CSI2532 - Labo 9 - Normalisation

Aperçu

L'objectif du tutoriel est de vous donner pratique sur des dépendances fonctionnelles (FD) et normalisation.

Règles de dépendance fonctionnelle

Règle réflexive	Règle d'augmentation	Règle de transitivité
if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$	if $\alpha \rightarrow \beta$, then $\gamma\alpha \rightarrow \gamma\beta$	if $\alpha \rightarrow \beta$ and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$

Règles dérivées

Règle de l'Union	Règle de décomposition	Règle de pseudo-transitivité
if $\alpha \rightarrow \beta$, $\alpha \rightarrow \gamma$, then $\alpha \rightarrow \beta\gamma$	if $\alpha \rightarrow \beta\gamma$, then $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$	if $\alpha \rightarrow \beta$ and $\gamma\beta \rightarrow \delta$, then $\alpha\gamma \rightarrow \delta$

Utilisez les règles ci-dessus pour prouver une dépendance. Ou trouver un contre-exemple pour prover un ambiguïté de sorte que α ne puisse pas identifier de manière unique β.

Reflexive rule

if $\beta \subseteq \alpha$,
then $\alpha \rightarrow \beta$

Augmentation rule

if $\alpha \rightarrow \beta$,
then $\gamma\alpha \rightarrow \gamma\beta$

Transitivity rule

if $\alpha \rightarrow \beta$ and $\beta \rightarrow \gamma$
then $\alpha \rightarrow \gamma$

Union rule

if $\alpha \rightarrow \beta$, $\alpha \rightarrow \gamma$
then $\alpha \rightarrow \beta\gamma$

Decomposition rule

if $\alpha \rightarrow \beta\gamma$
then $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$

Pseudo-transitivity rule

if $\alpha \rightarrow \beta$ and $\gamma\beta \rightarrow \delta$
then $\alpha\gamma \rightarrow \delta$

Super clé

Une super clé où K → R. Pour prouver, montrer la fermeture de K (K+) inclut toutes les relations (K+ = R).

Test superkey K

Test α = K

Check α+ → R

Clé candidat

Une clé candidate est une super clé minimisée. Il ne devrait donc pas y avoir $\alpha \subset K$ où $\alpha+ = R$.

Test α candidate K

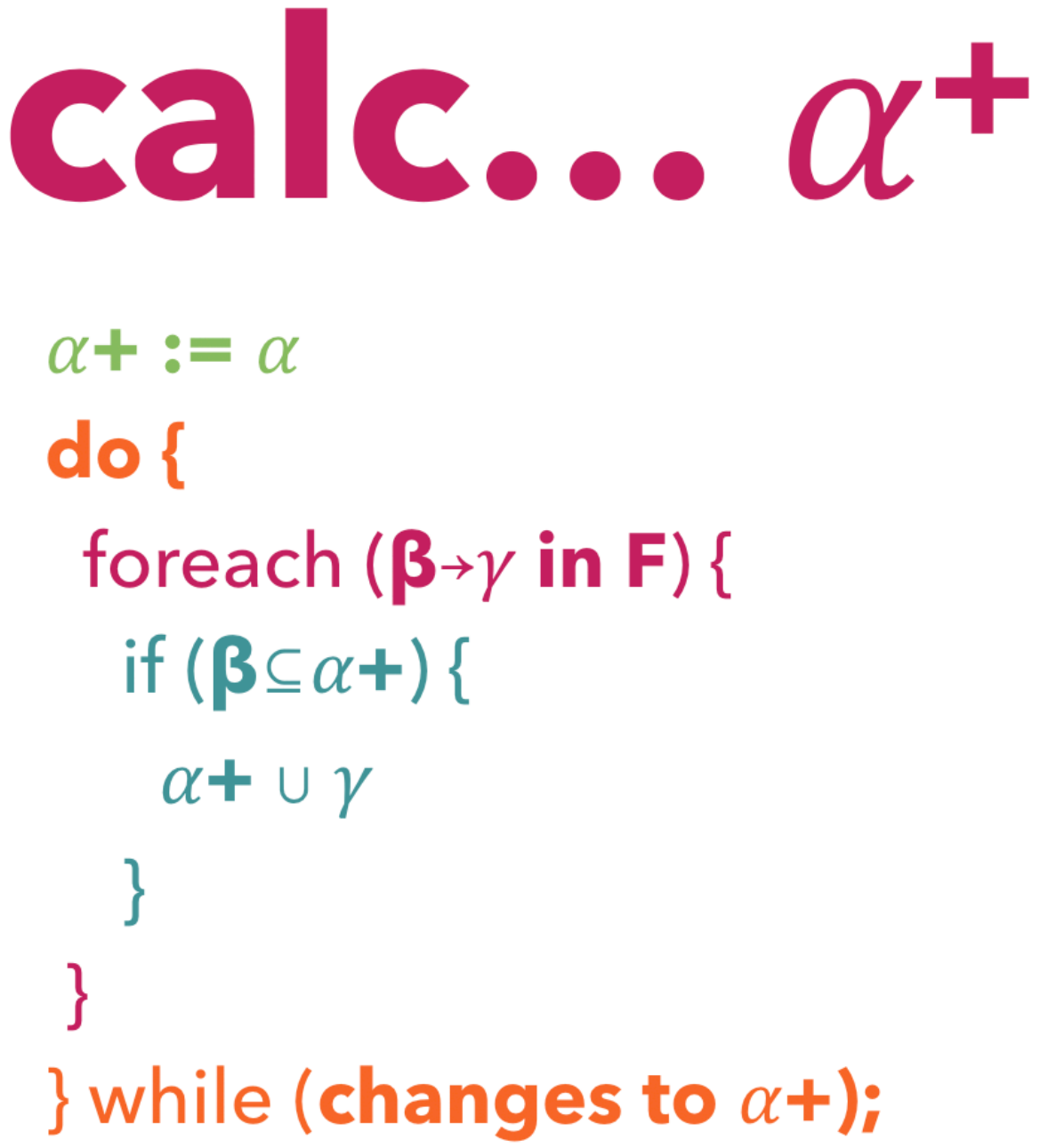
Test $\alpha+ \rightarrow K$

$\exists\ \beta \subset \alpha$ test ! $\beta+ \rightarrow R$

α size n, and β size n-1

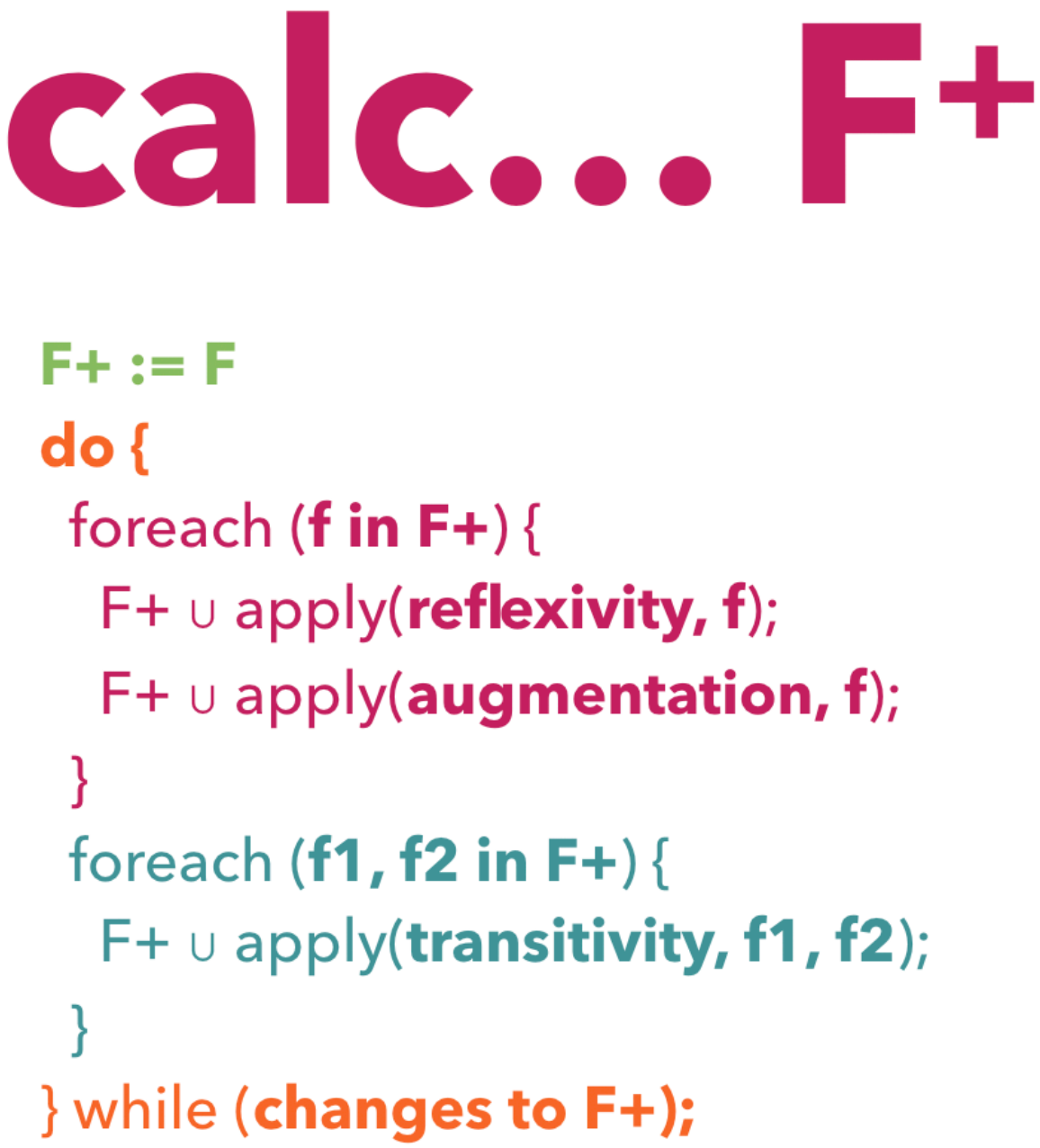
α+ (Fermeture de l'ensemble d'attributs)

Pour calculer $\alpha+$ determiner toutes les dépendances fonctionnelles $\beta \rightarrow \gamma$, si $\alpha \subseteq \beta$ alors ajouter γ à a+



F+ (fermeture des dépendances fonctionnelles)

Voici l'algorithme original pour calculer F+.



Mais, nous pouvons utiliser α+. Calculez α+ pour chaque combinaison, puis créez toutes les combinaisons de X → Y.

L'algorithme est

```
∃ γ⊆R find γ+
∃ S⊆γ+ output FD γ⊆
```

Compute F+

R=(A, B, C)

F={A→B, B→C}

∅+ = ∅ so ∅→∅

(A)+ = A
= AB
= ABC

(B)+ = B
= BC

(C)+ = C

(AB)+ = AB
= ABC

(AC)+ = AC
= ABC

(BC)+ = BC

(ABC)+ = ABC

Compute F+

R=(A, B, C)

F={A→B, B→C}

∅→∅

A→∅, A→A, A→B, A→C, A→AB, A→AC, A→BC, A→ABC

B→∅, B→B, B→C, B→BC

C→∅, C→C

AB→∅, AB→A, AB→B, AB→C, AB→AB, AB→AC, AB→BC, AB→ABC

AC→∅, AC→A, AC→B, AC→C, AC→AB, AC→AC, AC→BC, AC→ABC

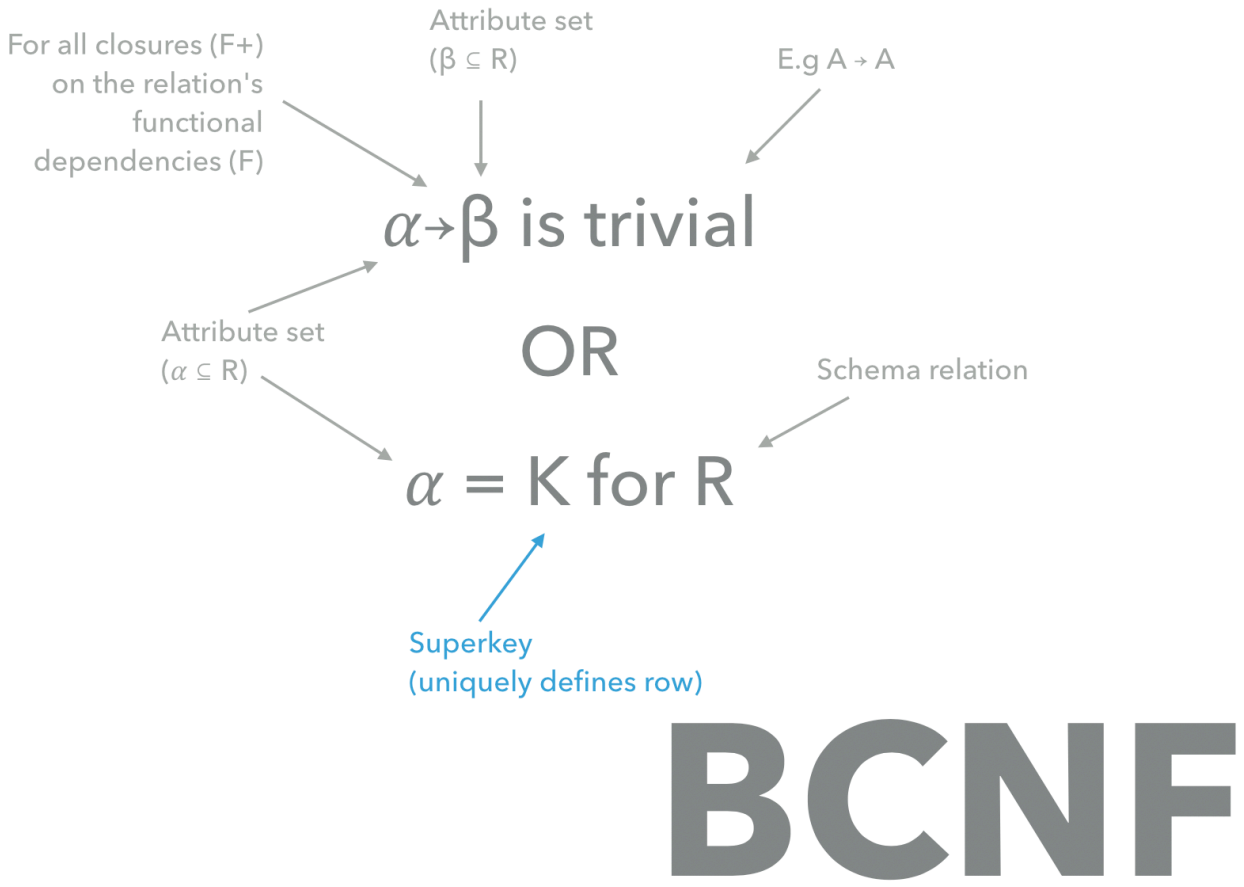
BC→∅, BC→B, BC→C, BC→BC

ABC→∅, ABC→A, ABC→B, ABC→C, ABC→AB, ABC→AC, ABC→BC, ABC→ABC

Test BCNF

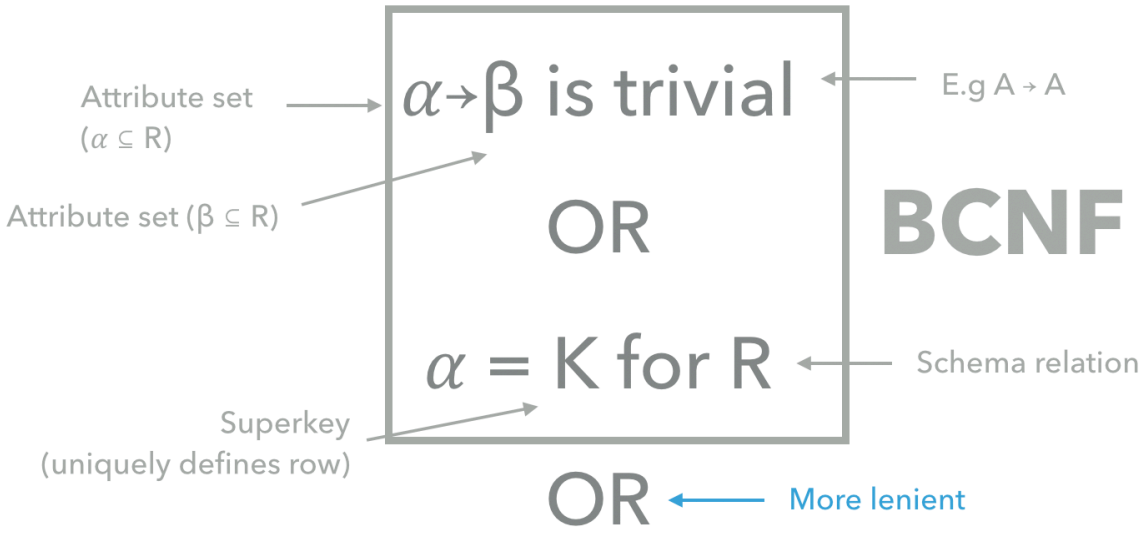
Pour tester BCNF, vous devez trouver toutes les fermetures de F (F+). Si vous trouvez une

relation $\alpha \rightarrow \beta$ où $\alpha \not\subset R$ alors la relation n'est PAS BCNF.



Test 3NF

Si ce n'est pas BCNF, trouver une relation $\alpha \rightarrow \beta$ où un attribut dans β (mais pas dans α, donc β - α) n'est pas dans K (n'importe quelle clé candidate) et ce n'est pas 3NF.



$$\forall A \in \beta\text{-}\alpha: A \subseteq K$$

3NF

Couverture canonique

Continuez jusqu'à ce que Fc ne change pas.

1. Appliquer la règle de l'union
2. Supprimez tout attribut étranger lorsque vous regardez tous les $\alpha \rightarrow \beta$

Canonical Cover

```
Fc := F
do {
 foreach (f1, f2 in Fc) {
  // α1→β1 and α1→β2
  // into α1→β1β2
  reduce(union, f1, f2);
 }
 foreach (α→β in Fc) {
  if with Fc find extraneous A in α or β {
   delete(A, α→β)
  }
 }
} while (changes to Fc);
```

Basé sur des tests externes

Testing for extraneous A in $\alpha \rightarrow \beta$

Remove "left"

$\gamma = \alpha - \{A\}$

Check if $\gamma \rightarrow \beta$
(Check $\beta \subseteq \gamma+$)

Remove "right"

$F' = (F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$

Check $A \in \alpha+$ under F'

Attribut étranger A dans α

Après avoir supprimé l'attribut, essayez de prouver la dépendance fonctionnelle originalle existe toujours.

Considérer

```
R=(A, B, C, D)
F={
  AB→C,
  A→D,
  D→C
}
```

Pouvons-nous (en toute sécurité) supprimer «B» dans AB → C?

Nous devons prouver que A → C peut être dérivé des autres dépendances fonctionnelles.

OUI nous pouvons (en utilisant les règles). A → D et D → C donc A → C. "B" est donc étranger.

OU OUI (en utilisant α+ avec F). (A)+ = ADC donc oui A → C.

Attribut étranger A dans β

Après avoir supprimé l'attribut, essayez de montrer que vous pouvez *recréer* la dépendance fonctionnelle la plus forte basée sur la plus faible.

Considérer

```
R=(A, B, C, D)
F={
  AB→CD,
  A→C
}
```

Pouvons-nous (en toute sécurité) supprimer «C» dans AB→CD?

Nous devons prouver qu'avec AB → D (une allégation plus faible) peut nous permettre de revenir à notre affirmation plus forte AB → CD.

OUI nous pouvons (en utilisant des règles). Avec A→C, nous l'augmentons à AB→C, puis en union avec l'AB→D (le plus faible règles), nous revenons à AB → CD.

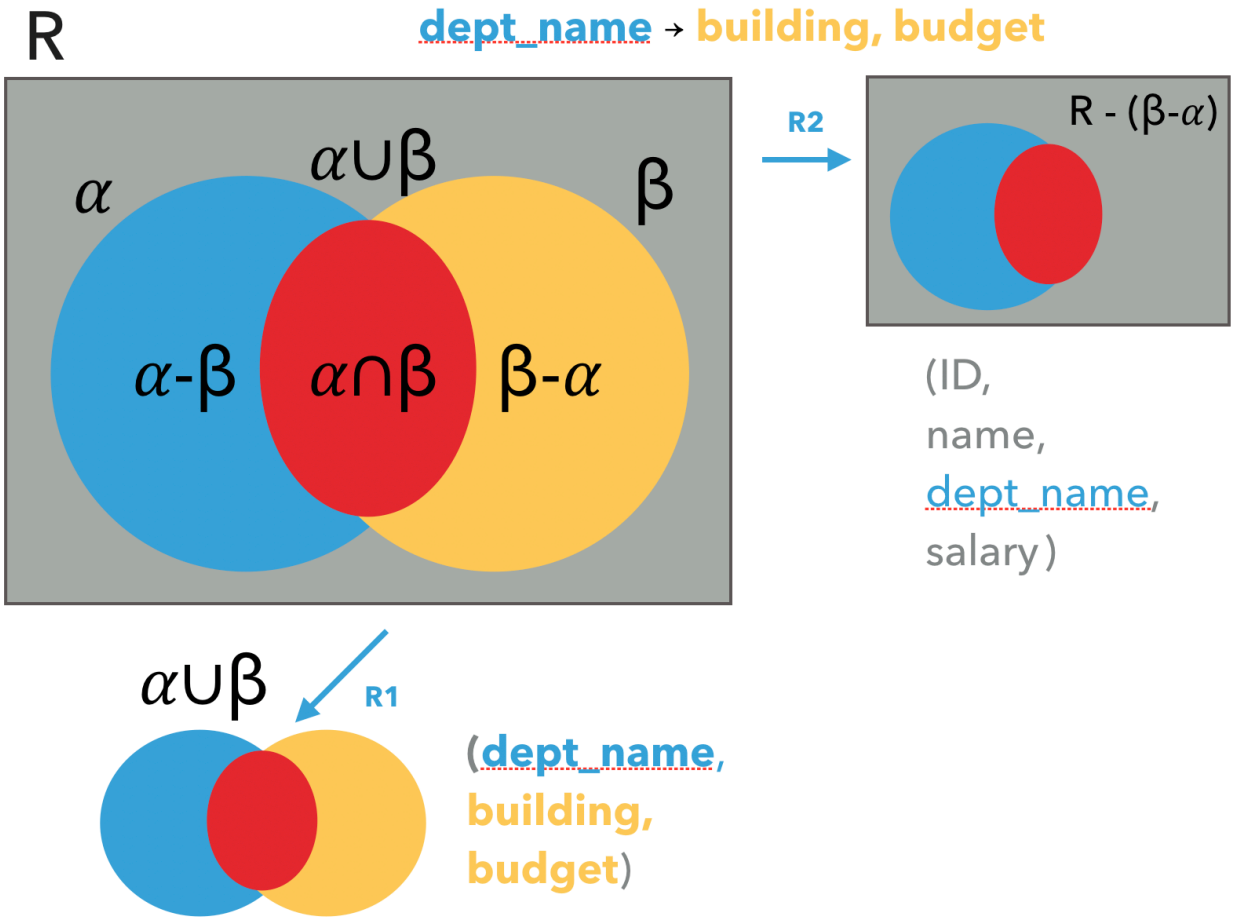
OU OUI (calcul de β+ en utilisant F'). Calculer (AB)+ = ABCD à partir de F' = {AB→D, A→C} qui contient AB→CD donc "C" est étranger.

Décomposition BCNF

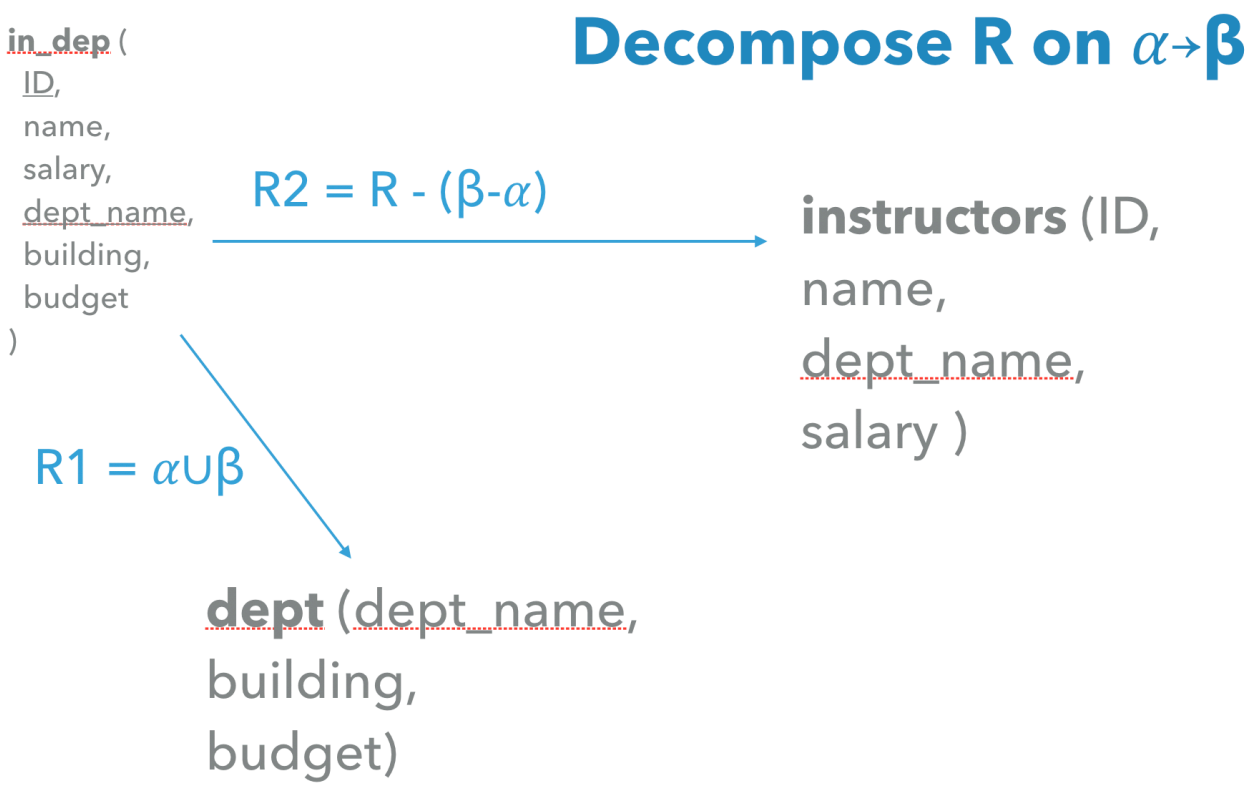
result := {*R* };
done := false;
compute F^+;
while (not *done)* **do**
    if (there is a schema R_i in *result* that is not in BCNF)
        then begin
            let $\alpha \rightarrow \beta$ be a nontrivial functional dependency that
            holds on R_i such that $\alpha \rightarrow R_i$ is not in F^+,
                and $\alpha \cap \beta = \varnothing$;
            result := (*result* – R_i) ∪ (R_i – β) ∪ (α, β);
        end
        else *done* := **true;**

Note: each R_i is in BCNF, and decomposition is lossless-join.

Pour la relation qui échoue BCNF



Par exemple,



Q1a: Test des formes normales

Considérez la relation et les dépendances fonctionnelles suivantes.

```
R=(A,B,C,D)
F={
  AB → C,
  C → D,
  D → A
}
```

a. Liste toutes les clés candidates de R.

b. Est-ce que R est dans 3NF? BCNF?

Q1b: Test des formes normales

Considérez la relation et les dépendances fonctionnelles suivantes.

```
R=(A,B,C,D)
F={
  A → B,
  B → C,
  C → D,
  D → A
}
```

a. Liste toutes les clés candidates de R.

b. Est-ce que R est dans 3NF? BCNF?

Q1c: Test des formes normales

Considérez la relation et les dépendances fonctionnelles suivantes.

```
S=(A,B,C,D)
F={
  B → C,
  C → A,
  C → D
}
```

a. Liste toutes les clés candidates de R.

b. Est-ce que R est dans 3NF? BCNF?

Q1d: Test des formes normales

Considérez la relation et les dépendances fonctionnelles suivantes.

```
R=(A,B,C,D)
F={
  ABC → D,
  D → A
}
```

a. Liste toutes les clés candidates de R.

b. Est-ce que R est dans 3NF? BCNF?

Q1e: Test des formes normales

Considérez la relation et les dépendances fonctionnelles suivantes.

```
R=(A,B,C,D)
F={
  A → C,
  B → D
}
```

a. Liste toutes les clés candidates de R.

b. Est-ce que R est dans 3NF? BCNF?

Q2a: Test de la dépendance fonctionnelle

Considérez la relation et les dépendances fonctionnelles suivantes.

```
R=(A,B,C,D,E,F)
F={
  AB → C,
  BC → AD,
  D → E,
  CF → B
}
```

Est AB → D valid? Si oui, montrez une preuve formelle; sinon, donnez un contre-exemple.

Q2b: Test de la dépendance fonctionnelle

Considérez la relation et les dépendances fonctionnelles suivantes.

```
R=(A,B,C)
F={
  AB → C
}
```

Est A → C valid? Si oui, montrez une preuve formelle; sinon, donnez un contre-exemple.

Q2c: Test de la dépendance fonctionnelle

Considérez la relation et les dépendances fonctionnelles suivantes.

```
R=(A,B,C)
F={
  AB → C
}
```

Est B → C valid? Si oui, montrez une preuve formelle; sinon, donnez un contre-exemple.

Q2d: Test de la dépendance fonctionnelle

Considérez la relation et les dépendances fonctionnelles suivantes.

```
R=(A,B,C)
F={
  AB → C
}
```

Est A → C OR B → C valid? Si oui, montrez une preuve formelle; sinon, donnez un contre-exemple.

Q3: Couverture canonique

Calculer une couverture canonique pour

```
F={
  B → A,
  D → A,
  AB → D
}
```

Q4: Décomposition BCNF

Produire une décomposition BCNF de R.

```
R = ABCDEFGH
F = {
  ABH → C,
  A → DE,
  BGH → F,
  F → ADH,
  BH → GE
}
```