Système d'information et base de données

Pr: MAJDOUB Soufyane

Université Sidi Mohamed Ben Abdellah de Fès
soufyane.majdoub@usmba.ac.ma

Laboratoire d'Informatique, Signaux, Automatique et Cognitivisme
30 mai 2025

Qu’est-ce qu’un système d’information ?

- Un système d’information (SI) est un ensemble organisé de ressources (matériel, logiciel, données, procédures, utilisateurs, …) permettant de collecter, stocker, traiter et diffuser l’information dans une organisation.
- Objectif : soutenir les processus métiers et la prise de décision en fournissant une information fiable et disponible.
- Composants : bases de données, réseaux, applications, utilisateurs, procédures.

Objectifs d'un SI

- Centraliser et fiabiliser les données pour éviter les doubles saisies.
- Automatiser des tâches et optimiser les processus (paie, gestion des stocks, etc.).
- Offrir un accès rapide à l'information pour la prise de décision.
- Faciliter la communication interne et externe (rapports, échanges de données).

Rôle d'un SI dans l'organisation

- Support des processus métiers : le SI prend en charge les opérations quotidiennes (gestion clients, ressources humaines, etc.).
- Système d'information décisionnel : exploitation des données pour produire des indicateurs, analyses (tableaux de bord, entrepôts de données).
- Alignement stratégique : un SI adapté aux besoins renforce la compétitivité de l'entreprise.
- Exemples : CRM (gestion clients), ERP (gestion intégrée), SGBD (gestion des bases de données).

Exemples de SI dans l'entreprise

- Système de gestion des ressources humaines (paie, congés, formation).
- Système de gestion des commandes et des stocks (ERP, logistique).
- Système d'information géographique (cartographie, agences).
- Applications métiers spécifiques (ex : bancaires, assurances, santé, ...).

Modélisation conceptuelle (MCD)

- Le Modèle Conceptuel de Données (MCD) formalise les données du domaine métier sans considération technique.
- Principaux concepts : **entité** (objet du monde réel), **attributs** (propriétés des entités), **association** (relation entre entités).
- But du MCD : définir clairement quelles données seront gérées et comment elles sont reliées.

Entités et attributs

- **Entité** : représente un objet ou concept (ex : *Etudiant*, *Cours*).
- **Attribut** : propriété descriptive d'une entité (ex : *noEtudiant*, *nom*, *prénom* pour l'entité *Etudiant*).
- **Identifiant** (clé primaire) : attribut (ou combinaison) identifiant de façon unique chaque occurrence.
- Par convention, on souligne l'attribut identifiant sur le MCD.

Associations et cardinalités

- **Association** : lien sémantique entre entités (ex : ≪ *S'inscrit* ≫ entre *Etudiant* et *Cours*).
- **Rôles** : chaque entité a un rôle dans l'association (ex : un *Etudiant* joue le rôle ≪ *S'inscrit* ≫).
- **Cardinalités** : contraintes min/max sur les participations (1, 0..1, 1..*n*, 0..*n*, etc.).
- Exemple : 1..*n* entre *Cours* et *Inscription* (un cours peut avoir plusieurs inscriptions).

Exemple de Modèle Conceptuel (MCD)

- Contexte universitaire : entités *Etudiant*, *Cours*, *Professeur*, *Inscription* avec attribut *note*.
- Associations : *Etudiant* `S'inscrit à` *Cours* (attribut : note), *Professeur* `Enseigne` *Cours*.
- Cardinalités : par exemple, un *Etudiant* peut s'inscrire à plusieurs *Cours* (1..n).
- Ce schéma serait représenté sous forme d'un diagramme entité-association.

Du MCD au MLD (passage au modèle logique)

- Chaque entité du MCD devient une table relationnelle.
- Les attributs deviennent les colonnes de la table.
- L'identifiant devient la clé primaire de la table.
- Une association 1..*n* se traduit par une clé étrangère dans la table du côté *n*.
- Une association *n*-*m* se traduit par une table d'association.

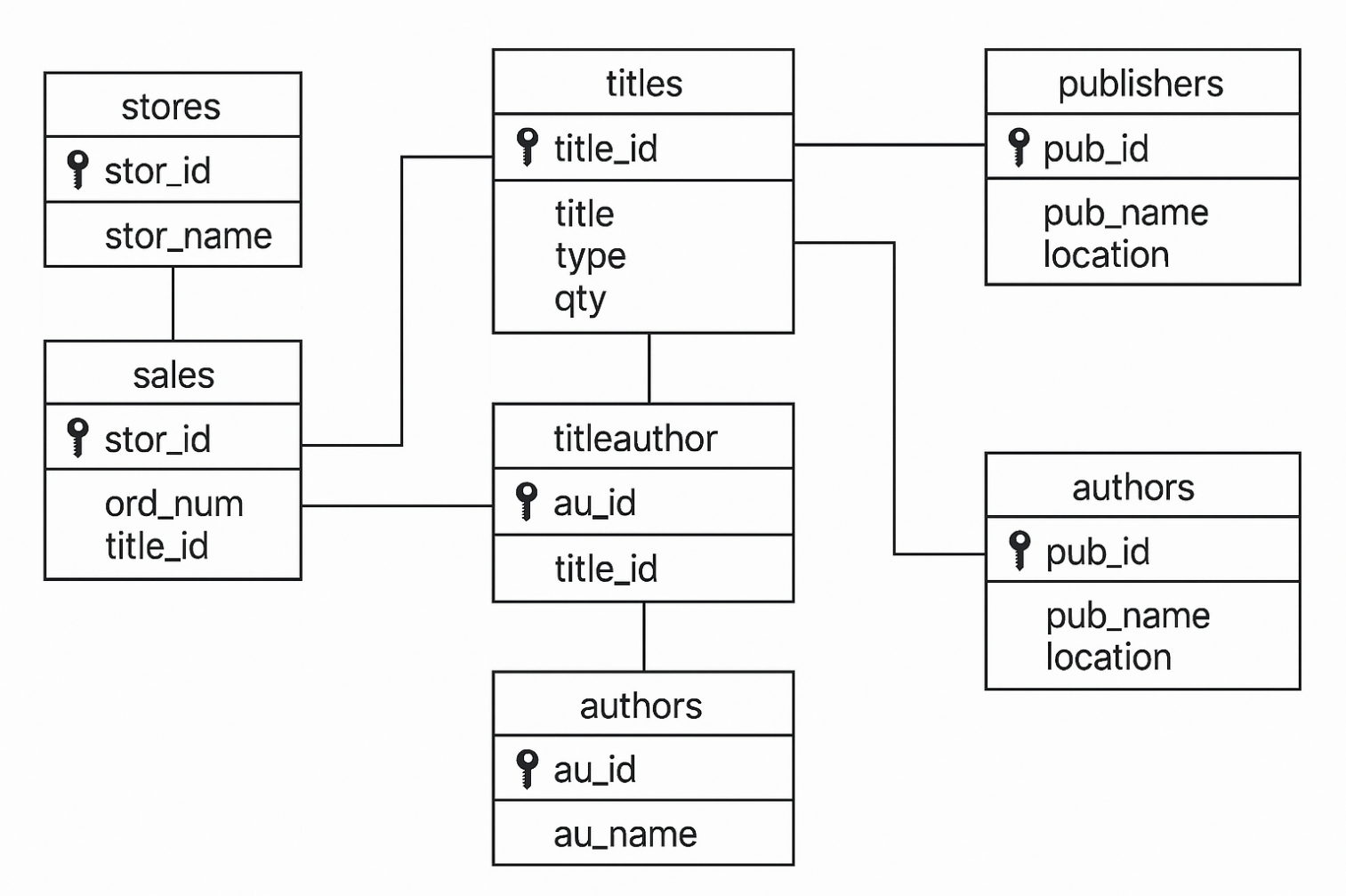
Modèle Logique de Données (MLD)

- Le MLD est l'équivalent relationnel du MCD (sous forme de tables).
- **Table** : représente un ensemble d'occurrences d'une entité.
- **Attributs** : colonnes de la table.
- **Clé primaire** (PK) : identifiant unique de la table.
- **Clé étrangère** (FK) : attribut référant la clé primaire d'une autre table.

Modèle Physique de Données (MPD)

- Le MPD est la définition technique des tables et indices selon un SGBD déterminé.
- Il spécifie les types physiques de colonnes, les tablespaces, les index, etc.
- Exemple : générer un script SQL pour MySQL ou PostgreSQL à partir du MLD.
- Objectif : optimiser les performances (index, partitions, vues matérialisées).

Exemple de Modèle Logique (tables & relations)



- Ce schéma montre des tables (*stores*, *sales*, *titles*, ...) avec leurs clés primaires (icônes clé) et relations par clés étrangères (liens).
- Par exemple, *sales* contient *stor_id* comme FK vers *stores*, et *title_id* vers *titles*.

Dépendances Fonctionnelles

- Une dépendance fonctionnelle exprime qu'une valeur d'attribut *détermine* une autre valeur.
- Notation : $X \rightarrow Y$ signifie que si deux enregistrements ont la même valeur pour X, ils ont la même valeur pour Y.
- Exemple : dans une table **ETUDIANT(noEt, Nom, Prénom, ville)**, on a $noEt \rightarrow \{\text{Nom, Prénom, ville}\}$.
- Ces dépendances servent à normaliser le schéma (1NF, 2NF, 3NF).

Normalisation (formes normales)

- **1NF** (Première Forme Normale) : chaque attribut est atomique, et les lignes sont uniques (pas de répétition).
- **2NF** : en 1NF et aucun attribut non-clé dépend d'une partie seulement de la clé primaire (pas de dépendance partielle).
- **3NF** : en 2NF et aucun attribut non-clé dépend transitivement de la clé (pas de dépendance transitive).
- Objectif : éviter les redondances et anomalies (insertion, suppression, mise à jour).

Le langage SQL

- SQL (Structured Query Language) : langage standard pour interagir avec des bases de données relationnelles.
- Deux parties principales :
  - DDL (Data Definition Language) : création et modification des schémas (`CREATE TABLE`, `ALTER`, etc.).
  - DML (Data Manipulation Language) : gestion des données (`SELECT`, `INSERT`, `UPDATE`, `DELETE`).
- Exemples de SGBD : MySQL, PostgreSQL, Oracle, SQL Server.

Requête SELECT (extraction)

- Syntaxe générale : `SELECT <attributs> FROM <table> [WHERE condition];`
- Permet d'interroger la base et de sélectionner des tuples selon des critères.
- Exemple :

```
SELECT nom, prenom
FROM ETUDIANT
WHERE ville = 'Paris';
```

- Résultat : liste des {nom, prénom} des *Etudiant*s habitant Paris.

Clause WHERE et filtrage

- `WHERE` permet de filtrer les résultats : conditions (`=`, `<`, `>`, `<=`, `>=`), `BETWEEN`, `IN`, `LIKE`, `AND`, `OR`.
- Exemple :

```
SELECT *
FROM ETUDIANT
WHERE annee_naissance >= 2000
AND ville IN ('Paris', 'Lyon');
```

- Récupère tous les *Etudiants* nés à partir de 2000 habitant Paris ou Lyon.

Jointure (JOIN) entre tables

- Les jointures relient plusieurs tables selon des clés communes.
- **INNER JOIN** : renvoie les lignes correspondantes dans les deux tables.
- Types de jointures : INNER, LEFT, RIGHT, FULL, etc.
- Syntaxe : `SELECT * FROM A INNER JOIN B ON A.cle = B.cle;`

Exemple de jointure

- *Objectif* : lister les cours suivis par chaque *Etudiant*.

```
SELECT E.nom, E.prenom, C.intituleCours
FROM ETUDIANT E
INNER JOIN INSCRIPTION I ON E.noEtudiant = I.noEtudiant
INNER JOIN COURS C ON I.idCours = C.idCours;
```

- Cette requête associe *ETUDIANT*, *INSCRIPTION*, *COURS* via leurs clés étrangères.

Insertion de données (INSERT)

- Commande `INSERT INTO <table>(<colonnes>) VALUES (<valeurs>);` pour ajouter un enregistrement.
- Exemple : ajouter un *Etudiant*.

```
INSERT INTO ETUDIANT(noEtudiant, nom, prenom,
annee_naissance)
VALUES (1234, 'Dupont', 'Alice', 2002);
```

- Insère une nouvelle ligne dans la table *ETUDIANT*.

Mise à jour de données (UPDATE)

- Commande `UPDATE <table> SET <colonne>=<valeur>, ... WHERE condition;` pour modifier des enregistrements existants.
- Exemple : modifier le nom d'un *Etudiant*.

```
UPDATE ETUDIANT
SET nom = 'Durand'
WHERE noEtudiant = 1234;
```

- Cette requête met à jour le nom de l'*Etudiant* dont l'ID vaut 1234.

Suppression de données (DELETE)

- Commande `DELETE FROM <table> WHERE condition;` pour effacer des enregistrements.
- Exemple : supprimer l'*Etudiant* n°1234.

```
DELETE FROM ETUDIANT
WHERE noEtudiant = 1234;
```

- Cette commande supprime l'enregistrement de l'*Etudiant* dont l'ID est 1234.

Contraintes d'intégrité

- **Clé primaire (PRIMARY KEY)** : garantit l'unicité et la non-nullité.
- **Clé étrangère (FOREIGN KEY)** : assure la cohérence référentielle entre tables.
- **UNIQUE** : force l'unicité sur un ou plusieurs attributs.
- **CHECK** : impose une contrainte de domaine (ex : `CHECK (note >= 0 AND note <= 20)`).
- **NOT NULL** : interdit les valeurs nulles sur un attribut.

Transactions (ACID)

- Transaction : ensemble d'opérations traitées comme un tout indivisible.
- Propriétés ACID : Atomicité, Cohérence, Isolation, Durabilité.
- Garantit la fiabilité des mises à jour lors d'opérations composées.
- Commandes SQL : `BEGIN` (début), `COMMIT` (validation), `ROLLBACK` (annulation).

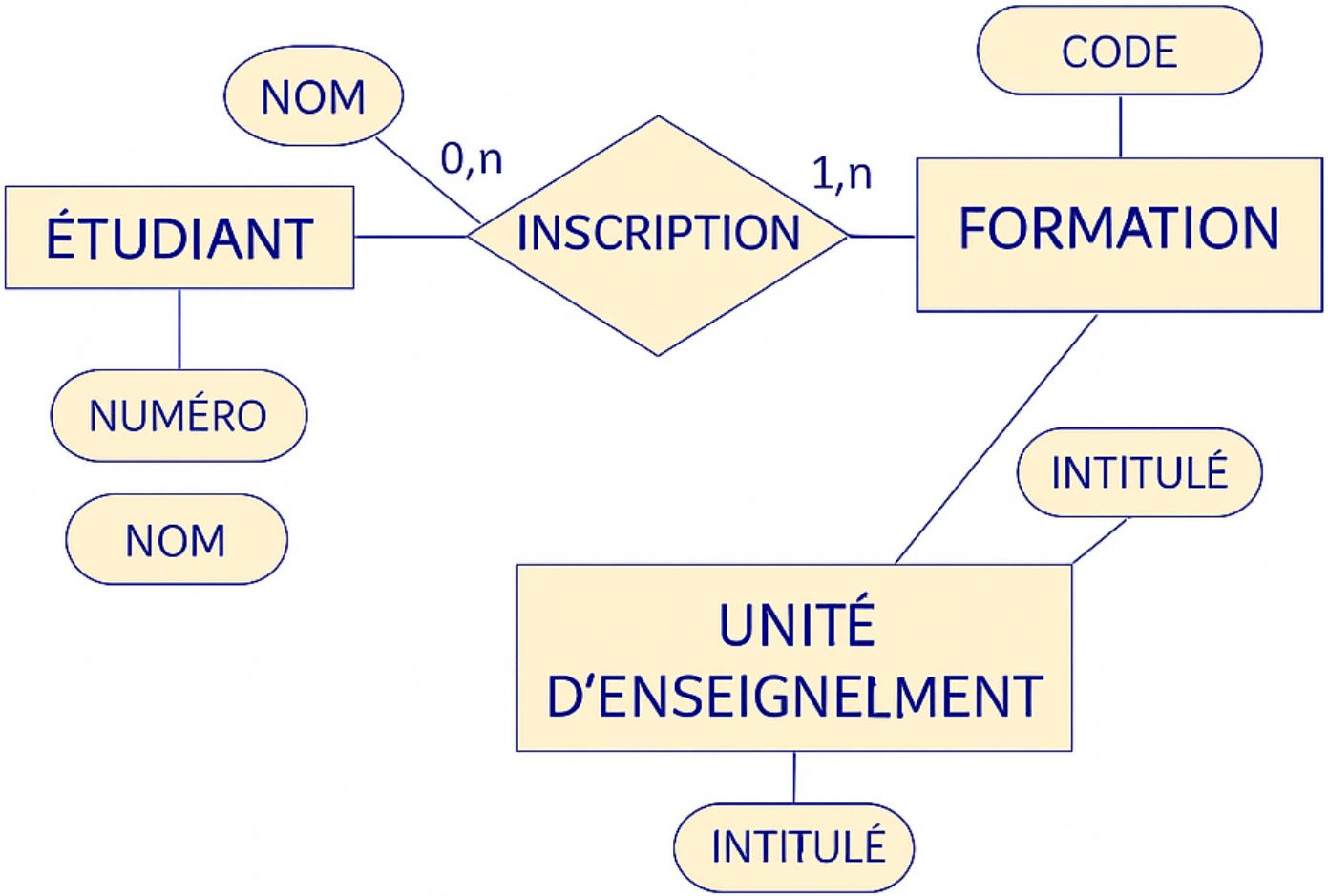
Présentation de PowerAMC

- PowerAMC (PowerDesigner) est un outil de modélisation MERISE/UML.
- Il permet de créer des MCD et de générer automatiquement des MLD ou du code SQL.
- Fonctionnalités : définir entités, attributs, associations ; vérifier les contraintes de modèle.
- Ex : modélisation d'une base de données et génération du script DDL.

Exemple : MCD dans PowerAMC

- Illustration d'un MCD simulé dans l'interface PowerAMC.
- Chaque entité et association est définie graphiquement avec ses attributs.
- Les identifiants sont soulignés (clés primaires), les cardinalités annotées.

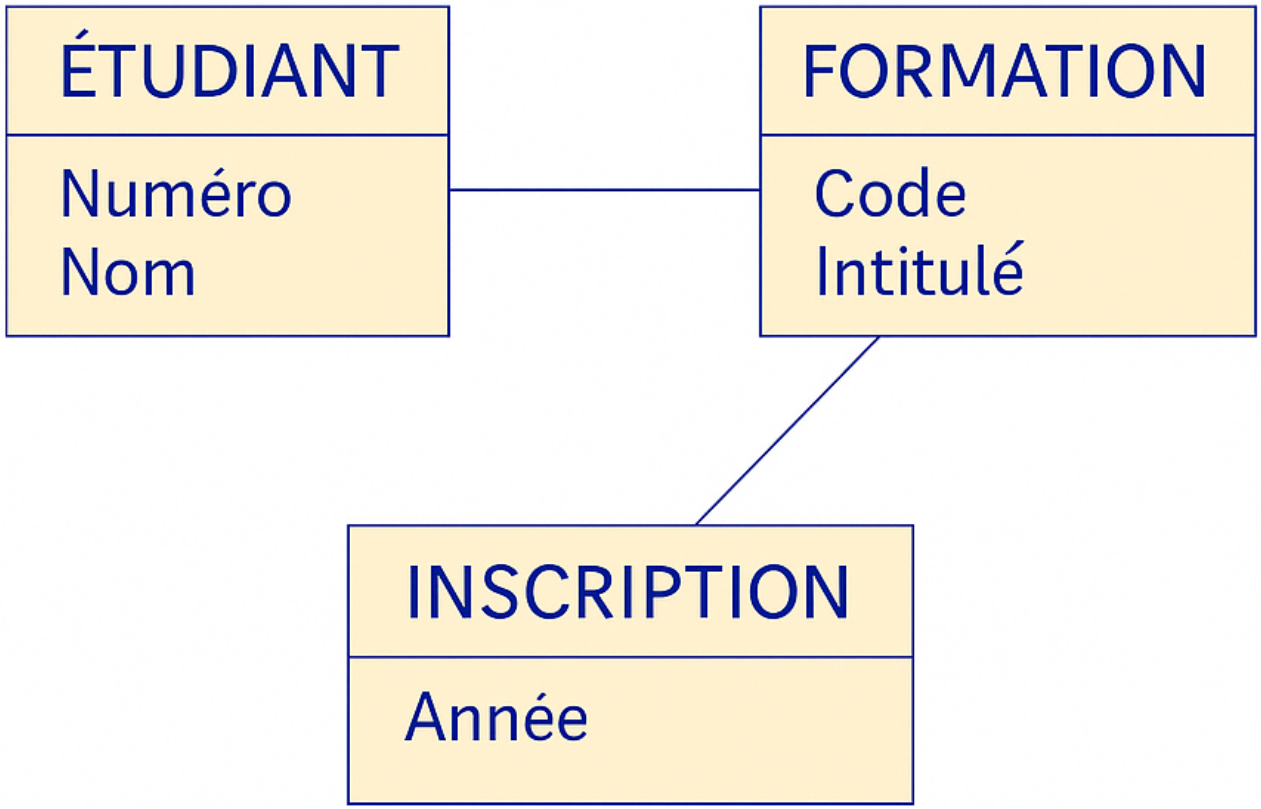
Capture d’écran : MCD (PowerAMC)



Exemple : MLD dans PowerAMC

- PowerAMC convertit le MCD en MLD via des règles (1-n, n-m, etc.).
- Exemple : tables avec clés primaires et étrangères générées automatiquement.

Capture d'écran : MLD (PowerAMC)



(Schéma *MLD* simulé avec PowerAMC)

Génération de SQL

- PowerAMC peut générer le script SQL de création des tables (`CREATE TABLE`, contraintes).
- Ex : `CREATE TABLE ETUDIANT(noEtudiant INT PRIMARY KEY, nom VARCHAR(50), ...);`
- Assure la cohérence entre modélisation et implémentation.

Contexte du cas d'étude

- Objectif : modéliser et manipuler une base de données pour gérer les étudiants et les cours.
- Entités principales : *ETUDIANT(noEtudiant, nom, prénom, dateNaissance)*, *COURS(idCours, intitule, crédits)*, *PROFESSEUR(idProf, nom)*, *INSCRIPTION(noEtudiant, idCours, dateInscription, note)*.
- Règle métier : un *Etudiant* peut suivre plusieurs *Cours*, un *Cours* peut être donné par plusieurs *Professeurs*.
- Le fil rouge suit tout le cycle : MCD, MLD, puis SQL.

Entités et attributs du cas

- **ETUDIANT**(*noEtudiant*, nom, prénom, dateNaissance, ...).
- **COURS**(*idCours*, intitule, crédits, ...).
- **PROFESSEUR**(*idProf*, nom, spécialité, ...).
- **INSCRIPTION**(*noEtudiant*, *idCours*, dateInscription, note).
- **ENSEIGNE**(*idProf*, *idCours*) pour les cours attribués aux professeurs (association n-m).

Schéma conceptuel (MCD) du cas

- **Etudiant** `S'inscrit à` **Cours** (*Inscription* avec attribut *note*).
- **Professeur** `Enseigne` **Cours**.
- Clés primaires soulignées, cardinalités indiquées sur les associations.
- (Schéma MCD illustré pour le cas d'étude universitaire.)

Schéma logique (MLD) du cas

- **ETUDIANT**(noEtudiant*****, nom, prénom, dateNaiss, ...)
  COURS(idCours*****, intitule, crédits, ...)
  PROFESSEUR(idProf*****, nom, spécialité)
  INSCRIPTION(noEtudiant*****, idCours*****, dateInscription, note)
  ENSEIGNE(idProf*****, idCours*****)
- (***** indique clé primaire composite)

Dépendances fonctionnelles (cas d'étude)

- Dans **ETUDIANT** : *noEtudiant* $\rightarrow$ {nom, prénom, dateNaiss, ...}.
- Dans **COURS** : *idCours* $\rightarrow$ {intitule, crédits, ...}.
- Dans **PROFESSEUR** : *idProf* $\rightarrow$ {nom, spécialité}.
- Dans **INSCRIPTION** : {*noEtudiant*, *idCours*} $\rightarrow$ {dateInscription, note}.
- Dans **ENSEIGNE** : {*idProf*, *idCours*} $\rightarrow$ rien d'autre (relation pure).

Exemple : Requête SELECT (cas d'étude)

- *Objectif* : Afficher les cours suivis par un étudiant.

```
SELECT E.nom, E.prenom, C.intitule
FROM ETUDIANT E
INNER JOIN INSCRIPTION I ON E.noEtudiant = I.noEtudiant
INNER JOIN COURS C ON I.idCours = C.idCours
WHERE E.noEtudiant = 2001;
```

- Résultat : liste des cours de l'étudiant 2001 (nom, prénom, intitule).

Exemple : Requête INSERT (cas d'étude)

- *Objectif* : Ajouter une nouvelle inscription.

```
INSERT INTO INSCRIPTION(noEtudiant, idCours, dateInscriptio
VALUES (2001, 300, '2025-05-01', NULL);
```

- Insère un enregistrement liant l'étudiant 2001 au cours 300 (note non saisie).

Exemple : Requête UPDATE (cas d'étude)

- *Objectif* : Modifier la note d'une inscription.

```
UPDATE INSCRIPTION
SET note = 15
WHERE noEtudiant = 2001 AND idCours = 300;
```

- Met à jour la note à 15 pour l'étudiant 2001 inscrit au cours 300.

Exemple : Requête DELETE (cas d'étude)

- *Objectif* : Supprimer une inscription.

```
DELETE FROM INSCRIPTION
WHERE noEtudiant = 2001 AND idCours = 300;
```

- Supprime l'enregistrement liant l'étudiant 2001 au cours 300.

Contraintes dans le cas d'étude

- Clés primaires et étrangères dans les tables :
  - **ETUDIANT**(*noEtudiant* **PK**), **COURS**(*idCours* **PK**), **INSCRIPTION**(**PK**(noEtudiant, idCours)).
  - **INSCRIPTION**(noEtudiant **FK** $\rightarrow$ ETUDIANT.noEtudiant, idCours **FK** $\rightarrow$ COURS.idCours).
- Exemple SQL :

```
CREATE TABLE INSCRIPTION (
noEtudiant INT,
idCours INT,
note DECIMAL(4,2),
PRIMARY KEY(noEtudiant,idCours),
FOREIGN KEY(noEtudiant) REFERENCES ETUDIANT(noEtudiant),
FOREIGN KEY(idCours) REFERENCES COURS(idCours)
);
```

Transaction (cas d'étude)

- Scénario : Inscrire un étudiant à un cours et enregistrer la note.
- Exigence : opération atomique (tout ou rien).
- Exemple SQL :

```
BEGIN;
INSERT INTO INSCRIPTION(...) VALUES (...);
UPDATE INSCRIPTION SET note = 12 WHERE ...;
COMMIT;
```

- En cas d'échec entre les opérations, on effectue un `ROLLBACK`.

Bilan du cas d'étude

- Nous avons illustré chaque étape : MCD (conceptuel), MLD (logique) et requêtes SQL.
- Le cas concret (université) montre comment modéliser les données et manipuler la base.
- Chaque partie du cours est accompagnée d'exemples concrets (extraits de MCD, scripts SQL, cas d'usage).
- Compétences visées : créer un MCD, le traduire en MLD, définir les relations, écrire des requêtes.

Points clés du cours

- Les systèmes d'information structurent l'organisation des données en entreprise.
- Le processus de modélisation passe du MCD au MLD, puis à l'implémentation (MPD).
- Le langage SQL permet de requêter et modifier les données dans une base relationnelle.
- Les outils (PowerAMC) facilitent la conception en automatisant le passage du conceptuel au SQL.

Perspectives

- Approfondir les normalisations (3NF, BCNF) et les techniques de dénormalisation.
- Découvrir les SGBD avancés (stockage, index, transactions distribuées).
- Explorer les nouvelles tendances (NoSQL, Big Data, Data Warehouse).
- Prochain cours : conception physique (MPD) et optimisation des requêtes.

Exercices suggérés

- Dessiner le MCD d'une bibliothèque (entités : Livre, Auteur, Emprunteur, Prêt).
- Écrire une requête SQL pour lister les emprunteurs d'un livre donné.
- Identifier les dépendances fonctionnelles du schéma de l'exercice précédent.

Fin

Questions ? Commentaires ?