Bases de données – cours 4
Construction de requêtes en SQL

Catalin Dima

Requêtes SQL et langage naturel

- Énoncés en langage naturel.
- Traduction en SQL ?
  - Correspondance entre syntagmes/phrases et opérations en algèbre relationnelle.
  - Identification des opérandes : relations, attributs.
  - Syntagmes définissant des opérations sur les tuples.
  - Ambigüité du langage naturel.
- Faisabilité d'un énoncé ? Limites de l'algèbre relationnelle.

Principes de traduction d'un énonce en SQL

- Tout énoncé demande de construire une nouvelle table (relation), dont les (types d') attributs doivent se trouver dans l'énoncé.
  - Les noms des attributs sont assez souvent repris des noms des attributs de tables existantes.
  - Le schéma de la table résultat peut impliquer plusieurs copies d'un attribut d'une des tables existantes (attention ! mais pas la même valeur pour les deux copies !).
- L'énoncé doit indiquer (souvent implicitement) quels attributs sont à utiliser pour construire la solution, ainsi que leur lien avec les attributs de la table à construire.
- Toutefois, assez souvent, les tables auxquelles ces attributs appartiennent ne sont pas données explicitement.
  - Cela reste donc à la charge de l'*opérateur* d'identifier les tables à inclure dans la requête.
  - Au cas où plusieurs tables possèdent les mêmes (noms d') attributs, cela sous-entend la nécessité d'utiliser toutes ces tables pour construire la solution.
  - La base de données à laquelle ces tables appartiennent est assez souvent sous-entendue.

Principes de traduction d'un énonce en SQL (2)

- La requête inclut les conditions que doivent satisfaire les attributs impliqués dans la construction de la solution.
  - Reste à la charge de l'opérateur d'identifier le de combien de tuples de chacune des tables existantes il a besoin pour construire la solution.
  - Les opérations à appliquer sur les tables existantes sont aussi à identifier dans l'énoncé.
  - Des constructions de relations auxiliaires peuvent être souvent utiles – modularisation de l'énoncé.
- L'énoncé peut indiquer aussi des opérations d'agrégation à effectuer sur les valeurs de certains attributs.
  - Une opération par attribut.
  - Certains contraintes peuvent être imposés sur les valeurs à prendre en considération dans l'agrégation.

Base de données pour les exemples

Base de donnée *Entrepôt* :

- Table *Quincaillerie(id, nom, prix, nbre-par-boite)*.
- Table *Peinture(id, nom, prix, poids)*.
- Table *Électricité(id, nom, prix, voltage)*.
- Table *Client(id, nom, adresse)*.
- Table *Commande(no, date, id-client)*.
- Table *DetailCde(no, nom-produit, qté)*.

Retrouver les attributs et tables impliquées

Les attributs sont donnés explicitement :

> *Afficher les noms des clients qui ont acheté des produits de quincaillerie plusieurs fois, et les dates et id des commandes respectives.*

- Nom client ds *Client*.
- Date commande ds *Commande*.
- Id commande ds *Commande* et *DetailCde* – lequel utiliser ?
  - A-t-on besoin des attributs de la table *Commande* ?
  - A-t-on besoin des attributs de la table *DetailCde* ?
- “acheté des produits plusieurs fois” : il faut retrouver le détail de chaque commande, pour vérifier si le même produit apparaît dans deux commandes différentes !
- “produit de quincaillerie” : utiliser la table *Quincaillerie*.

Retrouver les attributs et tables impliquées

- S'il y a ambigüité, c'est que toutes les tables qui possèdent l'attribut respectif doivent être inclues.

  Retrouver les prix de tous les produits achetés par M. Untel.

- Nom de client $\rightarrow$ Client.
- Prix dans trois tables : *Quincaillerie, Peinture, Électricité* !
  - Il faut les trois !
- Pour faire le lien entre client et prix, il faut utiliser aussi *Commande* et *DetailCde*.
  - C'est dans *DetailCde* qu'on trouve les produits commandés.
  - Mais c'est dans *Commande* qu'on retrouve celui qui a passé la commande.

Corrélations entre attributs des tables impliquées

Afficher les noms des clients qui ont acheté des produits de quincaillerie et les dates et id des commandes respectives.

- Tables et attributs à utiliser :
  - *Client.nom, Commande.no, Commande.date* : résultats.
  - *DetailCde.id, Quincaillerie.nom* : utilisés pour répondre à la commande.
- Corrélations :
  - Prendre chaque tuple ds *Client*, chaque tuple ds *Quincaillerie*, chaque tuple ds *Commande* et chaque tuple ds *DetailCde*.
  - Ne garder que les combinaisons pour lesquelles *Client.id=Commande.id-client* $\wedge$ *Commande.no=DetailCde.no* $\wedge$ *DetailCde.nom-produit=Quincaillerie.nom*.

Essayer aussi pour

Retrouver les revenus des réalisateurs des films dans lesquels joue Untel.

Conjonction, interprétation en tant que sélection sur un produit cartésien

Liste des noms d'utilisateurs ayant commandé des produits en quantité supérieure à 100 et avant 20/12/2011. (Parfois le et est sous-entendu !)

- Tables et attributs : *Client.nom, Commande.no, Commande.date, DetailCde.no, DetailCde.qté*.
- Produit cartésien, corrélation sous forme de conjonction ds clause `WHERE` :

```
SELECT // à compléter
FROM Client, Commande, DetailCde
WHERE Client.id=Commande.id-client AND
      Commande.no=DetailCde.no AND
      DetailCde.date < DATE '2011-12-20' AND
      DetailCde.qté > 100;
```

Conjonction, interprétation comme intersection

Liste des clients ayant passé des commandes de produits de quincaillerie et des produits électriques.

- Tables et attributs : *Client.nom, Commande.no, Commande.date, DetailCde.no, DetailCde.nom-produit, Quincaillerie.nom, Électricité.nom*.
- Cette fois-ci, le et nécessite `INTERSECT` !

```
(SELECT // quoi ?
FROM Client, Commande, DetailCde, Quincaillerie
WHERE Client.id=Commande.id-client AND
      Commande.no=DetailCde.no AND
      DetailCde.nom-produit=Quincaillerie.nom)
INTERSECT
(SELECT // quoi ?
FROM Client, Commande, DetailCde, Électricité
WHERE Client.id=Commande.id-client AND
      Commande.no=DetailCde.no AND
      DetailCde.nom-produit=Électricité.nom);
```

- Ne pas faire le produit cartésien de *Quincaillerie* avec *Électricité* !
- Les deux tables ne sont pas nécessaires en même temps pour construire un tuple de la relation résultat !

Disjonction, interprétation en tant que union

Liste des clients ayant passé des commandes de produits de quincaillerie ou des produits électriques.

```
(SELECT // quoi ?
FROM Client, Commande, DetailCde, Quincaillerie
WHERE Client.id=Commande.id-client AND
      Commande.no=DetailCde.no AND
      DetailCde.nom-produit=Quincaillerie.nom)
UNION
(SELECT // quoi ?
FROM Client, Commande, DetailCde, Électricité
WHERE Client.id=Commande.id-client AND
      Commande.no=DetailCde.no AND
      DetailCde.nom-produit=Électricité.nom);
```

Négation, interprétation en tant que différence

Liste des clients qui ont commandé de produits de quincaillerie mais n'ont pas acheté de produit électrique.

```
(SELECT // quoi ?
FROM Client, Commande, DetailCde, Quincaillerie
WHERE Client.id=Commande.id-client AND
      Commande.no=DetailCde.no AND
      DetailCde.nom-produit=Quincaillerie.nom)
EXCEPT
(SELECT // quoi ?
FROM Client, Commande, DetailCde, Électricité
WHERE Client.id=Commande.id-client AND
      Commande.no=DetailCde.no AND
      DetailCde.nom-produit=Électricité.nom);
```

Quantification existentielle

Liste des prix des produits pour lesquels il existe au moins une commande.

- Assez souvent énoncé : *commandés au moins une fois*.
- Tables et attributs : prix et produits
  - *Client.nom, Client.id, Commande.id, Commande.id-client, DetailCde.no, DetailCde.nom-produit*.
- Les produits qui apparaissent ds au moins une commande sont exactement ceux ds *DetailCde* !

Quantification universelle et énoncés contenant des négations

Liste des clients qui ont commandé tous les produits.

- La sélection, la projection, le produit cartésien, l'union et le renommage ne contiennent pas de quantificateur universel !
- Mais $\forall \varphi = \neg \exists \neg \varphi$!
- Construire d'abord la relation complément !

Liste clients/produits qui ne représentent pas des commandes.

- On l'obtient comme complément de la relation suivante :

Liste des clients/produits qui représentent une commande.

Quantification universelle et énoncés contenant des négations (2)

- Liste des clients/produits qui représentent une commande :

```
SELECT DISTINCT Client.nom, DetailCde.nom-produit
FROM Cient, Commande, DetailCde
WHERE Client.id=Commande.id-client AND Commande.id=DetailCde.id;
```

- Liste de tous les clients/produits ? (Distincts !)
- Différence entre les deux ?
- Et maintenant liste des clients qui n'apparaissent pas dans le résultat de la deuxième requête ?

Plusieurs attributs de la table résultat en relation avec le même attribut d'une table existante

Afficher les clients qui ont passé au moins deux commandes.

- Tables et attributs : *Client.nom, Client.id, Commande.id, Commande.id-client*.
- Mais il nous faut deux apparitions de l'attribut *Commande.id* !
- Utiliser deux copies de la table *Commande*, pour retrouver des paires de commandes.

```
SELECT //
FROM Client, Commande c1, Commande c2
WHERE Client.id=c1.id-client AND
      Client.id=c2.id-client AND
      C1.id < c2.id; // pourquoi ?
```

Énoncés qui peuvent être traduits en SQL en créant des sous-requêtes

Lister le nom de la pièce ayant la plus grande quantité de pièces livrée.

- Une seule table, *DetailCde*, nécessaire dans la requête.
- Mais utilisée deux fois :

```
SELECT nom-produit
FROM DetailCde
WHERE qté >= ALL
    (SELECT nom-produit
    FROM DetailCde);
```

Agrégations

- En général la/les fonction(s) d'agrégation sont présentes explicitement dans l'énoncé

  Quelle est la moyenne des produits achetés par Untel ?

- Parfois la traduction de l'énoncé nécessite un calcul impliquant une fonction d'agrégation :

  Quel est le pourcentage des commandes passées par des clients domiciliés à Paris ?

- Construire une relation avec un seul

```
SELECT f2 / f1
FROM (SELECT COUNT( * ) AS f1
      FROM Commandes) t1,
     (SELECT COUNT( * ) AS f2
      FROM Clients, Commandes
      WHERE id=id-client AND adresse='Paris') t2;
```

Agrégations par groupes

Quelle est la quantité totale des produits commandés pour chaque commande ?

- Remarquer qu'on peut avoir plusieurs produits commandés dans la même commande.
- L'opérateur d'agrégation doit s'appliquer à l'ensemble de valeurs de l'attribut *DetailCde.qté* correspondant à chaque valeur de l'attribut *DetailCde.no*.
- ... et pas sur l'ensemble des valeurs

```
SELECT SUM(qté)
FROM DetailCde
GROUP BY no;
```

Agrégation par groupes (2)

Quel est le montant des commandes d'au moins deux pièces de même type ?

- Attention ! la requête suivante ne traduit pas correctement l'énoncé !

```
SELECT SUM(qté)
FROM DetailCde
WHERE qté>=2
GROUP BY no;
```

Agrégation par groupes (3)

- Construire d'abord la liste des id de commandes dont au moins un des produits commandés est demandé en quantité supérieure à 2.
- Puis, combiner cette liste de commandes pour obtenir les sommes désirées.

```
SELECT SUM(qté)
FROM DetailCde
WHERE no IN
   (SELECT no
   FROM DetailCde
   WHERE qté >= 2)
GROUP BY no;
```

Agrégation par groupes avec des filtres sur les valeurs à prendre en compte dans chaque calcul

Afficher le prix moyen des produits de peinture ayant le même nom mais seulement pour les produits dont la somme des poids est supérieure à 20kg.

```
SELECT AVG(prix)
FROM Peinture
GROUP BY nom
HAVING SUM(poids)>=20;
```

- La clause `HAVING` agit sur les groupes créés par la clause `GROUP`,
- ... à la différence de la clause `WHERE`, qui agit sur les tuples de la relation initiale.

Division

Liste des clients qui ont commandé tous les produits.

- Considérons les deux tables suivantes :

$$R_1 = \pi_{no,id-client}(\text{Commande})$$
$$R_2 = \pi_{id}(\text{Quincaillerie} \cup \text{Peinture} \cup \text{Électricité})$$

- Quel est le résultat de la division suivante :

$$R_1 \div R_2 \ ?$$