TP01 - Révision SQL

Minh-Hoang DANG

November 19, 2018

Vous devez vous connecter sur la VM Bases de Données.

Présentation du cas d'étude. Dans plusieurs TP nous allons utiliser la base Sakila qui permet de gérer une entreprise de location de films sur plusieurs sites. Cette base assez complexe est composée de trois blocs principaux : - La gestion des films : films, catégories, acteurs et langues - La gestion des magasins : magasins, employés, inventaires et locations - La gestion des clients : clients, paiements

On n'utilisera pas l'intégralité de la base à chaque TP.

```
In [ ]: set datestyle = 'european'

        -- PostGre SQL Kernel pour Jupyter
        -- pip3 install git+https://github.com/bgschiller/postgres_kernel.git
        -- connection: host='localhost' dbname='Sakila' user='postgres' password='postgres'
```

1 Exercices LDD

1.0.1 Exercice 1.

Créez la base de données sous Postgresql. Pour utiliser postgreSQL vous disposez de l'interface pgadmin. Le mot de passe en salle de TP est «pgAdmin».

1.0.2 Exercice 2.

Ouvrez le fichier base-schema et ajoutez toutes les contraintes de clés primaires, uniques et étrangères à la main. Vous pouvez également tester ces contraintes en insérant des enregistrements incorrects. Pensez à indiquer les actions à effectuer en cas de modification ou de suppression (ON DELETE, ON UPDATE). Une fois que cela est fait, vous pouvez importer les données du fichier base-data.sql.

```
In [ ]: -- file: base-constrain.sql
        set datestyle = 'european'
        /*
        ALTER TABLE country ADD CONSTRAINT PK_COUNTRY PRIMARY KEY(country_id);
        ALTER TABLE city ADD CONSTRAINT PK_CITY PRIMARY KEY(city_id);
        ALTER TABLE address ADD CONSTRAINT PK_ADDRESS PRIMARY KEY(address_id);
        ALTER TABLE store ADD CONSTRAINT PK_STORE PRIMARY KEY(store_id);
        ALTER TABLE customer ADD CONSTRAINT PK_CUSTOMER PRIMARY KEY(customer_id);
        ALTER TABLE payment ADD CONSTRAINT PK_PAYMENT PRIMARY KEY(payment_id);
        ALTER TABLE rental ADD CONSTRAINT PK_RENTAL PRIMARY KEY(rental_id);

        ALTER TABLE city ADD CONSTRAINT FK_CITY_COUNTRY FOREIGN KEY(country_id) REFERENCES
        country(country_id);

        ALTER TABLE address ADD CONSTRAINT FK_ADDRESS_CITY FOREIGN KEY(city_id) REFERENCES
```

```
city(city_id);

ALTER TABLE staff ADD CONSTRAINT PK_STAFF PRIMARY KEY(staff_id);

ALTER TABLE store ADD CONSTRAINT FK_STORE_MANAGER FOREIGN KEY(manager_staff_id)
REFERENCES staff(staff_id);
ALTER TABLE store ADD CONSTRAINT FK_STORE_ADDRESS FOREIGN KEY(address_id) REFERENCES
address(address_id);

ALTER TABLE staff ADD CONSTRAINT FK_STAFF_ADDRESS FOREIGN KEY(address_id) REFERENCES
address(address_id);
ALTER TABLE staff ADD CONSTRAINT FK_STAFF_STORE FOREIGN KEY(store_id) REFERENCES
store(store_id);

ALTER TABLE customer ADD CONSTRAINT FK_CUSTOMER_STORE FOREIGN KEY(store_id) REFERENCES
store(store_id);
ALTER TABLE customer ADD CONSTRAINT FK_CUSTOMER_ADDRESS FOREIGN KEY(address_id)
REFERENCES address(address_id);

ALTER TABLE inventory ADD CONSTRAINT PK_INVENTORY PRIMARY KEY(inventory_id);
ALTER TABLE inventory ADD CONSTRAINT FK_INVENTORY_STORE FOREIGN KEY(store_id) REFERENCES
store(store_id);
--ALTER TABLE inventory ADD CONSTRAINT FK_INVENTORY_RENTAL FOREIGN KEY(film_id)
REFERENCES rental(rental_id);

ALTER TABLE payment ADD CONSTRAINT FK_PAYMENT_CUSTOMER FOREIGN KEY(customer_id)
REFERENCES customer(customer_id);
ALTER TABLE payment ADD CONSTRAINT FK_PAYMENT_STAFF FOREIGN KEY(staff_id) REFERENCES
staff(staff_id);
ALTER TABLE payment ADD CONSTRAINT FK_PAYMENT_RENTAL FOREIGN KEY(rental_id) REFERENCES
rental(rental_id);

ALTER TABLE rental ADD CONSTRAINT FK_RENTAL_INVENTORY FOREIGN KEY(inventory_id)
REFERENCES inventory(inventory_id);
ALTER TABLE rental ADD CONSTRAINT FK_RENTAL_CUSTOMER FOREIGN KEY(customer_id) REFERENCES
customer(customer_id);
ALTER TABLE rental ADD CONSTRAINT FK_RENTAL_STAFF FOREIGN KEY(staff_id) REFERENCES
staff(staff_id);
*/
```

2 Utiliser les dates

2.0.1 Exercice 3. Tester les requêtes suivantes qui pourront peut-être vous servir à un moment dans le TP.

```
In [3]: SELECT date '2001-10-01' - date '2001-09-28';
        SELECT timestamp '2001-09-28 01:00' + interval '18 hours';
        SELECT age(timestamp '2001-04-10', timestamp '1957-06-13');
        SELECT age(timestamp '1998-06-11');
        SELECT date_part('year', timestamp '2001-02-16 20:38:40');
        SELECT date_trunc('hour', timestamp '2001-02-16 20:38:40');
```

```
1 row(s) returned.
```

date_trunc
2001-02-16 20:00:00

3 Requêtes SQL

Ecrire des requêtes SQL pour répondre aux questions suivantes. Par défaut on ne demande jamais de trier les résultats ou de supprimer les doublons.

3.0.1 Exercice 4. Identifiant de tous les clients ayant un montant de paiement compris entre 1 et 2 euros.

```
In [4]: SELECT DISTINCT c.first_name, c.last_name, p.amount
        FROM customer c, payment p
        WHERE c.customer_id = p.customer_id and p.amount between 1 and 2
        LIMIT 10
```

```
10 row(s) returned.
```

first_name	last_name	amount
TED	BREAUX	1.99
GEORGIA	JACOBS	1.99
CAROL	GARCIA	1.99
NINA	SOTO	1.99
PEGGY	MYERS	1.99
NATHANIEL	ADAM	1.99
NICHOLAS	BARFIELD	1.99
PETER	MENARD	1.99
VIOLA	HANSON	1.99
DORIS	REED	1.99

3.0.2 Exercice 5. Locations (tous les attributs) qui ont été retournées moins de 24h après leur emprunt

```
In [5]: SELECT DISTINCT *
        FROM rental r
        WHERE r.return_date < r.rental_date + interval '24 hours'
        LIMIT 10
```

```
10 row(s) returned.
```

rental_id	rental_date	inventory_id	customer_id	return_date	staff_id
7188	2005-07-27 08:32:08	2199	259	2005-07-28 08:02:08	1
7933	2005-07-28 12:27:27	3175	297	2005-07-29 10:34:27	2
11923	2005-08-17 16:21:47	1907	72	2005-08-18 14:26:47	2
9004	2005-07-30 05:04:27	3775	178	2005-07-31 00:49:27	1
12843	2005-08-19 01:58:54	3279	128	2005-08-20 00:20:54	2
15667	2005-08-23 09:02:03	1302	73	2005-08-24 05:47:03	1
11270	2005-08-02 14:18:07	1882	418	2005-08-03 08:20:07	1
5360	2005-07-09 18:14:03	679	562	2005-07-10 15:17:03	2
3746	2005-07-06 12:10:51	3387	424	2005-07-07 11:36:51	2
11197	2005-08-02 11:45:07	3041	122	2005-08-03 09:07:07	1

3.0.3 Exercice 6. Locations (tous les attributs) qui ont été faites entre 23h et minuit (exclus)

```
In [6]: SELECT *
        FROM rental r
        WHERE date_part('hour', r.rental_date) between 23 AND 24
        LIMIT 10
```

```
10 row(s) returned.
```

rental_id	rental_date	inventory_id	customer_id	return_date	staff_id
3	2005-05-24 23:03:39	1711	408	2005-06-01 22:12:39	1
4	2005-05-24 23:04:41	2452	333	2005-06-03 01:43:41	2
5	2005-05-24 23:05:21	2079	222	2005-06-02 04:33:21	1
6	2005-05-24 23:08:07	2792	549	2005-05-27 01:32:07	1
7	2005-05-24 23:11:53	3995	269	2005-05-29 20:34:53	2
8	2005-05-24 23:31:46	2346	239	2005-05-27 23:33:46	2
139	2005-05-25 23:00:21	327	257	2005-05-29 17:12:21	1
140	2005-05-25 23:34:22	655	354	2005-05-27 01:10:22	1
141	2005-05-25 23:34:53	811	89	2005-06-02 01:57:53	1
142	2005-05-25 23:43:47	4407	472	2005-05-29 00:46:47	2

3.0.4 Exercice 7. Prénom et nom des clients dont le code postal commence par 47

```
In [7]: SELECT c.first_name, c.last_name
        FROM customer c, address a
        WHERE c.address_id = a.address_id and a.postal_code ~* '47.*'
```

```
21 row(s) returned.
```

first_name	last_name
PAMELA	BAKER
BERNICE	WILLIS
MARION	SNYDER
STACY	CUNNINGHAM
HOLLY	FOX
AGNES	BISHOP
COLLEEN	BURTON
TERRY	CARLSON
DAVID	ROYAL
KENNETH	GOODEN
NATHAN	RUNYON
KYLE	SPURLOCK
TROY	QUIGLEY
LLOYD	DOWD
DEREK	BLAKELY
MILTON	HOWLAND
ELMER	NOE
MITCHELL	WESTMORELAND
FELIX	GAFFNEY
MORRIS	MCCARTER
TRACY	HERRMANN

3.0.5 Exercice 8. Tous les identifiants distincts d'adresse en France

```
In [8]: SELECT DISTINCT a.address_id
        FROM address a, city cty, country c
        WHERE a.city_id = cty.city_id and c.country_id = cty.country_id and upper(c.country) =
        upper('france')
```

```
4 row(s) returned.
```

address_id
39
108
166
407

3.0.6 Exercice 9. Identifiant des films de l'inventaire qui n'ont jamais été loués

```
In [9]: SELECT i.inventory_id, i.film_id
        FROM inventory i
        WHERE i.inventory_id NOT IN (SELECT r.inventory_id FROM rental r );
```

```
1 row(s) returned.
```

inventory_id	film_id
5	1

```
In [10]: SELECT DISTINCT r.inventory_id
         FROM rental r
         WHERE r.inventory_id = 5;
```

```
0 row(s) returned.
```

3.0.7 Exercice 10. Identifiant des films et nombre de fois qu'ils ont été loués

```
In [11]: SELECT i.film_id, COUNT(r.rental_id)
         FROM inventory i, rental r
         WHERE i.inventory_id = r.inventory_id GROUP BY(i.film_id)
         LIMIT 10
```

```
10 row(s) returned.
```

film_id	count
652	14
273	25
51	23
951	28
70	17
839	7
350	20
758	8
539	10
278	7

3.0.8 Exercice 11. Prénom, nom et nombre de locations par client

```
In [12]: SELECT c.first_name, c.last_name, COUNT(DISTINCT r.rental_id)
         FROM customer c, rental r
         WHERE c.customer_id = r.customer_id
         GROUP BY c.first_name, c.last_name
         LIMIT 10
```

10 row(s) returned.

first_name	last_name	count
AARON	SELBY	24
ADAM	GOOCH	22
ADRIAN	CLARY	19
AGNES	BISHOP	23
ALAN	KAHN	26
ALBERT	CROUSE	23
ALBERTO	HENNING	21
ALEX	GRESHAM	33
ALEXANDER	FENNELL	36
ALFRED	CASILLAS	26

3.0.9 Exercice 12. Nombre de films distincts loués par une personne habitant en Allemagne

```
In [13]: SELECT COUNT(DISTINCT i.film_id)
         FROM inventory i, customer c, rental r, address a, country h, city cty
         WHERE i.inventory_id = r.inventory_id and c.customer_id = r.customer_id and a.city_id =
         cty.city_id
                 and cty.country_id = h.country_id and upper(h.country) = upper('Germany')
```

1 row(s) returned.

count
958

3.0.10 Exercice 13. Identifiant de magasin et nombre de locations

```
In [14]: SELECT s.store_id, COUNT(DISTINCT r.rental_id)
         FROM store s, inventory i, rental r
         WHERE s.store_id = i.store_id AND r.inventory_id = i.inventory_id
         GROUP BY s.store_id
```

2 row(s) returned.

store_id	count
1	7923
2	8121

3.0.11 Exercice 14. Prénom, nom, montant moyen et montant total dépensé par client

```
In [15]: SELECT c.first_name, c.last_name, AVG(p.amount), SUM(p.amount)
         FROM customer c, payment p
         WHERE c.customer_id = p.customer_id
         GROUP BY c.first_name, c.last_name
         LIMIT 10
```

10 row(s) returned.

first_name	last_name	avg	sum
JONATHAN	SCARBOROUGH	4.04556	72.82
TRACEY	BARRETT	4.39741	118.73
RUSSELL	BRINSON	3.79556	136.64
FRANKLIN	TROUTMAN	3.39909	74.78
CASSANDRA	WALTERS	4.32333	129.7
CECIL	VINES	4.45154	115.74
JORDAN	ARCHULETA	4.42333	132.7
THOMAS	GRIGSBY	4.23	105.75
RUBY	WASHINGTON	3.95429	110.72
STANLEY	SCROGGINS	4.65667	139.7

3.0.12 Exercice 15. Numéro de client, jour et nombre de location pour ce jour et ce client

```
In [16]: SELECT r1.customer_id, r1.rdate, r1.nbLocationClient, r2.nbLocationJour FROM
         (
             SELECT c.customer_id, r.rental_date::timestamp::date as rdate, COUNT(DISTINCT
         r.rental_id) as nbLocationClient
             FROM customer c, rental r
             WHERE c.customer_id = r.customer_id
             GROUP BY c.customer_id, rdate
         ) as r1
         JOIN
         (
             SELECT r.rental_date::timestamp::date as rdate, COUNT(DISTINCT r.rental_id) as
         nbLocationJour
             FROM rental r
             GROUP BY rdate
         ) as r2
         ON r1.rdate = r2.rdate
         LIMIT 10
```

```
10 row(s) returned.
```

customer_id	rdate	nblocationclient	nblocationjour
1	2005-05-25	1	137
1	2005-05-28	1	196
1	2005-06-15	3	348
1	2005-06-16	1	324
1	2005-06-18	2	344
1	2005-06-21	1	275
1	2005-07-08	2	512
1	2005-07-09	2	513
1	2005-07-11	1	461
1	2005-07-27	1	649

3.0.13 Exercice 16. Prénom et nom des clients ayant effectué strictement moins de 15 locations

```
In [17]: SELECT c.first_name, c.last_name, COUNT(DISTINCT r.rental_id) as nbLocations
         FROM customer c, rental r
         WHERE c.customer_id = r.customer_id
         GROUP BY c.first_name, c.last_name
         HAVING COUNT(DISTINCT r.rental_id) < 15
```

```
4 row(s) returned.
```

first_name	last_name	nblocations
BRIAN	WYMAN	12
KATHERINE	RIVERA	14
LEONA	OBRIEN	14
TIFFANY	JORDAN	14

3.0.14 Exercice 17. Nombre de clients qui ont effectué une location le 30 juillet 2005 ou habitent au code postal 35200 (ne pas utiliser OR)

```
In [18]: (
             SELECT c.customer_id
             FROM customer c, rental r
             WHERE r.rental_date::timestamp::date = '2005-07-30'::date
         )
         UNION DISTINCT
         (
             SELECT c.customer_id
             FROM customer c, address a
             WHERE c.address_id = a.address_id and a.postal_code = '35200'
         )
         LIMIT 10
```

```
10 row(s) returned.
```

customer_id
176
576
292
161
528
524
360
99
384
188

3.0.15 Exercice 18. Identifiant des clients ayant effectué une location le 11 d'un mois quelconque mais pas le 10 d'un mois quelconque

```
In [19]: SELECT r1.customer_id, r2.rdate FROM
         (
             SELECT c.customer_id, date_part('month', r.rental_date) as rdate
             FROM customer c, rental r
             WHERE c.customer_id = r.customer_id and
                 date_part('month', r.rental_date) = 11
         ) r1
         JOIN
         (
             SELECT c.customer_id, date_part('month', r.rental_date) as rdate
             FROM customer c, rental r
             WHERE c.customer_id = r.customer_id and
                 date_part('month', r.rental_date) != 10
         ) r2
         ON r1.customer_id = r2.customer_id
```

```
0 row(s) returned.
```

3.0.16 Exercice 19. Nombre maximum de location en une journée

```
In [20]: SELECT MAX(r1.nbLoc) FROM
         (
             SELECT r.rental_date::timestamp::date as rdate, COUNT(DISTINCT r.rental_id) as nbLoc
             FROM rental r
             GROUP BY rdate
         ) r1
```

```
1 row(s) returned.
```

max
679

3.0.17 Exercice 20. Numéro du client ayant effectué le plus de locations en une journée. Proposer au moins 3 solutions !

Solution 1:

```
In [21]: SELECT r1.customer_id, MAX(r1.nbLoc) as record FROM
         (
             SELECT c.customer_id, r.rental_date::timestamp::date as rdate, COUNT(DISTINCT
         r.rental_id) as nbLoc
             FROM rental r, customer c
             WHERE r.customer_id = c.customer_id
             GROUP BY c.customer_id, rdate
         ) r1
         GROUP BY r1.customer_id
         ORDER BY record desc
         LIMIT 1
```

```
1 row(s) returned.
```

customer_id	record
563	7

Solution 2:

```
In [22]: WITH r1 as (
             SELECT c.customer_id, r.rental_date::timestamp::date as rdate, COUNT(DISTINCT
         r.rental_id) as nbLoc
             FROM rental r, customer c
             WHERE r.customer_id = c.customer_id
             GROUP BY c.customer_id, rdate
         )
         SELECT r1.customer_id, r1.nbLoc
         FROM r1
         WHERE r1.nbLoc = (SELECT MAX(r1.nbLoc) FROM r1)
```

```
1 row(s) returned.
```

customer_id	nbloc
563	7

Solution 3:

```
In [23]: WITH r1 as (
             SELECT c.customer_id, r.rental_date::timestamp::date as rdate, COUNT(DISTINCT
         r.rental_id) as nbLoc
             FROM rental r, customer c
             WHERE r.customer_id = c.customer_id
             GROUP BY c.customer_id, rdate
         )
         SELECT r1.customer_id, r1.nbLoc
         FROM r1
         WHERE r1.nbLoc >= ALL(SELECT r1.nbLoc FROM r1 )
```

```
1 row(s) returned.
```

customer_id	nbloc
563	7

```
In [ ]:
```