LIFBDW2 – AIDE MÉMOIRE

Licence informatique – session 1 2020–2021

1 Dépendances Fonctionnelles (DF)

1.1 Définitions

Syntaxe $R : X \to Y$ ou simplement $X \to Y$ lu « X *détermine fonctionnellement* Y »

Sémantique $r \models X \to Y \Leftrightarrow \forall t_1, t_2 \in r. t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$

Cas dégénérés $X \to Y$ est *triviale* si $Y \subseteq X$, $X \to Y$ est *standard* si $X \neq \emptyset$

Modèle d'un ensemble de DFs

$$r \models \Sigma \Leftrightarrow \forall f \in \Sigma. r \models f$$

Implication logique de DFs

$$\Sigma \models f \Leftrightarrow \forall r. (r \models \Sigma \Rightarrow r \models f)$$

Fermeture de DFs $\Sigma^+ = \{f \mid \Sigma \models f\}$

1.2 Axiomatisation d'Armstrong

- Le *système d'Armstrong* est l'ensembles des règles $\mathcal{A} = \{\text{Reflex., Aug., Trans.}\}$ (fig. 1).
- Les règles $\{\text{Compo., Decompo., PseudoTrans.}\}$ sont déductibles de $\mathcal{A}$ (fig. 2) et donc correctes.

Preuve formelle une *séquence* $\langle f_0, \ldots, f_n \rangle$ de DFs telles que $f_n = f$ et $\forall i \in [0..n]$ soit $f_i \in \Sigma$; soit f_i est la conséquence d'une règle de $\mathcal{A}$ dont toutes les prémisses $f_0 \ldots f_p$ apparaissent avant f_i dans la séquence. On note $\Sigma \vdash f$ s'il existe une preuve finissant par f avec Σ comme ensemble d'hypothèses.

Fermeture d'un ensemble d'attributs

sémantique $X^+ = \{A \mid \Sigma \models X \to A\}$

syntaxique $X^\star = \{A \mid \Sigma \vdash X \to A\}$

Lemme 1. $\Sigma \models X \to Y \Leftrightarrow Y \subseteq X^+$

Lemme 2. $\Sigma \vdash X \to Y \Leftrightarrow Y \subseteq X^\star$

1.3 Correction et complétude

Le système $\mathcal{A}$ est correct et complet.

Théorème 1 (Correction)**.**

$$\Sigma \vdash X \to Y \Rightarrow \Sigma \models X \to Y$$

Théorème 2 (Complétude)**.**

$$\Sigma \models X \to Y \Rightarrow \Sigma \vdash X \to Y$$

Corollaire 1 (Equivalence des fermetures)**.**

$$X^+ = X^\star$$

Clef de la preuve de complétude Supposant $\Sigma \not\vdash X \to Y$ exhiber une instance r telle que $r \models \Sigma \wedge r \not\models X \to Y$. Avec $X^\star = X_1 \ldots X_n$ et $Z_1 \ldots Z_p = R \setminus X^\star$, la voici :

r	X_1	...	X_n	Z_1	...	Z_p
s	x_1	...	x_n	z_1	...	z_p
t	x_1	...	x_n	y_1	...	y_p

1.4 Algorithmes de fermeture

Algorithme 1 : $Closure(\Sigma, X)$

```
1  CI := X;
2  done := false;
3  while (¬done) do
4      done := true;
5      forall W → Z ∈ Σ do
6          if W ⊆ CI ∧ Z ⊈ CI then
7              CI := CI ∪ Z;
8              done := false;
9  return CI
```

Algorithme 2 : $Closure'(\Sigma, X)$ linéaire

```
1  for W → Z ∈ Σ do
2      count[W → Z] := |W|
3      for A ∈ W do
4          list[A] := list[A] ∪ W → Z
5  closure := X, update := X
6  while (update ≠ ∅) do
7      update := update \ {A}
8      for W → Z ∈ list[A] do
9          count[W → Z] := count[W → Z] − 1
10         if count[W → Z] = 0 then
11             update := update ∪ (Z \ closure)
12             closure := closure ∪ Z
13 return closure
```

Théorème 3. *Les algorithmes 1 et 2 sont corrects pour le calcul de fermeture des DFs standards*[1] *:*

$$Closure(\Sigma, X) = X^\star = X^+ = Closure'(\Sigma, X)$$

Théorème 4 (Résumé)**.**

$$\begin{array}{lll} & Y \subseteq Closure(\Sigma, X) & \\ \equiv & Y \subseteq X^\star & (\textit{théorème 3}) \\ \equiv & \Sigma \vdash X \subseteq Y & (\textit{lemme 1}) \\ \equiv & \Sigma \models X \subseteq Y & (\textit{théorèmes 1 et 2}) \\ \equiv & Y \subseteq X^+ & (\textit{lemme 2}) \end{array}$$

1. l'algorithme 2 peut être modifié pour traiter les DFs non-standards.

$$\frac{Y \subseteq X}{X \to Y}\ \text{Reflex.} \qquad \frac{X \to Y}{WX \to WY}\ \text{Aug.} \qquad \frac{X \to Y \quad Y \to Z}{X \to Z}\ \text{Trans.}$$

FIGURE 1 – Axiomatisation d'Armstrong pour les DFs

$$\frac{X \to Y \quad X \to Z}{X \to YZ}\ \text{Compo.} \qquad \frac{X \to YZ}{X \to Y}\ \text{Decompo.} \qquad \frac{X \to Y \quad WY \to Z}{WX \to Z}\ \text{PseudoTrans.}$$

FIGURE 2 – Règles admissibles pour les DFs

1.5 Relation d'Armstrong

Ensemble des fermés l'ensemble des fermés $Cl(\Sigma)$ d'un ensemble de DFs Σ est défini par $Cl(\Sigma) = \{X^+ \mid X \subseteq R\}$

Définition équivalente $Cl(\Sigma)$ est défini de façon équivalente par $Cl(\Sigma) = \{X \mid X \subseteq R \wedge X^+ = X\}$

L'algorithme 3 répète la construction clef de la preuve de complétude (théorème 2) pour chaque élément de $Cl(\Sigma)$.

Algorithme 3 : *Armstrong*(Σ, R)

```
1  r := ∅ ; i := 0
2  for A ∈ R do  t[A] := 0 ;
3  r := r ∪ {t}
4  for X ∈ Cl(Σ) \ R do
5  |  for A ∈ R do
6  |  |  if A ∈ X then  t[A] := 0 ;
7  |  |  else  t[A] := i ;
8  |  r := r ∪ {t}
9  |  i := i + 1
10 return r
```

Théorème 5. *Soit $r = Armstrong(\Sigma, R)$ une instance obtenue avec l'algorithme 3 :*

$$\forall f.\Sigma \models f \Rightarrow r \models f \text{ et } \forall f.\Sigma \not\models f \Rightarrow r \not\models f$$

2 Autres dépendances

2.1 Dépendances d'Inclusion (DI)

Soient $R, S \in \mathbf{R}$, X et Y des *séquences d'attributs distincts* respectivement de R et de S, avec $|X| = |Y|$.

Syntaxe $R[X] \subseteq S[Y]$

Sémantique $r, s \models R[X] \subseteq S[Y] \Leftrightarrow \forall t_r \in r, \exists t_s \in s.t_r[X] = t_s[Y] \Leftrightarrow \pi_X(r) \subseteq \pi_Y(s)$

Théorème 6. *L'axiomatisation de Casanova pour les DIs $\mathcal{C} =$* {Reflex., Trans., Proj.} *(fig. 3) est correcte et complète.*

Lemme 3. *Les propriétés suivantes d'interactions entre DFs et DIs sont vérifiées :*

- $\{R[XY] \subseteq S[TU], S : T \to U\} \models R : X \to Y$
- $\{R[XY] \subseteq S[TU], R[XZ] \subseteq S[TV], S : T \to U\} \models R[XYZ] \subseteq S[TUV]$

2.2 Dépendances MultiValuées (DMV)

Syntaxe $X \twoheadrightarrow Y$ lu « X *multidétermine* Y »

Sémantique $r \models X \twoheadrightarrow Y$ *ssi* $\forall t_1, t_2 \in r$ tels que $t_1[X] = t_2[X]$ $\exists t_3, t_4 \in r$ tels que :

- $t_3[XY] = t_1[XY]$ et $t_3[R \setminus Y] = t_2[R \setminus Y]$
- $t_4[XY] = t_2[XY]$ et $t_4[R \setminus Y] = t_1[R \setminus Y]$

Lemme 4. *Toute DF est une DMV.*

Théorème 7. *L'axiomatisation* {Reflex., Aug., Complement. , Trans.} *est correcte et complète pour l'inférence des DMVs (fig. 4).*

Théorème 8. *L'axiomatisation précédente à laquelle on ajoute les règles* {Gene., Mix.} *(fig. 5) est correcte et complète pour l'inférence des DFs et des DMVs considérées ensemble.*

3 Normalisation

3.1 Définitions

DF élémentaire une DF $X \to Y$ est *élémentaire* ssi $\forall .X' \subsetneq X \Rightarrow X' \not\to Y$

DF directe une DF $X \to Y$ est *directe* ssi $\nexists Z.X \to Z \wedge Z \not\to X \wedge Z \to Y$ (pas de transivité)

Clé C'est un ensemble d'attributs X tels que $X \to R$

Clé minimale C'est une clé X avec $X \to R$ élémentaire.

Attribut premier Un attribut $A \in R$ est *premier* s'il appartient à *au moins* une clé minimale de R.

Couverture Soient Σ et Γ deux ensembles de DFs, Γ est une couverture de Σ *ssi* $\Gamma^+ = \Sigma^+$.

Algorithme 4 : *Minimize*(Σ)

```
1  G := ∅
   /* Fermeture des parties droites                    */
2  for X → Y ∈ Σ do
3  |  G := G ∪ {X → X+};
   /* Suppression des redondances                      */
4  for X → X+ ∈ G do
5  |  if G − {X → X+} ⊢ X → X+ then
6  |  |  G := G − {X → X+};
7  return G
```

Théorème 9. *Soit $F = Reduce(Minimize(\Sigma))$ donnés par les algorithmes 4 et 5, alors :*

- *F est une couverture de Σ ($F^+ = \Sigma^+$)*
- *F est minimal en nombre de dépendances ($\forall G.G^+ = \Sigma^+ \Rightarrow |F| \leq |G|$)*
- *toutes les parties gauches sont réduites ($\forall X \to Y \in F.\forall .X'.X' \subsetneq X \Rightarrow X' \not\to Y$)*
- *toutes les parties droites sont réduites ($\forall X \to Y \in F.\forall Y' \subsetneq Y.F \setminus \{X \to Y\} \cup \{X \to Y'\} \not\models X \to Y$)*

Pertes d'information et de dépendances

Décomposition Une *décomposition* d'un ensemble d'attributs R est un schéma de base de données $\mathbf{R} = \{R_1, \ldots, R_n\}$ avec $R_i \subseteq R$ et $\bigcup R_i = R$

$$\frac{}{R[X] \subseteq R[X]}\ \text{Reflex.} \qquad \frac{R[X] \subseteq S[Y] \qquad S[Y] \subseteq T[Z]}{R[X] \subseteq T[Z]}\ \text{Trans.} \qquad \frac{R[A_1 \dots A_n] \subseteq S[B_1 \dots B_n]}{R[A_{\sigma(1)} \dots A_{\sigma(k)}] \subseteq S[B_{\sigma(1)} \dots B_{\sigma(k)}]}\ \text{Proj.}$$

Avec σ une permutation d'un sous-ensemble de $\{1 \dots n\}$

FIGURE 3 – Axiomatisation de Casanova pour les DIs

$$\frac{Y \subseteq X}{X \twoheadrightarrow Y}\ \text{Reflex.} \qquad \frac{X \twoheadrightarrow Y}{WX \twoheadrightarrow WY}\ \text{Aug.} \qquad \frac{X \twoheadrightarrow Y}{X \twoheadrightarrow R \setminus XY}\ \text{Compl.} \qquad \frac{X \twoheadrightarrow Y \qquad Y \twoheadrightarrow Z}{X \twoheadrightarrow Z \setminus Y}\ \text{Trans.}$$

FIGURE 4 – Axiomatisation pour les DMVs

Algorithme 5 : *Reduce*(Σ)

```
1  Min := F
   /* Réduction des parties gauches                    */
2  for X → Y ∈ Min do
3      W := X
4      for A ∈ X do
5          if Min ⊨ (W − A) → Y then W := W − {A} ;
6      Min := (Min − {X → Y}) ∪ {W → Y}
   /* Réduction des parties droites                    */
7  for X → Y ∈ Min do
8      W := Y
9      for A ∈ Y do
10         G := (Min − {X → Y}) ∪ {X → (W − A)}
11         if G ⊨ X → Y then W := W − {A};
12         ;
13     Min := (Min − {X → Y}) ∪ {X → W};
14 return Min;
```

Perte d'information Une décomposition $\mathbf{R} = \{R_1, \dots, R_n\}$ est *sans perte d'information* (ou de *jointure*) *ssi* $\forall r. r = \pi_{R_1}(r) \bowtie \dots \bowtie \pi_{R_n}(r)$

Projection de DFs (1) soit $S \subseteq R$ et Σ en un ensemble de DFs, la *projection de* Σ *sur* S est $\Sigma[S] = \{X \to Y \mid X \to Y \in \Sigma^+ \wedge XY \subseteq S\}$.

Projection de DFs (2) La *projection de* Σ *sur un schéma de base de données* $\mathbf{R}$ est $\Sigma[\mathbf{R}] = \bigcup\{\Sigma[R] \mid R \in \mathbf{R}\}$

Perte de dépendances Une décomposition $\mathbf{R}$ est *sans perte de dépendances ssi* $(\Sigma[\mathbf{R}])^+ = \Sigma^+$

3.2 Formes normales

On considère les paires $\langle R, \Sigma \rangle$ formées d'un schéma de relation R et d'un ensemble de DFs Σ sur R.

2FN $\langle R, \Sigma \rangle$ est en 2FN *ssi il n'existe pas de DF non-triviale* $X \to A \in \Sigma^+$ *avec* A *non-premier et* X *sous-ensemble propre d'une clé minimale.*

3FN $\langle R, \Sigma \rangle$ est en 3FN, de façons équivalentes[2] :

- *ssi elle est en 2FN et qu'il n'existe pas d'attribut non-premier qui dépende transitivement d'une clé minimale ;*
- *ssi pour toute DF non-triviale* $X \to A \in \Sigma^+$, A *n'est pas-premier implique que* X *est une (super)clé .*

FNBC $\langle R, \Sigma \rangle$ est en FN de Boyce-Codd (FNBC) *ssi pour toute DF non-triviale* $X \to A \in \Sigma^+$, X *est une (super)clé.*

4FN $\langle R, \Sigma \rangle$ est en 4FN *ssi pour toute DMV non-triviale* $X \twoheadrightarrow A \in \Sigma^+$, X *est une (super)clé.*

2. On préfèrera la seconde définition, sans référence à la 2FN.

Contre-exemples

- $\langle ABC, \{AB \to C, B \to C\} \rangle$ n'est *pas* 2FN.
- $\langle ABC, \{A \to B, B \to C\} \rangle$ est 2FN mais *pas* 3FN.
- $\langle ABC, \{AB \to C, C \to B\} \rangle$ est 3FN mais *pas* FNBC.
- $\langle ABC, \{A \twoheadrightarrow B\} \rangle$ est FNBC mais *pas* 4FN.

Théorème 10. *La 4FN implique la FNBC, la FNBC implique la 3FN, la 3FN implique la 2FN et les inclusions sont strictes.*

Lemme 5. *Toute relation en 3FN avec une unique clef minimale est en FNBC. Toute relation à deux attributs est en FNBC.*

Forme normale d'une base de données Un schéma de base de données $\mathbf{R} = \{R_1 \dots R_n\}$ et un ensemble de dépendances Σ sur ces relations est en 2FN (respectivement 3FN, FNBC, 4FN) ssi $\langle R_i, (\Sigma[R_i]) \rangle$ est en 2FN (resp. 3FN, FNBC, 4FN) pour tout $1 \leq i \leq n$.

3.3 Algorithmes de normalisation

Algorithme 6 : *Synthesis*(Σ, U)

```
   /* 1. minimisation et réduction                     */
1  F := Reduce(Minimize(Σ))
   /* 2. une relation pour chaque DF                   */
2  for X → Y ∈ F do
3      R := R ∪ {XY}
   /* 3. suppression des non-maximaux                  */
4  for R ∈ R do
5      if ∃R'.R ⊊ R' then R := R \ {R};
   /* 4. pertes de jointure                            */
6  Cle := {X | X → U ∧ ∀Z.Z ⊊ X ⇒ Z ↛ U}
7  if ∀R ∈ R. ∄K ∈ Cle.K ⊆ R then
       /* ajout d'une clé si nécessaire                */
8      choisir K ∈ Cle
9      R := R ∪ {K}
10 return R
```

Algorithme 7 : *Decompose*(Σ, U)

```
1  F := Reduce(Minimize(Σ))
2  R = {U};
   /* tant que tout n'est pas en BCNF                  */
3  while (∃R ∈ R.¬BCNF(R)) do
       /* trouver une DF non-triviale non-clef         */
4      let X → Y with Y ⊈ X and F ⊭ X → U;
       /* remplacer R par R1 = X+ et
          R2 = (R \ X+) ∪ X                            */
5      R := R \ {R} ∪ {X+, (R \ X+) ∪ X};
6  return R
```

$$\frac{X \to Y}{X \twoheadrightarrow Y}\ \text{Gene.} \qquad \frac{X \twoheadrightarrow Y, Z \subseteq Y \qquad W \cap Y = \emptyset, W \to Z}{X \to Z}\ \text{Mix.}$$

FIGURE 5 – Axiomes supplémentaires pour les DMVs et DFs considérées ensembles

Théorème 11. *La décomposition obtenue par l'algorithme 6 de synthèse termine en 3FN sans perte d'information ni de dépendances et en FNBC si c'est possible sans perte de dépendances.*

Théorème 12. *La décomposition obtenue par l'algorithme 7 de décomposition termine en FNBC sans perte d'information, avec éventuellement perte de dépendances.*

4 Programmation PL/SQL

4.1 Exceptions

`NO_DATA_FOUND` aucun résultat (dans un `SELECT ...INTO`).

`TOO_MANY_ROWS` plusieurs résultats.

`VALUE_ERROR` érreur numérique.

`ZERO_DIVIDE` division par zéro

`OTHERS` toutes erreurs non interceptées.

4.2 Structures

4.2.1 Branchements

```
IF v1 = c1 THEN
  ...
ELSIF v1 = c2 THEN
  ...
END IF;
```

```
val := CASE c
  WHEN c1 THEN v1
  WHEN c1 THEN v2
  ELSE v_default
END;
```

4.2.2 Boucles

```
LOOP
  instructions;
EXIT [WHEN condition];
  instructions;
END LOOP;
```

```
WHILE condition LOOP
  instructions;
END LOOP;
```

```
FOR variable IN [REVERSE] debut..fin
LOOP
   instructions;
END LOOP;
```

4.3 Déclarations types

4.3.1 Exceptions

```
DECLARE
  MY_EXCEPTION EXCEPTION;
BEGIN
  ...
    RAISE MY_EXCEPTION;
  ...
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    ...
  WHEN MY_EXCEPTION THEN
    ...
  WHEN OTHERS THEN --optionnel
    ...
END;
```

4.3.2 Procédures

```
CREATE OR REPLACE PROCEDURE
  nomP(arg1 IN type1, arg2 IN OUT type2,...)
IS
BEGIN
  DECLARE
  ...
  BEGIN
  ...
  END;
END;
```

4.3.3 Fonctions

```
CREATE OR REPLACE FUNCTION
  nomF(arg1 IN type1, arg2 IN type2,...)
  RETURN typeRetour IS
BEGIN
  DECLARE
  ...
  BEGIN
  ...
  END;
END;
```

4.3.4 Curseurs

```
DECLARE
  CURSOR c(param type) IS
    SELECT...
    FROM ...
    WHERE ...;
BEGIN
  FOR v_c in c(10) LOOP
          ...
  END LOOP;
END;
```

4.3.5 Triggers

```
CREATE [OR REPLACE ] TRIGGER trigger_name
 {BEFORE | AFTER}
 {INSERT [OR] | UPDATE [OR] | DELETE}
 [OF col_name] ON table_name
 [REFERENCING OLD AS o NEW AS n]
 [FOR EACH ROW]
 WHEN (condition)

BEGIN
  ...
   IF UPDATING('col') THEN
  ...
END;
```

4.3.6 Vues

```
CREATE [OR REPLACE ] VIEW view_name AS
 /* SELECT QUERY */;
```