Université Rennes I - IUT de Lannion Département Informatique
Bases de Données

Aide mémoire Tutorial D

LMD (Langage de Manipulation des Données)

■ UNION
`s = r1 UNION r2`
— `s`, `r1` et `r2` sont des relations de même type
— la valeur de `s` est l'ensemble des n-uplets, où chacun d'entre eux est soit un n-uplet de `r1`, soit un n-uplet de `r2` (ou des deux).

■ INTERSECTION
`s = r1 INTERSECT r2`
— `s`, `r1` et `r2` sont des relations de même type
— la valeur de `s` est l'ensemble des n-uplets, où chacun d'entre eux est un n-uplet de `r1` et de `r2`.

■ DIFFERENCE
`s = r1 MINUS r2`
— `s`, `r1` et `r2` des relations de même type
— la valeur de `s` est l'ensemble des n-uplets, où chacun d'entre eux est un n-uplet de `r1` et n'est pas un n-uplet de `r2`.

■ RESTRICTION
`s = r WHERE c`
— `s` et `r` sont de même type
— `c` est une condition, expression possiblement vraie sur un ou plusieurs n-uplet de `r`
— la valeur de `s` est l'ensemble des n-uplets de `r` qui vérifient (chacun) la condition `c`.

■ PROJECTION
`s = r {a1, a2, ..., an}`
`= r {ALL BUT b1, ... , bm}`
— `{a1, a2, ..., an}` et `{b1, ... , bm}` sont des sous-ensembles disjoints de l'ensemble des attributs de `r`
— l'entête de `s` est le sous-ensemble de l'entête de `r` donné par `{a1, a2, ..., an}`
— la valeur de `s` est l'ensemble des n-uplets de `r` dont on retire les attributs de `{b1, ... , bm}` (ou dont on n'affiche que les attributs de `{a1, a2, ..., an}`).

■ INSERTION : (exemples)
```
INSERT produit RELATION {
   TUPLE { no_prod 'P8',
      qte_stock 150, categ 's' } };
```
ou par affectation
```
produit := produit UNION RELATION{TUPLE{...}, ...}
```

■ JOINTURE
`s = r1 join r2`
— `r1` et `r2` sont des relations dont les types ont une intersection non-nulle (un attribut ou plusieurs en commun : même non, même type).
— le type de `s` est l'union des types de `r1` et de `r2`.
— la valeur de `s` est l'ensemble des n-uplets de `r1` combinés avec l'ensemble des n-uplets de `r2` qui ont une valeur qui correspond sur l'ensemble des attributs en commun.
— on appelle l'égalité des attributs en commun la condition de jointure

■ AGREGATS
`EXTEND r : {x:=f1, y:=f2}`
— où `f1` et `f2` sont des formules arbitraires, basées par exemple sur des fonctions d'agrégat
— permet d'étendre l'entête d'une relation avec ces formules.

`SUMMARIZE r BY {b1,...,bm}:`
`{x:=f1, ..., z:=fn}` où
— `r` est une relation
— `b1,...,bm` sont des noms d'attributs de `r`, sur lesquels on construits les regroupements (dans l'ordre de leur énumération)
— `f1, ...,fn` sont des formules arbitraires impliquant des fonctions d'agrégat
Les fonctions d'agrégats peuvent être AVG, SUM, COUNT, MAX, MIN, *etc.*

Une commande bien utile pour déclarer des variables relationnelles intermédiaires (et locales) :
```
WITH (pv := produit
   WHERE couleur = DCOULEUR('Vert'),
         ftpv := fourniture JOIN pv) :
SUMMARIZE ftpv BY {nofour} :
   {produit := count()}
```

■ SUPPRESSION : (exemple)
```
DELETE produit
   WHERE no_prod = 'P1';
```

LDD (Langage de Description des données)

■ Création de relation :
```
VAR nom_relation BASE RELATION{
   nom_attr1 type_attr1,
   nom_attr2 type_attr2, ... }
   KEY {liste_attr_dans_clé1}
   KEY {liste_attr_dans_clé2};
```

■ Clé étrangère :
```
CONSTRAINT relation_référençante_fk1
   relation_référençante{liste_attributs} <=
       relation_référençée{liste_attributs}
```

■ Suppression :
`DROP VAR nom_relation;`
Quand il s'agit d'un type, remplace `VAR` par `TYPE`.
De même, pour `CONSTRAINT`.