TD 7 : Contraintes (suite)

Définition de données et Contraintes

2024-11-08

Avec solutions

- **L3 MIASHS/Ingémath**
- **Université Paris Cité**
- Année 2024-2025
- Course Homepage

- Moodle

Ojectifs

Cette séance est consacrée à l'enrichissement des schémas `pagila` et `nycflights`.
L'objectif est d'ajouter des contraintes aux schémas, en particulier, des contraintes d'exclusion.
Cette séance est l'occasion de se familiariser avec les types `timestamp` et `tsrange` et avec les opérateurs de recouvrement.

Contraintes SQL (suite)

Schéma `pagila`

Question

Imposer la contrainte suivante : un client ne peut emprunter plusieurs DVD simultanément.

Écrire d'abord une requête correspondant à la contrainte.
La requête est facile à écrire si on est prêt à utiliser le type `tsrange`, l'opérateur de recouvrement `&&` (`overlap`).
Postgresql documentation on range types
Dans l'instance du schéma `pagila` disponible sur le serveur `etu-pgsql`, la contrainte est-elle vérifiée ?

Solution

```
select
  r1.customer_id,
  r1.inventory_id,
  r1.rental_date,
  r1.return_date,
  r2.inventory_id,
  r2.rental_date,
  r2.return_date
from
  rental r1
join
  rental r2
on (r1.customer_id=r2.customer_id AND r1.rental_id<>r2.rental_id)
where
  tsrange(r1.rental_date, r1.return_date, '[)') &&
  tsrange(r2.rental_date, r2.return_date, '[)') ;
```

☠ La contrainte n'est pas du tout satisfaite par l'instance de `pagila`

```
ALTER TABLE
  pagila.rental
ADD CONSTRAINT
  xcl_simul_rental
EXCLUDE USING gist (
  customer_id with =,
  rental_id with <>,
  tsrange(rental_date, return_date) with &&
) ;
```

👉 Sur l'instance disponible de `pagila`, cette contrainte ne peut pas être surimposée.

```
ERROR:  could not create exclusion constraint "xcl_simul_rental"
DETAIL:  Key (customer_id, rental_id, tsrange(rental_date, return_date))=(408, 3, ["2005-05-24
```

Question

Imposer la contrainte : un film est identifié par son titre, son année de sortie et sa langue originale.

Solution

La requête correspondante est

```
SELECT
  COUNT(*)
FROM
  pagila.film AS f1
JOIN
  pagila.film AS f2
ON (
    f1.title = f2.title AND
    f1.release_year = f2.release_year AND
    f1.original_language_id = f2.original_language_id AND
    f1.film_id < f2.film_id)
```

Il s'agit d'une contrainte d'unicité. Elle est satisfaite dans l'instance de `pagila`

```
ALTER TABLE pagila.film
ADD CONSTRAINT uq_film_title_release_year_original_language_id
UNIQUE (title, release_year, original_language_id) ;
```

La description du schéma de `pagila.film` (`\d pagila.film`) nous renvoie alors :

```
...
Indexes:
    "film_pkey" PRIMARY KEY, btree (film_id)
    "film_fulltext_idx" gist (fulltext)
    "idx_fk_language_id" btree (language_id)
    "idx_fk_original_language_id" btree (original_language_id)
    "idx_title" btree (title)
    "uq_film_title_release_year_original_language_id" UNIQUE CONSTRAINT, btree (title, release
...
```

Question

Imposer la contrainte : le prix de location doit être croissant en fonction de la durée du film

Solution

```
SELECT
  COUNT(*)
FROM
  pagila.film AS f1
JOIN
  pagila.film AS f2
ON (
  f1.rental_rate < f2.rental_rate AND
  f1.length > f2.length
) ;
```

☠ La contrainte n'est pas satisfaite dans l'instance courante de `pagila`.
La contrainte devrait pouvoir être formulée comme une contrainte d'exclusion.

```
ALTER TABLE
  pagila.film
ADD CONSTRAINT
  xcl_anomalous_rental_rate
EXCLUDE USING gist (
  "length" with <,
  rental_rate with >
) ;
```

mais :

```
ERROR:  operator <(smallint,smallint) is not commutative
DETAIL:  Only commutative operators can be used in exclusion constraints.
```

Est-ce que l'impossiblité d'utiliser un opérateur non-commutatif est de principe ou technique ?
👉 On peut utiliser un contournement laborieux en utilisant une contrainte de ligne (`CHECK`) qui va simuler la contrainte de table (voir td6).

Question

Imposer la contrainte : un même DVD ne peut pas être loué simultanément à deux clients différents.

Solution

```
ALTER TABLE
  pagila.rental
ADD CONSTRAINT
  xcl_simul_rental_inventory
EXCLUDE USING gist (
  inventory_id with =,
  rental_id with <>,
  tsrange(rental_date, return_date) with &&
) ;
```

L'examen de la deéfinition de la table `rental` nous révèle maintenant :

```
Indexes:
    "rental_pkey" PRIMARY KEY, btree (rental_id)
    "idx_fk_inventory_id" btree (inventory_id)
    "idx_unq_rental_rental_date_inventory_id_customer_id" UNIQUE, btree (rental_date, inventor
    "xcl_simul_rental_inventory" EXCLUDE USING gist (inventory_id WITH =, rental_id WITH <>, t
te) WITH &&)
Foreign-key constraints:
    "rental_customer_id_fkey" FOREIGN KEY (customer_id) REFERENCES customer(customer_id) ON UP
    "rental_inventory_id_fkey" FOREIGN KEY (inventory_id) REFERENCES inventory(inventory_id) O
    "rental_staff_id_fkey" FOREIGN KEY (staff_id) REFERENCES staff(staff_id) ON UPDATE CASCADE
```

Schéma `nycflights`

Question

Dans `flights` et `weather` imposer les contraintes
`year, month, day, hour  time_hour`
`time_hour  year, month, day, hour`

Solution

```
ALTER TABLE nycflights_sandbox.weather
ADD CONSTRAINT weather_fd_1
EXCLUDE USING gist (
  year WITH =,
  month WITH =,
  day WITH =,
  hour WITH =,
  time_hour WITH <>
) ;
```

```
ALTER TABLE nycflights_sandbox.flights
ADD CONSTRAINT flights_fd_1
EXCLUDE USING gist (
  year WITH =,
  month WITH =,
  day WITH =,
  hour WITH =,
  time_hour WITH <>
) ;
```

👉 Nous sommes dans une situation où une colonne peut être calculée à partir d'autres colonnes. PostgreSQL propose des genrated columns pour traîter ce genre de situations.

> Une colonne générée est une colonne spéciale qui est toujours calculée à partir d'autres colonnes. Elle est donc pour les colonnes ce qu'une vue est pour les tables. Il existe deux types de colonnes générées : les colonnes stockées et les colonnes virtuelles. Une colonne générée stockée est calculée lorsqu'elle est écrite (insérée ou mise à jour) et occupe l'espace de stockage comme s'il s'agissait d'une colonne normale. Une colonne générée virtuelle n'occupe pas d'espace de stockage et est calculée lorsqu'elle est lue. Ainsi, une colonne générée virtuelle est similaire à une vue et une colonne générée stockée est similaire à une vue matérialisée (sauf qu'elle est toujours mise à jour automatiquement). PostgreSQL n'implémente actuellement que les colonnes générées stockées.

Question

Dans `flights` imposer la contrainte : un aéronef ne peut pas effectuer deux missions simultanément.

Question

Dans `flights` imposer la contrainte : un aéronef ne peut pas être exploité par deux compagnies différentes à la même date.

Solution

Question

Dans `planes`, imposer les contraintes `model manufacturer` et `model type`

Solution

```
create function nycflights.lldistance(
  p_lat_1 float8,
  p_lon_1 float8,
  p_lat_2 float8,
  p_lon_2 float8
) returns float8
language SQL AS
$$
select
  earth_distance(
    ll_to_earth(p_lat_1, p_lon_1),
    ll_to_earth(p_lat_2, p_lon_2)
  ) ;
$$ ;
```

```
select
  A_1.faa, A_2.faa, nycflights.lldistance(A_1.lat, A_1.lon, A_2.lat, A_2.lon) as dista
from
  (select
    *
  from
    nycflights.airports
  where
    faa in ('LGA', 'JFK', 'EWR')
  ) A_1
cross join
  (select
    *
  from
    nycflights.airports
  where
    not faa in ('LGA', 'JFK', 'EWR')
  ) A_2 ;
```

```
select
  a_1.faa, a_2.faa,
  (point(a_1.lon, a_1.lat) <@> point(a_2.lon, a_2.lat))/1.609
FROM
  nycflights.airports a_1
cross join
  nycflights.airports a_2
where
  a_1.faa in ('EWR', 'LGA', 'JFK')  AND
  a_2.faa in ('LAX', 'ATL')
ORDER BY a_2.faa, a_1.faa ;
```