IUT Sénart / Fontainebleau

NORMALISATION

Maggie LEKPA

Objectifs de ce cours

Manipuler une base de données via des requêtes SQL

Comment construire une bonne base de données relationnelle?

Mauvaise modélisation

Quels sont les problèmes liés à une mauvaise modélisation ?

Considérons la relation ci-dessous :

R(nom_agence, num_pret, client, montant, chif_affaire)

Nom_agence	Num_pret	client	montant	Chif_affaire
Fontainebleau	01	Dupont	1500	60M
Melun	10	Durant	10000	90M
Fontainebleau	02	Achille	6000	60M
Melun	11	Nestor	5000	90M
…	…	…	…	…

Que remarquez vous ?

Mauvaise modélisation

Une mauvaise conception d'une base de données relationnelle peut entrainer :

- Redondance des données (mise jour difficile de la données, gaspillage d'espace disque)
- Dépendance incohérente
- Compréhension difficile

NORMALISATION

D’où le besoin de définir un ensemble de règles permettant de définir un bon schéma et définir des contraintes sur les données.

La mise en place de ces règles s’appelle la **normalisation**.

Les contraintes sur les données sont définies par les **dépendances fonctionnelles**.

IUT Sénart / Fontainebleau

DEPENDANCES FONCTIONNELLES

Dépendance fonctionnelle

Soit une relation R ayant un ensemble d'attribut U;

Soient A et B des sous ensemble de U

On dit que **A détermine B ou B dépend fonctionnellement de A** si pour toute valeur de A une **seule valeur** de B lui est associé. Ça se note A->B

Exemple :

Soit la relation ETUDIANT(Num_Etu, Nom, Prenom,Adresse)

On a les dépendances fonctionnelles suivantes :

Num_Etu -> Nom

Num_Etu ->Prenom

Num_Etu ->Adresse

Nom -> Prenom n'est pas une dépendance fonctionnelle

Dépendance fonctionnelle

Soit une relation R; Soient A et B deux attributs de R.

On a A->B dans R si, pour tout tuple t1 et t2 de R,

t1[A] = t2[A] ⇒ t1[B] = t2[B]

Où ti[X] désigne la projection sur X du tuple ti de la relation R (la valeur de l'attribut X pour le tuple ti de la relation R)

Les dépendances fonctionnelles définissent les contraintes de données.

Exemple : si Id_Emp -> Nom_Employe

Pour deux employés X et Y, si on Id_Emp(X) = Id_Emp(Y) alors les deux employés ont forcement le même nom.

Propriétés des dépendances fonctionnelles

A, B, C sont des attributs ou des groupes d'attributs.

Les dépendances fonctionnelles obéissent aux propriétés suivantes :

Propriété 1: reflexivité

A->A
si il existe B ⊆ A alors A->B

Tout ensemble d'attribut détermine lui-même ou une partie de lui-même

Propriété 2: augmentation

Si A -> B alors A,C -> B,C

Si A détermine B, les deux ensembles peuvent être enrichis par un troisième.

Propriété 3: transitivité

Si A -> B et B -> C alors A->C

Propriétés des dépendances fonctionnelles

Propriété 4 : Union

Si A -> B et A -> C alors A -> B,C

Propriété 5 : pseudo-transitivité

Si A -> B et B,C -> D alors A,C -> D

Propriété 6 : décomposition

Si A->B et C ⊆ B alors A->C

Dépendance fonctionnelle directe

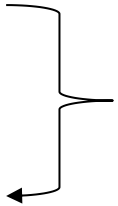
Une DF **A->B est directe** s'il n'existe aucun C tel que A->C et C->B

Exemple : soient les DF

Id_Fact -> Id_Client

Id_Client ->Adresse

Id_Fact->Adresse



Id_Fact -> Id_Client -> Adresse

Id_Fact -> Adresse n'est pas directe car peut être obtenue par transitivité

Dépendance fonctionnelle - élémentaire

Soit une relation R ayant un ensemble d'attribut U;

Soient A un sous ensemble de U et B un attribut de U.

Une **dépendance fonctionnelle A –> B est élémentaire** si pour toute partie A' ⊂ A, A' -> B n'est pas vrai.

En gros A est le plus petit ensemble qui détermine B.

Exemple :

Soit R(Id_emp, nom_emp, date_embauche, salaire)

{Id_Emp, Nom_Emp} -> Date_Embauche n'est pas une relation élémentaire car il suffit d'avoir Id_Emp (⊂ Id_Emp, Nom_Emp) pour obtenir la date d'embauche de l'employé.

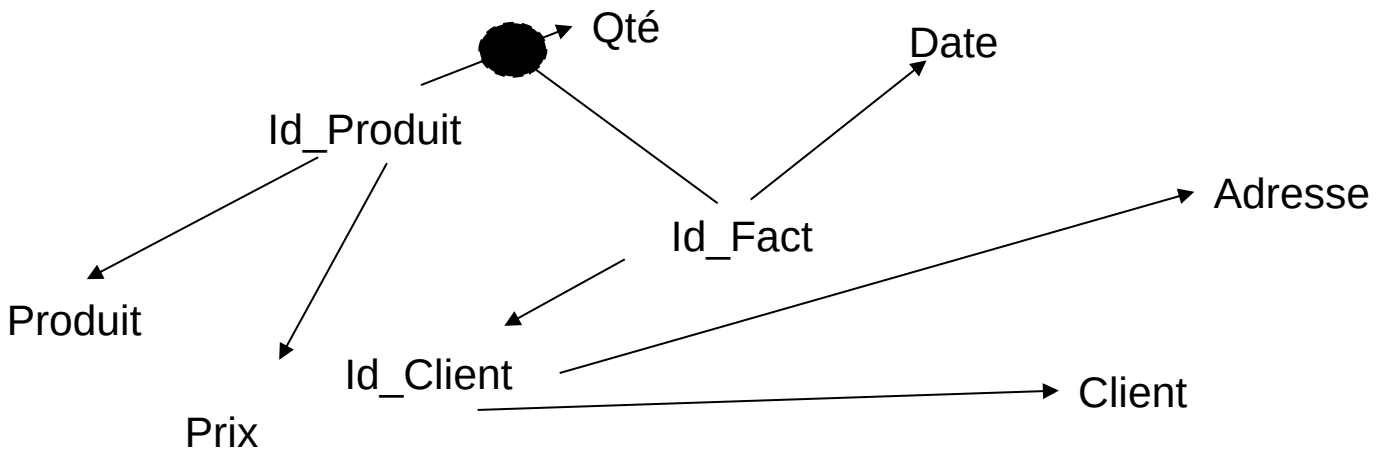
Dépendance fonctionnelle - graphe

L'ensemble des DF **élémentaires** d'une relation peut être représenté par un graphe dont les nœuds sont les attributs et les arc les dépendances.

Exemple :

Soient la relation R(id_fact, date, id_client,nom_client, adresse, produit, id_produit, prix, qte) et DF{id_produit → produit, prix; id_produit,id_fact→qt; id_fact→date,id_client; id_client→adresse, nom_client} l'ensemble des dépendances fonctionnelles élémentaires.

Le graphe des dépendances fonctionnelles est le suivant :



Dépendance fonctionnelle – Fermeture transitive

La **fermeture transitive F+ d'un ensemble de DFs F** est l'ensemble des DF élémentaires enrichi de toutes les DF élémentaires déduites par transitivité. Elle se note F+

Deux ensembles de DF sont équivalents s'ils ont la même fermeture transitive (si ils ont le même graphe maximal).

Exemple :

Soit l'ensemble des DF F tel que

F = {Id_Fact->Id_Client, Id_Client ->Nom, Id_Client ->Adresse}

F+ = F ∪ {Id_Fact -> Adresse} (fermeture transitive)

Dépendance fonctionnelle – couverture minimale

Un ensemble F de dépendance fonctionnelle est une **couverture minimale** si :

- Pour toute dépendance fonctionnelle f de F, F – {f} n'est pas équivalent à F
- En remplaçant une DF X->A de F par une DF Y->A avec Y ⊂ X, alors

F – {Y ->A} n'est pas équivalent à F

En gros F est une couverture minimale si elle n'inclue pas de dépendances fonctionnelles redondantes.

Toute relation a une couverture minimale aussi appelée couverture irrédondante (pas obligatoirement unique)

Dépendance fonctionnelle
fermeture d'un ensemble d'attributs

Soit F un ensemble de DF sur un ensemble d'attributs A

Soit X un ensemble d'attributs avec X ⊆ A

La fermeture de X sur F notée X+|F est l'ensemble d'attribut Y de A tel que X -> Y et {X→Y} ⊂ F+

Exemple :

F+ = {Id_Fact -> Id_Client, Id_Fact -> Date, Id_Client->Nom,
Id_Client->Adresse, Id_Fact->Nom, Id_Fact->Adresse}

{Id_Fact}+|F= {Id_Fact,Id_Client, Date, Adresse, Nom}

{Date}+|F={Date}

Dépendance fonctionnelle – clé primaire

La clé d'une relation est un attribut ou un ensemble d'attribut qui permet de caractériser de façon unique chaque tuple de la relation.

Soit Z un ensemble d'attributs

Z est la clé de la relation R(A1, A2,…,An) si

- Z+ ={A1,A2,…,An} : Z détermine fonctionnellement tous les attributs de R
- Il n'existe pas Z' ⊂ Z telle que Z'+ ={A1,A2,…,An}

En d'autres terme, Z doit être le plus petit ensemble d'attributs qui définit fonctionnellement R.

Dépendance fonctionnelle – clé primaire

Exemple : Soit une relation R(Id_Emp, Nom_Emp ,Date_Embauche), A l'ensemble des attributs de R

et l'ensemble de ses dépendances fonctionnelles

F = {Id_Fact->Id_Client, Id_Client ->Nom, Id_Client ->Adresse}

On a F+ = {Id_Fact -> Id_Client, Id_Fact -> Date, Id_Client->Nom, Id_Client->Adresse, Id_Fact->Nom, Id_Fact->Adresse}

{Id_Emp}+|F = {Id_Emp,Nom_Emp, Date_Embauche}

{Nom_Emp}+|F = {Nom_Emp}

{Date_Embauche}+|F = {Date_Embauche}

On peut conclure que {Id_Emp} est la clé de la relation R car {id_emp}+|F=A

Dépendance fonctionnelle – récap

Définition

Propriétés des dépendances fonctionnelles

Dépendance fonctionnelle directe – élémentaire

Fermeture transitive d'un ensemble de dépendances fonctionnelles

Fermeture élémentaire d'un ensemble d'attributs

Clé d'une relation

NORMALISATION

Dépendances fonctionnelles – contraintes sur les données

Comment modéliser une base de données ?

Partant d'une relation ayant un ensemble d'attributs, comment fait on pour la décomposer en relation ayant une bonne forme?

→ Décomposition selon un ensemble de règles

NORMALISATION – Decomposition d'une relation

La **decomposition d'un schema de relation** R est son remplacement par un ensemble de schema de relation R1, R2, … Rn telle que

$R = R1 \cup R2 \cup \ldots \cup Rn$

NORMALISATION – Decomposition d'une relation

Une **decomposition (R1, R2,…, Rn) de R est sans perte d'information** si la fermeture transitive R+ de R est la même que l'union des fermetures transitives des Ri.

R+=R1+∪ R2+ ∪ … ∪ Rn+

Condition suffissante : **la decomposition d'une relation R en R1, R2, …, Rn est sans perte d'informations si l'attribut de jointure entre Ri et Rj est la clé de Ri ou Rj**

Exemple : R(**Id_Fact**, Date, Id_Client, Nom)

R1(**Id_Fact**, Date, Id_Client)

R2(**Id_Client**, Nom)

NORMALISATION

La **normalisation** est le processus d'organisation des données dans la base de données. Cela passe par la création des tables à la création des relations entre ces dernières conformément à une ensemble de règles appelées **formes normales**.

Objectifs :

- S'assurer de la non redondance des données

- Eliminer les dépendances incohérentes : permet de faciliter l'accès aux données lorsqu'elles sont bien définies notamment via les index.

NORMALISATION – formes normales

La normalisation est basée sur 5 formes normales dans l'ordre 1FN, 2FN, 3FN, 4FN, 5FN.

La troisième forme normale est considérée comme étant le niveau le plus élevé nécessaire pour la plupart des applications.

Les 4FN et 5FN sont rarement prises en compte en pratique. Le non respect de ces règles peut engendrer une structure imparfaite de base de données, mais la fonctionnalité de la base de données ne devrait pas en souffrir.

Seules les trois premières formes normales seront abordées

Mmmais cela ne vous empêche pas d'aller vous documenter

☺

NORMALISATION : schema normalisé

Deux méthodes de normalisation :

- Par **decomposition sans perte de l'information**
- Par **synthèse** (graphe des DFs -> 3FN)

NORMALISATION : DECOMPOSITION - 1FN

Une relation respecte la 1FN si

- elle possède une clé primaire
- Tous les attributs sont atomiques (non décomposables).

Un attribut atomique est un attribut n'ayant à tout instant donné qu'une seule valeur ou ne regroupant pas un ensemble de valeurs.

Son but est d'éliminer les groupes répétitifs d'une table.

Exemple 1 : EmployeDenor (**Id_Emp**, Nom, Prenom, Projet, Durée(Jr) ,Desc)

Id_Emp	Nom	Prenom	Projet	Durée(Jr)	Desc
1	Martin	Paul	11	2	xxxx

NORMALISATION : DECOMPOSITION - 1FN

EmployeDenor (**Id_Emp**, Nom, Prenom, Projet1, Durée1,Desc1,… , Projetn, Duréen, Descn)

Id_Emp	Nom	Prenom	Projet1	Durée1	Desc	…	Projetn	Duréen	Descn
1	Martin	Paul	11	2	XXX		1n	4	YYY

Que se passe t'il lorsqu'un employé se voit octroyé n+1 projet?

Ajouter de nouvelles colonnes -> modification de la structure de la table

Mettre l'ensemble des projet dans la colonne Projet → Attribut non atomique

NORMALISATION : DECOMPOSITION - 1FN

Solution : Créer une seconde table contenant le groupe répétitif – faire une première décomposition (sans perte de l'information)

Employe(**Id_Emp**, Nom, Prenom)

Projet_Empl (**Id_Projet**, Durée(Jr), Desc, #**Id_Emp**)

Id_Emp	Nom	Prenom
1	Martin	Paul

Id_Projet	Durée(Jr)	Desc	Id_Emp
11	2	XXX	1
12	1	YYY	1
13	4	ZZZ	1

La jointure entre Employe et Projet se fait via Id_Emp qui est la clé de Employe

NORMALISATION : DECOMPOSITION - 2FN

Une relation respecte la 2FN si

- La 1FN est respectée
- Tous les attributs de la relation dépendent fonctionnellement de toute la clé (cas des clés composées)

Exemple :

Employe(Id_Emp, Nom, Prenom)

Projet(Id_Projet, Durée(Jr), Desc, Id_Emp)

La clé de la relation Projet est Id_Projet, Id_Emp

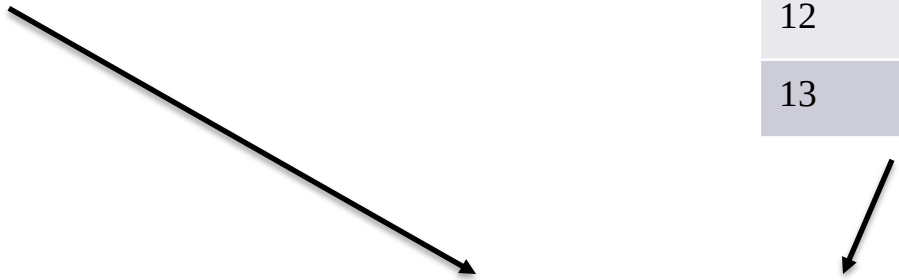
Id_Projet -> Desc et Id_Projet ->Durée(Jr) donc 2FN non respectée.

NORMALISATION : DECOMPOSITION - 2FN

Solution : Conserver dans la table les attributs qui dépendent de la totalité de la clé. Créer une nouvelle table avec les attributs qui dépendent d'une partie de la clé et faire de cette partie de la clé a clé de la nouvelle table.

Id_Emp	**Nom**	**Prenom**
1	Martin	Paul

Id_Projet	**Durée(Jr)**	**Desc**
11	2	XXX
12	1	YYY
13	4	ZZZ



Id_Emp	**Id_Projet**
1	11
1	12
1	13

On se rassure que les DF sont conservées et que les jointures entre les relations se fait via des clés.

NORMALISATION : DECOMPOSITION - 3FN

Une relation respecte la 3FN si

- elle respecte la 2FN
- Tous les champs qui ne dépendent pas **directement** de la clé ne sont pas stockés dans la table.

Exemple :

Employe (Id_Emp, Nom, Prenom, CP, Ville) ne respecte pas le 3FN

Ville -> CP : le code postal ne dépend pas directement de l'employé mais de la ville

Solution : on conserve dans la table initiale les champs dépendants directement de la clé. On crée une autre table avec les attributs dépendants transitivement. Le champ de transition devient la clé de la nouvelle table.

Employe (Id_Emp, Nom, Prenom, Ville) et Ville (Ville, CP)

NORMALISATION : ALGORITHME DE SYNTHESE

Algorithme de synthèse

Soit R=<A,F> où A est l'ensemble d'attributs et F les DFs

1. Représentons la couverture minimale
2. Tant qu'il y'a des dépendances fonctionnelles, faire ProduireGroupe
3. Construire la relation composée des attributs restants.

Procédure ProduireGroupe

• Rechercher le plus grand ensemble d'attributs X tel que X –> Am (m=1...p)

• Construire la relation Ri dont les attributs sont X ∪ A1 ∪ A2 ∪... ∪ Ap. La clé est X

• Retirer les DF utilisées

• Retirer de U les sommets de X, A1, A2..., Ap qui sont devenus isolés dans IRR(F) (ni source ni cible d'une DF)

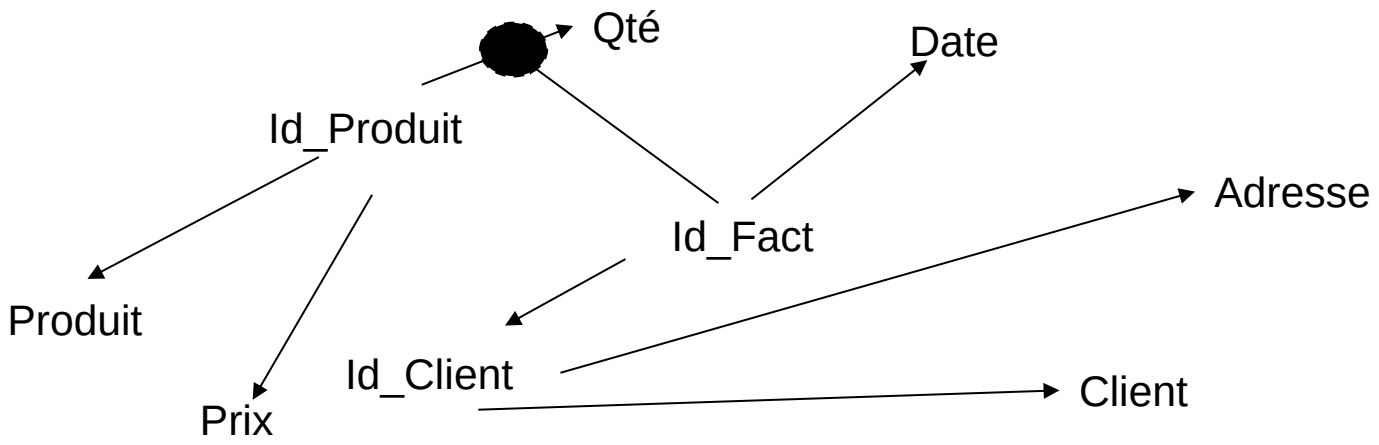
Fin procédure

NORMALISATION : ALGORITHME DE SYNTHESE

Soient la relation R(id_fact, date, id_client,nom_client, adresse, produit, id_produit, prix, qte)

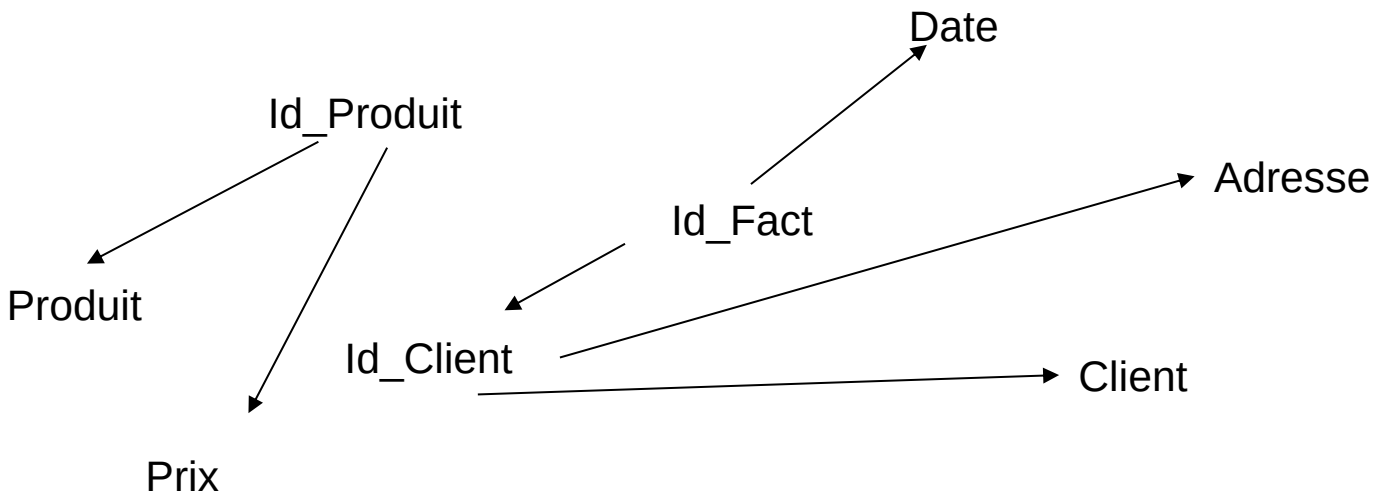
F={id_produit → produit, prix; id_produit,id_fact→qt; id_fact→date,id_client; id_client→adresse, nom_client} sa couverture minimale.

1. Représentons la couverture minimale



Itération 1 : R1(**Id_Produit**, **Id_Fact**, Qté)

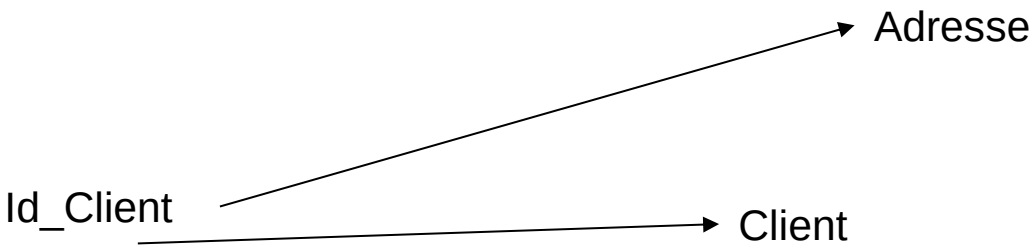
NORMALISATION : ALGORITHME DE SYNTHESE



Itération 1 : R1(**Id_Produit**, **Id_Fact**, Qté)

Itération 2 : R2(**Id_produit**, Produit, Prix) et R3(**Id_Fact**, Date, Id_Client)

NORMALISATION : ALGORITHME DE SYNTHESE



Itération 1 : R1(**Id_Produit**, **Id_Fact**, Qté)

Itération 2 : R2(**Id_produit**, Produit, Prix) et R3(**Id_Fact**, Date, Id_Client)

Itération 4 : R4(**Id_client**, Adresse, Client)

IUT Sénart / Fontainebleau

Fin