Bases de données : un kit de survie

1 Schémas relationnels

Ils peuvent être donnés sous différentes syntaxe/présentation, et donnent la structure d'une base, avec les différentes tables, chacune possédant différents attributs/champs avec son type/domaine.

1. Une base constituée d'une table de films :

   `films( nom:text, realisateur:text, annee:entier )`

films
`nom` : texte
`realisateur` : texte
`annee` : entier

2. Une base constituée d'une table d'élèves et d'une table de classes :

   `eleves( nom:texte, prenom:texte, classe:entier )`
   `classes( id:entier, nom:texte, prof_maths:texte )`

eleves
`nom` : texte
`prenom` : texte
`classe` : entier

classes
`id` : entier
`nom` : texte
`prof_maths` : texte

3. Une base constituée de trois tables communes/départements/régions :

   `communes( id:entier, dep:entier, nom:texte, pop:entier )`
   `departements( id:entier, reg:entier, nom:texte )`
   `regions( id:entier, nom:texte )`

communes
`id` : entier
`dep` : entier
`nom` : texte
`pop` : entier

departements
`id` : entier
`reg` : entier
`nom` : texte

regions
`id` : entier
`nom` : texte

Les champs soulignés sont des clés primaires, utilisés dans les jointures de tables.
Voici quelques lignes des différentes tables :

nom	realisateur	annee
'Star Wars'	'George Lucas'	1977
'Le gendarme de St Tropez'	'Jean Girault'	1964
...	...	...

nom	prenom	classe
'aze'	'qsd'	945
'rty'	'fgh'	933
...	...	...

id	nom	prof_maths
933	'MPBG'	'Appel"
945	'PSIG'	'Gonnord'
...	...	...

id	dep	nom	pop
79147	79	'Largeasse'	726
69266	69	'Villeurbanne'	145150
79037	79	'Boësse'	1
...	...	...	

id	reg	nom
69	82	'Rhône'
42	82	'Loire'
79	54	'Deux-Sèvres'
...	...	...

id	nom
82	'Rhône-Alpes'
54	'Poitou-Charentes'
...	...

2 Les requêtes SQL d'interrogation

2.1 Sélection et projection

On sélectionne tels champs des lignes vérifiant telles conditions :

```
SELECT champ_1, ..., champ_n FROM table WHERE condition
```

– Si on met `*` pour les champs, ils sont tous sélectionnés.
– Si on ne met pas de condition `WHERE`, toutes les lignes sont sélectionnées.
– La condition peut utiliser les connecteurs `AND, OR` et la fonction `NOT`
Exemples :

1. Quels sont les films de l'année 1977 ?

   ```
   SELECT * FROM films WHERE annee = 1977
   ```

2. Constituer la table des noms/populations des communes des Deux-Sèvres de plus de 10000 habitants :

   ```
   SELECT nom, pop FROM communes WHERE dep = 79 AND pop >= 10000
   ```

3. Quel sont les noms des élèves de 945 ?

   ```
   SELECT nom FROM eleves WHERE classe = 945
   ```

2.2 Jointure de tables

On peut lier deux tables via une « clé primaire » d'une table (deux éléments distincts de la table ont toujours des clés primaires distinctes) qui est pointé par un champ d'une autre table. Plutôt que de construire le produit cartésien des deux tables, on ne retient que les couples de lignes qui sont cohérents.

```
SELECT * FROM table1 JOIN table2 ON cle1 = cle2
```

– Si deux tables ont des attributs partageant le même nom, on préfixe par le nom de la table.
– Certains conseillent même, pour des raisons de lisibilité, de le faire même quand ce n'est pas nécessaire.
– On peut joindre trois tables ou plus sans problème !
– Autre syntaxe via le produit cartésien : `SELECT * FROM table1, table2 WHERE cle1=cle2`

Exemples (réécrivez les schémas relationnels sur cette page !) :

1. Constituer la table d'appel d'Appel [1]

   ```
   SELECT eleves.nom, prenom
   FROM eleves JOIN classes
   ON classe = id
   WHERE prof_maths = 'Appel'
   ```

2. Quel est le nom et la population des villes de plus de 10000 habitants dans la région Limousin ?

   ```
   SELECT communes.nom, pop
   FROM communes JOIN departements JOIN regions
   ON communes.dep = departements.id AND departements.reg = regions.id
   WHERE regions.nom = 'Limousin' AND pop > 10000
   ```

2.3 Fonction d'agrégation

On commence par regrouper les entrées selon certains attributs (même numéro de département) par exemple. Ensuite, pour chaque groupe, on va appliquer une fonction (symétrique) sur certain des arguments.

```
SELECT A1,..,Ak,f1(B1),...,fi(Bi)
FROM table
GROUP BY A1,...,Ak
```

On récupère ainsi une table contenant autant d'éléments qu'il y a de groupes. Attention, si on projette sur autre chose que des champs ayant permis de regrouper ou qui ont été calculés par agrégation, on obtient probablement n'importe quoi (mais pas d'erreur), sauf si ces champs sont uniques par groupement (voir les noms de régions, plus loin).
– En plus de la sélection en amont (WHERE), on peut réaliser une sélection en aval des calculs par agrégation via HAVING.

1. Désolé...

– Les fonctions d'agrégation sont : COUNT (compter le nombre de lignes du groupe), SUM (sommer un champ sur tous les éléments du groupe), MAX et MIN (trouver le maximum ou le minimum d'un champ sur un groupe) et AVG (pour average : moyenne d'un champ sur un groupe).

1. Nombre de films réalisés par chaque réalisateur ?

```
SELECT realisateur, COUNT() FROM films GROUP BY realisateur
```

2. Population des différents départements ?

```
SELECT dep, SUM(pop) FROM communes GROUP BY dep
```

3. La même chose mais par région, avec le nom de la région, et par ordre décroissant de population :

```
SELECT rg.nom, SUM(pop) AS totale
FROM communes AS cm JOIN departements AS dp JOIN regions AS rg
ON cm.dep = dp.id AND dp.reg = rg.id
GROUP BY rg.nom
ORDER BY totale DESC
```

On a noté ici l'utilisation d'alias (renommage) pour les tables ou les champs. Le AS est optionnel.

4. Nom des professeurs ayant au moins 60 élèves (ils peuvent avoir plusieurs classe !) :

```
SELECT prof_maths, COUNT() AS nb_eleves
FROM eleves JOIN classes
ON classe = id
GROUP BY prof_maths
HAVING nb_eleves >= 60
```

5. Table des régions constituées d'au moins 6 départements ?

```
SELECT rg.nom, COUNT() AS nb_dep
FROM departements AS dp JOIN regions AS rg
ON dp.reg = rg.id
GROUP BY rg.nom
HAVING nb_dep >= 6
```

6. Donner la liste des départements par ordre croissant de population moyenne des villes.

```
SELECT dp.nom, AVG(pop) AS moyenne
FROM communes AS cm JOIN departements AS dp
ON cm.dep = dp.id
GROUP BY dp.nom
ORDER BY moyenne
```

7. Population de la plus grande ville de chaque département :

```
SELECT departements.nom, MAX(pop) AS pop_maxi
FROM communes JOIN departements
ON communes.dep = departements.id
GROUP BY departements.nom
```

8. Pour chaque département, déterminer la ville avec la plus grande population :

```
SELECT bar, nom, pop_maxi FROM
   (SELECT departements.id AS foo, departements.nom AS bar, MAX(pop) AS pop_maxi
    FROM communes JOIN departements
    ON communes.dep = departements.id
    GROUP BY departements.id)
JOIN communes
ON communes.dep = foo
WHERE pop = pop_maxi
```

2.4 Mais aussi

– On peut faire l'union, l'intersection ou la différence de deux tables, via les opérateurs UNION, INTERSECT, EXCEPT. Avec la syntaxe moisie de SQL, c'est assez pénible, mais bon...
– On peut limiter un résultat aux k premières lignes, via `LIMIT(k)` ; c'est par exemple une façon d'obtenir la ligne réalisant le maximum de tel attribut, après avoir ordonné par ordre décroissant (`ORDER BY atr DESC`).

3 Algèbre relationnelle

Il s'agit d'une (vague) formalisation mathématique du langage SQL. On manipule des relations, c'est-à-dire des n-uplets. Un exemple de formule de l'algèbre relationnelles est :

$$\gamma_{MAX(c)}\left({}_{dep.id}\gamma_{COMPTAGE:c}\left(dep \underset{J_{CD}}{\bowtie} (\sigma_{pop\geqslant 10^4}(com))\right)\right)$$

Qui dit en français : « donne moi le nombre maximum de communes de plus de dix mille habitants qu'on trouve dans les départements français ». En SQL, on l'aurait écrit :

```
SELECT MAX(c) FROM
 (SELECT departements.nom, count() AS c
  FROM communes JOIN departements
  ON communes.dep = departements.id
  WHERE pop > 10000
  GROUP BY departements.id)
```

L'objectif est de faire la triple correspondance entre :
– une requête racontée en français ;
– une requête en SQL ;
– une formule de l'algèbre relationnelle.
Il y a essentiellement quatre choses à connaître :

1. La **sélection** : `SELECT * FROM t WHERE c` devient
$$\sigma_c(t)$$
2. La **projection** : `SELECT A1,...,Ak FROM t` devient
$$\Pi_{A_1,\ldots,A_k}(t)$$
3. La **jointure** : `t1 JOIN t2 ON c` devient
$$t_1 \underset{c}{\bowtie} t_2$$
4. Le **calcul d'agrégats** : `SELECT A1,..,Ak,f1(B1),...,fi(Bi) FROM table GROUP BY A1,...,Ak` devient
$${}_{A_1,\ldots,A_k}\gamma_{f_1(B_1),\ldots,f_i(B_i)}table$$

Attention, la composition entre opérateurs est souvent implicite (on omet les ronds et les parenthèses). Exemples :

1. La traduction de `SELECT * FROM films WHERE annee = 1977` est
$$\sigma_{an\geqslant 1977}(films)$$
2. La traduction de `SELECT nom, annee FROM films` est
$$\Pi_{nom,an}(films)$$
3. La traduction de `SELECT nom, pop FROM communes WHERE dep = 79` est
$$\Pi_{nom,pop}\sigma_{dep=79}(com)$$
4. La traduction de
```
SELECT eleves.nom, prenom
FROM eleves JOIN classes
ON classe = id
WHERE prof_maths = 'Appel'
```
est :
$$\Pi_{el.nom,pre}\sigma_{prof='Appel'}\left(eleves \underset{classe=id}{\bowtie} classes\right)$$
5. La traduction de `SELECT dep, SUM(pop) FROM communes GROUP BY dep` est
$${}_{dep}\gamma_{SUM(pop)}com$$