SQL
et
Bases de données

Cours 8

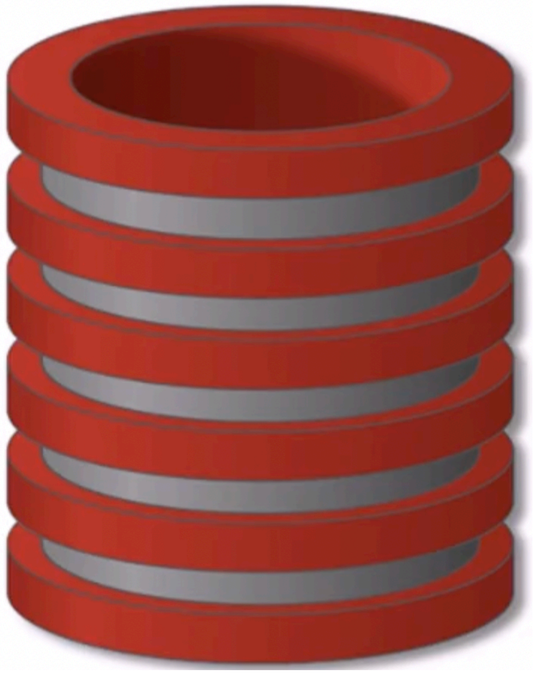
Jean-Jacques Lévy

jean-jacques.levy@inria.fr

http://jeanjacqueslevy.net/lp-sql

Plan

- exercices (suite)
- requêtes récursives
- interface Python
- serveur SQLite avec interface HTML



- deux bons tutoriels

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

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

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

3 manières de nommer

- localement avec `WITH`

```
with clientParis
 as (select CID, nom, actif
 from client
 where ville = 'Paris')
select * from clientParis;
```

- localement avec `AS`

```
select * from
 (select CID, nom, actif
 from client
 where ville = 'Paris')
as clientParis;
```

- globalement avec `VUE`

```
create view clientParis as
select cID, nom, actif
from client
where ville = 'Paris';

select * from clientParis;
```

pour plus de sécurité

```
create view if not exists clientParis as
select cID, nom, actif
. . .
```

Requête récursive

- localement avec `WITH`, on peut nommer des requêtes

```
WITH R1 AS (requête-1),
     R2 AS (requête-2),
     …
     Rn AS (requête-n)
requête avec R1, R2, …, Rn (et autres tables);
```

- et aussi faire des requêtes récursives

```
WITH RECURSIVE
     R1 AS (requête-1),
     R2 AS (requête-2),
     …
     Rn AS (requête-n)
requête avec R1, R2, …, Rn (et autres tables);
```

R1, R2, .. Rn aussi possibles dans leurs définitions

Requête récursive

- exemple 1: relation « ancêtre de » dans une famille
- exemple 2: relation « supérieur de » dans une entreprise
- exemple 3: relation « meilleur chemin » dans un déplacement
- ces exemples font intervenir la notion de fermeture transitive inexprimable sans récursivité

Requête récursive

- exemple 1: relation « ancêtre de » dans une famille

```
create table "parentDe" (
[parent] text,
[enfant] text
);
```

- relation « grand-père »

```
select P1.parent, P2.enfant
from parentDe as P1, parentDe as P2
where P1.enfant = P2.parent;
```

- relation « arrière-grand-père »

```
select P1.parent, P3.enfant
from parentDe as P1, parentDe as P2, parentDe as P3
where P1.enfant = P2.parent
and P2.enfant = P3.parent;
```

parentDe

parent	enfant
Roger	Jean-Jacques
Jacques	Roger
Marthe	Roger
Rose	Marthe
Camille	Marthe
Jean-Jacques	Laurent
Ghislaine	Laurent
Jean-Jacques	Rémi
Léa	Raymonde
Raymonde	Ghislaine
Achille	Raymonde
Rémi	Gabriel
Suzanne	Jean-Jacques
Rémi	Léonard
Michel	Suzanne
Antonio	Michel
Pura	Suzanne
Roger	Marianne
Suzanne	Marianne
Marthe	Jacqueline
Jacqueline	Marjo
Rose	Hélène
Jacqueline	Claire
Ghislaine	Rémi
Jacques2	Ghislaine

Requête récursive

- relation « arrière-grand-père »

```
with grandParentDe (a, d)
 as (select P1.parent as a, P2.enfant as d
     from parentDe as P1, parentDe as P2
     where P1.enfant = P2.parent)
select a, enfant from grandParentDe, parentDe
where d = parentDe.parent;
```

- relation « ancètre »

```
with recursive ancetreDe (a, d)
 as (select parent as a, enfant as d from parentDe
     union
     select ancetreDe.a, parentDe.enfant
     from ancetreDe, parentDe
     where ancetreDe.d = parentDe.parent)
select a from ancetreDe
where d = 'Jean-Jacques';
```

parentDe

parent	enfant
Roger	Jean-Jacques
Jacques	Roger
Marthe	Roger
Rose	Marthe
Camille	Marthe
Jean-Jacques	Laurent
Ghislaine	Laurent
Jean-Jacques	Rémi
Léa	Raymonde
Raymonde	Ghislaine
Achille	Raymonde
Rémi	Gabriel
Suzanne	Jean-Jacques
Rémi	Léonard
Michel	Suzanne
Antonio	Michel
Pura	Suzanne
Roger	Marianne
Suzanne	Marianne
Marthe	Jacqueline
Jacqueline	Marjo
Rose	Hélène
Jacqueline	Claire
Ghislaine	Rémi
Jacques2	Ghislaine

Requête récursive

Exercice 2

- dans l'exemple 2: on crée 3 relations « employe », « chefDe », « projet »
  et on chercher à calculer la somme des salaires dans un projet

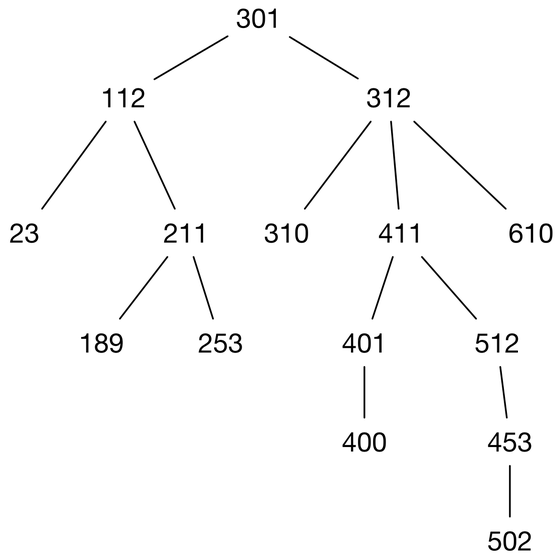
Exercice 3

- dans exemple 3: relation « vol » avec 4 attributs « départ », « destination », « compagnie », « coût »
  et on cherche le trajet le moins cher entre 2 villes

Requête récursive

Exercice 2

- dans l'exemple 2: on crée 3 relations « employe », « chefDe », « projet » et on chercher à calculer la somme des salaires dans un projet



employe

ID	salaire
301	5000
112	3000
211	2600
123	4000
312	1100
411	400
401	3500
512	900
453	3050
610	2100
189	2890
253	430
310	270
400	4810
502	3000

chefDe

mID	eID
301	112
112	23
112	211
211	189
211	253
301	312
312	310
312	411
411	401
401	400
411	512
512	453
453	502
312	610

projet

nom	mgrID
X	312
Y	211
Z	453

Requête récursive

Exercice 2

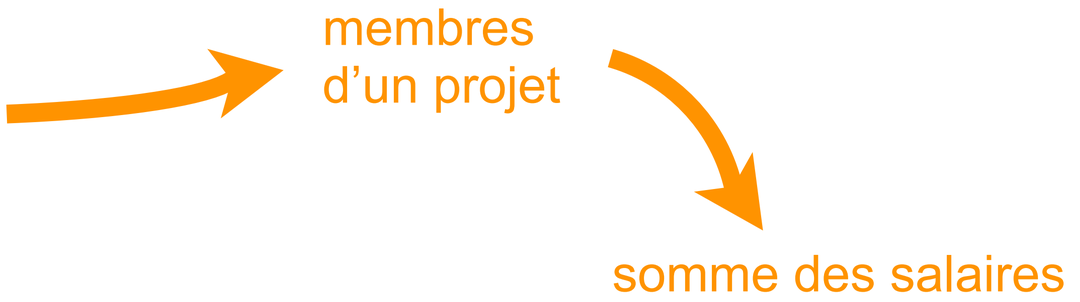
```
create table "employe" (
[ID] integer,
[salaire] integer
);
```

```
create table "chefDe" (
[mID] integer,
[eID] integer
);
```

```
create table "projet" (
[nom] text,
[mgrID] integer
);
```

• la requête récursive `superieurDe` est la fermeture transitive de la relation `chefDe`

```
with recursive superieurDe (c, e)
as (select mID as c, eID as e from chefDe
    union
    select superieurDe.c, chefDe.eID
    from superieurDe, chefDe
    where superieurDe.e = chefDe.mID)
select * from superieurDe;
```



• comment faire une requête plus efficace pour calculer les membres d'un projet ?

Requête récursive

Exercice 3

- relation « metro » avec 4 attributs « départ », « arrivée », « ligne », « temps »
  et on cherche le trajet le plus rapide
  entre 2 stations

depart	arrivee	ligne	temps
Pont de l'Alma	Invalides	C	5
Invalides	Musée d'Orsay	C	5
Musée d'Orsay	St-Michel	C	5
Invalides	La Tour Maubourg	8	2
La Tour Maubourg	Ecole Militaire	8	2
Ecole Militaire	La Motte Picquet	8	2
La Motte Picquet	Commerce	8	2
Commerce	Félix Faure	8	2
Dupleix	La Motte Picquet	6	2
La Motte Picquet	Cambronne	6	2
Cambronne	Sèvres Lecourbe	6	2
Sèvres Lecourbe	Pasteur	6	2
Pasteur	Montparnasse	6	2
Notre Dame des Champs	Edgar Quinet	6	2
Charles Michels	Emile Zola	10	2
Emile Zola	La Motte Picquet	10	2
La Motte Picquet	Ségur	10	2
Ségur	Duroc	10	2
Duroc	Vaneau	10	2
Vaneau	Sèvres Babylone	10	2

Requête récursive

Exercice 3

- relation « metro » avec 4 attributs « départ », « arrivée », « ligne », « temps »
  et on cherche le trajet le plus rapide
  entre 2 stations


```
with recursive chemin (d, a, t)
as (select depart as d, arrivee as a, temps as t from metro
    union
    select metro.depart, chemin.a, metro.temps + chemin.t
    from metro, chemin
    where metro.arrivee = chemin.d)
select * from chemin
where d = 'Montparnasse' and a = 'Pasteur' LIMIT 4;
```

- expliquer pourquoi mettre le mot clé LIMIT ?

Interface Python — SQLite

- on importe le paquetage `sqlite3` dans Python

```
import sqlite3
con = sqlite3.connect(‘minibank.db')
```

- on définit un curseur qui permet d’accéder à SQL

```
cur = con.cursor()

# Créer une table
cur.execute('''create table rlv
               (nOp integer, cID integer, delta integer, opDate text)''')

# Insérer un nouveau n-uplet
cur.execute("insert into rlv values (101, 243, +200, ‘2022-04-22’)" )

# Sauver les changements
con.commit()

# On peut fermer la connexion à la fin du programme,
# Et être ainsi sûr que tous les changements ont été validés.
con.close()
```

Interface Python — SQLite

- on peut alors utiliser les valeurs de SQL dans le programme Python

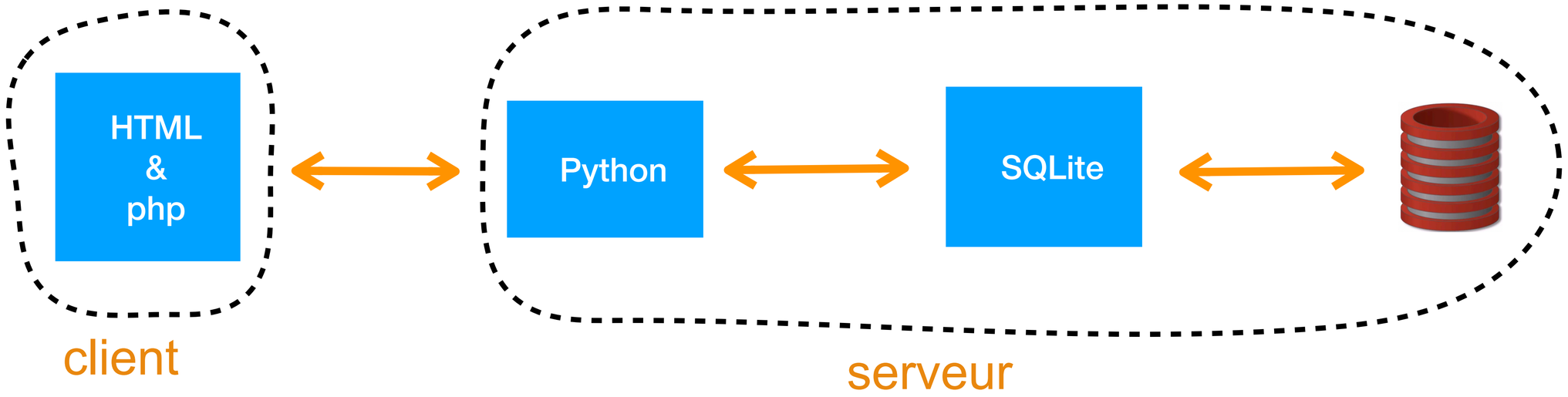
```
for row in cur.execute('select * from rlv order by opDate'):
        print(row)
```

- exemple avec notre première base de donnée

```
import sqlite3
con = sqlite3.connect("minisample.db")

cur = con.cursor()

for row in cur.execute ("select * from client"):
    print(row)

con.commit()
```

Interface HTML — SQLite

- avec PHP, on peut appeler un serveur Python
- le serveur Python peut s'interfacer avec SQLite




```
for row in cur.execute('select * from rlv order by opDate'):
        print(row)
```

Interface HTML — Python — SQLite

- on peut faire un petit serveur SQLite avec un interface HTML

```
<html><body>

<form method="get" action="http://localhost/cgi-bin/py1.cgi" target="_blank">

Red<input type="checkbox" name="color" value="red">
Green<input type="checkbox" name="color" value="green">
<input type="submit" value="Submit">

</form>

</body></html>
```

Interface HTML — Python — SQLite

- le serveur py1-cgi commence par traiter le formulaire envoyé

```
#!/opt/local/bin/python

print ("Content-type: text/html;charset=utf-8\n\n")

print ("<html>")
print ('<link rel="stylesheet" href="http://localhost/cgi-bin/w3.css">')
print ("<body>")

# ---------------- début du traitement du formulaire
import cgi, cgitb
cgitb.enable()
form = cgi.FieldStorage()

print ("<h2> Requête envoyée: </h2>")

print ('<div class="w3-container w3-margin">')
for key in form.keys() :
    r = key + ": " + "&".join (form.getlist(key)) + "<br>"
    print (r)
print ("<br>")
print ("</div>")
# ---------------- début du traitement SQL
```

Interface HTML — Python — SQLite

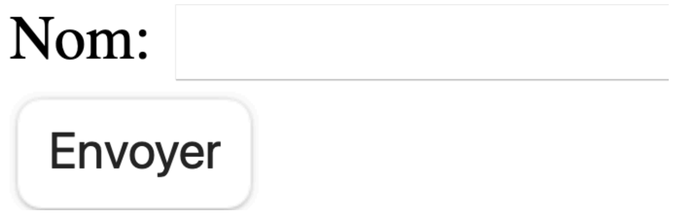
- le serveur `py1-cgi` imprime la couleur choisie et la table client

```
# ---------------- début du traitement SQL
import sqlite3

con = sqlite3.connect("minisample.db")
cur = con.cursor()

print ('<div class="w3-container w3-margin">')
for row in cur.execute ("select * from client"):
    print(row)
    print("<br>")

con.commit()
print ("</div>")
print ("</body>\n</html>")
```

Interface HTML — Python — SQLite

- un autre interface avec un formulaire texte

```
<html><body>

<form method="get" action="http://localhost/cgi-bin/py1.cgi" target="_blank">

Red<input type="checkbox" name="color" value="red">
Green<input type="checkbox" name="color" value="green">
<input type="submit" value="Submit">
</form>

<br><br>

<form action="http://localhost/cgi-bin/py2.cgi" method="get">
Nom: <input type="text" name="nom"><br>
<input type="submit" value="Envoyer">
</form>


</body></html>
```



Interface HTML — Python — SQLite

- le serveur `py2-cgi` utilise le formulaire pour la requête SQL

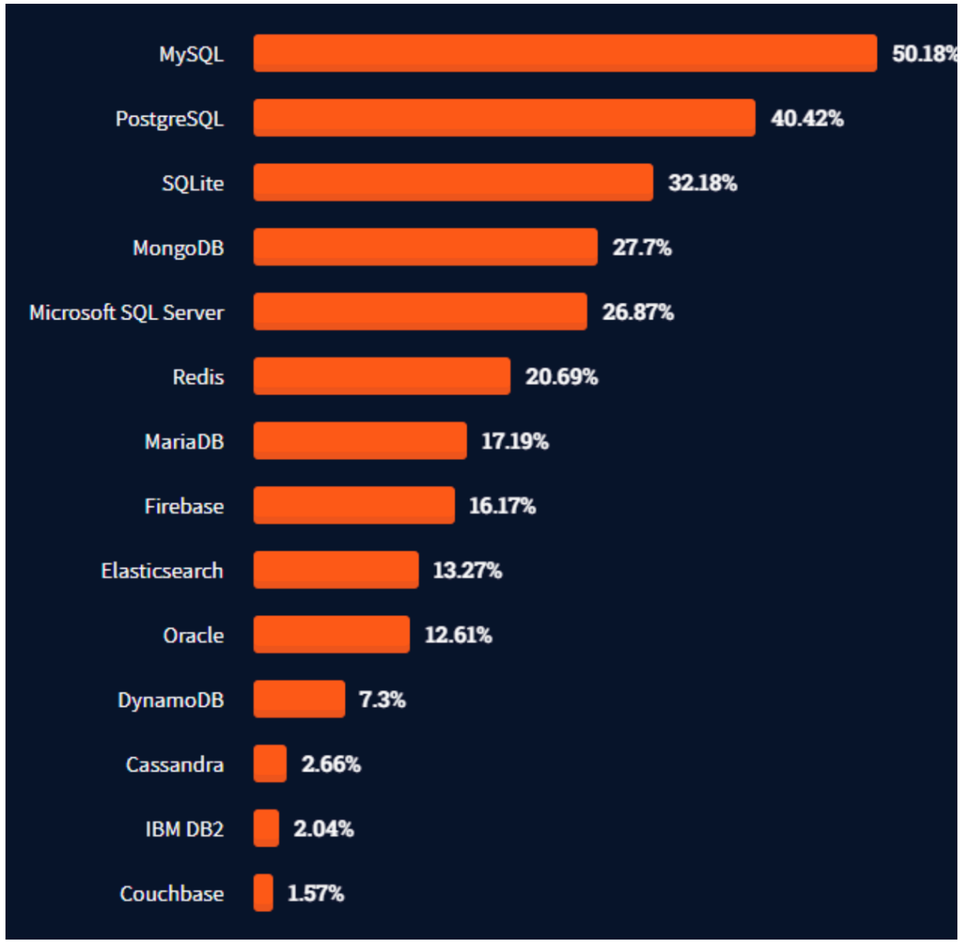
```
#!/opt/local/bin/python
print ("Content-type: text/html;charset=utf-8\n\n")

print ("<html>")
print ('<link rel="stylesheet" href="http://localhost/cgi-bin/w3.css">')
print ("<body>")
# ---------------- début du traitement fichier CGI
import cgi, cgitb
cgitb.enable()

form = cgi.FieldStorage()
for key in form.keys() :
    r = "".join (form.getlist(key))

print ("<h2> Welcome " + r + " ! </h2>")

print ('<div class="w3-container w3-margin">')

# ---------------- fin de l'en-tête
```

```
# ---------------- fin de l'en-tête
import sqlite3

con = sqlite3.connect("minisample.db")
cur = con.cursor()

print ('<div class="w3-container w3-margin">')

for row in cur.execute ('select * from client where nom = ' + '"' + r + '"'):
    print(row)
    print("<br>")

con.commit()
print ("</div>")

print ("</body>")
print ("</html>")
```

Alternatives pour un serveur SQL

- mySQL (Oracle)
- SQL server (Microsoft)
- PostgreSQL (public)
- critères:
  - passage à l’échelle
  - efficacité
  - sécurité
  - facilité d’installation
  - respect des standards

développeurs



Prochains cours

- vues (suite)
- droits d’accès
- correspondance avec la logique du 1er ordre
- dépendances et formes normales
- au-delà de SQL: graphes, orientation objet, XML