NORMALISATION AVANCÉE

INF3710

Objectifs d'apprentissage

- Comprendre et appliquer les règles d'inférences sur les DFs
- Comprendre les propriétés et les implications des décompositions relationnelles

Récapitulation

- **1NF**: Tous les attributs de chaque relation sont atomiques
- **2NF**: Une relation est en 2NF si elle est en 1NF et si chaque attribut qui ne fait pas partie d'une clé candidate dépend fonctionnellement complètement de toute clé candidate (on supprime les DF partielles)
- **3NF**: Une relation en 3NF est une relation en 2NF dans laquelle aucun attribut (qui ne fait pas partie d'une clé candidate) n'est transitivement dépendant d'une clé candidate (On supprime les DF transitives)
- **BCNF**: Une relation est en BCNF ssi chaque déterminant est une clé candidate
- **4NF**: Relation en BCNF qui ne contient pas de MVD non triviale

A −>> B dans une relation R est dite triviale si

(a) B est un sous-ensemble de A *ou*

(b) $A \cup B = R$.

RÈGLES D'INFÉRENCE POUR LES DÉPENDANCES FONCTIONNELLES

La clôture de l'ensemble des DFs

- On spécifie généralement les dépendances fonctionnelles évidentes à partir de la description du problème mais il n'est généralement pas possible de déterminer toutes les DFs possibles dans une situation.
  - Pour cela on utilise la notion de clôture qui se base sur l'inférence pour déterminer les autres DFs
- On dénote par F l'ensemble des dépendances fonctionnelles
- On dénote par F+ la fermeture ou la clôture de F
- F+ nous sert à déterminer quelles autres DF sont valides dans notre schéma et inclut les DF de F ainsi que celles qui sont inférées

Règles d'inférence

- On dénote une DF inférée par |=
- Les axiomes d'Armstrong:
  1. Règle de réflexivité: Si Y ⊆ X alors X →Y (DF triviale)
  2. Règle d'augmentation {X→Y} |= X, Z →Y, Z
  3. Règle de transitivité {X→Y, Y→Z} |= X → Z
- La clôture de F peut être définie simplement en appliquant les axiomes d'Armstrong jusqu'à ne plus pouvoir dériver de nouvelle DF

Autres règles d'inférences

- Trois autres règles découlent des axiomes d'Armstrong:
  - La règle de décomposition: {X→ Y, Z} |= {X →Y, X→Z}
  - La règle d'union: {X→Y, X→Z} |= X → Y, Z
  - La règle pseudo-transitive: {X→Y, W,Y →Z} |= W,X →Z

- Attention: X,Y → A n'implique pas nécessairement que X →A ou Y →A

La clôture de F

- Elle nous sert à

1. Trouver toutes les DF de F. Typiquement, on spécifie tout d'abord les DFs évidentes à trouver puis on utilise l'algorithme de calcul de la clôture pour trouver toutes les DFs qui peuvent être inférées à partir d'un ensemble d'attributs X sous F.
2. Tester l'existence d'une DF
   - Pour vérifier si on a la DF $X \rightarrow A$
     - Calculer la clôture de X, dénotée X+
     - Vérifier si $A \subseteq X+$
3. Tester si un déterminant X est une clé

Algorithme de calcul de la clôture

```
function ComputeX+(X, F)
begin
    X+ := X;
    while true do
        if there exists (Y → Z) ∈ F such that
            (1) Y ⊆ X+, and
            (2) Z ⊈ X+
        then X+ := X+ ∪ Z
        else exit;
    return X+;
end
```

X: l'ensemble d'attributs pour lesquels on veut trouver les dépendants fonctionnels

F: l'ensemble des DF

X+: La fermeture de X sous F

Exemple 1

- Soit l'ensemble des DFs:

  F = {nas → eNom; pnum → pNom, pVille; pVille, heures → bonus}

- En appliquant l'algorithme ComputeX+ (X, F) avec X = {pnum, heures}:

DF	X+
Ensemble Initial	pnum, heures
Pnum→ pNom, pVille	pnum, heures, pNom, pVille
pVille, Heures → bonus	pnum, heures, pNom, pVille, bonus

Exemple 2

$$\mathcal{F} = \{$$
$$A_1 \rightarrow A_2,$$
$$A_3 \rightarrow \{A_4, A_5\},$$
$$\{A_1, A_3\} \rightarrow A_6,$$
$$\}$$

Alors

$$A_1^+ = \{A_1, A_2\}$$
$$A_3^+ = \{A_3, A_4, A_5\},$$
$$\{A_1, A_3\}^+ = \{A_1, A_2, A_3, A_4, A_5, A_6\}$$

La clôture de F

- Elle nous sert à

1. Trouver toutes les DF de F. Typiquement, on spécifie tout d'abord les DFs évidentes à trouver puis on utilise l'algorithme de calcul de la clôture pour trouver toutes les DFs qui peuvent être inférées à partir d'un ensemble d'attributs sous F.
2. Tester l'existence d'une DF
   - Pour vérifier si on a la DF $X \rightarrow A$
     - Calculer la clôture de X, dénotée X+
     - Vérifier si $A \subseteq X+$
3. Tester si un déterminant X est une clé

Fermeture et clés

- Soit F un ensemble de DF dans une relation R.
  - X →Y appartient à F+ ssi Y est inclus dans `computeX+(X,F)`
  - X est une super-clé de R si et seulement si X → R autrement dit : X est une super-clé de R ssi `computeX+ (X, F) = R`
    - On teste si X+ contient tous les attributs de la relation R
  - X est une clé candidate de R si et seulement si X→ R et pour aucun $\alpha \subset X$, on a $\alpha \rightarrow R$
  - Pour trouver la clé candidate/ primaire d'une relation, on utilise l'algorithme sur la diapositive suivante

Détection d'une clé avec les DF et la clôture

Algorithm 16.2(a). Finding a Key K for R Given a set F of Functional Dependencies

Input: A relation R and a set of functional dependencies F on the attributes of R.

1. Set $K := R$.
2. For each attribute A in K

   {compute $(K - A)^+$ with respect to F;

   if $(K - A)^+$ contains all the attributes in R, then set $K := K - \{A\}$ };

Source: Elmasri and Navathe. Fundamentals of Database Systems- Pearson

Exercice

Etudiant (numEtud, nom, adresse, numEcole, nomEcole, villeEcole, moy, priorité)

numEtud → nom, adresse, moy

moy→ priorite

numEcole→ nomEcole, villeEcole

Fermeture de {numEtud, numEcole} ?

Clé?

Réponse

- {numEtud, numEcole} + = {numEtud, nom, adresse, moyenne, priorite, numEcole, nomEcole, villeEcole}
- Clé: {numEtud, numEcole}

COUVERTURE MINIMALE

D’un ensemble de dépendances fonctionnelles

Couverture d'un ensemble de DFs

- Un ensemble de DFs F couvre un autre ensemble de DFs E si chaque DF dans E est aussi dans F+.
  - On dit aussi que E est couvert par F.
- Deux ensembles de dépendances fonctionnelles E et F sont dits équivalents si E+ = F+

Ensemble minimal de dépendances fonctionnelles

- Il est également possible de réduire un ensemble de dépendances fonctionnelles F à un ensemble minimal qui est équivalent à l'ensemble original F.
- On s'intéresse à la couverture minimale pour pouvoir répondre à la question suivante:
- *Pour préserver toutes les dépendances fonctionnelles F, quel est l'ensemble minimal de dépendances fonctionnelles que l'on doit préserver?*

- De manière informelle, une couverture minimale d'un ensemble de DFs E est un ensemble de DFs F qui ont la propriété que chaque DF de E est dans la fermeture F+ de F.

Couverture minimale

- On définit formellement un ensemble de DF F comme étant minimal s'il satisfait les conditions suivantes:

1. Chaque dépendance dans F a un attribut unique sur son côté droit
2. On ne peut remplacer aucune dépendance X→A dans F avec une dépendance Y→A où Y est un sous-ensemble propre de X et avoir toujours un ensemble de dépendances qui soit équivalent à F
3. On ne peut pas supprimer une dépendance de F et avoir toujours un ensemble de dépendances équivalent à F.

- On peut penser à un ensemble minimal comme étant un ensemble de dépendances sous forme canonique (condition 1) et sans attributs superflus ou redondances (conditions 2 et 3).

Algorithme

Algorithm 16.2. Finding a Minimal Cover F for a Set of Functional Dependencies E

Input: A set of functional dependencies E.

1. Set $F := E$.
2. Replace each functional dependency $X \rightarrow \{A_1, A_2, \ldots, A_n\}$ in F by the n functional dependencies $X \rightarrow A_1, X \rightarrow A_2, \ldots, X \rightarrow A_n$.
3. For each functional dependency $X \rightarrow A$ in F
   for each attribute B that is an element of X
   if $\{ \{F - \{X \rightarrow A\}\} \cup \{ (X - \{B\}) \rightarrow A\} \}$ is equivalent to F
   then replace $X \rightarrow A$ with $(X - \{B\}) \rightarrow A$ in F.
4. For each remaining functional dependency $X \rightarrow A$ in F
   if $\{F - \{X \rightarrow A\}\}$ is equivalent to F,
   then remove $X \rightarrow A$ from F.

Source: Elmasri and Navathe. Fundamentals of Database Systems- Pearson

Exercice

- Soit le schéma relationnel R(A, B, C, D), sur lequel est défini l'ensemble de dépendances fonctionnelles suivant F = {A→ C, B→C, D→BC, AC→D}
- Donnez une couverture minimale de F. Expliquez votre démarche en détail.

- Voir le fichier couverture_minimale_illustration.pdf

Couverture minimale – Stratégie 2

- On peut aussi trouver la couverture minimale en utilisant les axiomes d'Armstrong
- On trouve la forme canonique de F
- On utilise les axiomes d'Armstrong pour trouver les attributs superflus
- On utilise les axiomes d'Armstrong pour trouver les dépendances fonctionnelles redondantes

Exercice

- Soit l'ensemble E des DFs: {B→A; D→A; A,B→D}
- Trouvez la couverture minimale de E.

Réponse

- **Etape 1** : Toutes les DFs sont sous forme canonique: elles ont un attribut à droite
- **Etape 2**: Pour la DF A, B → D on doit déterminer si un des attributs A, B est redondant. Autrement dit, on doit tester si on peut remplacer A, B → D par B →D ou A→D.

E : {B→A; D→A; A,B→D}

- Comme on a B →A
  - On peut l'augmenter par B, i.e. B, B→ A, B (règle d'addition), ce qui est équivalent à B → A, B
  - Or on a aussi A, B → D
  - Par la règle de transitivité, on obtient B → D
  - Donc A, B → D peut être remplacé par B → D
  - E' = {B →A; D →A; B →D} est un ensemble minimal équivalent qui ne peut pas être plus décomposé car toutes les DFs ont un attribut unique sur la gauche.

Source: Elmasri and Navathe. Fundamentals of Database Systems- Pearson

Réponse

- **Etape 3** : On recherche les DFs redondante dans E'.
- E' = {B →A; D →A; B →D}
- En utilisant la règle de transitivité sur B → D et D →A, on infère B → A
- Donc on peut supprimer B →A de E' (elle est redondante)

- Enfin, la couverture minimale de E est {B →D, D → A}

Source: Elmasri and Navathe. Fundamentals of Database Systems- Pearson

PROPRIÉTÉS DES DÉCOMPOSITIONS RELATIONNELLES

Introduction

- Les formes normales ne garantissent pas à elles seules d'avoir une bonne modélisation de la BD
  - Il ne suffit pas de dire que vos relations sont en 3NF ou BCNF séparément par exemple

Décomposition

- La normalisation se base sur la décomposition pour diviser une table qui contient des anomalies en plusieurs tables basées sur les DFs
- On doit être capable de retrouver notre table initiale après la décomposition (via une jointure naturelle)
- Une bonne décomposition:
  - ne perd PAS d'information
  - ne complique PAS la vérification des contraintes
  - contient MOINS d'anomalies que la table initiale

Propriétés

- Il existe plusieurs propriétés essentielles à conserver lors des décompositions successives:

1. La préservation d'attributs: on doit s'assurer qu'aucun attribut n'est perdu lors de la décomposition. Chaque attribut doit se retrouver dans une des relations décomposées.
2. La préservation des dépendances fonctionnelles: les DFs initiales à partir du « schéma universel » doivent pouvoir être retrouvées à partir des dépendances fonctionnelles des relations décomposées
3. La jointure sans perte et sans addition(Non additive lossless join property): Aucun tuple en trop ne doit être généré par la jointure naturelle des relations décomposées

Préservation des dépendances

- La propriété de préservation des dépendances garantit que chaque dépendance fonctionnelle initiale est représentée dans une des relations individuelles après le processus de décomposition
- Chaque DF représente une contrainte de notre BD et la perte d'une DF sur une relation individuelle implique qu'on ne peut pas imposer cette contrainte!
- Soit S l'ensemble des relations obtenues dans notre modélisation finale.
- F l'ensemble des DFs initiales
- F0 l'ensemble des DF préservées dans les relations décomposées
- La modélisation S préserve les dépendances si F0+ = F+

Définition formelle

- Soit F l'ensemble des dépendances fonctionnelles d'une relation R
- Soit Ri un sous-ensemble de R
- La projection de F sur Ri, dénotée πRi(F) est l'ensemble des dépendances $X \rightarrow Y$ dans F+ tel que tous les attributs dans $X \cup Y$ sont contenus dans Ri
- On dit qu'une décomposition D = {R1, R2, … , Rm} de R préserve les dépendance par rapport à F si l'union des projections de F sur chaque Ri dans D est équivalent à F, c'est-à-dire

$$((\pi R1(F)) \cup \ldots \cup (\pi Rm(F)))+ = F+.$$

Exemple

- ClientSuccursale (clientId, employeId, succursaleId)
- DF1: employeId → succursaleId
- DF2: clientId, succursaleId → employeId

- La relation n'est pas en BCNF. Si on la décompose en:

- SuccursaleEmploye (employeId, succursaleId)
- ClientEmploye (clientId, employeId)

- On constate que DF2 est perdue!
- ((πSuccursaleEmploye(F)) ∪ (πClientEmploye(F)))+ ≠ F+.

Exercice

- Soit le schéma R(A, B, C, D) avec F = {A → B, B → C}.
- Si on décompose ce schéma en R1(AB), R2(AC), est-ce que les DFs sont préservées ?

Réponse

- Ensemble initial F = {A → B, B → C}.
- S = {R1, R2}.
- R1(AB), R2(AC)
- F' = {A → B, A → C}
- On voit bien que F+ n'est pas équivalent à F'+
- Donc la décomposition S ne préserve pas les dépendances.

Lossless Join

- La propriété lossless join garantit qu'aucun tuple additionnel ne soit généré suite à une jointure naturelle des relations de base i.e. on doit retrouver uniquement les tuples originaux des relations de base
- Lossless réfère à une perte d'information, pas à une perte de tuples!
- C'est une propriété qu'il faut conserver!

Exemple

- Soit la table Marks

Student	Assignment	Group	Mark
Ann	A1	G1	80
Ann	A2	G3	60
Bob	A1	G2	60

- On la décompose en:

Student	Group	Mark
Ann	G1	80
Ann	G3	60
Bob	G2	60

Assignment	Mark
A1	80
A2	60
A1	60

Que se passe-t-il si vous faites la jointure de ces deux tables?

Exemple

Student	Assignment	Group	Mark
Ann	A1	G1	80
Ann	A2	G3	60
Ann	A1	G3	60
Bob	A2	G2	60
Bob	A1	G2	60

- Si on effectue la jointure des nouvelles tables, on obtient des tuples additionnels!
- Ceci n'est pas une bonne décomposition. La propriété *lossless join* n'est pas respectée.

Décomposition Lossless-Join binaire

- Une décomposition {R1, R2} sur un ensemble de dépendances F a la propriété lossless join si les attributs communs à R1 et R2 forment une super-clé de R1 ou R2 c'est-à-dire:
  - R1 ∩ R2 → R1-R2 est dans F+

    ou
  - R1 ∩ R2 → R2-R1 est dans F+
- Autrement dit, étant donné l'ensemble de DFs F, on doit avoir au moins une de ces dépendances R1 ∩ R2 → R1-R2 ou R1 ∩ R2 → R2-R1 dans F+

Exemple

Student	Assignment	Group	Mark
Ann	A1	G1	80
Ann	A2	G3	60
Bob	A1	G2	60

- Dans l'exemple on a:
  - R = {Student, Assignment, Group, Mark}
    - F = {(Student, Assignment →Group, Mark)}
  - R1 = {Student, Group, Mark}
  - R2 = {Assignment, Mark}

Student	Group	Mark
Ann	G1	80
Ann	G3	60
Bob	G2	60

Assignment	Mark
A1	80
A2	60
A1	60

- La décomposition {R1, R2} perd des informations car:
  - R1 ∩ R2 = {mark}
  - R1-R2 = {Student, Group}
  - R2-R1 = {Assignment}
- On n'a ni Mark → Student, Group ni Mark→Assignment dans F+
- Cette décomposition n'a donc pas la propriété lossless join!

Exercice 1

- Soit R = (A, B, C), F = {A → B, B → C)
- Supposons les deux décompositions:
  a. R1 = (A, B), R2 = (B, C)
  b. R1 = (A, B), R2 = (A, C)
- Dans chaque cas indiquez
  - Si la décomposition est lossless-join
  - Si les dépendances sont préservées

Réponse

- Soit R = (A, B, C), F = {A → B, B → C)
- R1 = (A, B), R2 = (B, C)
- La décomposition est Lossless-join car
    - R1 ∩ R2 = {B} ; R1-R2 = {A} ; R2-R1={C}
        - A-t-on B → A ou B → C dans F+ ?
        - On a B →C
        - Lossless join!
- Les dépendances sont préservées

Réponse

- Soit R = (A, B, C), F = {A → B, B → C)

  R1 = (A, B), R2 = (A, C)

  - La décomposition est lossless-join car
  - R1 ∩ R2 = {A} ; R1-R2 = {B} ; R2-R1={C}
    - A-t-on A → B ou A → C dans F+ ?
    - Oui, lossless join!
- Mais on perd la dépendance ! (B → C est perdue)

Exercice 2

- Soit R=ABC et F = { A→B }.
- On veut évaluer la décomposition en R1 =(A, B) et R2 =(B, C).
- Propriété lossless join préservée?

Réponse

- Soit R=ABC et F = { A→B }.
- On veut évaluer la décomposition en R1 ={AB} et R2 ={BC}. Propriété lossless join?

- R1 ∩ R2 = B ; R1 – R2 = A; R2 – R1 = C
- A-t-on B→ A ou B→ C dans F+ ?
- Non.
- La propriété lossless join n'est pas respectée par cette décomposition.

Décomposition 3NF

- Il est toujours possible de décomposer une relation en un ensemble de relations 3NF de telle manière que la décomposition soit *lossless* et que les dépendances soient préservées
- Pour ceux qui voudraient savoir comment le faire, consultez l'algorithme **15.4 d'Elmasri et Navathe (7ème édition)**

- Mais la forme 3NF comporte toujours certaines anomalies comme on l'a vu!

Décomposition BCNF – Perte de DF

- La décomposition BCNF vue en cours garantit la propriété lossless join mais peut nous faire perdre certaines dépendances fonctionnelles
  - Ces DFs perdues peuvent introduire des anomalies
  - Une option est alors de considérer une forme plus faible de forme normale si on veut les préserver

Rappel: Décomposition BCNF

Entrée: relation R + DFs de R
Sortie: décomposition de R en relations BCNF (lossless)

Trouver les clés de R
Répéter jusqu'à ce que toutes les relations soient en BCNF:
    Prendre n'importe quelle R' avec $A \rightarrow B$ qui viole BCNF
    Décomposer R' en $R_1(A, B)$ et $R_2(A, reste)$
    Trouver les DFs de R_1 et R_2
    Trouver les clés de R_1 et R_2

Exemple

Inscription(numEtud, universite, date, programme)

- un étudiant à une université donnée ne peut pas s'inscrire à plus d'un programme ni à plus d'une date
  - DF1:

- Chaque date d'inscription correspond à une université unique
  - DF2:

- Clé ?
- 3NF? BCNF?
- Bonne modélisation?

Réponse

- Inscription(numEtud, universite, date, programme)
- DF1: numEtud, universite → date, programme
- DF2: date → universite
- Clé: numEtud, universite
- La relation est en 3NF
- La relation n'est pas en BCNF! On décompose:
  - R1 (date, universite)
  - R2 (numEtud, date, programme)
  - Ceci n'est pas une « bonne » modélisation car:
    - On a séparé l'étudiant de l'université où il s'inscrit
    - DF1 n'est pas préservée

Rappel: 3NF permet des DF A→B ou A n'est PAS une clé candidate et B fait partie d'une clé candidate!

Après la décomposition, il n'y a aucune garantie que les dépendances pourront être vérifiées sur les relations décomposées : on voit que DF1 ne peut pas être vérifiée sans faire un join entre R1 et R2!

Récapitulation sur la normalisation

- Théorie pour la conception des BDs relationnelles
  - Formes Normales → “Bonnes” relations
  - Modélisation par décomposition
  - Généralement intuitif et marche convenablement
- Limites
  - Respect des dépendances
  - Surcharge des requêtes
  - Trop de décompositions
- Parfois, on peut donc vouloir “dénormaliser”
  - Si on a peu d’opérations d’insertion, suppression et modification de données qui pourraient introduire des anomalies
  - Si on a beaucoup de requêtes qui nécessitent des jointures

Quelques limites

- Les algorithmes de normalisation se basent sur l'identification des DFs
  - L'oubli de DFs importantes peut conduire à une mauvaise modélisation
- Certains algorithmes ne sont pas déterministes : on peut avoir plusieurs solutions valides!
  - Exemple 1: la recherche de la couverture minimale : en fonction de la couverture minimale trouvée, on peut aboutir à des modélisations différentes, certaines meilleures que d'autres!
  - Exemple 2 : La décomposition issue de BCNF peut être différente en fonction de l'ordre dans lequel sont traitées les DFs!

Références

- Connolly et Begg, chapitre 14
- Elmasri and Navathe. Fundamentals of Database Systems- Pearson – Chapitre 15, 16