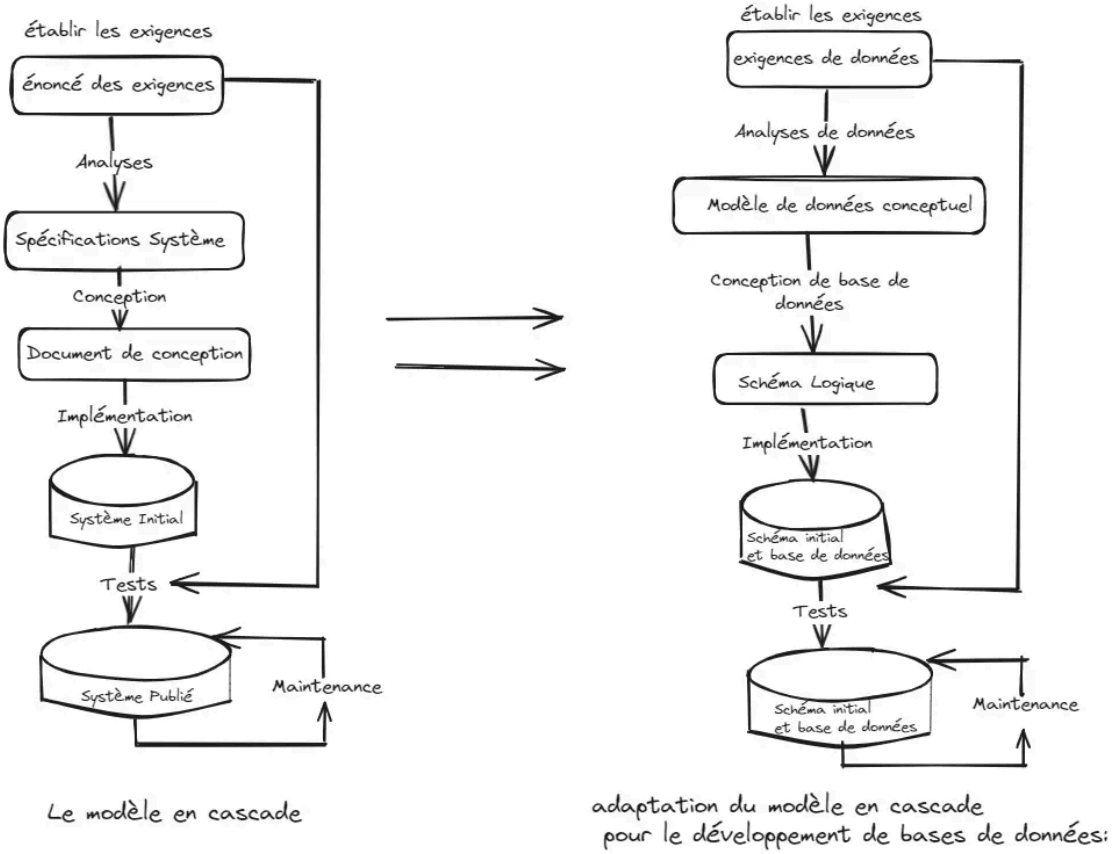
SIBD cours 6 -Développement des bases de données - processus, dépendances fonctionnelles et normalisation

1- Processus de développement de BD

- Parmi les méthodes de développement de système d'information, nous avons abordé la méthode de cycle de vie de développement logiciel.
- Cette méthodologie peut être modélisée en utilisant le model Waterfall (Modèle en Cascade).
- Le modèle en cascade illustre le processus comme une séquence stricte d'étapes où la sortie d'une étape est l'entrée de la suivante et où toute une étape doit être terminée avant de passer à la suivante.
- Ce modèle en cascade peut être adaptée pour le développement de bases de données:



Le modèle en cascade

adaptation du modèle en cascade pour le développement de bases de données:

- Pour utiliser ce modèle, on suppose que:
  - On peut séparer la spécification et la création d'un schéma pour définir les données dans une base de données - des processus utilisateur qui utilisent la base de données.
  - L'architecture à trois schémas peut servir de base pour distinguer les activités associées à un schéma (Conceptuel, Logique, et Physique).
  - Les contraintes peuvent être représentées une fois dans la base de données plutôt que dans chaque processus utilisateur qui utilise les données.

- Les étapes peuvent être résumées comme suit:

1-1 Collecte des exigences

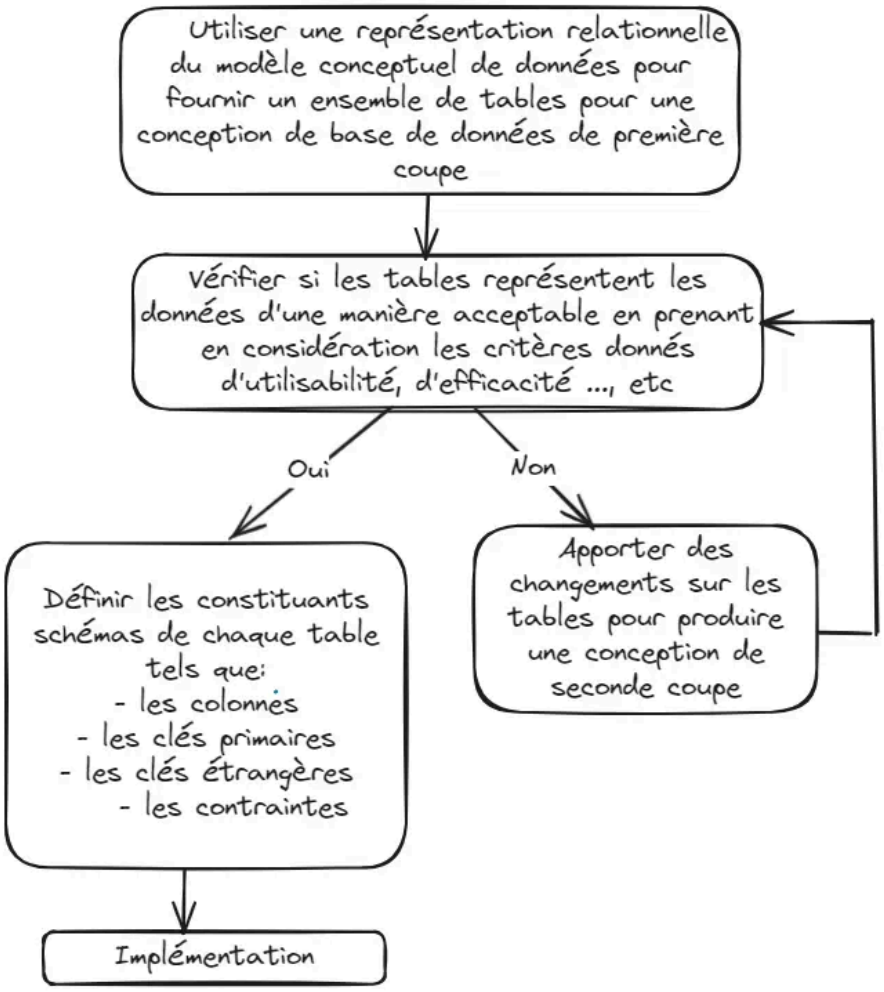
- Les concepteurs de bases de données doivent interviewer les utilisateurs pour comprendre leurs besoins.
- Le résultat est un document qui comprend les exigences détaillées fournies par les utilisateurs.

1-2 Analyse des données

- L'objectif est d'obtenir une description détaillée des données qui répondront aux exigences des utilisateurs.
- La description doit traiter à la fois les propriétés des données et leur utilisation.
- cette étape va produire le modèle de conceptuel de données.
- le modèle conceptuel de données s’intéresse à la signification et à la structure des données, mais pas aux détails affectant la manière dont elles sont mises en œuvre.

1-3 Conception logique

- Il est possible d'utiliser une représentation relationnelle du modèle de données conceptuel comme entrée dans le processus de conception logique.
- Le résultat de cette étape est une spécification relationnelle détaillée, le schéma logique, de toutes les tables et contraintes nécessaires pour satisfaire la description des données dans le modèle de données conceptuel.



1-4 Mise en œuvre (Implémentation) et Réalisation du schéma:

- Construit une base de données selon la spécification d'un schéma logique.
- La mise en œuvre est influencée par le choix du SGBD, des outils de base de données et de l'environnement d'exploitation.
- Création de la base de données selon le schéma logique. Peut se faire:
  - Soit en utilisant SQL:
    - Utilisation de SQL pour créer des tables et des contraintes
    - Enregistrement des instructions SQL dans un fichier texte
  - Soit en utilisant un outil de base de données comme SQL Server Management Studio ou Microsoft Access

1-5 Remplissage de la base de données

- Deux approches pour peupler les tables de base de données :
  - Utiliser des données existantes
  - Utiliser des applications utilisateur développées pour la base de données
- Sources de données existantes :=> utiliser les outils d'importation et d'exportation utilisés déja dans les SGBD
  - Autres bases de données

- Fichiers de données
- Conversion de documents papier en fichiers informatiques

2- Dépendance Fonctionnelle

2-1 Définition

- Une dépendance fonctionnelle (FD) est une relation entre deux attributs, généralement entre la clé primaire et d'autres attributs non clés au sein d'une table. Pour toute relation R, l'attribut Y dépend fonctionnellement de l'attribut X (généralement la clé primaire PK), si pour chaque instance valide de X, cette valeur de X, détermine de manière unique la valeur de Y
- La notation pour une FD est X → Y, où X est le déterminant et Y est le dépendant.
  - *exemple:*
    - SSN ———-> Nom, Adresse, Date_Naissance
    - ISBN ———> Titre

2-2 Règles d'inférence et Axiomes d'Armstrong de dépendance fonctionnelles

- Les axiomes d'Armstrong sont un ensemble de règles qui peuvent être utilisées pour inférer toutes les FD dans une table.
- Soit R(U) une relation définie sur l'ensemble d'attributs U. Les lettres X, Y, Z représenteront n'importe n'importe quel sous ensemble et l'union de deux ensembles d'attributs.
- Il existe 3 axiomes

2-2-1 Axiome de réflexivité :

- Si Y est un sous-ensemble de X, alors X → Y.

$$\text{If } Y \subseteq X, \text{ then } X \rightarrow Y$$

2-2-2 Axiome d'augmentation :

Si X → Y, alors XZ → YZ pour tout Z.

- Autrement dit:
  - Tout attribut non clé doit être entièrement dépendant de la clé primaire (PK).
- *Exemple:*
- Dan Dans l'exemple ci-dessous, EtudiantNom, Adresse, Cité, Province et CodePostal ne dépendent que de EtudiantNo. Et ne dépendent pas du cours.

EtudiantCours(EtudiantNo, Cours, EtudiantNom, Adresse, Cité, Province, CodePostal, Note, DateAccomplissement)
EtudiantNo, Cours ---> EtudiantNom, Adresse,Cité, Province, CodePostal, Note, DateAccomplissement

-Cette situation n'est pas souhaitable car chaque attribut non clé doit être entièrement dépendant de la PK. Dans ce cas, les informations sur les étudiants ne sont que partiellement dépendantes de la PK (EtudiantNo).

- Pour résoudre ce problème, nous devons diviser la table d'origine en deux comme suit :
  - **Table 1:** EtudiantNo, Cours, Note, DateAccomplissement
  - **Table 2:** EtudiantNo, EtudiantNom, Adresse, Cité, Province, CodePostal

2-2-3 Axiome de transitivité :

Si X → Y et Y → Z, alors X → Z.

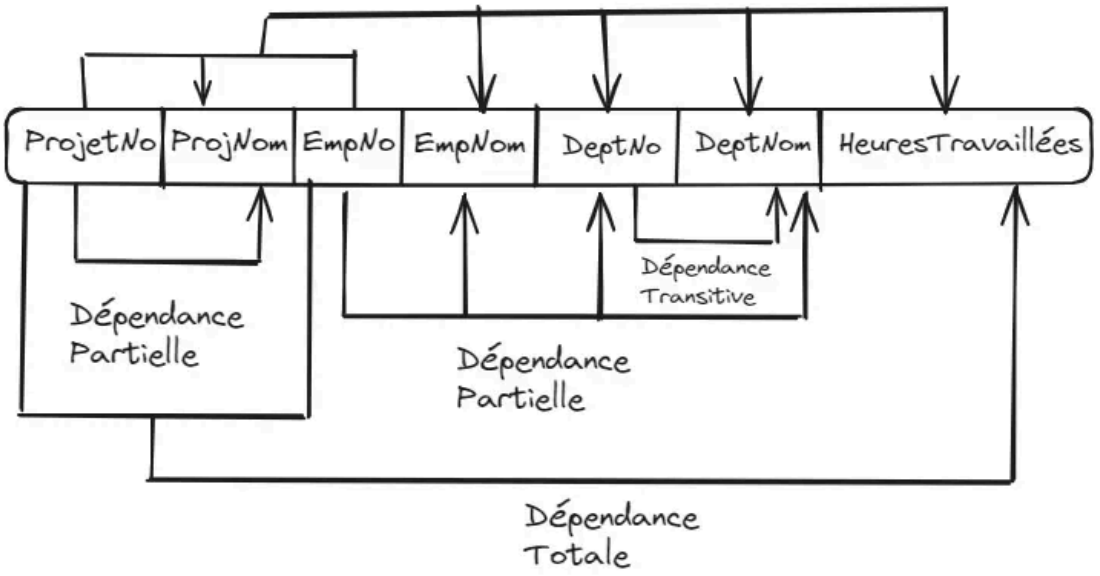
- *Exemple:*
  - La table ci-dessous contient des informations qui ne sont pas directement liées à l'étudiant:
    EtudiantNo —> EtudiantNom, Adresse, Cité, Province, CodePostal, ProgrammeID, ProgrammeNom
  - par exemple, ProgrammeID et ProgrammeNom devraient avoir leur propre table. ProgrammeNom ne dépend pas de EtudiantNo; il dépend de ProgrammeID.
  - Cette situation n'est pas souhaitable car un attribut non clé (ProgrammeNom) dépend d'un autre attribut non clé (ProgrammeID).
  - Pour résoudre ce problème, nous devons diviser cette table en deux : une pour contenir les informations sur l'étudiant et l'autre pour contenir les informations sur le programme.
    - Table 1 : EtudiantNo → EtudiantNom, Adresse, Cité, Province, CodePostal, ProgrammeID
    - Table 2 : ProgrammeID → ProgrammeNom
    - Cependant, nous devons quand même laisser une clé étrangère dans la table Etudiant afin de pouvoir identifier le programme dans lequel l'étudiant est inscrit.

2-2-4 Règles d'Union et de décomposition

- La règle d'union stipule que si X → Y et X → Z, alors X → YZ.
- La règle de décomposition stipule que si X → Y Z, alors X → Y et X → Z séparément.
- Lorsque deux tables sont liées par un attribut clé, elles peuvent être fusionnées en une seule table.
- Certains administrateurs de base de données (DBA) préfèrent conserver les tables séparées pour les raisons suivantes :
  - Chaque table décrit une entité différente, et les entités doivent être séparées.
  - Si la valeur de l'attribut en question est souvent nul, il n'est pas nécessaire de l'inclure dans la même table que celle de l'attribut associé.

2-3 Diagramme de dépendance

- Un diagramme de dépendance est une représentation graphique des FD dans une table



- ProjetNo and EmpNo, combinés, forment la clé primaire(PK).
- Dependences Partielles:
  - ProjectNo —> ProjNom
  - EmpNo —> EmpNom, DeptNo,
- Dépendance transitive:
  - DeptNo —> DeptNom
- Dépendance totale:
  - ProjectNo, EmpNo —> HeuresTravaillées

3- Normalisation

- La normalisation devrait faire partie du processus de conception de bases de données.
- La normalisation permet de réduire la redondance dans les données, ce qui peut améliorer les performances, l'intégrité et la maintenabilité de la base de données.
- Le résultat du processus de normalisation est la transformation des relations en formes normales (NF)
- Il existe six formes normales (NF), chacune présentant des exigences spécifiques en matière de dépendances fonctionnelles.
- Les formes normales les plus couramment utilisées sont la première forme normale (1NF), la deuxième forme normale (2NF) et la troisième forme normale (3NF).

3-1 Première forme normale (1NF)

- Une relation est en 1NF si elle ne comporte pas de groupes répétés.
- Un groupe répété est un ensemble d'attributs qui peuvent prendre plusieurs valeurs pour une même valeur d'un autre attribut.
- Pour normaliser une relation en 1NF, il faut supprimer les groupes répétés et créer de nouvelles relations.
- exemple:

- Etudiant_Note_Rapport (EtudiantNo, EtudiantNom, Spécialité, CoursNo, CoursNom, InstructeurNo, InstructeurNom, InstructeurEmplacemnt, Note)
- => Etudiant (EtudiantNo, EtudiantNom, Spécialité)
- EtudiantCours (EtudiantNo, CoursNo, CoursNom, InstructeurNo, InstructeurNom, InstructeurEmplacemnt, Note)

3-2 Deuxième forme normale (2NF)

- Une relation est en 2NF si elle est en 1NF et que tous les attributs non clés sont entièrement dépendants de la clé primaire.
- Une dépendance partielle est une dépendance dans laquelle un attribut non clé est dépendant d'un sous-ensemble de la clé primaire.
- Pour normaliser une relation en 2NF, il faut éliminer les dépendances partielles.
- exemple:
  - Etudiant (EtudiantNo, EtudiantNom, Spécialité)
  - CoursNote (EtudiantNo, CoursNo, Note)
  - CoursInstructeur (CoursNo, CoursNom, InstructeurNo, InstructeurNom, InstructeurEmplacemnt)

3-3 Troisième forme normale (3NF)

- Une relation est en 3NF si elle est en 2NF et que toutes les dépendances transitives sont éliminées.
- Une dépendance transitive est une dépendance dans laquelle un attribut non clé est dépendant d'un autre attribut non clé.
- Pour normaliser une relation en 3NF, il faut éliminer les dépendances transitives.
- exemple:
  - Etudiant (EtudiantNo, EtudiantNom, Spécialité)
  - CoursNote (EtudiantNo, CoursNo, Note)
  - Cours (CoursNo, CoursNom, InstructeurNo)
  - Instructeur (InstructeurNo, InstructeurNom, InstructeurEmplacemnt)

4- Développement de diagrammes Entité Association

- Consignez toutes les entités découvertes lors de la phase de collecte d'informations.
- Documentez tous les attributs appartenant à chaque entité. Sélectionnez les clés candidates et primaires. Assurez-vous que tous les attributs non clés de chaque entité sont entièrement dépendant de la clé primaire.
- Développez un diagramme ER initial et examinez-le avec le personnel approprié.
- Créez de nouvelles entités pour les attributs multivalents et les groupes répétés.
- Vérifiez le modèle ER en normalisant les tables.

Références Bibliographiques

- Watt, Adrienne, and Nelson Eng. *Database design*. 2nd Edition. BCcampus, 2014