Auteur VARCHAR(50)
);
```

3.2 Les cycles de vie d'un système d'information

MERISE distingue deux cycles principaux :

3.2.1 Cycle d'étude

1. **Schéma directeur** : Définition des objectifs et de la stratégie.
2. **Étude préalable** : Analyse de l'existant et des besoins.
3. **Étude détaillée** : Conception détaillée du système.
4. **Étude technique** : Choix des solutions techniques.

3.2.2 Cycle de réalisation

1. **Réalisation** : Développement du système.
2. **Mise en œuvre** : Déploiement et formation des utilisateurs.

3. **Maintenance** : Corrections et évolutions du système.

Exercice 3 : Pour un projet de création d'une application de gestion de tâches, identifiez une activité pour chaque étape du cycle d'étude.

Correction :

1. Schéma directeur : Définir les objectifs (ex: améliorer la productivité de l'équipe)
2. Étude préalable : Analyser les méthodes actuelles de gestion des tâches
3. Étude détaillée : Concevoir les modèles de données et de traitements pour l'application
4. Étude technique : Choisir la plateforme de développement (ex: web, mobile)

Chapitre 4 : Étude de cas introductive

Pour illustrer les concepts de MERISE, nous allons analyser un cas simple : la gestion d'une petite bibliothèque.

4.1 Énoncé du problème

Une petite bibliothèque municipale souhaite informatiser son système de gestion. Le système doit permettre :

- L'enregistrement des livres et des membres
- La gestion des emprunts et des retours
- Le suivi des livres disponibles et empruntés

4.2 Analyse avec MERISE

4.2.1 Niveau conceptuel

MCD (Modèle Conceptuel de Données)

[LIVRE] (0,n)	———	EMPRUNTER	———	(0,n)	[MEMBRE]
ISBN		Date_Emprunt			ID_Membre
Titre		Date_Retour			Nom
Auteur					Prénom
Année_Publication					Adresse

MCT (Modèle Conceptuel des Traitements) pour l'emprunt

```
Opération : EMPRUNTER_LIVRE
Événement déclencheur : Demande d'emprunt par un membre
Résultat : Livre emprunté et enregistré
Actions :
1. Vérifier la disponibilité du livre
2. Vérifier l'éligibilité du membre
3. Enregistrer l'emprunt
4. Mettre à jour le statut du livre
```

4.2.2 Niveau organisationnel

MOT (Modèle Organisationnel des Traitements) pour l'emprunt

```
1. Membre : Présente le livre et sa carte de membre
2. Bibliothécaire : Scanne le livre et la carte
3. Système : Vérifie la disponibilité et l'éligibilité
4. Système : Enregistre l'emprunt
5. Bibliothécaire : Remet le livre au membre
```

4.2.3 Niveau logique

MLD (Modèle Logique de Données)

```
LIVRE (ISBN, Titre, Auteur, Année_Publication, Disponible)
MEMBRE (ID_Membre, Nom, Prénom, Adresse)
EMPRUNT (ID_Emprunt, ISBN#, ID_Membre#, Date_Emprunt, Date_Retour)
```

4.2.4 Niveau physique

MPD (Modèle Physique de Données) - Extrait

```
CREATE TABLE LIVRE (
  ISBN VARCHAR(13) PRIMARY KEY,
  Titre VARCHAR(100) NOT NULL,
  Auteur VARCHAR(50),
  Année_Publication INT,
  Disponible BOOLEAN DEFAULT TRUE
);

CREATE TABLE MEMBRE (
```

```
ID_Membre INT AUTO_INCREMENT PRIMARY KEY,  
Nom VARCHAR(50) NOT NULL,  
Prénom VARCHAR(50) NOT NULL,  
Adresse TEXT  
);  
  
CREATE TABLE EMPRUNT (  
ID_Emprunt INT AUTO_INCREMENT PRIMARY KEY,  
ISBN VARCHAR(13),  
ID_Membre INT,  
Date_Emprunt DATE NOT NULL,  
Date_Retour DATE,  
FOREIGN KEY (ISBN) REFERENCES LIVRE(ISBN),  
FOREIGN KEY (ID_Membre) REFERENCES MEMBRE(ID_Membre)  
);
```

4.3 Exercice final

Exercice 4 : En vous basant sur l'étude de cas de la bibliothèque, proposez un MCD pour un système de gestion d'une petite librairie. La librairie doit gérer les livres, les auteurs, les commandes clients et les fournisseurs.

Correction : Voici un MCD simplifié pour la gestion d'une petite librairie :  Module-1-4

Ce MCD représente les principales entités et relations d'une petite librairie. Il inclut :

- Les LIVRES avec leurs caractéristiques (ISBN, Titre, Prix, Stock)
- Les AUTEURS liés aux livres par la relation ÉCRIRE
- Les COMMANDES passées par les CLIENTS
- Les FOURNISSEURS qui approvisionnent les livres

Cette structure permet de gérer l'inventaire des livres, les ventes aux clients, et les approvisionnements auprès des fournisseurs.

Conclusion du Module 1

Dans ce module d'introduction à MERISE, nous avons couvert les concepts fondamentaux des systèmes d'information et de la méthode MERISE. Nous avons exploré :

Cours : Introduction à MERISE et Analyse Préliminaire des Systèmes d'Information

Module 2 : Analyse préliminaire et étude des besoins (3 heures)

1. Étude des besoins

L'étude des besoins est une étape cruciale dans tout projet de système d'information. Elle permet de comprendre les attentes des utilisateurs et les objectifs du projet.

1.1 Techniques de recueil des besoins

Plusieurs techniques peuvent être utilisées pour recueillir les besoins :

a) **Entretiens** : Discussions individuelles ou en groupe avec les parties prenantes.

- Avantages : Permet d'obtenir des informations détaillées et de clarifier les points ambigus.
- Inconvénients : Peut être chronophage et subjectif.

b) **Questionnaires** : Formulaires écrits ou en ligne pour collecter des informations.

- Avantages : Permet de toucher un grand nombre de personnes rapidement.
- Inconvénients : Les réponses peuvent manquer de profondeur.

c) **Observation directe** : Observation des utilisateurs dans leur environnement de travail.

- Avantages : Permet de comprendre les processus réels et les difficultés rencontrées.
- Inconvénients : Peut être intrusive et modifier le comportement des utilisateurs.

d) **Brainstorming ou Atelier** : Sessions de groupe pour générer des idées.

- Avantages : Favorise la créativité et l'émergence de nouvelles idées.
- Inconvénients : Peut être difficile à gérer et à synthétiser.

Exemple : Imaginons que nous devons développer un système de gestion des bibliothèques. Nous pourrions organiser des entretiens avec les bibliothécaires, distribuer des questionnaires aux utilisateurs, observer les processus actuels de gestion des livres, et organiser des ateliers pour discuter des fonctionnalités souhaitées.

1.2 Analyse de l'existant

L'analyse de l'existant consiste à étudier le système d'information actuel pour comprendre ses forces et ses faiblesses.

Étapes de l'analyse de l'existant :

1. **Inventaire des ressources** : Matériel, logiciels, données, personnel.
2. **Cartographie des processus** : Identification et description des processus actuels.
3. **Analyse des flux d'information** : Comment l'information circule dans l'organisation.
4. **Évaluation des performances** : Mesure de l'efficacité du système actuel.

5. **Identification des problèmes** : Repérage des dysfonctionnements et des points d'amélioration.

Exemple : Pour le système de gestion des bibliothèques, nous pourrions lister les systèmes actuels de gestion des prêts, décrire les processus de prêt et de retour des livres, et identifier les problèmes comme les retards dans les retours de livres.

1.3 Définition des objectifs

La définition des objectifs permet de clarifier ce que le nouveau système d'information doit accomplir.

Nous pouvons les classer en deux catégories :

Objectifs fonctionnels : Ce que le système doit faire. **Objectifs non fonctionnels** : Comment le système doit le faire (performance, sécurité, etc.).

Caractéristiques des objectifs bien définis (méthode SMART) :

- Spécifiques : Clairement définis et compréhensibles.
- Mesurables : Quantifiables pour évaluer leur réalisation.
- Atteignables : Réalistes compte tenu des ressources disponibles.
- Pertinents : En accord avec les besoins de l'organisation.
- Temporellement définis : Avec une échéance précise.

Exemple : Pour le système de gestion des bibliothèques, les objectifs fonctionnels pourraient inclure la gestion des prêts et des retours de livres, tandis que les objectifs non fonctionnels pourraient inclure la rapidité des transactions et la sécurité des données des utilisateurs.

2. Documentation initiale

La documentation initiale formalise les résultats de l'étude des besoins et sert de base pour la suite du projet.

2.1 Rédaction du cahier des charges

Le cahier des charges est un document clé qui décrit les spécifications fonctionnelles et techniques du projet.

Éléments essentiels d'un cahier des charges :

1. Contexte et objectifs du projet
2. Description fonctionnelle du système
3. Contraintes techniques ,organisationnelles , budgétaires , et temporels
4. Livrables attendus
5. Critères d'acceptation

Exemple : Pour le système de gestion des bibliothèques, le cahier des charges pourrait inclure une introduction présentant le projet, une description des besoins en termes de gestion des prêts et des retours, les contraintes budgétaires, et les critères de succès comme la réduction des retards dans les retours de livres.

2.2 Identification des acteurs et des flux

Cette étape consiste à identifier tous les intervenants du système et à comprendre comment l'information circule entre eux.

a) **Identification des acteurs :** Les Acteurs sont les personnes ou les systèmes impliqués dans le projet.

- Acteurs internes : employés, départements, etc.
- Acteurs externes : clients, fournisseurs, partenaires, etc.

b) **Analyse des flux :**

Le flux représente les échanges d'informations entre les acteurs.

- Flux d'information : données échangées entre les acteurs
- Flux physiques : mouvements de biens ou de personnes
- Flux financiers : transactions monétaires

Exemple : Pour le système de gestion des bibliothèques, les acteurs pourraient inclure les bibliothécaires, les utilisateurs, et le système de gestion des prêts. Les flux pourraient inclure les demandes de prêt, les confirmations de prêt, et les notifications de retard.

2.3 Définition du périmètre du projet

Le périmètre du projet délimite clairement ce qui est inclus et exclu du projet.

Éléments à considérer :

- Fonctionnalités à développer
- Interfaces avec d'autres systèmes
- Contraintes techniques et organisationnelles
- Limites géographiques ou organisationnelles

Exemple : Pour le système de gestion des bibliothèques, le périmètre pourrait inclure la gestion des prêts et des retours, mais exclure la gestion des achats de livres. Les livrables pourraient inclure le système de gestion des prêts et un rapport de suivi des retards.

Exercice 2 : Étude de cas simplifiée

Contexte : Une petite librairie souhaite informatiser sa gestion des stocks et des ventes.

Consigne :

1. Identifiez 3 techniques de recueil des besoins appropriées pour ce projet et justifiez votre choix.
2. Listez 5 objectifs SMART pour ce projet.
3. Énumérez les principaux acteurs et flux d'information pour ce système.

Correction :

1. Techniques de recueil des besoins : a) Entretiens avec le propriétaire et les employés : pour comprendre en détail les processus actuels et les besoins spécifiques. b) Observation directe : pour voir comment les tâches sont réellement effectuées et identifier les points d'amélioration. c) Questionnaire aux clients : pour comprendre leurs attentes en termes de service.
2. Objectifs SMART : a) Réduire le temps de traitement d'une vente de 5 minutes à 2 minutes en moyenne d'ici 6 mois. b) Diminuer les ruptures de stock de 30% dans les 3 mois suivant l'implémentation du système. c) Augmenter la précision de l'inventaire à 98% dans les 2 mois suivant la mise en place. d) Former 100% du personnel à l'utilisation du nouveau système dans le mois précédant son lancement. e) Réduire le temps consacré à la gestion administrative de 25% dans les 4 mois suivant l'implémentation.
3. Acteurs et flux d'information : Acteurs :
4. Propriétaire de la librairie
5. Employés (vendeurs, gestionnaire de stock)
6. Clients
7. Fournisseurs de livres

Flux d'information :

1. Commandes de livres (Librairie → Fournisseurs)
2. Livraisons de stock (Fournisseurs → Librairie)
3. Ventes de livres (Clients → Librairie)
4. Mises à jour du stock (Employés → Système)
5. Rapports de ventes et de stock (Système → Propriétaire)

Conclusion du Module 2

L'analyse préliminaire et l'étude des besoins sont des étapes cruciales dans le développement d'un système d'information. Elles permettent de comprendre les attentes des utilisateurs, d'analyser l'existant, et de définir clairement les objectifs et le périmètre du projet. Une documentation initiale bien rédigée assure une base solide pour les phases suivantes du projet.

Module 3 : Le Modèle Conceptuel des Données (MCD)

Introduction

Le Modèle Conceptuel des Données (MCD) est une étape cruciale dans la conception d'un système d'information. Il permet de représenter de manière structurée et visuelle les données d'un système, ainsi que leurs relations, indépendamment des contraintes techniques. Dans ce module, nous allons explorer en détail les concepts fondamentaux du MCD, la méthodologie de construction, et nous terminerons par un atelier pratique pour consolider vos connaissances.

1. Concepts fondamentaux du MCD

1.1 Définition et importance du MCD

Le MCD est une représentation graphique et structurée des données à stocker dans un système d'information. Il fait partie de la phase de conception dans le cycle de développement d'un projet informatique.

Importance du MCD :

- Visualisation claire de la structure des données
- Facilitation de la communication entre les différents acteurs du projet
- Base pour la création de la base de données
- Aide à la détection précoce des erreurs de conception

Exemple concret : Imaginons que vous êtes chargé de concevoir un système de gestion pour une bibliothèque municipale. Le MCD vous permettra de représenter visuellement toutes les entités importantes (livres, auteurs, adhérents, emprunts) et leurs relations. Cela aidera l'équipe de développement à comprendre comment les données sont structurées, et permettra aux bibliothécaires de vérifier que tous leurs besoins en termes de gestion de données sont pris en compte.

1.2 Principes de conception du MCD

Les principes fondamentaux de la conception d'un MCD sont :

1. **Simplicité** : Le modèle doit être facile à comprendre, même pour des non-techniciens.
2. **Exhaustivité** : Toutes les données pertinentes doivent être représentées.
3. **Non-redondance** : Éviter la duplication des informations pour maintenir l'intégrité des données.

4. **Évolutivité** : Le modèle doit pouvoir s'adapter aux changements futurs sans nécessiter une refonte complète.

Exemple concret : Dans notre système de gestion de bibliothèque, le principe de non-redondance nous empêcherait de stocker le nom de l'auteur à la fois dans l'entité "Livre" et dans une entité séparée "Auteur". Nous créerions plutôt une relation entre ces deux entités.

1.3 Logiciel de modélisation

Il existe plusieurs logiciels pour créer des MCD. Voici une comparaison détaillée :

Logiciel	Avantages	Inconvénients	Cas d'utilisation idéal
Looping	Gratuit, simple d'utilisation	Fonctionnalités limitées	Projets académiques, petits projets
PowerAMC	Puissant, supporte tout le cycle de vie	Coûteux, courbe d'apprentissage raide	Grands projets d'entreprise
MySQL Workbench	Gratuit, intégré avec MySQL	Spécifique à MySQL	Projets utilisant MySQL

Pour ce cours, nous utiliserons Looping pour sa simplicité et sa gratuité.

1.4 Entités, associations, attributs

a) Entités

Une entité représente un ensemble d'objets ayant des propriétés communes et une existence propre dans le système que l'on modélise.

Contraintes d'une entité :

- Un nom unique au singulier
- Au moins un identifiant (clé primaire)
- Au moins une propriété

Exemple concret : Dans notre système de gestion de bibliothèque, nous aurions les entités suivantes :

- LIVRE
- AUTEUR
- ADHERENT
- EMPRUNT

Représentation graphique d'une entité :

```
+-----+
|  LIVRE  |
+-----+
```

b) Attributs

Les attributs sont les propriétés qui décrivent une entité.

Les propriétés peuvent être de différents types :

- **Simple**s : une seule valeur (ex: nom, prénom)
- **Composées** : regroupement de propriétés (ex: adresse = numéro + rue + ville + code postal)
- **Multivaluées** : plusieurs valeurs possibles (ex: Un utilisateur peut avoir plusieurs numéros de téléphone, Une commande peut contenir plusieurs produits.)
- **Calculées** : dérivées d'autres propriétés (ex: âge calculé à partir de la date de naissance)

Exemple concret : Pour l'entité "LIVRE", les attributs pourraient être :

- ISBN (Identifiant unique)
- Titre
- Année de publication
- Nombre de pages
- Éditeur

Représentation graphique avec attributs :

```
+-----+
|  LIVRE  |
+-----+
| ISBN    |
| Titre   |
| Année_publication |
| Nombre_pages |
| Editeur  |
+-----+
```

c) Associations

Une association représente un lien sémantique entre deux ou plusieurs entités.

Elle peut avoir :

- Un nom unique (verbe à l'infinitif)
- Des cardinalités pour chaque entité liée
- Des propriétés propres (dates, quantités, etc.)

Exemple concret : Dans notre système de bibliothèque, nous aurions les associations suivantes :

- "Écrit" entre AUTEUR et LIVRE
- "Emprunte" entre ADHERENT et LIVRE
- "Concerne" entre EMPRUNT et LIVRE

Représentation graphique d'une association :



d) Occurrences

Une occurrence est une instance concrète de cette entité.

Chaque occurrence est définie par une valeur unique de sa clé primaire ou identifiant.

Exemple concret : Dans notre système de bibliothèque, une occurrence de l'entité "LIVRE" pourrait être :

- ISBN : 9782222222222
- Titre : "Le livre de la vie"
- Éditeur : "Hachette"
- Année de publication : 2020
- Nombre de pages : 300

1.5 Les cardinalités et les types de relations

Les cardinalités expriment le nombre minimum et maximum de fois qu'une entité peut participer à une association.

Types de relations : Les relations dans une base de données définissent comment les tables sont connectées entre elles. Elles sont exprimées par des cardinalités qui sont des combinaisons de:

0 : Optionnel 1 : Exactement un N (ou *) : Plusieurs

1. **Relation (0,1)** : Une occurrence de l'entité peut être associée à 0 ou 1 occurrence de l'autre entité.

EMPLOYÉ $\xrightarrow{0,1}$ VOITURE

Exemple : Un employé peut avoir ou non une voiture de fonction.

2. **Relation (1,1)** : Une occurrence de l'entité doit être associée à exactement 1 occurrence de l'autre entité.

PERSONNE $\xrightarrow{1,1}$ PASSEPORT

Exemple : Une personne a exactement un passeport.

3. **Relation (0,n)** : Une occurrence de l'entité peut être associée à 0, 1 ou plusieurs occurrences de l'autre entité.

CLIENT $\xrightarrow{0,n}$ COMMANDE

Exemple : Un client peut n'avoir fait aucune commande ou en avoir fait plusieurs.

4. **Relation (1,n)** : Une occurrence de l'entité doit être associée à au moins 1 occurrence de l'autre entité.

ÉQUIPE $\xrightarrow{1,n}$ JOUEUR

Exemple : Une équipe doit avoir au moins un joueur.

Représentation graphique avec cardinalités :

```

+-----+  0,n   Écrit   1,n  +-----+
| LIVRE  | <-----> | AUTEUR |
+-----+              +-----+

```

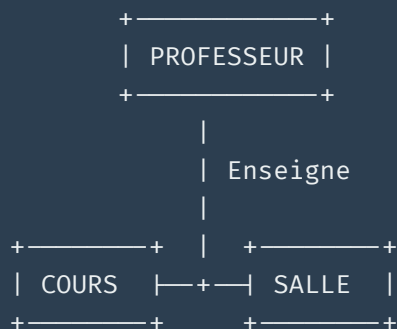
Ici, un livre peut être écrit par un ou plusieurs auteurs (1,n), et un auteur peut écrire zéro ou plusieurs livres (0,n).

1.6 Les relations ternaires

Une relation ternaire implique trois entités.

Exemple concret : Dans un système de gestion d'université, on pourrait avoir une relation ternaire "Enseigne" entre les entités "PROFESSEUR", "COURS" et "SALLE".

Représentation graphique :



Cette relation indique quel professeur enseigne quel cours dans quelle salle.

1.7 Contraintes d'intégrité (CIF) et règles de modélisation

Les Contraintes d'Intégrité Fonctionnelle (CIF) sont des règles qui garantissent la cohérence des données.

Exemple concret de CIF : Dans notre système de bibliothèque, une CIF pourrait être : "Un adhérent ne peut pas emprunter plus de 5 livres simultanément."

Représentation graphique d'une CIF :



Règles de modélisation

Définir clairement les cardinalités des relations : Chaque relation doit être accompagnée de cardinalités (1,1 ; 1,N ; 0,1 ; N,N) qui spécifient combien d'instances d'une entité peuvent être associées à une instance de l'autre entité.

Éviter les entités vides ou inutiles : Ne modélisez pas d'entités qui ne contiennent aucune information ou qui ne sont pas nécessaires pour le modèle de données.

Choisir des noms d'entités et d'attributs explicites : Les noms des entités et des attributs doivent être clairs et significatifs pour que l'on puisse facilement comprendre ce qu'ils représentent (par exemple, **Client**, **Date_Commande**, etc.).

Respecter l'intégrité des données : Assurez-vous que les entités respectent les contraintes d'intégrité, telles que l'intégrité référentielle, ce qui garantit que les données restent cohérentes et valides.

Utiliser des entités faibles si nécessaire : Lorsqu'une entité dépend d'une autre pour son identification, utilisez une entité faible, identifiée par une clé composée de l'entité forte et d'un attribut supplémentaire.

2. Méthodologie de construction

2.1 Analyse du cahier des charges

1. **Lire attentivement** le cahier des charges.
2. **Identifier** les entités principales.
3. **Lister** les attributs de chaque entité.
4. **Déterminer** les relations entre les entités.

Exemple concret : Imaginons que nous devons concevoir un système de gestion pour un petit hôtel. Voici un extrait du cahier des charges :

"L'hôtel dispose de 20 chambres, chacune ayant un numéro unique, un type (simple, double, suite) et un tarif. Les clients réservent des chambres pour des périodes spécifiques. Chaque client a un nom, un prénom, une adresse email et un numéro de téléphone. L'hôtel propose également des services optionnels comme le petit-déjeuner, le service de chambre et la blanchisserie."

Analyse :

- Entités identifiées : CHAMBRE, CLIENT, RESERVATION, SERVICE
- Attributs (exemples) :
 - CHAMBRE : Numéro, Type, Tarif
 - CLIENT : Nom, Prénom, Email, Téléphone
- Relations :
 - Un CLIENT fait une RESERVATION
 - Une RESERVATION concerne une CHAMBRE
 - Une RESERVATION peut inclure des SERVICES

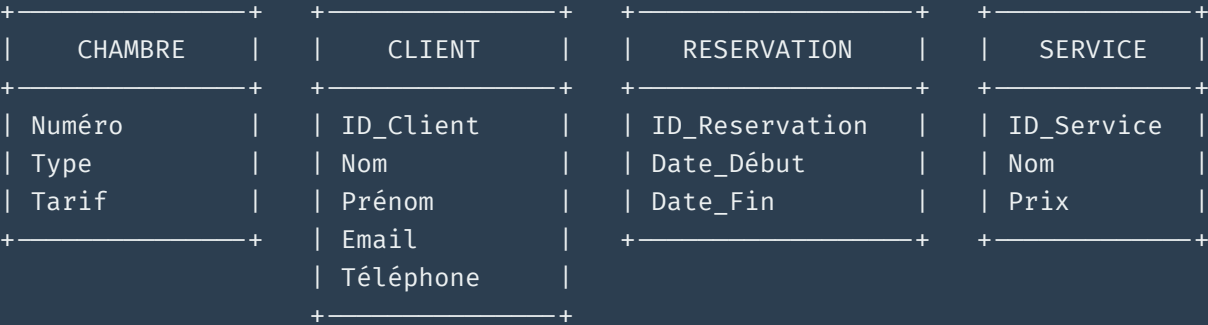
2.2 Création d'un MCD étape par étape

Prenons l'exemple de l'hôtel pour illustrer chaque étape :

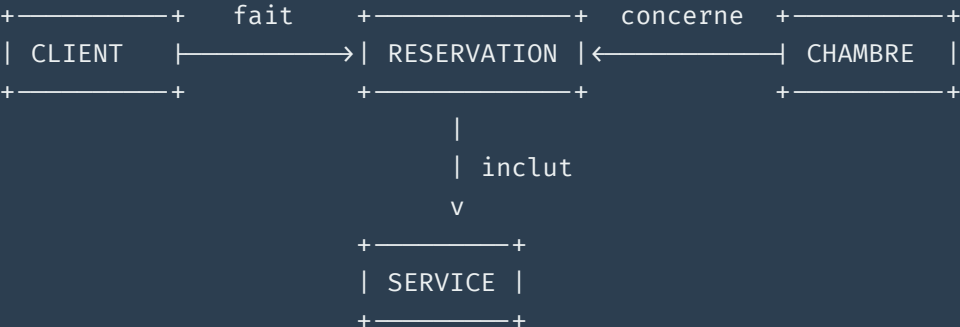
1. **Dessiner les entités** :



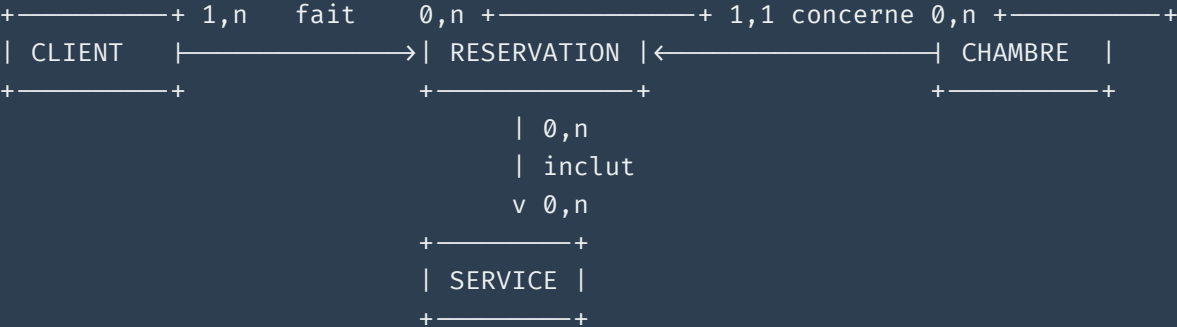
2. Ajouter les attributs :



3. Établir les associations :



4. Définir les cardinalités :



5. Vérifier et optimiser :

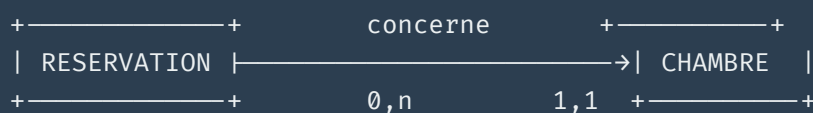
6. S'assurer que toutes les entités ont un identifiant unique.
7. Vérifier que les cardinalités reflètent correctement les règles métier.
8. S'assurer qu'il n'y a pas de redondance dans le modèle.

2.3 MCD - CIF (Contraintes d'Intégrité Fonctionnelle)

Les CIF sont des règles qui garantissent la cohérence des données. Elles sont généralement représentées par des flèches dans le MCD.

Exemple concret : Dans notre système hôtelier, une CIF pourrait être : "Une chambre ne peut être réservée que si elle est disponible pour la période demandée."

Représentation graphique :



Cette CIF indique qu'une réservation doit obligatoirement concerner une et une seule chambre.

2.4 MCD - CIM (Contraintes d'Intégrité Multiples)

Les CIM sont des contraintes qui impliquent plusieurs associations ou entités.

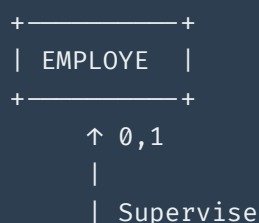
Exemple concret : Dans notre système hôtelier, une CIM pourrait être : "Le montant total d'une réservation doit être égal à la somme du tarif de la chambre pour la durée du séjour plus le coût des services additionnels."

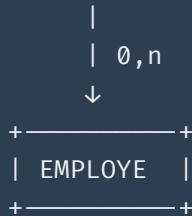
Cette contrainte implique les entités RESERVATION, CHAMBRE et SERVICE, ainsi que leurs associations.

2.5 MCD - Réflexivité

Une association réflexive relie une entité à elle-même.

Exemple concret : Dans un système de gestion des employés d'un hôtel, on pourrait avoir une association réflexive "Supervise" sur l'entité EMPLOYE.





Cela indique qu'un employé peut superviser plusieurs autres employés, et qu'un employé est supervisé par au plus un autre employé.

2.6 MCD - Dépendance fonctionnelle

Une dépendance fonctionnelle existe lorsque la valeur d'un attribut détermine de façon unique la valeur d'un autre attribut.

Exemple concret : Dans notre entité RESERVATION, le numéro de réservation (ID_Reservation) détermine de façon unique les dates de début et de fin du séjour.

2.7 MCD - Règles de normalisation

Les règles de normalisation aident à structurer les données de manière à minimiser la redondance et les anomalies de mise à jour.

- 1. **Première forme normale (1NF) :** Éliminer les groupes répétitifs. **Exemple :** Au lieu d'avoir un attribut "Téléphones" dans l'entité CLIENT qui contiendrait plusieurs numéros, créer une entité séparée TELEPHONE reliée à CLIENT.
- 2. **Deuxième forme normale (2NF) :** Éliminer les dépendances partielles. **Exemple :** Dans une entité RESERVATION_SERVICE qui lierait RESERVATION et SERVICE, s'assurer que tous les attributs dépendent de la clé complète (ID_Reservation, ID_Service) et non d'une partie seulement.
- 3. **Troisième forme normale (3NF) :** Éliminer les dépendances transitives. **Exemple :** Si dans l'entité CHAMBRE, le type de chambre détermine le tarif, on pourrait créer une entité séparée TYPE_CHAMBRE avec les attributs Type et Tarif.

2.8 Dictionnaire des données

Le dictionnaire des données est un document qui liste et décrit tous les éléments du MCD. C'est un outil essentiel pour maintenir la cohérence et la clarté du modèle. **Table CHAMBRE :**

Nom de l'entité	Nom de l'attribut	Type de données	Contraintes	Description
CHAMBRE	Numéro	Entier	Clé primaire	Numéro unique de la chambre
CHAMBRE	Type	Chaîne	Non nul	Type de chambre (simple, double, suite)

Nom de l'entité	Nom de l'attribut	Type de données	Contraintes	Description
CHAMBRE	Tarif	Décimal	Non nul	Tarif par nuit de la chambre

Table CLIENT :

Nom de l'entité	Nom de l'attribut	Type de données	Contraintes	Description
CLIENT	ID_Client	Entier	Clé primaire	Identifiant unique du client
CLIENT	Nom	Chaîne	Non nul	Nom de famille du client
CLIENT	Prénom	Chaîne	Non nul	Prénom du client
CLIENT	Email	Chaîne	Unique	Adresse email du client
CLIENT	Téléphone	Chaîne	Non nul	Numéro de téléphone du client

Table RESERVATION :

Nom de l'entité	Nom de l'attribut	Type de données	Contraintes	Description
RESERVATION	ID_Reservation	Entier	Clé primaire	Identifiant unique de la réservation
RESERVATION	Date_Début	Date	Non nul	Date de début du séjour
RESERVATION	Date_Fin	Date	Non nul	Date de fin du séjour

Table SERVICE :

Nom de l'entité	Nom de l'attribut	Type de données	Contraintes	Description
SERVICE	ID_Service	Entier	Clé primaire	Identifiant unique du service
SERVICE	Nom	Chaîne	Non nul	Nom du service proposé
SERVICE	Prix	Décimal	Non nul	Prix du service

Module 4 : Le Modèle Logique des Données (MLD)

Introduction

Le Modèle Logique des Données (MLD) est une étape cruciale dans la conception d'un système d'information. Il permet de traduire le Modèle Conceptuel de Données (MCD) en un modèle exploitable par un système de gestion de base de données (SGBD). Ce module vise à vous familiariser avec les concepts clés du MLD, à vous apprendre à construire un MLD à partir d'un MCD, et à vous montrer comment optimiser et vérifier ce modèle.

1. Introduction au MLD

1.1. Passage du MCD au MLD : principes et règles de transformation

Le passage du MCD au MLD consiste à transformer les entités, associations et attributs du MCD en tables, clés primaires, clés étrangères et relations dans le MLD. Voici les principes de base :

- **Entité → Table** : Chaque entité du MCD devient une table dans le MLD.
- **Attribut → Colonne** : Les attributs de l'entité deviennent des colonnes de la table.
- **Association → Relation** : Les associations entre entités sont traduites en relations entre tables via des clés étrangères.

Exemple :

MCD	MLD
Entité : Client	Table : Client
Attributs : ID, Nom, Adresse	Colonnes : ID (clé primaire), Nom, Adresse

1.2. Concepts clés

- **Tables** : Structures de stockage des données dans une base de données relationnelle.
- **Clé primaire** : Identifiant unique d'une table (ex : ID Client).
- **Clé étrangère** : Colonne qui fait référence à la clé primaire d'une autre table pour établir une relation.
- **Relations** : Liens entre tables établis via des clés étrangères.

Exemple concret :

- **Table Client** : ID (clé primaire), Nom, Adresse.
- **Table Commande** : ID (clé primaire), Date, Montant, ID_Client (clé étrangère).

2. Construction du MLD

2.1. Définition des tables et des relations

Pour construire un MLD, suivez ces étapes :

1. **Identifier les tables** : À partir des entités du MCD.
2. **Définir les colonnes** : À partir des attributs de chaque entité.
3. **Ajouter les clés primaires** : Chaque table doit avoir une clé primaire unique.
4. **Établir les relations** : Utiliser des clés étrangères pour relier les tables.

Exemple :

- **MCD** :
 - Entité : Client (ID, Nom, Adresse)
 - Entité : Commande (ID, Date, Montant)
 - Association : Un client peut passer plusieurs commandes.
- **MLD** :
 - Table Client : ID (PK), Nom, Adresse.
 - Table Commande : ID (PK), Date, Montant, ID_Client (FK).

2.2. Schéma et schéma de données

- **Schéma de base de données** : Représentation visuelle des tables et de leurs relations.
- **Schéma de données** : Description détaillée des tables, colonnes, types de données, et contraintes.

Exemple de schéma :

```
Client (ID, Nom, Adresse)
Commande (ID, Date, Montant, ID_Client)
```

Relation : Une commande est associée à un client via ID_Client.



Règles de Passage du MCD au MLD

✓ 1. Transformation des Entités en Tables

- Chaque **entité** du MCD devient une **table** dans le MLD.
- Les **attributs** de l'entité deviennent les **colonnes** de la table.
- L'**identifiant de l'entité** (clé primaire) est conservé comme **clé primaire** (PK) de la table.

Exemple :

MCD → Entité : Client (id_client, nom, email)

MLD → Table Client (id_client [PK], nom, email)

Passage d'un MCD (Modèle Conceptuel de Données) à un MLD (Modèle Logique de Données)

Pour traduire un MCD en un MLD, il suffit d'appliquer cinq règles simples. Voici comment procéder, avec des exemples clairs pour chaque cas.

Notations des Associations

Une association binaire (entre deux entités) peut être de trois types :

- **1:1 (un à un)** : aucune des deux cardinalités maximales n'est "n".
- **1:n (un à plusieurs)** : une des deux cardinalités maximales est "n".
- **n:m (plusieurs à plusieurs)** : les deux cardinalités maximales sont "n".

Remarque : Le MLD ne fait pas de distinction entre (0,n) et (1,n), mais distingue (0,1) de (1,1).

Règle 1 : Transformation des Entités

Principe : Chaque entité devient une table. Les attributs de l'entité deviennent des colonnes, et l'identifiant de l'entité devient la clé primaire.

Exemple :

L'entité **Article** possède :

- Numéro d'article (identifiant)
- Désignation
- Prix unitaire

Cela donne la table **Article** avec les colonnes correspondantes.

Règle 2 : Association Binaire de Type 1:n

Principe : L'association disparaît. On ajoute une clé étrangère dans la table correspondant au côté "n", qui référence la clé primaire de la table du côté "1".

- Si la cardinalité est (1,1), la clé étrangère ne peut pas être vide.
- Les attributs de l'association sont ajoutés à la table du côté "n".

Exemple :

Un **Fournisseur** peut livrer plusieurs **Livraisons**, mais chaque livraison est réalisée par un seul fournisseur.

La table **Livraison** contient une référence vers le fournisseur.

Règle 3 : Association Binaire de Type n:m

Principe : L'association devient une table supplémentaire (appelée table de jonction). La clé primaire de cette table est composée des deux clés étrangères référençant les entités associées.

- Les attributs de l'association deviennent des colonnes de cette table.

Exemple :

Un **Article** peut apparaître dans plusieurs **Commandes**, et une **Commande** peut contenir plusieurs articles.

Une table intermédiaire **Ligne de Commande** relie les deux, avec un attribut comme la quantité commandée.

Règle 4 : Association Binaire de Type 1:1

Principe :

- L'association est traitée comme une association 1:n, mais avec une contrainte d'unicité sur la clé étrangère.
- Si les deux cardinalités sont (0,1), la clé étrangère peut être placée indifféremment dans l'une des deux tables.

Exemple :

Chaque **Service** est dirigé par un seul **Employé**, et chaque **Employé** peut diriger un seul service.

La clé étrangère est ajoutée à la table du service, avec une contrainte d'unicité.

Règle 5 : Association Non Binaire (Ternaires ou Plus)

Principe : L'association devient une table supplémentaire. La clé primaire est composée des clés étrangères référençant les entités concernées. Les attributs de l'association deviennent des colonnes.

Exemple :

Un **Film** peut être projeté dans une **Salle** à un **Créneau** horaire spécifique.

La table **Projection** contiendra des références aux trois entités et un attribut supplémentaire comme le tarif.

Conclusion

Le Modèle Logique des Données (MLD) est une étape essentielle dans la conception d'un système d'information. Il permet de traduire le MCD en un modèle exploitable par un SGBD, en définissant des tables, des clés primaires, des clés étrangères et des relations. La construction du MLD nécessite une attention particulière à la normalisation et à la vérification des relations pour garantir l'intégrité et l'efficacité de la base de données.

Annexes

Tableau récapitulatif des transformations MCD → MLD

MCD	MLD
Entité	Table

MCD	MLD
Attribut	Colonne
Association	Relation (clé étrangère)
Identifiant	Clé primaire

Exemple complet de MCD et MLD

MCD :

- Entité : Étudiant (ID, Nom, Prénom)
- Entité : Cours (ID, Nom, Description)
- Association : Un étudiant peut s'inscrire à plusieurs cours.

MLD :

- Table Étudiant : ID (PK), Nom, Prénom.
- Table Cours : ID (PK), Nom, Description.
- Table Inscription : ID_Étudiant (FK), ID_Cours (FK).

Schéma de données :

```
Étudiant (ID, Nom, Prénom)
Cours (ID, Nom, Description)
Inscription (ID_Étudiant, ID_Cours)
```