SQL
et
Bases de données

Cours 3

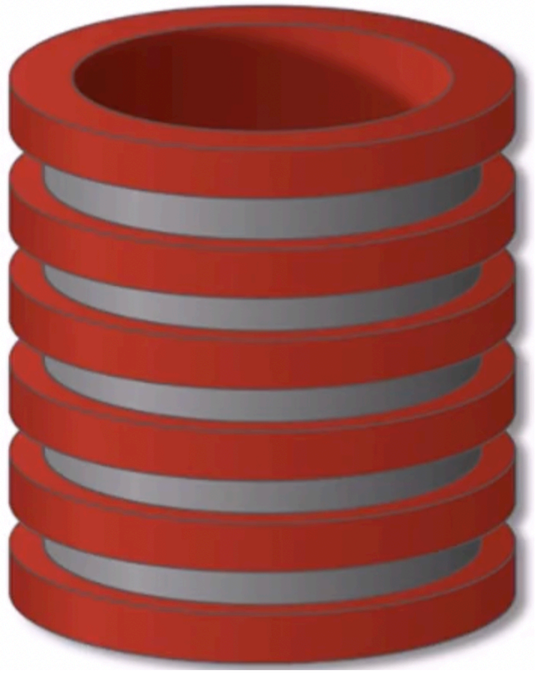
Jean-Jacques Lévy

jean-jacques.levy@inria.fr

http://jeanjacqueslevy.net/lp-sql

Plan

- installations de sqlite et VScode
- requêtes élémentaires
- requêtes imbriquées
- requêtes plus élaborées
- JOIN



- deux bons tutoriels

  http://www.w3schools.com/sql/default.asp

  http://www.programiz.com/sql

Création d'une base de données en SQL

- définir les schémas

client

```
create table if not exists "client"
(
	[cID] integer primary key autoincrement NOT NULL,
	[nom] nvarchar(20),
	[actif] integer,
	[ville] nvarchar(10)
);
```

produit

```
create table if not exists "produit"
(
	[pID] integer primary key autoincrement NOT NULL,
	[pNom] nvarchar(10),
	[pCat] nvarchar(10),
	[pVille] nvarchar(10),
	[prix] integer
);
```

devis

```
create table if not exists "devis"
(
	[cID] integer NOT NULL,
	[pNom] nvarchar(10),
	[dCat] nvarchar(10),
	[commande] integer  -- 0 si devis en cours, pId du produit accepté
);
```

Initialisation des valeurs

- `autoincrement` met à jour automaitquement le numéro de série

```
INSERT INTO client (nom, actif, ville) VALUES ('Tom', 23000, 'Bordeaux');
INSERT INTO client (nom, actif, ville) VALUES ('Jean-Jacques', 38000, ‘Paris’);
…
```

- déclarer un champ `primary key` signifie que sa valeur est unique (utile pour la formation d’indexs)

```
INSERT INTO produit (pNom, pCat, pVille, prix) VALUES ('clio', 'auto', 'Paris', 13000);
INSERT INTO produit (pNom, pCat, pVille, prix) VALUES ('audi', 'auto', 'Paris', 45000);
…
```

```
INSERT INTO devis (cID, pNom, dCat, commande) VALUES (2, 'peugeot', 'velo', 0);
INSERT INTO devis (cID, pNom, dCat, commande) VALUES (7, NULL, 'velo', 0);
…
```

http://www.w3schools.com/sql/default.asp

client

cID	nom	actif	ville
1	Tom	23000	Bordeaux
2	Jean-Jacques	38000	Paris
3	Martin	51000	Nice
4	Kiki	54000	Pekin
5	Iteki	84000	Tokyo
6	Bob	6100	Nice
7	Albert	12000	Bordeaux
8	Manu	8150	Paris
9	Valou	10300	Bordeaux
10	Joe	32500	Nice
11	Helmut	8150	Paris
12	Martine	11200	Bordeaux
13	Marina	9150	Nice
14	Masha	10290	Nice
15	Julia	32000	Paris
17	Bob	38000	Nice

produit

pID	pNom	pCat	pVille	prix
1	clio	auto	Paris	13000
2	audi	auto	Paris	45000
3	tesla	auto	Pekin	70000
4	tesla	auto	Nice	40000
5	yamaha	moto	Tokyo	8000
6	kawasaki	moto	Tokyo	8000
7	megamo	velo	Paris	3240
8	shimano	velo	Paris	1900
9	btwin	velo	Nice	990
10	triban	velo	Nice	690
11	peugeot	velo	Paris	750
12	bertin	eVelo	Paris	1190
13	trek	eVelo	Bordeaux	1390
14	trek	eVelo	Paris	1350

devis

cID	pNom	dCat	commande
2	peugeot	velo	0
7	NULL	velo	0
6	trek	eVelo	0
8	NULL	auto	0
8	NULL	velo	0
8	NULL	eVelo	0
9	honda	auto	0
11	NULL	auto	0
4	honda	moto	0
5	NULL	moto	0
8	triban	velo	0
13	NULL	velo	0
11	yaris	auto	0
12	NULL	velo	0
1	NULL	eVelo	0

Quelques requêtes

- lister les noms des clients qui figurent dans la table des devis (solution)

```
select nom,pNom,dCat
from devis,client
where client.cID = devis.cID;
```

- lister les noms des clients avec leurs actifs en euros et en RMBs
- idem mais pour ceux qui ont plus de 30000 RMBs

```
select nom, actif as actifEuro,
   actif*7.01 as actifRMB, ville
from client
where actifRMB > 300000;
```

- lister toutes les villes des clients et des produits

```
select distinct ville from client
union
select distinct pVille from produit;
```

http://www.w3schools.com/sql/default.asp

Quelques requêtes - bis

- lister dans l'ordre croissant ou décroissant

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

- valeur NULL de certainese données

```
SELECT column_names
FROM table_name
WHERE column_name IS NULL;
```

- renommer des attributs avec AS (*alias*)

```
SELECT column_name AS alias_name
FROM table_name;
```

```
select nom, actif as actifEuro,
   actif*7.01 as actifRMB, ville
from client
where actifRMB > 300000;
```

Renommer les tables

- renommer les tables avec `AS` (facultatif .. on peut ne pas le mettre)

```
select C1.*, C2.*
from client as C1, client as C2
where C1.actif = C2.actif;
```

```
select C1.*, C2.*
from client C1, client C2
where C1.actif = C2.actif;
```

```
select C1.*, C2.*
from client as C1, client as C2
where C1.actif = C2.actif
and C1.cID <> C2.cID;
```

```
select C1.*, C2.*
from client as C1, client as C2
where C1.actif = C2.actif
and C1.cID < C2.cID;
```

Requêtes imbriquées

- imbriquer dans FROM

```
select *
from (select cID, nom, actif, actif*7.01 as actifRMB
from client) as C
where C.actifRMB > 200000;
```

- imbriquer dans SELECT (à condition de ne retourner qu'une seule valeur)

```
select pNom, pVille,
(select distinct max(actif)
 from devis, client
 where produit.pCat = devis.dCat
   and devis.cID = client.cID) as dMax, pCat
from produit;
```

Requêtes imbriquées

- imbriquer dans WHERE

```
select cID, nom
from client
where cID in (select cID from devis where dCat = 'auto')
  and cID not in (select cID from devis where dCat = 'velo');
```

- lister le client avec le plus grand actif (avec max)

```
select nom, max(actif) from client;
```

- lister le client avec le plus grand actif (sans utiliser max)

```
select nom, actif
from client as C1
where not exists (
  select * from client as C2
  where C1.actif < C2.actif);
```

- lister le produit, la ville, l'actif et le nom du client avec le plus grand actif pour acheter ce produit

Quelques requêtes - ter

- appartenance à un ensemble avec IN

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

ou encore

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (SELECT STATEMENT);
```

- présence dans un ensemble avec EXISTS

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);
```

Modifications

- ajouter des n-uplets

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

ou encore

```
INSERT INTO table2 (column1, column2, column3, ...)
SELECT column1, column2, column3, ...
FROM table1
WHERE condition;
```

- supprimer des n-uplets

```
DELETE FROM table_name WHERE condition;
```

- modifier les valeurs de n-uplets

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Modifications

- jour de chance: tous les clients recoivent 10000€ -> modifier la table `client`.
- spécial `Bordeaux`: tous les clients recoivent `10000€` -> modifier la table `client`.
- créer la table des clients de `Bordeaux`
- retirer `Helmut` et `Masha` de la table des clients

Attributs de groupe

- on peut regrouper les n-uplets avec `GROUP` selon des attributs et utiliser une commande de groupe

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

- commandes de groupe `MIN, MAX, COUNT, AVG, SUM`

```
select ville, count(*)
from client
group by
ville;
```

[ne pas oublier `DISTINCT`]

Attributs NULL

- les attributs inexistants ont la valeur `NULL`

```
select * from devis
where pNom is NULL;
```

[remarque: ne pas utiliser `=`, mais utiliser `IS`]

- le résultat de tests avec des attributs nuls peut être aléatoire

Jokers

- on peut tester partiellement une valeur avec `LIKE`

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

[caractères spéciaux: ne pas utiliser `_` et `%` pour un caractère quelconque ou une série de tels caractères]

- exemple des clients dont le nom commence par la lettre M:

```
select * from client
where nom like 'M%';
```

JOIN

- déjà vu avec les conditions dans `WHERE`
- jointure à gauche
- jointure interne `INNER`
- jointure externe `OUTER`
- description complète en

  http://www.w3schools.com/sql/sql_join.asp

  http://www.programiz.com/sql/join

Prochain cours

- indexs
- correspondance avec la logique du 1er ordre
- introduction au requêtes concurrentes