المدرسة الوطنية العليا للذكاء الاصطناعي وعلوم المعطيات - تارودانت
ÉCOLE NATIONALE SUPÉRIEURE DE L'INTELLIGENCE ARTIFICIELLE
ET SCIENCES DES DONNÉES - TAROUDANT

Systèmes d'Information et Bases de Données

Chapitre 2 : La structuration d'une BD

Pr. Sara El-Ateif, 19 Février 2025

Introduction

La structuration d'une BD

poser les bases d'une architecture solide

La structuration d'une base de données est une étape fondamentale dans la conception d'un système d'information performant et cohérent. Un bon modèle de données garantit l'intégrité, la cohérence et la facilité d'accès aux informations.

Ce chapitre explore les étapes clés de ce processus, du recensement des données à la normalisation des relations.

Les objectifs de la structuration

- La structuration vise plusieurs objectifs :
  - **Minimiser la redondance :** Éviter la répétition des informations (gain de place, cohérence). **Exemple :** stocker l'adresse d'un client une seule fois.
  - **Faciliter les mises à jour :** une modification d'information ne doit être effectuée qu'à un seul endroit.
  - **Simplifier les requêtes : r**etrouver facilement l'information.
  - **Assurer l'intégrité :** garantir la validité et la cohérence des données (ex : un âge ne peut être négatif).
  - **Améliorer les performances :** accès rapide aux données.

Le recensement des données

Connaître son terrain

- Cette étape consiste à identifier toutes les données nécessaires au système d'information. Il est important d'être exhaustif et précis.
  - **Sources de données** : documentation existante, entrevues avec les utilisateurs, formulaires, etc.

Exemple : Pour une BD de gestion de bibliothèque, recenser les informations sur les livres (titre, auteur, ISBN), les adhérents (nom, adresse), les emprunts, etc.

Normalisation

Problématique

- Combiner des informations relatives à plusieurs entités au sein d'une même relation (table) engendre généralement des redondances de données. Ces redondances sont à l'origine d'anomalies qui compromettent l'intégrité et la cohérence de la base de données. On distingue notamment trois types d'anomalies :
  - Anomalies d'**insertion** : L'impossibilité d'ajouter certaines informations sans en ajouter d'autres, non directement liées.
  - Anomalies de **mise à jour** : La nécessité de modifier une information à plusieurs endroits, ce qui peut conduire à des incohérences si la mise à jour n'est pas effectuée partout.
  - Anomalies de **suppression** : La perte d'informations non directement liées à l'information que l'on souhaite supprimer.

Exemple

isbn	titre	éditeur	pays
2-212-09283-0	Bases de données – objet et relationnel	Eyrolles	France
2-7117-8645-5	Fondements des bases de données	Vuibert	USA
0-201-70872-8	Databases and Transaction Processing	Addison Wesley	USA
2-212-09069-2	Internet/Intranet et bases de données	Eyrolles	France

- Cette relation qui décrit des livres et leurs éditeurs, contient des redondances provoquant les anomalies suivantes:
  - **insertion** :il n'est pas possible d'insérer un livre sans connaître son éditeur. Attribuer une valeur nulle aux attributs éditeur et pays violerait l'intégrité d'entité : un attribut appartenant à la clé, ne doit pas avoir de valeur nulle.
  - **mise à jour** : si un éditeur change de pays, il faut modifier ce pays pour chacun des livres qu'il a édités.
  - **suppression** : si l'unique livre publié par un éditeur est supprimé, l'information sur cet éditeur est perdue. Attribuer une valeur nulle aux attributs isbn et titre, violerait l'intégrité d'entité.

Normalisation

Définition

- La normalisation consiste en des **méthodes de décomposition** des relations. Cette décomposition repose sur l'**analyse de dépendances** entre attributs qui sont à l'origine de phénomènes de redondance.
- Un **schéma relationnel normalisé** doit répondre aux exigences minimales suivantes:
  - **Non redondance:** un attribut n'appartient qu'à une seule relation, donc à une seule table, à moins qu'il n'agisse comme clé étrangère pour assurer l'association avec une autre table ;
  - **Cohérence:** les attributs qui décrivent le même objet appartiennent à la même table et dépendent chacun fonctionnellement et totalement de la clé primaire de la table.

Un attribut b dépend fonctionnellement d'un attribut a si à une valeur de a correspond au plus une valeur de b. La dépendance fonctionnelle est notée a -> b.

Le membre droit de l'écriture s'appelle le dépendant, le membre gauche s'appelle le déterminant.

Dépendance fonctionnelle

Comprendre les liens entre les données

- Une dépendance fonctionnelle décrit un lien de détermination entre deux attributs ou ensembles d'attributs.
- On dit qu'un attribut B dépend fonctionnellement d'un attribut A **(noté A → B) si et seulement si chaque valeur de A est associée à une et une seule valeur de B**.
- A est appelé le déterminant, et B le déterminé (ou dépendant).
  - *Analogie : Imaginez une machine à fabriquer des pièces. Chaque réglage de la machine (A) produit un type de pièce spécifique (B). Modifier le réglage change le type de pièce produite.*

Un attribut b dépend fonctionnellement d'un attribut a si à une valeur de a correspond au plus une valeur de b. La dépendance fonctionnelle est notée a -> b.

Le membre droit de l'écriture s'appelle le dépendant, le membre gauche s'appelle le déterminant.

Exemples
⍰ Supposons les exemples suivants, qui concernent des pilotes ayant un numéro, un nom, une fonction (copilote, commandant, instructeur…), et un nombre d'heures de vol pour un jour particulier:
numPilote,jour->nbHeuresVol est une DF, car à un couple (numPilote, jour) correspond au plus un nombre d'heures de vol ;
numPilote -> nomPilote,fonction est équivalente aux écritures numPilote -> nomPilote et numPilote -> fonction qui sont deux DF. En conséquence numPilote -> nomPilote, fonction est une DF ;
nomPilote -> fonction est une DF s'il n'y a pas d'homonymes dans la population des pilotes enregistrés dans la base de données. Dans le cas contraire, ce n'est pas une DF, car à un nom de pilote peuvent correspondre plusieurs fonctions ;
fonction -> nomPilote n'est pas une DF, car à une fonction donnée correspondent éventuellement plusieurs pilotes.

Dépendance fonctionnelle

Comprendre les liens entre les données

- ***Exemple :*** Considérons une table **Produits(CodeProduit, NomProduit, Prix)** où **CodeProduit** est la clé primaire.
  - **CodeProduit → NomProduit** : Chaque code produit identifie un seul nom de produit.
  - **CodeProduit → Prix** : Chaque code produit a un seul prix.
  - **NomProduit → Prix** : Ceci est faux en général, car deux produits différents pourraient avoir le même nom mais des prix différents.

Un attribut b dépend fonctionnellement d'un attribut a si à une valeur de a correspond au plus une valeur de b. La dépendance fonctionnelle est notée a -> b.

Le membre droit de l'écriture s'appelle le dépendant, le membre gauche s'appelle le déterminant.

Dépendance fonctionnelle

Comprendre les liens entre les données

- **A ne détermine pas forcément B dans le monde réel, seulement dans le contexte du modèle de données que l'on crée.**
- On pourrait imaginer un produit dont le prix change au cours du temps. Dans ce cas, pour un même CodeProduit, on aurait plusieurs prix.
- Mais si on décide de ne s'intéresser qu'au prix actuel du produit, alors la DF **CodeProduit → Prix** est valide dans le contexte de notre modèle.

Un attribut b dépend fonctionnellement d’un attribut a si à une valeur de a correspond au plus une valeur de b. La dépendance fonctionnelle est notée a -> b.

Le membre droit de l’écriture s’appelle le dépendant, le membre gauche s’appelle le déterminant.

Dépendance fonctionnelle

Dépendances Fonctionnelles Triviales

- Une **DF est triviale** si l'attribut déterminé (B) est inclus dans le déterminant (A). En d'autres termes, A → B est triviale si B est un sous-ensemble de A.
  - **Exemple :**
    - **(CodeProduit, NomProduit) → CodeProduit**
    - **NumEmployé → NumEmployé**

Un attribut b dépend fonctionnellement d’un attribut a si à une valeur de a correspond au plus une valeur de b. La dépendance fonctionnelle est notée a -> b.

Le membre droit de l’écriture s’appelle le dépendant, le membre gauche s’appelle le déterminant.

Dépendance fonctionnelle

Dépendances Fonctionnelles Partielles

- Une **DF est partielle** si **une partie seulement de l'attribut déterminant (qui est composé) est suffisante pour déterminer l'attribut déterminé**. Ceci se produit généralement lorsque la clé primaire est composée de plusieurs attributs.
  - **Exemple :** Dans la table **Inscriptions(NumEtudiant, CodeCours, DateInscription, NomEtudiant)** avec une clé primaire **(NumEtudiant, CodeCours)**, si on a la DF : **(NumEtudiant, CodeCours) → NomEtudiant**, alors la DF est partielle, car **NumEtudiant → NomEtudiant** est aussi vrai.

En résumé, une dépendance fonctionnelle représente une contrainte forte dans un modèle de données : elle indique qu'un attribut détermine de manière unique un autre attribut. Il est important de bien comprendre ce concept pour concevoir des schémas relationnels normalisés et éviter les redondances.

DF élémentaire

Une DF a,b → c est **élémentaire** si ni a → c, ni b → c ne sont des DF.

OU

Une DF **A → B est élémentaire** si :

- A est un attribut unique (ou atomique) et non pas un ensemble d'attributs.
- Aucun sous-ensemble propre de A ne détermine B. Autrement dit, A est minimal.

La définition d'une dépendance fonctionnelle (DF) élémentaire est la suivante : Une DF X → Y est élémentaire si :

Y est un attribut unique (ou atomique) et non pas un ensemble d'attributs.

Aucun sous-ensemble propre de X ne détermine Y. Autrement dit, X est minimal.

En termes plus simples, une DF élémentaire est la plus petite DF possible : on ne peut ni réduire le déterminant ni augmenter le déterminé sans perdre la validité de la dépendance.

Exemples de DF élémentaires :

CodeProduit → NomProduit: CodeProduit détermine NomProduit. On ne peut pas réduire CodeProduit, et on ne peut pas ajouter d'autres attributs à NomProduit sans que la DF ne soit plus vraie (en supposant que CodeProduit est une clé primaire).

ISBN → Titre: ISBN détermine Titre. On ne peut pas réduire ISBN, et on ne peut pas ajouter d'autres attributs à Titre (en supposant que ISBN est une clé).

NuméroSécuritéSociale → DateNaissance: Dans ce cas, on suppose qu'un numéro de sécurité sociale est associé à une seule date de naissance. On ne peut réduire NuméroSécuritéSociale, et on ne peut pas ajouter d'autres attributs à DateNaissance.

Exemples de DF non élémentaires et comment les rendre élémentaires :

NuméroClient, DateCommande → MontantTotal, AdresseLivraison: Non élémentaire, car Y n’est pas atomique.

Solution: Décomposer en deux DF élémentaires :

NuméroClient, DateCommande → MontantTotal

NuméroClient, DateCommande → AdresseLivraison

NuméroEmployé, NomEmployé → Département: Non élémentaire, car X n’est pas minimal (si on suppose que NuméroEmployé est une clé).

Solution : Simplifier en : NuméroEmployé → Département

Immatriculation → Marque, Modèle, Couleur: Non élémentaire car Y n'est pas atomique.

Solution: Décomposer en trois DF élémentaires :

Immatriculation → Marque

Immatriculation → Modèle

Immatriculation → Couleur

En résumé, pour vérifier si une DF est élémentaire :

Assurez-vous que le côté droit de la flèche (le déterminé) ne contient qu'un seul attribut.

Assurez-vous qu’aucune partie du côté gauche de la flèche (le déterminant) ne peut, à elle seule, déterminer l'attribut à droite.

DF élémentaire

Exemples

- la dépendance fonctionnelle **numPilote,jour → nbHeuresVol** **est élémentaire**, car **numPilote → nbHeuresVol** n'est pas une DF (un pilote vole différents jours, donc pour un pilote donné, il existe plusieurs nombres d'heures de vol), pas plus que **jour → nbHeuresVol** (à un jour donné, plusieurs vols sont programmés) ;
- la dépendance fonctionnelle **numPilote,nomPilote → fonction** **n'est pas élémentaire**, car le numéro du pilote suffit pour retrouver sa fonction (**numPilote → fonction** est une DF).
- **ISBN → Titre:** **est élémentaire** car ISBN détermine Titre. On ne peut pas réduire ISBN, et on ne peut pas ajouter d'autres attributs à Titre (en supposant que ISBN est une clé).

Cet exemple illustre le concept de dépendance fonctionnelle (DF) élémentaire en montrant qu'une DF peut être valide uniquement lorsqu'on combine plusieurs attributs déterminants. Décomposons l'explication :

numPilote, jour → nbHeuresVol: Cette DF signifie que si on connaît à la fois le numéro du pilote (numPilote) et le jour (jour), on peut déterminer de manière unique le nombre d'heures de vol (nbHeuresVol) de ce pilote pour ce jour spécifique.

numPilote → nbHeuresVol n'est pas une DF : Si on connaît seulement le numéro du pilote, on ne peut pas déterminer de manière unique son nombre d'heures de vol. En effet, un pilote peut voler un nombre d'heures différent chaque jour. Un même pilote peut accumuler 5 heures de vol un jour et 2 heures un autre jour.

jour → nbHeuresVol n'est pas une DF : Si on connaît seulement le jour, on ne peut pas déterminer un nombre d'heures de vol unique. Plusieurs pilotes peuvent voler ce jour-là, chacun avec un nombre d'heures potentiellement différent. Un jour donné, il peut y avoir plusieurs vols, chacun avec une durée différente.

Conclusion : La DF numPilote, jour → nbHeuresVol est élémentaire: Puisqu'aucun des attributs numPilote ou jour ne peut, à lui seul, déterminer nbHeuresVol, leur combinaison est minimale pour déterminer nbHeuresVol. C'est la définition d'une DF élémentaire : on ne peut retirer aucun attribut du déterminant sans invalider la dépendance fonctionnelle.

En résumé :

Cet exemple souligne qu'il est parfois nécessaire de combiner plusieurs attributs pour obtenir une dépendance fonctionnelle valide. L'attribut composé (numPilote, jour) est le déterminant minimal pour nbHeuresVol, ce qui rend la DF élémentaire. Si on essayait de n'utiliser que numPilote ou que jour, on perdrait l'unicité de la

détermination, et la DF ne serait plus valide.

DF directe

Une DF a → c **est directe** si elle n'est pas déduite par transitivité, c'est-à-dire s'il n'existe pas de DF a → b et b → c.

Autrement dit, il n'y a pas d'intermédiaire entre X et Y. X détermine *directement* Y, et non pas via un autre attribut.

Parlons des dépendances fonctionnelles directes.

Une dépendance fonctionnelle X → Y est directe si et seulement si il n'existe aucun ensemble d'attributs Z tel que :

X → Z

Z → Y

Z n'est pas X, et Z n'est pas Y.

Autrement dit, il n'y a pas d'intermédiaire entre X et Y. X détermine directement Y, et non pas via un autre attribut.

Exemples de DF directes:

Numéro de sécurité sociale → Date de naissance: Le numéro de sécurité sociale détermine directement la date de naissance. Il n'y a pas d'autre attribut intermédiaire.

Code produit → Prix (actuel): Le code produit détermine directement le prix actuel.

Immatriculation → Couleur du véhicule: L'immatriculation détermine directement la couleur du véhicule.

Exemples de DF indirectes et comment les transformer en directes:

Numéro de commande → Ville du client: Indirecte. Le numéro de commande détermine l'identité du client, qui à son tour détermine l'adresse du client, et donc la ville. On pourrait décomposer cela en :

Numéro de commande → NumClient

NumClient → AdresseClient

AdresseClient → VilleClient
Ici, les DF sont directes. On pourrait aussi faire Numéro de commande → AdresseClient mais ce ne serait pas direct car le numéro de client est impliqué dans la relation.

Numéro d'étudiant → Nom du professeur principal : Indirecte. Le numéro d'étudiant détermine la classe, qui à son tour détermine le professeur principal. Décomposer en deux DF directes :

Numéro d'étudiant → CodeClasse

CodeClasse → NumProfesseurPrincipal

ISBN → Ville de l’éditeur : Indirecte. L'ISBN détermine l’éditeur, qui détermine la ville de l'éditeur. Décomposer :

ISBN → NomEditeur

NomEditeur → VilleEditeur

Comment identifier une DF indirecte ?

Cherchez un "chemin" logique entre les attributs. Si vous pouvez passer par un ou plusieurs autres attributs pour aller de X à Y, alors X → Y est probablement indirecte. La normalisation vise notamment à éliminer ces dépendances transitives (3FN).

En résumé, une DF directe est un lien de détermination sans intermédiaire entre deux attributs. Elle est fondamentale pour la troisième forme normale (3FN).

DF directe

Exemples

- Considérons l'exemple d'un avion avec les attributs suivants : (immat : numéro d'immatriculation d'un avion ; typeAvion : type de l'avion; nomConst : nom du constructeur de l'avion).
  - La dépendance **immat → nomConst** n'est pas une DF directe.
  - La dépendance **immat → typeAvion** est une DF directe.
  - La dépendance **typeAvion → nomConst** est une DF directe.

Propriétés de DF

- Les propriétés suivantes sont appelées «axiomes d'Armstrong »
  - **Réflexivité :** si b est un sous-ensemble de a alors a → b est une DF.
  - **Augmentation :** si a → b est une DF, alors a,c → b,c est une DF.
  - **Transitivité :** si a → b et b → c sont des DF, alors a → c est une DF.
  - **Union :** si a → b et a → c sont des DF, alors a → b,c est une DF.
  - **Pseudo-transitivité :** si a → b et b,c → d sont des DF, alors a,c → d est une DF.
  - **Décomposition :** si a → b,c est une DF, alors a → b et a → c sont des DF.

Les formes normales (FN)

Les étapes de la normalisation

Les classifications formelles utilisées pour décrire le niveau de normalisation d'une base de données relationnelle sont appelées les formes normales (FN).

- Il existe huit formes normales. Les quatre premières sont les plus pratiques et sont à connaître:
  - la 1ère forme normale **(1FN)**,
  - la 2ème forme normale **(2FN)**,
  - la 3ème forme normale **(3FN),**
  - la forme normale de Boyce-Codd qui est la plus aboutie **(FNBC).**

Les formes normales (FN)

1FN

Une relation est en **1FN si tous ses attributs ont une valeur atomique**.

pas en 1FN

isbn	auteur
2-212-09283-0	Gardarin
2-7117-8645-5	Abiteboul, Hull, Vianu

en 1FN

isbn	auteur
2-212-09283-0	Gardarin
2-7117-8645-5	Abiteboul
2-7117-8645-5	Hull
2-7117-8645-5	Vianu

Une fois on passe à la 1FN il faut designer la clé primaire !

Première forme normale (1NF)

Une table dans une base de données relationnelle répond à la première forme normale (1NF) lorsqu'elle remplit les critères suivants :

- Toutes les données sont atomiques
- Toutes les colonnes du tableau contiennent des valeurs identiques

Un ensemble de données est considéré comme atomique si chaque élément d'information est affecté à un champ de données distinct.

La phrase "Un ensemble de données est considéré comme atomique si chaque élément d'information est affecté à un champ de données distinct" décrit le concept fondamental de la première forme normale (1FN) dans le cadre de la normalisation des bases de données relationnelles. Décomposons cette explication :

Atomique : Dans ce contexte, "atomique" signifie indivisible. Un élément d'information atomique ne peut pas être décomposé en sous-éléments plus petits qui auraient une signification propre dans le modèle de données.

Champ de données distinct : Chaque élément d'information atomique doit être stocké dans une colonne (attribut) séparée de la table.

Exemple illustratif : Imaginez une table Clients avec un attribut Adresse contenant des valeurs comme "10 rue de la Paix, 75000 Paris, France". Cet attribut n'est pas atomique, car il contient plusieurs informations : rue, numéro, ville, code postal, et pays. Pour respecter la 1FN, il faudrait décomposer cet attribut en plusieurs attributs distincts : Rue, Numéro, Ville, CodePostal, et Pays.

Pourquoi l'atomicité est-elle importante ?

Éviter les redondances : Stocker plusieurs informations dans un même champ peut entraîner la répétition des mêmes données à plusieurs endroits de la table, gaspillant ainsi de l'espace disque et augmentant le risque d'incohérences.

Faciliter les requêtes : Il est plus facile d'interroger et de filtrer les données lorsque chaque information est stockée séparément. Par exemple, il est plus simple de rechercher tous les clients habitant à Paris si la ville est stockée dans un champ Ville dédié.

Simplifier les mises à jour: Modifier une information atomique est plus simple et plus sûr : il suffit de modifier la valeur dans la colonne correspondante, sans avoir à manipuler d'autres informations dans le même champ.

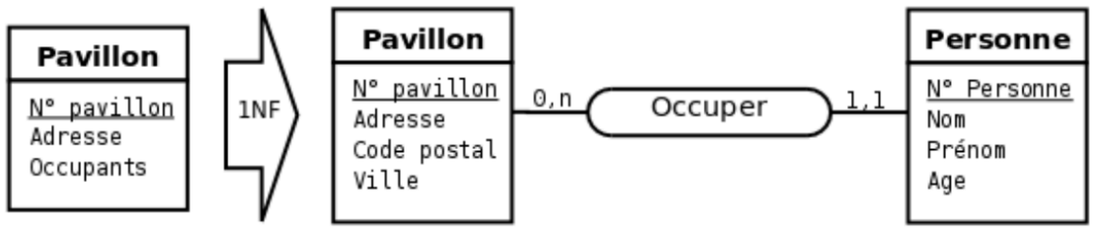
En résumé, la 1FN impose l'atomicité des attributs, ce qui signifie que chaque information élémentaire doit être stockée dans sa propre colonne. C’est la première étape du processus de normalisation, et une condition préalable aux formes normales supérieures (2FN, 3FN, etc.).

⚠️ Note

Une valeur est considérée comme atomique selon le contexte de son utilisation. S’il n’est pas nécessaire de séparer le prénom et le nom de famille, le nom complet d’une personne peut être considéré comme atomique. Mais dans la pratique, il est préférable de diviser les valeurs en plusieurs parties en unités aussi petites que possible.

Les formes normales (FN)

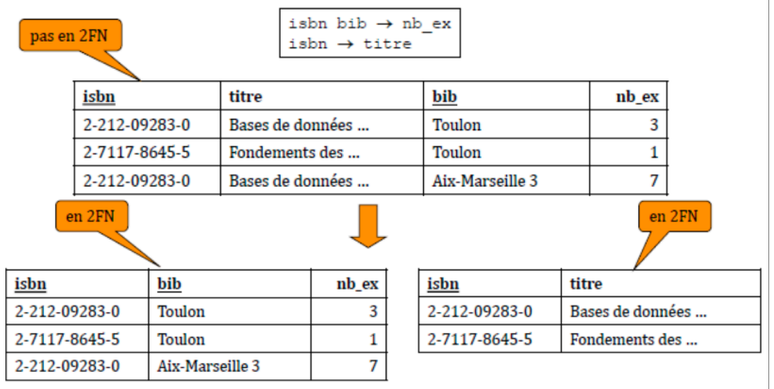
1FN



Les formes normales (FN)

2FN

Une relation est en 2FN si elle est en 1FN et si c**haque attribut non clé dépend** totalement et non partiellement **de la clé primaire** (c-à-d que la dépendance est élémentaire).



isbn	titre	bib	nb_ex
2-212-09283-0	Bases de données ...	Toulon	3
2-7117-8645-5	Fondements des ...	Toulon	1
2-212-09283-0	Bases de données ...	Aix-Marseille 3	7

isbn	bib	nb_ex
2-212-09283-0	Toulon	3
2-7117-8645-5	Toulon	1
2-212-09283-0	Aix-Marseille 3	7

isbn	titre
2-212-09283-0	Bases de données ...
2-7117-8645-5	Fondements des ...

Pour convertir une table de base de données en la deuxième forme normale, vous devez non seulement déterminer la clé primaire et tous les attributs non clés, mais également leur relation les uns aux autres. Suivez ces étapes :

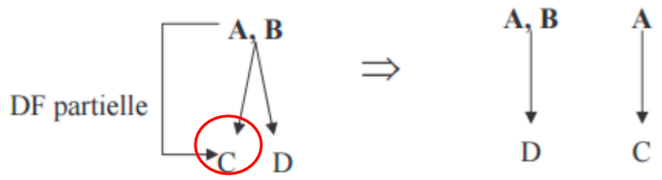
- Vérifiez si tous les attributs non clés dépendent entièrement de la fonction de la clé primaire. Une telle dépendance n’existe que si tous les attributs clés primaires sont nécessaires pour identifier de façon unique l’attribut non clé. Cela signifie également que les tables avec clés primaires monobloc correspondent automatiquement à la deuxième forme normale si toutes les conditions préalables pour la première forme normale sont remplies.
- Déplacez tous les attributs non clés qui ne dépendent pas entièrement fonctionnellement de la clé primaire complète dans des tables séparées.

Les formes normales (FN)

2FN

Comment normaliser en 2FN:

- Isoler la DF partielle dans une nouvelle relation
- Eliminer l'attribut cible (le dépendant) de DF de la relation initiale

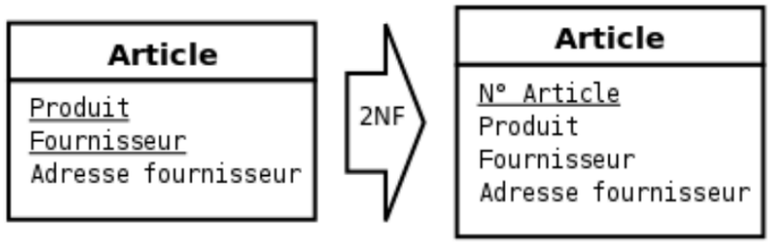


Remarque: Une relation en 1FN dont la clé est atomique (mono-attribut) est forcément en 2FN

Les formes normales (FN)

2FN

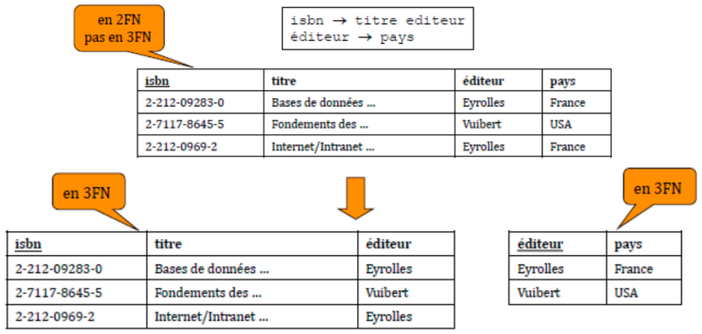
Si la **clé est atomique**, alors on est sûr que la relation est en deuxième forme normale !



Les formes normales (FN)

3FN

Une relation est en 3FN si elle respecte la 2FN et si les **dépendances fonctionnelles** entre la clé primaire et les autres attributs sont **directes**.



Troisième forme normale (3NF)

Si une table doit être convertie à la troisième forme normale, toutes les conditions préalables de la première et de la deuxième forme normale doivent être remplies ainsi que les conditions suivantes :

Aucun attribut non clé ne peut dépendre de façon transitoire d’un candidat clé.
Une dépendance transitive se produit lorsqu’un attribut non clé dépend d’un autre attribut non clé et donc indirectement de son candidat clé.

⚠️ Note

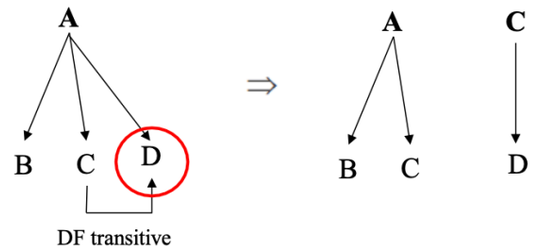
Les doublons dans les bases de données relationnelles sont souvent inévitables. En examinant l’exemple au fur et à mesure que la conversion se déroule, il est évident que la liaison des tables de base de données par des clés étrangères peut être liée à des redondances. Il s’agit des redondances clés.

Les formes normales (FN)

3FN

Comment normaliser en 3FN:

- Isoler la DF transitive dans une nouvelle relation
- Eliminer la cible de DF transitive de la relation initiale

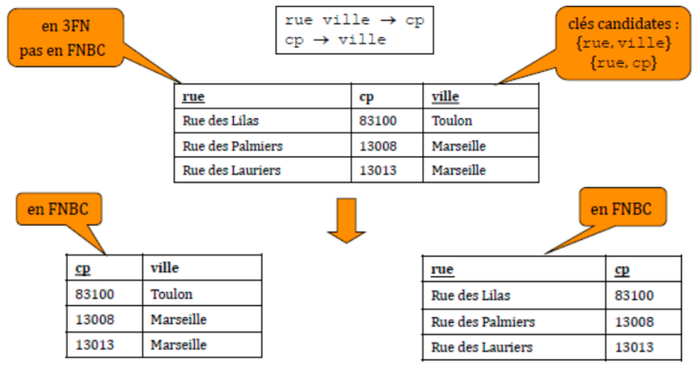


Même si la normalisation des bases de données nécessite un effort de programmation plus important, la troisième forme normale 3NF, est généralement considérée comme la norme pour les formules de base de données relationnelles, et n’est déviée que dans des cas exceptionnels. Par exemple, les bases de données conformes à la troisième forme normale sont parfois dénormalisées sous la deuxième forme normale. C’est parce que les jointures entre plusieurs tables prennent beaucoup de temps pour de très grandes bases de données. La dénormalisation réduit le nombre de tables et avec elle le temps de requête.

Les formes normales (FN)

FNBC

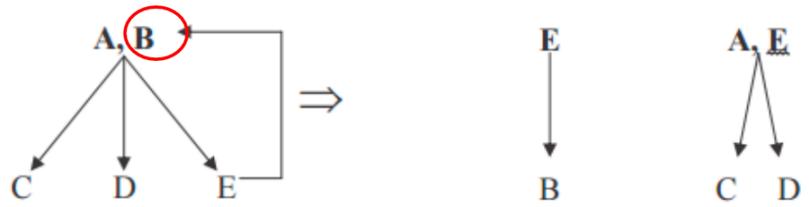
Une relation est en forme normale de Boyce-Codd si elle est en 3FN et que le **seul déterminant** (membre gauche des DF) existant dans la relation **est la clé primaire**.



Les formes normales (FN)
FNBC

Comment normaliser en FNBC:

- Isoler la DF problématique dans une nouvelle relation
- Eliminer la cible de DF problématique et la remplacer par sa source dans la relation initiale.



Les formes normales (FN)

Exemple

Considérons l'exemple des assurances de véhicules. Décomposer cette relation de manière à obtenir un schéma normalisé.

Ventes

num	nom	{Vehicules}					
		nimm	type	ncons	nomCons	dateAchat	prixAchat
1	Soutou	6748-XW-31	320 i	1	BMW	12-05-1999	62 500
		734-AJH-31	1200 GSF	3	SUZUKI	19-07-2000	46680
		358-ALZ-31	320 Ci	1	BMW	25-01-2001	213 970
2	Bidal	955-NEH-75	Scenic	2	RENAULT	16-01-1995	105 000
		52-AIM-31	Beetle	4	VOLKSWAGEN	16-01-2002	109 500

FN Exemple

1FN

En partant de l'hypothèse qu'un assuré puisse vendre puis racheter sa voiture, la clé de cette relation sera la concaténation des attributs **num, nimm et dateAchat**.

Ventes1

num	nom	**nimm**	type	ncons	nomCons	**dateAchat**	prixAchat
1	Soutou	6748-XW-31	320 i	1	BMW	12-05-1999	62 500
1	Soutou	734-AJH-31	1200 GSF	3	SUZUKI	19-07-2000	46680
1	Soutou	358-ALZ-31	320 Ci	1	BMW	25-01-2001	213 970
2	Bidal	955-NEH-75	Scenic	2	RENAULT	16-01-1995	105 000
2	Bidal	52-AIM-31	Beetle	4	VOLKSWAGEN	16-01-2002	109 500

FN Exemple

2FN

La question à se poser: « **De quoi dépend élémentairement chaque attribut de la relation ?** ».

Ventes1

num	nom	nimm	type	ncons	nomCons	dateAchat	prixAchat
1	Soutou	6748-XW-31	320 i	1	BMW	12-05-1999	62 500
1	Soutou	734-AJH-31	1200 GSF	3	SUZUKI	19-07-2000	46680
1	Soutou	358-ALZ-31	320 Ci	1	BMW	25-01-2001	213 970
2	Bidal	955-NEH-75	Scenic	2	RENAULT	16-01-1995	105 000
2	Bidal	52-AIM-31	Beetle	4	VOLKSWAGEN	16-01-2002	109 500

Considérons tous les attributs :

- **num, nimm, dateAchat → prixAchat** (prixAchat dépend de la clé) ;
- **nimm → ncons,type, nomCons**;
- **num → nom**.

Les relations déduites de ces DF sont par définition en 2FN

FN Exemple

2FN

Ventes2

num#	nimm#	dateAchat	prixAchat
1	6748-XW-31	12-05-1999	62 500
1	734-AJH-31	19-07-2000	46680
1	358-ALZ-31	25-01-2001	213 970
2	955-NEH-75	16-01-1995	105 000
2	52-AIM-31	16-01-2002	109 500

Assures2

num	nom
1	Soutou
2	Bidal

Vehicules2

nimm	type	ncons	nomCons
6748-XW-31	320 i	1	BMW
734-AJH-31	1200 GSF	3	SUZUKI
358-ALZ-31	320 Ci	1	BMW
955-NEH-75	Scenic	2	RENAULT
52-AIM-31	Beetle	4	VOLKSWAGEN

FN Exemple

3FN

- La question à se poser: « **De quoi dépend directement chaque attribut de la relation ?** ».
- Cherchons les DF indirectes. On trouve la DF **nimm → nomCons** qui est indirecte car issue d'une transitivité.:
  - **nimm → ncons**
  - **ncons → nomCons**

FN Exemple

3FN

Ventes3

num#	nimm#	dateAchat	prixAchat
1	6748-XW-31	12-05-1999	62 500
1	734-AJH-31	19-07-2000	46680
1	358-ALZ-31	25-01-2001	213 970
2	955-NEH-75	16-01-1995	105 000
2	52-AIM-31	16-01-2002	109 500

Assures3

num	nom
1	Soutou
2	Bidal

Vehicules3

nimm	type	ncons#
6748-XW-31	320 i	1
734-AJH-31	1200 GSF	3
358-ALZ-31	320 Ci	1
955-NEH-75	Scenic	2
52-AIM-31	Beetle	4

Constructeurs3

ncons	nomCons
1	BMW
3	SUZUKI
2	RENAULT
4	VOLKSWAGEN

FN Exemple

FNBC

- Les relations sont déjà en FNBC !

Exercice

- On considère la relation suivante comme la base de données initiale d'une agence immobilière. Décomposer cette relation de manière à obtenir un schéma normalisé.

Num-Client	Nom-Client	NumApp	AdrApp	DateDLoc	DateFLoc	Montant	NumProp	NomProp
CR76	Jamal Khiyati	PG4	12, rue de la Gare	01.07.2003	31.08.2005	3500	CX40	Farid Assri
		PG16	7, av. République	01.09.2005	01.09.2010	4500	CX93	Ayoub Badr
CR56	Ahmed Zitouni	PG4	12, rue de la Gare	01.09.2002	10.06.2003	3500	CX40	Farid Assri
		PG36	3, Grande Rue	10.10.2003	01.12.2004	3800	CX93	Ayoub Badr
		PG16	7, av. République	01.01.2005	10.08.2005	4500	CX93	Ayoub Badr

- **Remarque :** On suppose qu'une personne peut louer plusieurs appartements en même temps mais elle ne peut pas relouer le même appartement une fois qu'elle le quitte.

Voici des exemples pour chaque concept du chapitre 2, avec exercices et leurs solutions :

1. Recensement des données

- Exemple 1 : Bibliothèque
  - Livres : Titre, Auteur, ISBN, Date de publication, Éditeur, Nombre de pages, Genre, Résumé.
  - Adhérents : NumAdhérent, Nom, Prénom, Adresse, Téléphone, Date d'inscription.
  - Emprunts : NumAdhérent, ISBN, DateEmprunt, DateRetour.
- Exemple 2 : Hôpital
  - Patients : NumPatient, Nom, Prénom, DateNaissance, Sexe, Adresse, MédecinTraitant.
  - Médecins : NumMédecin, Nom, Prénom, Spécialité.
  - Consultations : NumPatient, NumMédecin, DateConsultation, Diagnostic.
- Exemple 3: E-commerce
  - Produits: CodeProduit, NomProduit, Description, Prix, Catégorie, Stock.
  - Clients : NumClient, Nom, Prénom, Adresse, Email.
  - Commandes: NumCommande, NumClient, DateCommande, Total.
- Exercice : Recenser les données nécessaires pour une application de gestion d'une école (élèves, classes, professeurs, matières, notes).
- Réponse possible :
  - Élèves : NumÉlève, Nom, Prénom, DateNaissance, Classe.
  - Classes : CodeClasse, NomClasse, Niveau.
  - Professeurs : NumProfesseur, Nom, Prénom, Spécialité.
  - Matières : CodeMatière, NomMatière, Coefficient, ProfesseurResponsable (NumProfesseur).

- Notes : NumÉlève, CodeMatiere, Note, TypeEvaluation, DateEvaluation.

2. Objectifs de la structuration

- Exemple 1 : Minimiser la redondance (stockage, cohérence). Si l'adresse d'un client est stockée dans plusieurs tables, une modification d'adresse doit être effectuée dans chaque table, ce qui est source d'erreurs et de gaspillage d'espace disque.
- Exemple 2 : Faciliter les mises à jour (cohérence, maintenance). En centralisant les informations, les modifications sont plus rapides et plus fiables.
- Exemple 3 : Simplifier les requêtes (performance, ergonomie). Une structure de données bien pensée permet de formuler des requêtes plus simples et plus performantes.
- Exercice : Pourquoi est-il important de minimiser la redondance dans une base de données ?
- Réponse : Minimiser la redondance permet d'économiser de l'espace de stockage, de réduire les incohérences dues à des mises à jour partielles, et de simplifier la maintenance de la base.

3. Dépendance Fonctionnelle

- Exemple 1 : NumPasseport → Nom, Prénom, DateNaissance (Un numéro de passeport identifie une seule personne et donc ses informations).
- Exemple 2 : ISBN → Titre, Auteur, Éditeur (Un ISBN identifie un seul livre).
- Exemple 3 : CodePostal → Ville (Un code postal détermine une unique ville)
- Exercice : Identifier les dépendances fonctionnelles dans la table Employés(NumEmployé, Nom, Département, Salaire, NomManager).
- Réponse : NumEmployé → Nom, Département, Salaire, NomManager

4. Normalisation des relations

- Exemple 1: Table non normalisée Clients(NumClient, Nom, Adresse, Produits[liste], PrixTotal) La liste de produits achetés est multivaluée.
- Exemple 2: Table non normalisée Employés(NumEmployé, Nom, Département, Salaire, Projets[liste]) La liste de projets est multivaluée.
- Exemple 3: Table non normalisée Commandes(NumCommande, Client, AdresseClient, Produit, Prix, Quantité, Total) Total est un attribut dérivé et redondant, calculable à partir de Prix et Quantité.
- Exercice: Normaliser la table Livres(ISBN, Titre, Auteurs[liste], Editeur, AdresseEditeur)
- Réponse possible:
  - Livres(ISBN, Titre, Editeur)
  - Auteurs(NumAuteur, NomAuteur)
  - LivresAuteurs(ISBN, NumAuteur)
  - Editeurs(NomEditeur, AdresseEditeur) (Si on considère que l'éditeur a une existence indépendante du livre.)
  - Note: Si on considère que chaque livre n'a qu'un seul éditeur, alors Editeur et AdresseEditeur peuvent rester dans la table Livres.

5. Formes Normales

- Exemple 1 : 1FN – Adresses(Rue, Numéro, Ville, CodePostal, Habitants[liste]) viole la 1FN. Habitants est multivalué. Solution : créer une table Habitants(NumHabitant, Nom, Rue, Numéro) et une table intermédiaire AdresseHabitant(Rue, Numéro, NumHabitant).
- Exemple 2 : 2FN – Commandes(NumCommande, NumProduit, NomProduit, Quantité, PrixUnitaire, Total) avec clé primaire (NumCommande, NumProduit) viole la 2FN. NomProduit dépend uniquement de NumProduit. Solution : créer une table Produits(NumProduit, NomProduit)
- Exemple 3 : 3FN – Employés(NumEmployé, Département, NomDépartement, Salaire) avec clé primaire NumEmployé viole la 3FN. NomDépartement dépend de Département, qui n’est pas clé. Solution : créer une table Départements(CodeDépartement, NomDépartement).
- Exercice : Est-ce que la table Films(IdFilm, Titre, Réalisateur, NationalitéRéalisateur) est en 3FN ?
- Réponse possible : Non, car la NationalitéRéalisateur dépend du Réalisateur et pas directement de la clé IdFilm. Pour la normaliser, il faut créer une table Réalisateurs(IdRéalisateur, NomRéalisateur, Nationalité) et remplacer Réalisateur dans Films par IdRéalisateur.

Conclusion

- La structuration des données est essentielle pour un SI performant.
- Le recensement des données permet d'identifier les informations nécessaires.
- Les dépendances fonctionnelles révèlent les liens entre les attributs.
- La normalisation, à travers les formes normales, optimise la structure des tables pour réduire la redondance, améliorer l'intégrité et faciliter les mises à jour.

Références

- Cours de Professeur Asmaa El Hannani.
- Formes Normales: https://sgbd.developpez.com/tutoriels/cours-complet-bdd-sql/?page=normalisation

Resources

- Formes Normales Explications: https://www.ionos.fr/digitalguide/hebergement/aspects-techniques/normalisation-base-de-donnees/