Requêtes en langage SQL

Motivation [CERV 3] Au XIXe au US la population augmente de 30% tous les 10 ans. Recenser la population prend 10 ans.

I. Creation d'un table en SQL [VACHE 3.1.1]

Définition 1 **Une Table** (ou Relation) est un multi-ensemble de n-uplets ou enregistrements représenté ligne par ligne.

Definition 2 **Le SQL** (Structured query langage) est un langage déclaratif de requête permettant d'interagir avec une base de données.

Définition 3 **Les types SQL** de bases sont nombreux. On utilisera notamment `varchar` pour une chaine de charactère, `int` pour un entier ou `real` pour un nombre réel.

Exemple 4 [VACHE 5.11.1 p.162] **Sailors, Boats et Reserves** sont les 3 tables que l'on va étudier et se définissent comme suit :

- Marin(*sid* : **int**, snom : **varchar**, note : **int**, age: **real**)
- Bateau(*bid*: **int**, bnom : **varchar**, couleur: **varchar**)
- Reservation(*sid*: **int**, *bid* : **int**, jour : **date**)

Définition 5 [VACHE 5.11.1 p.162] **La création de ces tables** se fait comme ceci :

```sql
create table MARIN(
  MID int primary key,
  MNOM varchar,
  NOTE int,
  AGE real
);
```

```sql
create table BATEAU(
  BID int primary key,
  BNOM varchar,
  COULEUR varchar
);
```

```sql
create table RESERVATION(
  MID int not null,
  BID int,
  JOUR date,
  foreign key (BID)
    references BATEAU(BID),
  foreign key (MID)
    references MARIN(MID),
  primary key (BID, JOUR)
);
```

Remarque 6 **Clé primaire et étrangère.** Les identifiants sont utilisés comme clé primaire dans `MARIN` et `BATEAU`. La clé `MID` est une clé étrangère dans la table `RESERVATION`.

Définition 7 **Les insertions** nous permettent d'ajouter des éléments dans une table :

```sql
insert into BATEAU values(101, 'Interlake', 'blue');
insert into BATEAU values(102, 'Interlake', 'red');
...
insert into MARIN values(22, 'Dustin', 7, 45.0);
....
insert into RESERVATION values(22, 101, '2023-01-06');
...
```

Exemple 8

BID	BNOM	COULEUR
101	Interlake	blue
102	Interlake	red
103	Clipper	red
104	Marine	red

MID	MNOM	NOTE	AGE
22	Dustin	7	45
23	Brutus	5	34
24	Dustin	8	55.5
25	Andy	6	41.5
26	Rusty	10	53.5
27	Horatio	7	47
28	Andy	2	41.5
29	Brutus	8	NULL
30	Brutus	9	54

MID	BID	JOUR
22	101	06
22	102	01
22	103	14
22	104	02
23	104	14
24	102	07
24	103	02
24	104	16
25	101	07
25	102	02
27	102	10
27	104	22
28	104	06
28	103	15
29	102	03
30	102	25
30	103	16

Définition 9 **Update et Delete** peuvent aussi être utilisées pour mettre à jour une table ou supprimer des enregistrements.

II. Un problème concret [VACHE 5]

Problème 10 Supposons que les bateaux rouges posent problèmes. On cherche alors pour chacun d'entre eux le nom d'un marin expérimenté (plus de 40 ans, a une note de plus de 7, a reservé plus de 2 bateaux et n'a jamais reservé un bateau en premier) et la moyenne de notes de tous les marins qu'ils l'ont reservé.

Remarque 11 `select` et `from` nous permettent de faire une requête sur une table donnée. `distinct` permet de ne pas avoir de doublon dans la table finale.

Exemple 12 Noms de tous les marins sans doublons.

```
1 select distinct M.MNOM from MARIN as M;
```

MNOM
Dustin
Brutus
Andy
Rusty
Horatio

Remarque 13 **where** permet de faire une selection sur les éléments d'une table selon une condition. Cette clause n'est pas obligatoire.

Exemple 14 Tous noms et id des bateaux rouges.

```
1 select B.BID, B.BNOM
2 from BATEAU as B
3 where B.COULEUR = 'red';
```

BID	BNOM
102	Interlake
103	Clipper
104	Marine

Remarque 15 Dans une clause `where` la condition booléenne peut être complexifiée avec `and` ou `or`.

Exemple 16 Tous les marins qui ont plus de 40 ans et une note d'au moins 5.

```
1 select M.MNOM, M.MID from MARIN as M
2 where M.NOTE >= 5 and M.AGE >= 40;
```

MNOM	MID
Dustin	22
Dustin	24
Andy	25
Rusty	26
Horatio	27
Brutus	30

Remarque 17 **Valeur `NULL`.** Certaines valeurs dans une table peuvent être `NULL` si non indiqué autrement à la création. Comme vu dans l'exemple précédent, les marins dont on ne connait pas l'âge ne sont pas affichés.

Définition 18 **Les opérateurs d'agrégats** disponibles en SQL sont `max`, `min`, `count`, `sum`, `avg` etc. Utilisés avec `group by`, ils permettent de faire une opération sur le groupe spécifié.

Exemple 19 Marins avec plus de 2 reservations.

```
1 select R.MID, count(*) as R_NB
2 from RESERVATION as R
3 group by R.MID
4 having count(*) >= 2;
```

MID	R_NB
22	4
24	3
25	2
27	2
28	2
30	2

Remarque 20 **Jointure.** Le mot clé `join` permet de récupérer les enregistrements qui ont les mêmes valeurs pour les champs de même nom dans le produit cartésien entre deux tables.

DEV

Remarque 21 [DSC 6] Les jointures sont une briques de base de l'évaluation des requêtes les plus importantes et couteuses. Différents algorithmes existent pour faire des jointures en fonction de la situation.

Exemple 22 Les reservations avec les notes des marins.

```
1 select R.BID, M.NOTE, R.JOUR
2 from RESERVATION as R
3 natural join MARIN as M;
```

BID	NOTE	JOUR
101	7	06
102	7	01
103	7	14
104	7	02
104	5	14
102	8	07
103	8	02
104	8	16
101	6	07
102	6	02
102	7	10
104	7	22
104	2	06
103	2	15
102	8	03
102	9	25
103	9	16

Remarque 23 **Requête imbriquée.** La clotûre de SQL nous permet très facilement de faire des requêtes imbriquées. Ces requêtes peuvent utiliser des champs de la table qui appelle la requête imbriquée.

Remarque 24 **Le langage SQL est clos** car toute requête sur une ou plusieurs tables s'évalue en une nouvelle table.

Remarque 25 **Table comme un ensemble.** Chaque table étant un multi ensemble d'enregistrements, des opérations ensemblistes comme `union`, `intersect`, `exists`, `in`, `all` ou `any` peuvent être utilisés en SQL.

Exemple 26 Premier marin qui a réservé chaque bateau.

```
1 select R1.BID, R1.MID, R1.JOUR
2 from RESERVATION as R1
3 where R1.JOUR in ( -- "=" possible
4     select min(R2.JOUR)
5     from RESERVATION R2
6     where R1.BID = R2.BID
7 )
8 group by R1.BID;
```

BID	MID	JOUR
101	22	06
102	22	01
103	24	02
104	22	02

Exemple 27 Tous les identifiants des marins expérimentés.

```
select MID, MNOM, NOTE from
    (select * from
        (select * from MARIN_M5_P40
        intersect
        select * from MARIN_PLUS_2_BATEAU)
    except
    select * from MARIN_PREMIER_RESERVATION)
natural join MARIN;
```

MID	MNOM	NOTE
25	Andy	6
27	Horatio	7
30	Brutus	9

Exemple 28 Pour chaque réservation la note du navigateur qui a fait la réservation.

```
select R.BID, avg(M.NOTE) as NOTE
from RESERVATION as R
natural join MARIN as M
group by R.bid;
```

BID	NOTE
101	6.5
102	7.5
103	6.5
104	5.8

Exemple 29 Requête complète finale.

```
 with NOTE_MOY(BID, NOTE, BNOM) as (
      select R.BID, avg(M.NOTE) as NOTE, B.BNOM
      from RESERVATION as R
      natural join M_EXP as M
      natural join BATEAU_RED as B
      group by R.bid
 ), M1_EXP(MID, BID) as (
      select R.MID, R.BID
      from RESERVATION as R
      where R.MID in (select M_EXP.MID from M_EXP)
      group by R.BID
 )
select NOTE_MOY.BNOM, NOTE_MOY.NOTE, MARIN.MNOM
from NOTE_MOY
natural join M1_EXP
join MARIN on MARIN.MID = M1_EXP.MID;
```

BNOM	NOTE	MNOM
Interlake	7.33	Andy
Clipper	9	Brutus
Marine	7	Horatio

III. Calcul (Traduction) et Optimisation de Requête SQL [SCHWARZ 14.3] [DSC 6]

A. Traduction en algèbre relationnelle

Définition 30 **Algèbre relationelle** L'algèbre relationnelle consiste en un ensemble d'opérations prenant une ou deux relations en entrée et produisant une nouvelle relation en sortie.

Exemple 31 **Sélection** $\sigma_{\text{NOTE}>=5}(\text{MARIN})$ renvoie une nouvelle relation correspondant aux marins ayant eu la moyenne.

Définition 32 Il y a différentes opérations algébriques:

- $\sigma_{\text{cond}}(\text{R})$: sélection des lignes de R vérifiant *cond*
- $\pi_{\text{att1, att2,...}}(\text{R})$: projection sur les colonnes *att1*, *att2*, etc..
- $\text{R}_1 \bowtie_{\text{att1=att2}} \text{R}_2$: jointure entre R_1 et R_2 selon *att1* et *att2*

Définition 33 **Calcul relationnel** Le calcul relationnel est le fragment syntaxique de la logique du premier ordre où il n'y a que des symboles de relation, des conjonctions, et le quantificateur $\exists$.

Theorème 34 **Codd(Admis)** Il y a équivalence d'expressivité entre l'algèbre relationnelle et le calcul relationnel.

B. Optimisation de requêtes

Définition 35 **Plan d'évaluation** Un plan d'évaluation consiste en arbre d'algèbre relationnelle.

Définission 36 **Équivalence** Deux plans sont dits équivalents si pour une base de données vérifiant les mêmes contraintes, ces deux plans donnent le même résultats.

Exemple 37 **Ces deux plans sont équivalents:**

DEV

Remarque 38 [DSC] On peut optimiser les requêtes en trouvant des formules logiques équivalentes en minimisant une fonction de coût.

Requêtes en langage SQL

I. Creation d'un table en SQL [VACHE 3.1.1]

1 Def Une Table
2 Def Le SQL
3 Def Les types SQL
4 Ex [VACHE 5.11.1 p.162] Sailors, Boats et Reserves
5 Def [VACHE 5.11.1 p.162] La création de ces tables
6 Rem Clé primaire et étrangère.
7 Def Les insertions
8 Ex
9 Def Update et Delete

II. Un problème concret [VACHE 5]

10 Prob
11 Rem
12 Ex
13 Rem where
14 Ex
15 Rem
16 Ex
17 Rem Valeur NULL.
18 Def Les opérateurs d'agrégats
19 Ex
20 Rem Jointure.
21 Rem [DSC 6]
22 Ex
23 Rem Requête imbriquée.
24 Rem Le langage SQL est clos
25 Rem Table comme un ensemble.
26 Ex
27 Ex
28 Ex
29 Ex

III. Calcul (Traduction) et Optimisation de Requête SQL [SCHWARZ 14.3] [DSC 6]

A. Traduction en algèbre relationnelle

30 Def Algèbre relationelle
31 Ex Sélection
32 Def
33 Def Calcul relationnel
34 Theorème Codd(Admis)

B. Optimisation de requêtes

35 Def Plan d'évaluation
36 Définission Équivalence
37 Ex Ces deux plans sont équivalents:
38 Rem [DSC]

Programmes

Première Gestion de BDD par un tableau doublement indexé. Lecture csv dans un langage comme python : Indexation, Recherche, Tri, Fusion de tables.

Terminale SQL : SELECT, FROM, WHERE, JOIN, UPDATE, INSERT, DELETE, DISTINCT, ORDER BY. Base de données relationnelles. SQBD.

Prepa Modèle entité–association. SQL99 maximum. Requête imbriquée.

Complémentaire Création, suppression, modification en SQL. Opérateurs Algèbre Relationnelle : application à l'optimisation de requêtes. Bases de données : calcul relationnel et théorème de Codd.

Remarque

- Cette leçon peut suivre un quelconque livre de base de données. Le choix ici (motivé car c'est un choix naturel pour un prof) est de faire un exemple filé qui permet de faire comprendre dans la pratique le langage SQL. En effet tous les programmes accentuent sur le fait que l'apprentissage du SQL doit se faire pas l'axe pratique plutôt que l'analyse théorique du langage.
- Les développements eux un peu plus haut niveau permettent de montrer sa technicité et sa compréhension fine du SQL.
- Le théorème de Codd demande au calcul relationnel d'être indépendant du domaine, mais cette subtilité est flouté par le caractère admis du théorème et son but pédagogique de montrer le lien entre l'algèbre relationelle et le langage SQL

Sources

- [DSC] : Complet avec beaucoup de schémas. Bien pour les devs jointure et optimisation de requêtes.
- [TOR] : bien SQL résumé, des exemples de code bien.
- [VACHE] : Interessant et complet. Sources possible pour les devs.
- [SCHWARZ] : bien et concis pour algèbre relationnelle et première ordre.

Bibliographie

[CERV] B. Christian & T. Griffiths, *Penser en Algorithmes.*

[VACHE] R. Ramakrishnan & J. Gehrke, *Database Management Systems 3rd edition.*

[DSC] A. Silberchatz & H. F. Korth & S. Sudarshan, *Database System Concepts.*

[SCHWARZ] P. l. Barbenchon & S. Pinchinat & F. Schwarzentruber, *Logique: Fondements et application.*

[TOR] T. Balabonski & S. Conchon & J. Filliâtre & K. Nguyen & L. Sartre, *MP2I MPI, Informatique Cours et exercices corrigés.*

Paul Adam - Adrien Decosse