Bases de données et Data mining

Partie 3

Filière Modélisation & Calcul Scientifique – 2ème année
Année 2004-2005
Jérôme Darmont
http://eric.univ-lyon2.fr/~jdarmont/

Plan du cours

- Introduction
- L'algèbre relationnelle
- Le langage SQL

Objectifs du cours

- Logique de l'interrogation de bases de données

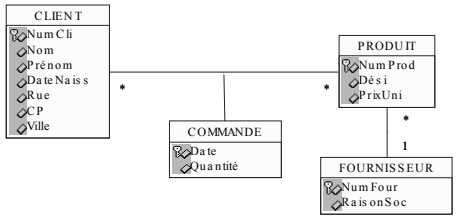
⇒ Algèbre relationnelle

- Mise en œuvre (*création*, *mise à jour* et *interrogation* de bases de données)

⇒ Langage SQL

Base de données exemple

Modèle conceptuel UML



Base de données exemple

Modèle logique relationnel

CLIENT (NumCli, Nom, Prénom, DateNaiss, Rue, CP, Ville)
PRODUIT (NumProd, Dési, PrixUni, NumFour#)
FOURNISSEUR (NumFour, RaisonSoc)

COMMANDE (NumCli#, NumProd#, Date, Quantité)

Clés primaires Clés étrangères#

Plan du cours

- ✓Introduction
- L'algèbre relationnelle
- Le langage SQL

Définition

- Ensemble d'opérateurs qui s'appliquent aux relations
- Résultat : nouvelle relation qui peut à son tour être manipulée

⇒ L'algèbre relationnelle permet d'effectuer des recherches dans les relations.

Opérateurs ensemblistes

- Union : T = R ∪ S ou T = UNION (R, S)
  R et S doivent avoir même schéma.

  ex. R et S sont les relations PRODUIT de deux sociétés qui fusionnent et veulent unifier leur catalogue.

  Notation graphique :

Opérateurs ensemblistes

- Intersection : T = R ∩ S ou
  T = INTERSECT (R, S)
  R et S doivent avoir même schéma.

  ex. Permet de trouver les produits communs aux catalogues de deux sociétés.

  Notation graphique :

Opérateurs ensemblistes

- Différence : T = R - S ou T = MINUS (R, S)
  R et S doivent avoir même schéma.

  ex. Permet de retirer les produits de la relation S existant dans la relation R.

  Notation graphique :

Opérateurs ensemblistes

- Produit cartésien : T = R x S
  ou T = PRODUCT (R, S)

  Associe chaque n-uplet de R à chaque n-uplet de S.

  Notation graphique :

Produit cartésien

ex.

NumProd	*Dési*
0	P1
1	P2

X

NumFour	*RaisonSoc*
10	F1
20	F2
30	F3

=

NumProd	*Dési*	*NumFour*	*RaisonSoc*
0	P1	10	F1
1	P2	10	F1
0	P1	20	F2
1	P2	20	F2
0	P1	30	F3
1	P2	30	F3

Opérateurs ensemblistes

- Division : T = R ÷ S ou T = DIVISION (R, S)

  R (A_1, A_2, …, A_n) S (A_{p+1}, …, A_n)
  T (A_1, A_2, …, A_p) contient tous les n-uplets tels que leur concaténation à chacun des n-uplets de S donne toujours un n-uplet de R.

  Notation graphique :

Division

NumCli	*Date*	*Quantité*
1	22/09/99	1
1	22/09/99	5
1	10/10/99	2
2	15/10/99	9
3	22/09/99	1
3	10/10/99	2

÷

Date	*Quantité*
22/09/99	1
10/10/99	2

=

NumCli
1
3

Opérateurs spécifiques

- Projection : T = Π <A, B, C> (R)
  ou T = PROJECT (R / A, B, C)

  T ne contient que les attributs A, B et C de R.
  ex. Noms et prénoms des clients.

  Notation graphique :

Opérateurs spécifiques

- Restriction : T = σ <C> (R)
  ou T = RESTRICT (R / C)

  T ne contient que les attributs de R qui satisfont la condition C.
  ex. C = Clients qui habitent Lyon.

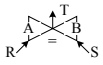
  Notation graphique :

Opérateurs spécifiques

- Jointure naturelle : T = R >< S
  ou T = JOIN (R, S)

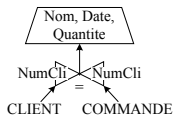
  Produit cartésien R x S et restriction A = B sur les attributs A $\in$ R et B $\in$ S.

  Notation graphique :



Jointure

Exemple : *Commandes avec le nom du client et pas seulement son numéro*



Requête : enchaînement d'opérations

Jointure

Décomposition des opérations

CL

NumCli	*Nom*
1	C1
2	C2
3	C3

X

CO

NumCli	*Date*	*Quantité*
1	22/09/99	1
3	22/09/99	5
3	22/09/99	2

Jointure

=

CL.NumCli	*Nom*	*CO.NumCli*	*Date*	*Quantité*
1	C1	1	22/09/99	1
2	C2	1	22/09/99	1
3	C3	1	22/09/99	1
1	C1	3	22/09/99	5
2	C2	3	22/09/99	5
3	C3	3	22/09/99	5
1	C1	3	22/09/99	2
2	C2	3	22/09/99	2
3	C3	3	22/09/99	2

Jointure

CL >< CO

CL.NumCli	*Nom*	*CO.NumCli*	*Date*	*Quantité*
1	C1	1	22/09/99	1
3	C3	3	22/09/99	5
3	C3	3	22/09/99	2

Nom	*Date*	*Quantité*
C1	22/09/99	1
C3	22/09/99	5
C3	22/09/99	2

Π <Nom, Date, Quantité> (CL >< CO)

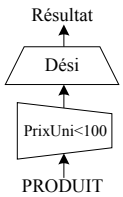
(Projection sur les attributs Nom, Date, Quantité)

Exemples de requêtes

Ex. 1 : Désignation et prix unitaire de tous les produits

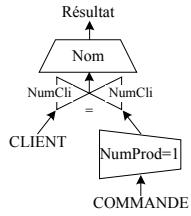
Exemples de requêtes

Ex. 2 : Désignation des produits de prix inférieur à 100 €



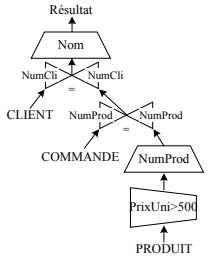
Exemples de requêtes

Ex. 3 : Nom des clients qui ont commandé le produit n° 1



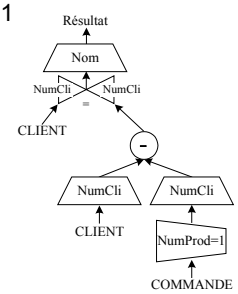
Exemples de requêtes

Ex. 4 : Nom des clients qui ont commandé au moins un produit de prix supérieur à 500 €



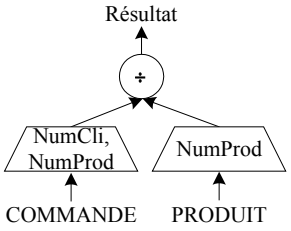
Exemples de requêtes

Ex. 5 : Nom des clients qui n'ont pas commandé le produit n° 1



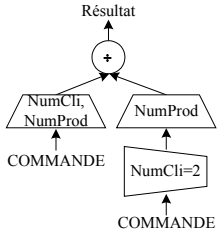
Exemples de requêtes

Ex. 6 : Numéro des clients qui ont commandé tous les produits



Exemples de requêtes

Ex. 7 : Numéro des clients qui ont commandé tous les produits commandés par le client n° 2



Classification des SGBDR

- Niveau 1 : Systèmes non relationnels. Supportent uniquement la structure tabulaire.
- Niveau 2 : Systèmes relationnellement minimaux. Permettent les opérations de sélection, projection et jointure.
- Niveau 3 : Systèmes relationnellement complets. Toutes les opérations de l'algèbre relationnelle.
- Niveau 4 : Systèmes relationnellement pleins. Permettent la définition des contraintes d'intégrité.

Plan du cours

- ✓Introduction
- ✓L'algèbre relationnelle
- Le langage SQL

Généralités

- SQL : *Structured Query Language*, issu de SEQUEL (*Structured English as a QUery Language*)
- SQL permet la définition, la manipulation et le contrôle d'une base de données relationnelle. Il se base sur l'algèbre relationnelle.
- SQL est un standard ANSI depuis 1986.
- Nous adoptons dans cette partie la syntaxe du SQL d'Oracle (très proche de la norme).

Généralités

SQL se subdivise en trois sous-langages :

- **LDD** (*Langage de Définition de Données*) : création, modification et suppression des définitions des tables
- **LMD** (*Langage de Manipulation de Données*) : ajout, suppression, modification et interrogation des données
- **LCD** (*Langage de Contrôle de Données*) : gestion des protections d'accès

Généralités

- Caractère de fin d'instruction : ;
- Commentaires :
  - -- Ligne commentée
  - /* Bloc de texte commenté */
- Clauses optionnelles : Notées entre [ ] dans ce support de cours

Tables

```
CREATE TABLE nom_table (    Attribut1 TYPE,
                            Attribut2 TYPE, …,
                            contrainte_integrité1,
                            contrainte_integrité2,
                            …);
```

Principaux type de données :

- NUMBER(n) : Entier à n chiffres
- NUMBER(n, m) : Réel à n chiffres au total (virgule comprise), m après la virgule
- VARCHAR(n) : Chaîne de n caractères (entre ' ')
- DATE : Date au format 'JJ-MM-AAAA'
- BLOB : *Binary Large Object*

Contraintes d'intégrité

- Clé primaire :

CONSTRAINT nom_contrainte PRIMARY KEY (attribut_clé [, attribut_clé2, …])

- Clé étrangère :

CONSTRAINT nom_contrainte FOREIGN KEY (attribut_clé_ét) REFERENCES table(attribut)

- Contrainte de domaine :

CONSTRAINT nom_contrainte CHECK (condition)

Exemple

```
CREATE TABLE Produit (      NumProd NUMBER(3),
                            Dési VARCHAR(30),
                            PrixUni NUMBER(8,2),
                            NumFour NUMBER(3),

CONSTRAINT produit_cle_pri PRIMARY KEY (NumProd),

CONSTRAINT produit_cle_etr FOREIGN KEY (NumFour)
        REFERENCES Fournisseur(NumFour),

CONSTRAINT prix_ok CHECK (PrixUni > 0)          );
```

Modifications structurelles

- Ajout d'attributs
  ALTER TABLE nom_table ADD (attribut TYPE, …);
  ex. ALTER TABLE Client ADD (tel NUMBER(8));
- Modifications d'attributs
  ALTER TABLE nom_table MODIFY (attribut TYPE, …);
  ex. ALTER TABLE Client MODIFY (tel NUMBER(10));
- Suppression d'attributs
  ALTER TABLE nom_table DROP COLUMN attribut, ...;
  ex. ALTER TABLE Client DROP COLUMN tel;

Modifications structurelles

- Ajout de contrainte
  ALTER TABLE nom_table ADD CONSTRAINT nom_contrainte définition_contrainte;
  ex. ALTER TABLE Client
  ADD CONSTRAINT sal_ok CHECK (salaire>0);
- Suppression de contrainte
  ALTER TABLE nom_table DROP CONSTRAINT nom_contrainte;
  ex. ALTER TABLE Client
  DROP CONSTRAINT sal_ok;

Copie et destruction de tables

- Destruction : DROP TABLE nom_table;

  ex. DROP TABLE Client;
- Copie : CREATE TABLE copie AS requête;

  ex. CREATE TABLE Client_Copie AS
  SELECT * FROM Client;

Index

- Création d'index (accélération des accès)
  CREATE [UNIQUE] INDEX nom_index ON nom_table (attribut [ASC|DESC], …);
  UNIQUE ⇒ pas de double
  ASC/DESC ⇒ ordre croissant ou décroissant
  ex. CREATE INDEX Idx_cli ON Client (Nom);
- Destruction : DROP INDEX nom_index;
  ex. DROP INDEX Idx_cli;

Insertion et mise à jour

- Ajout d'un n-uplet
  INSERT INTO nom_table
  VALUES (val_att1, val_att2, …);
  ex. INSERT INTO Produit
  VALUES (400, 'Nouveau produit', 78.90, 30);
- Mise à jour d'un attribut
  UPDATE nom_table SET attribut=valeur
  [WHERE condition];
  ex. UPDATE Client SET Nom='Dudule'
  WHERE NumCli = 3;

Suppression

- Suppression de n-uplets

  DELETE FROM nom_table [WHERE condition];

  ex. DELETE FROM Produit;

  DELETE FROM Client
  WHERE Ville = 'Lyon';

Requêtes simples

Forme générale d'une requête

SELECT [ALL|DISTINCT] attribut(s) FROM table(s)
[WHERE condition]
[GROUP BY attribut(s) [HAVING condition]]
[ORDER BY attribut(s) [ASC|DESC]];

- Tous les n-uplets d'une table
  ex. SELECT * FROM Client;
- Tri du résultat
  ex. *Par ordre alphabétique inverse de nom*
  SELECT * FROM Client
  ORDER BY Nom DESC;

Ou… par l'exemple

Requêtes simples

- Calculs ex. *Calcul de prix TTC*
  SELECT PrixUni+PrixUni*0.196 FROM Produit;
- Projection
  ex. *Noms et Prénoms des clients, uniquement*
  SELECT Nom, Prenom FROM Client;
- Restriction
  ex. *Clients qui habitent à Lyon*
  SELECT * FROM Client
  WHERE Ville = 'Lyon';

Requêtes simples

ex. *Commandes en quantité au moins égale à 3*
SELECT * FROM Commande
WHERE Quantite >= 3;

ex. *Produits dont le prix est compris entre 50 et 100 €*
SELECT * FROM Produit
WHERE PrixUni BETWEEN 50 AND 100;

ex. *Commandes en quantité indéterminée*
SELECT * FROM Commande
WHERE Quantite IS NULL;

Requêtes simples

ex. *Clients habitant une ville dont le nom se termine par* sur-Saône

SELECT * FROM Client
WHERE Ville LIKE '%sur-Saône';

'sur-Saône%' ⇒ commence par *sur-Saône*
'%sur%' ⇒ contient le mot *sur*

Prédicat ensembliste IN

ex. *Prénoms des clients dont le nom est Dupont, Durand ou Martin*

SELECT Prenom FROM Client
WHERE Nom IN ('Dupont', 'Durand', 'Martin');

NB : Possibilité d'utiliser la négation pour tous ces prédicats : NOT BETWEEN, NOT NULL, NOT LIKE, NOT IN.

Fonctions d'agrégat

Elles opèrent sur un ensemble de valeurs.

- AVG(), VARIANCE(), STDDEV() : moyenne, variance et écart-type des valeurs
- SUM() : somme des valeurs
- MIN(), MAX() : valeur minimum, valeur maximum
- COUNT() : nombre de valeurs

ex. *Moyenne des prix des produits*
SELECT AVG(PrixUni) FROM Produit;

Fonctions d'agrégat

Opérateur DISTINCT

ex. *Nombre total de commandes*
SELECT COUNT(*) FROM Commande;
SELECT COUNT(NumCli) FROM Commande;

ex. *Nombre de clients ayant passé commande*
SELECT COUNT(DISTINCT NumCli)
FROM Commande;

Fonctions d'agrégat

Table COMMANDE (simplifiée)

NumCli	*Date*	*Quantite*
1	22/09/99	1
3	22/09/99	5
3	22/09/99	2

COUNT(NumCli) ⇒ Résultat = 3

COUNT(DISTINCT NumCli) ⇒ Résultat = 2

Jointures

ex. *Liste des commandes avec le nom des clients*

```
SELECT Nom, Date, Quantite
FROM Client, Commande
WHERE    Client.NumCli =
         Commande.NumCli;
```


Jointures

ex. *Idem avec le numéro de client en plus*

```
SELECT C1.NumCli, Nom, Date, Quantite
FROM Client C1, Commande C2
WHERE C1.NumCli = C2.NumCli
ORDER BY Nom;
```

NB : Utilisation d'alias (C1 et C2) pour alléger l'écriture + tri par nom.

Jointures

Jointure exprimée avec le prédicat IN

ex. *Nom des clients qui ont commandé le 23/09*

```
SELECT Nom FROM Client
WHERE NumCli IN (
    SELECT NumCli FROM Commande
    WHERE Date = '23-09-1999');
```

NB : Il est possible d'imbriquer des requêtes.

Prédicats d'existence

- Prédicats EXISTS / NOT EXISTS

  ex. *Clients qui ont passé au moins une commande [n'ont passé aucune commande]*

```
SELECT * FROM Client C1
WHERE [NOT] EXISTS (
    SELECT * FROM Commande C2
    WHERE C1.NumCli = C2.NumCli  );
```

Prédicats de dénombrement

- Prédicats ALL / ANY

  ex. *Numéros des clients qui ont commandé au moins un produit en quantité supérieure à chacune [à au moins une] des quantités commandées par le client n° 1.*

```
SELECT DISTINCT NumCli FROM Commande
WHERE Quantite > ALL [ANY] (
    SELECT Quantite FROM Commande
    WHERE NumCli = 1);
```


Groupement

ex. *Quantité totale commandée par chaque client*

```
SELECT NumCli, SUM(Quantite)
FROM Commande
GROUP BY NumCli;
```

ex. *Nombre de produits différents commandés...*

```
SELECT NumCli, COUNT(DISTINCT NumProd)
FROM Commande
GROUP BY NumCli;
```


Groupement

ex. *Quantité moyenne commandée pour les produits faisant l'objet de plus de 3 commandes*

```
SELECT NumProd, AVG(Quantite)
FROM Commande
GROUP BY NumProd
HAVING COUNT(*)>3;
```

Attention : La clause HAVING ne s'utilise qu'avec GROUP BY.

Groupement

Différence entre HAVING et WHERE :

- HAVING : évaluation de condition sur un résultat de groupement (*a posteriori*)
- WHERE : évaluation de condition *a priori* (avant GROUP BY)

Opérations ensemblistes

Opérateurs ensemblistes :

INTERSECT, MINUS, UNION

ex. *Numéro des produits qui soit ont un prix inférieur à 100 €, soit ont été commandés par le client n° 2*

```
SELECT NumProd FROM Produit WHERE PrixUni<100
UNION
SELECT NumProd FROM Commande WHERE NumCLi=2;
```


Les vues

- Vue : table *virtuelle* calculée à partir d'autres tables grâce à une requête
- Création d'une vue
  CREATE VIEW nom_vue AS requête;

  ex. CREATE VIEW Noms AS
  SELECT Nom, Prenom FROM Client;

Intérêt des vues

- Simplification de l'accès aux données en masquant les opérations de jointure

ex. CREATE VIEW Prod_com AS
SELECT P.NumProd, Dési, PrixUni, Date, Quantite
FROM Produit P, Commande C
WHERE P.NumProd=C.NumProd;

SELECT NumProd, Dési FROM Prod_com
WHERE Quantite>10;

Intérêt des vues

- Sauvegarde indirecte de requêtes complexes
- Présentation de mêmes données sous différentes formes adaptées aux différents usagers particuliers
- Support de l'indépendance logique
  ex. Si la table Produit est remaniée, la vue Prod_com doit être refaite, mais les requêtes qui utilisent cette vue n'ont pas à être remaniées.

Intérêt des vues

- Renforcement de la sécurité des données par masquage des lignes et des colonnes sensibles aux usagers non habilités

Restrictions des vues

Pour que la mise à jour de données à travers une vue soit possible :

- Le mot clé DISTINCT doit être absent.
- La clause FROM doit faire référence à une seule table.
- La clause SELECT doit faire référence directement aux attributs de la table concernée (pas d'attribut dérivé).
- Les clauses GROUP BY et HAVING sont interdites.

Tutoriel SQL

Pour approfondir SQL en ligne…

http://eric.univ-lyon2.fr/~jdarmont/tutoriel-sql/

Plan du cours

- ✓Introduction
- ✓L'algèbre relationnelle
- ✓Le langage SQL