LIF10 – FONDEMENTS DES BASES DE DONNÉES
TD8 – normalisation

Licence informatique – Automne 2015–2016

Les questions marquées du symbole (†) sont à préparer pour la séance

Exercice 1 : Dépendances et décomposition

Soit $R = XYZ$ un schéma de relation avec $\{X, Y, Z\}$ une partition de R et r une instance sur R.

1. Supposons que $r \models X \to Y$. Montrer que la décomposition en $\{XY, XZ\}$ est sans perte, c'est-à-dire que $r = \pi_{XY}(r) \bowtie \pi_{YT}(r)$ en montrant les deux inclusions.
2. Transformer la preuve précédente pour montrer que $r \models X \twoheadrightarrow Y$ implique que la décomposition en $\{XY, XZ\}$ est sans perte.
3. Prouver que si la décomposition en $\{XY, XZ\}$ est sans perte alors $r \models X \twoheadrightarrow Y$. Pour la preuve, on montrera la contraposée, c'est-à-dire que l'on montre $P \Rightarrow Q$ en prouvant $\neg Q \Rightarrow \neg P$.

Exercice 2 : Décompositions sans pertes (*[1]*)

1. Soit $R(ABCDE)$ une relation et $\Sigma = \{A \to BC, CD \to E, B \to D, E \to A\}$ un ensemble de dépendances. Calculer les clefs minimales (dites aussi candidates).
2. Montrer que la décomposition de R en $R_1(ABC)$ et $R_2(ADE)$ est sans perte d'information.
3. On rappelle la définition de la projection de dépendance sur une relation $\Sigma[S] = \{X \to Y | X \to Y \in \Sigma^+ \wedge XY \subseteq S\}$. La décomposition précédente est-elle sans perte de dépendance ?
4. Montrer que la décomposition en $R_1(ABC)$ et $R_2(CDE)$ n'est pas sans perte d'information en proposant un contre-exemple.
5. Proposer une décomposition en 3FN qui soit sans perte d'information ni perte de dépendances en utilisant l'algorithme de synthèse.
6. Proposer une décomposition en deux relations qui soient en FNBC et sans perte d'information.

Exercice 3 : Forme normale de Boyce-Codd

1. Montrer que toute relation en 3FN n'ayant qu'une seule et unique clé minimale est en FNBC.
2. Montrer que toute relation à deux attributs est en FNBC. On procédera par exhaustivité en énumérant toutes les dépendances possibles qui peuvent exister.
3. Soit le schéma $R = ABCD$ et l'ensemble de DFs $F = \{A \to B, C \to D, B \to C\}$. Simplifier cet ensemble de DFs. Donner des décompositions *différentes* de cette base, toutes en FNBC, mais avec perte de dépendances possible.
4. Soit le schéma $R = ABCD$ et l'ensemble de DFs $F = \{AB \to C, C \to D, D \to A\}$. Calculer les clés minimales de cet ensemble
5. Identifier toutes les violations de la FNBC.
6. Décomposer R en 3FN en utilisant l'algorithme de synthèse. Préciser la forme normale obtenue.
7. Soit la décomposition AD, BC, CD. Quelle est la forme normale de ce schéma ? Préciser les dépendances qui ne sont pas préservées.

Exercice 4 : Le *chase*

On va considérer dans cette exercice l'algorithme 1 appelé *chase* qui permet de déterminer si une décomposition est sans perte d'information. Le *chase* prend en entrée un ensemble fini d'attributs $R = A_1 ... A_n$, un ensemble F de dépendances fonctionnelles et une décomposition $\mathbf{R} = \{R_1 ... R_k\}$ et détermine si pour toute instance r de R on a bien $r = \pi_{R_1}(r) \bowtie ... \bowtie \pi_{R_k}(r)$. Pour chaque attribut $A \in R$, on considère un domaine associé $D_A = \{d^A, v_0^A, v_1^A, ...\}$ avec l'ordre total $d^A \sqsubset v_0^A \sqsubset v_1^A ... \ d^A$ est appelée la *constante* pour l'attribut A et $v_0^A, v_1^A, ...$ sont appelées les *variables* pour l'attribut A.

Le principe du *chase* est de construire une instance symbolique r à partir de $\mathbf{R}$, lignes 1 à 7 de l'algorithme 1 : pour chaque relation $S \in \mathbf{R}$ on ajoute un tuple t dans r tel que t a une constante que chaque attribut de S et une variable pour les autres. Ensuite, lignes 9 à 16, une boucle essaie d'appliquer successivement les dépendances de F pour rendre égales certaines variables afin de s'assurer que r soit bien un modèle de F. Finalement, ligne 17, quand plus aucune dépendance n'est applicable, on vérifie si r contient une ligne qui ne contient que des constantes, si c'est le cas alors la décomposition est sans perte.

1. Justifier que l'algorithme termine.
2. Soit $R = ABC$, $\mathbf{R} = \{AB, BC\}$ et $F = \{B \to C\}$. Construire l'instance symbolique, appliquer la dépendance $B \to C$ et déterminer si la décomposition est sans perte.
3. Même question avec $R = ABCDE$, $\mathbf{R} = \{AD, AB, BE, CDE, AE\}$ et $F = \{A \to C, B \to C, C \to D, DE \to C, CE \to A\}$.
4. Montrer que si l'algorithme renvoie *false*, alors la décomposition proposée est *avec* perte. L'idée est que l'instance produite par le *chase* est un contre-exemple où il y a perte.
5. Pouvez-vous conclure sur la correction de l'algorithme ?

Algorithme 1 : *chase*

Data : $R = A_1 ... A_n$ un schéma
Data : F un ensemble de DFs
Data : $\mathbf{R}$ une décomposition de R
Result : *true* si la décomposition est sans perte, *false* sinon

```
1  r := ∅;
2  i := 0;
3  for S ∈ R do
4      for A ∈ R do
5          if A ∈ S then  t[A] = d^A ;
6          else  t[A] = v_i^A ;
7      r := r ∪ {t};
8      i := i + 1;
9  done := false;
10 while (¬done) do
11     done := true;
12     foreach X → Y ∈ F do
13         if ∃t0, t1 ∈ r.t0[X] = t1[X] ∧ ∃A ∈ R.t0[A] ⊏ t1[A] then
14             r := r \ {t1};
15             t1[A] := t0[A];
16             r := r ∪ {t1};
17             done := false;
18 return (∃t ∈ r.t = ⟨d^A1 ... d^An⟩)
```

Corrections

Solution de l'exercice 1

1. Pour montrer que la décomposition est sans perte, $r = \pi_{XY}(r) \bowtie \pi_{XZ}(r)$, on montre la double inclusion :

   $r \subseteq \pi_{XY}(r) \bowtie \pi_{XZ}(r)$: il est assez évident que tout $t \in r$ appartient aussi à $\pi_{XY}(r) \bowtie \pi_{XZ}(r)$. Intuitivement, on obtient par reconstruction *au moins* ce que l'on avait initialement. Supposons $t \in r$ et notons $t_1 = t[XY] \in \pi_{XY}(r)$ et $t_2 = t[XZ] \in \pi_{XZ}(r)$: on retrouve bien t avec t_1 et t_2 par jointure sur X. Noter que l'on a utilisé aucune propriété particulière et que donc cette propriété est toujours vraie quelque soit l'instance r.

   $\pi_{XY}(r) \bowtie \pi_{XZ}(r) \subseteq r$: soit t un tuple de $\pi_{XY}(r) \bowtie \pi_{XZ}(r)$. D'après la définition de la jointure, ils existent deux tuples t_1 et t_2 de r tels que $t_1[XY] = t[XY]$ et $t_2[XZ] = t[XZ]$. On a donc $t_1[X] = t_2[X] = t[X]$, comme $r \models X \rightarrow Y$, on a également $t_1[Y] = t_2[Y] = t[Y]$. Ainsi $t_2 = t_2[XYZ] = t[XYZ] = t$ et donc $t \in r$.

2. Comme remarqué à la réponse précédente, il suffit de montrer que $\pi_{XY}(r) \bowtie \pi_{XZ}(r) \subseteq r$ car l'inclusion inverse est toujours vraie. Soit t un tuple de $\pi_{XY}(r) \bowtie \pi_{XZ}(r)$ et $t_1[XY] = t[XY]$ et $t_2[XZ] = t[XZ]$ tel que précédemment. Comme $t_1[X] = t_2[X] = t[X]$, d'après la définition de $r \models X \twoheadrightarrow Y$ [1], il existe un tuple t_3 tel que $t_3[X] = t[X]$, $t_3[Y] = t_1[Y] = t[Y]$ et $t_3[Z] = t_2[Z] = t[Z]$ et ainsi $t_3 = t$.
3. La décomposition est sans perte implique la DMV. Supposons que $r \not\models X \twoheadrightarrow Y$, c'est-à-dire que pour au moins une certaine paire de tuples t_1 et t_2 de r tels que $t_1[X] = t_2[X]$ alors soit t_3 tel que $t_3[XY] = t_1[XY]$ et $t_3[R \setminus Y] = t_2[R \setminus Y]$ n'appartient pas à r, soit c'est t_4 tel que $t_4[XY] = t_2[XY]$ et $t_4[R \setminus Y] = t_1[R \setminus Y]$ qui n'appartient pas à r. Supposons sans perte de généralité que ce soit $t_3 \notin r$. Or en faisant la jointure de $t_1[XY] \in \pi_{XY}(r)$ avec $t_2[XZ] \in \pi_{XZ}(r)$ on retrouve le tuple $t_3 \in \pi_{XY}(r) \bowtie \pi_{XZ}(r)$ qui n'est pas censé exister (on a « crée » t_3 par jointure). La jointure est donc avec perte car $\pi_{XY}(r) \bowtie \pi_{XZ}(r) \not\subseteq r$.

Solution de l'exercice 2

1. On trouve A, E, BC et CD.
2. On utilise le résultat de l'exercice 1. $\{A, BC, DE\}$ est une partition de R. La décomposition en $R_1(ABC)$ et $R_2(ADE)$ est sans perte car on a $A \rightarrow BC$.
3. Si on projette Σ sur R_1 et R_2, on perd $B \rightarrow D$. En effet $\Sigma_1 = \Sigma[R_1]$ ne contient aucune dépendance avec D en partie droite et $\Sigma_2 = \Sigma[R_2]$ ne contient aucune dépendance avec B en partie gauche. Pour que $B \rightarrow D$ soit préservée il faudrait qu'il existe un attribut $\alpha \in R_1 \cap R_2$ tel que $B \rightarrow \alpha \in \Sigma_1^+$ et $\alpha \rightarrow D \in \Sigma_2^+$. Or $R_1 \cap R_2 = \{A\}$ et $B \rightarrow A \notin \Sigma_1^+$.
4. On remarque que $R_1 \cap R_2 = \{C\}$, donc pour obtenir un contre-exemple il faut arriver à faire se combiner des valeurs de AB avec des valeurs de DE qui n'existaient pas initialement dans la relation, pour obtenir $r \neq \pi_{ABC}(r) \bowtie \pi_{CDE}(r)$. Voici un tel exemple

	A	B	C	D	E
u	a_0	b_0	c_0	d_0	e_0
u	a_1	b_1	c_0	d_1	e_1

5. On vérifie que l'ensemble Σ est bien déjà une couverture minimale. On obtient par synthèse $R' = \{\underline{A}BC, \underline{CD}E, \underline{B}D, \underline{E}A\}$ qui est en 3FN.
6. $B \rightarrow D$ est la seule dépendance dont la partie gauche ne soit pas une clef. On peut décomposer en $R_1(ABCE)$ et $R_2(BD)$.

Solution de l'exercice 3

1. Soit $X \rightarrow A$ une dépendance non-triviale et K l'unique clé minimale de R. Deux cas sont possibles :
   — soit $A \in K$, comme $X \rightarrow A$ n'est pas triviale, on a $A \notin X$, mais alors, $K \setminus \{A\} \cup X$ serait clé et on obtiendrait par minimisation de la partie gauche une nouvelle clé miminale différente de K, ce qui est impossible.

1. A savoir $r \models X \twoheadrightarrow Y$ ssi $\forall t_1, t_2 \in r$ avec $t_1[X] = t_2[X]$ alors $\exists t_3, t_4 \in r$ tels que $t_3[X] = t_4[X] = t_1[X]$, $t_3[Y] = t_1[Y]$ et $t_3[R \setminus Y] = t_2[R \setminus Y]$ et $t_4[Y] = t_2[Y]$ et $t_4[R \setminus Y] = t_1[R \setminus Y]$ (on obtient toujours t_3 et t_4 en « croisant » t_1 et t_2).

— soit $A \notin K$, dès lors, la condition de 3FN est applicable car A n'appartient à aucune clé minimale et donc K est (super)clé et aussi clé minimale.

2. Sans perte de généralité, posons $R = AB$. Il n'y a que deux DFs non-triviales possibles : $A \to B$ et $B \to A$, dès lors on regarde les quatre cas possibles :
   — $F = \emptyset$ la propriété FNBC est trivialement satisfaite.
   — $F = \{A \to B\}$. La clé est A et on a bien B qui dépend directement de la clé.
   — $F = \{B \to A\}$. Similaire au cas précédent.
   — $F = \{B \to A, A \to B\}$. On a deux clés minimales : A et B et ainsi pour chacune des DFs la partie gauche est bien clé.
3. Cet ensemble de DF est déjà simplifié. Par synthèse on va obtenir le schéma de base de données $\mathbf{R}_1 = \{AB, BC, CD\}$. Pour que la décomposition soit sans perte d'information, on va utiliser le résultat de l'exercice 1 et s'assurant que l'intersection d'une paire de relations est bien clé de l'une des deux. On va trouver les décompositions suivantes où les DFs ne sont pas préservées : $\mathbf{R}_2 = \{AB, AC, CD\}$, $\mathbf{R}_3 = \{AD, AB, BC\}$ etc.
4. L'attribut B n'apparaissant jamais en partie droite de F, il appartient nécessairement aux clés minimales. Comme $B^+ = B$, B seul ne suffit pas. On calcule $AB^+ = R$, $BC^+ = R$ et $BD^+ = R$ qui sont clés minimales. Comme tout ensemble de cardinalité trois doit au moins contenir l'une de ces clés, ce sont les seules minimales.
5. On considère chacune des DFs de $F = \{AB \to C, C \to D, D \to A\}$:
   — $AB \to C$ ne viole pas la FNBC car AB est clé minimale et donc (super)clé.
   — $C \to D$ viole la FNBC car C n'est pas (super)clé.
   — $D \to A$ viole la FNBC car C n'est pas (super)clé.
6. Le principe de la synthèse est de créer une relation XY pour chaque DF $X \to Y \in F$, puis supprimer les relations inutiles et enfin ajouter une relation avec une clé minimale si nécessaire. On obtient ici les relations AD, CD, ABC dès la première étape. On est en 3FN car il n'y a plus de transitivité (on a éliminé D qui poserait ce problème dans ABC) mais pas en FNBC à cause de $AB \to C$ et $C \to A$.
7. Cette décompo est en FNBC : on peut le vérifier ou utiliser l'argument de la deuxième question car chaque schéma est de cardinalité deux.

Solution de l'exercice 4 En fait, le *chase* est une méthode de type *tableaux* qui peut être aussi utilisé pour l'inférence des DFs et même d'autres classes de dépendances plus générales.

1. Après l'initialisation on a une instance r telle que $|r| = |\mathbf{R}|$. Après chaque étape de la boucle, les tuples de r sont plus définis qu'à l'étape précédente car la ligne 14 prend l'élément le plus petit selon l'ordre $\sqsubset$. Comme les domaines sont bornés par la constante d^A on ne pourra pas décroître indéfiniment.
2. On génère t_0 avec AB puis t_1 avec BC pour obtenir l'instance suivante :

r	A	B	C
t_0	d^A	d^B	v_0^C
t_1	v_1^A	d^B	d^C

   On a qu'une seule DF $B \to C$ on va l'appliquer et obtenir l'instance suivante où on remplace v_0^C par d^C dans t_0 :

r	A	B	C
t_0	d^A	d^B	d^C
t_1	v_1^A	d^B	d^C

   La première ligne ne contient que des constantes et ainsi la décomposition est sans perte.
3. Selon les choix des dépendances appliquées les résultats intermédiaires peuvent différer mais l'algorithme est Church-Rosser et on arrive finalement toujours au même résultat. L'algorithme laisse des trous dans l'utilisation des variables mais ce n'est pas grave.

r	A	B	C	D	E
t_0	d^A	v_0^B	v_0^C	d^D	v_0^E
t_1	d^A	d^B	v_1^C	v_1^D	v_1^E
t_2	v_2^A	d^B	v_1^C	v_1^D	d^E
t_3	v_3^A	v_3^B	d^C	d^D	d^E
t_4	d^A	v_4^B	v_4^C	v_4^D	d^E

On applique $A \to C$ pour remplacer v_1^C et v_4^C par v_0^C. Ensuite on applique $B \to C$ pour remplacer v_2^C par v_0^C :

r	A	B	C	D	E
t_0	d^A	v_0^B	v_0^C	d^D	v_0^E
t_1	d^A	d^B	v_0^C	v_1^D	v_1^E
t_2	v_2^A	d^B	v_0^C	v_2^D	d^E
t_3	v_3^A	v_3^B	d^C	d^D	d^E
t_4	d^A	v_4^B	v_0^C	v_4^D	d^E

Ensuite on applique $C \to D$ pour remplacer v_1^D, v_2^D et v_4^D par d^D.

r	A	B	C	D	E
t_0	d^A	v_0^B	v_0^C	d^D	v_0^E
t_1	d^A	d^B	v_0^C	d^D	v_1^E
t_2	v_2^A	d^B	v_0^C	d^D	d^E
t_3	v_3^A	v_3^B	d^C	d^D	d^E
t_4	d^A	v_4^B	v_0^C	d^D	d^E

Enfin on applique $DE \to C$ pour remplacer v_0^C par d^C et ensuite $CE \to A$ pour remplacer v_2^A et v_3^A par d^A :

r	A	B	C	D	E
t_0	d^A	v_0^B	v_0^C	d^D	v_0^E
t_1	d^A	d^B	v_0^C	d^D	v_1^E
t_2	d^A	d^B	d^C	d^D	d^E
t_3	d^A	v_3^B	d^C	d^D	d^E
t_4	d^A	v_4^B	d^C	d^D	d^E

On peut encore appliquer $A \to C$ mais déjà à cette étape on remarque que la ligne t_2 ne contient que des constantes : la décomposition est sans perte.

4. Si le *chase* retourne *false* c'est que l'instance r obtenue à la fin de la boucle ne contient pas de tuple $\langle d^{A_1} \dots d^{A_n} \rangle$ composé uniquement de constantes. Or, par construction, pour chaque $S \in \mathbf{R}$ il y a un tuple de r qui ne contient que des constantes et ce sont toujours les mêmes utilisées, donc $\langle d^{A_1} \dots d^{A_n} \rangle \in \pi_{R_1}(r) \bowtie \dots \bowtie \pi_{R_k}(r)$. r et $\pi_{R_1}(r) \bowtie \dots \bowtie \pi_{R_k}(r)$ ne sont pas égales car $\langle d^{A_1} \dots d^{A_n} \rangle$ appartient à la deuxième mais pas à la première.
5. Il manque l'autre direction pour terminer la preuve de correction de l'algorithme : s'il n'y a pas de perte alors l'algorithme répond *true*. C'est vrai mais plus difficile et ça ne sera pas démontré ici.

Références

[1] A. Silberschatz, H. F. Korth, and S. Sudarshan. *Database System Concepts, Sixth Edition*. McGraw-Hill, 2010.