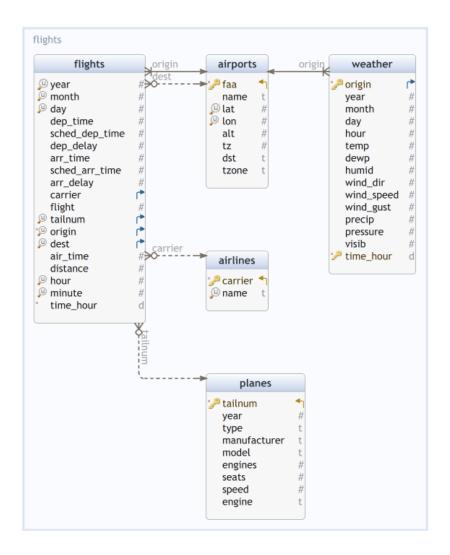
- L3 MIASHS/Ingémath
- Université Paris Cité
- Année 2023-2024
- Course Homepage
- Moodle



🛕 Les requêtes portent sur le schéma nycflights légèrement nettoyé.



 ${\it Fig.}\ 1:{\it NYCFlights}$ en relationel à pattes de corbeau

Définition du schéma en SQL

```
CREATE TABLE weather (
CREATE TABLE airlines (
                                              origin text NOT NULL,
    carrier text NOT NULL,
                                              "year" int4 NULL,
    "name" text NULL,
                                              "month" int4 NULL,
    CONSTRAINT airlines_pk
                                              "day" int4 NULL,
        PRIMARY KEY (carrier),
                                              "hour" int4 NULL,
    CONSTRAINT airlines_un
                                              "temp" float8 NULL,
        UNIQUE (name)
                                              dewp float8 NULL,
);
                                              humid float8 NULL,
                                              wind_dir float8 NULL,
CREATE TABLE airports (
                                              wind_speed float8 NULL,
    faa text NOT NULL,
                                              wind_gust float8 NULL,
    "name" text NULL,
                                              precip float8 NULL,
    lat float8 NULL,
                                              pressure float8 NULL,
    lon float8 NULL,
                                              visib float8 NULL,
    alt float8 NULL,
                                              time_hour timestamptz NOT NULL,
    tz float8 NULL,
                                              CONSTRAINT weather_pk
    dst text NULL,
                                                  PRIMARY KEY (origin, time_hour)
    tzone text NULL,
                                          );
    CONSTRAINT airports_pk
        PRIMARY KEY (faa),
                                          ALTER TABLE weather ADD
    CONSTRAINT airports_un
                                              CONSTRAINT weather_fk
       UNIQUE (name),
                                              FOREIGN KEY (origin)
    CONSTRAINT airports_un_ll
                                              REFERENCES airports(faa)
        UNIQUE (lat, lon)
                                              ON DELETE CASCADE
);
                                              ON UPDATE CASCADE;
CREATE TABLE planes (
    tailnum text NOT NULL,
    "year" int4 NULL,
    "type" text NULL,
    manufacturer text NULL,
    model text NULL,
    engines int4 NULL,
    seats int4 NULL,
    speed int4 NULL,
    engine text NULL,
    CONSTRAINT planes_pk PRIMARY KEY (tailnum)
);
```

Dans le schéma nycflights, on a aussi les dépendances fonctionnelles suivantes :

Table airports

• faa, name, et (lon, lat) sont des clés.

Table airlines

• carrier et name sont des clés

Table weather

- origin, time_hour est une clé
- time_hour → year, month, day, hour
- year, month, day, hour → time_hour

Table planes

- tailnum est une clé
- model → manufacturer, engines, engine, type

Table flights

• tailnum, time_hour → carrier

```
ALTER TABLE flights ADD
CREATE TABLE flights (
                                               CONSTRAINT flights_fk
    "year" int4 NULL,
                                               FOREIGN KEY (carrier)
    "month" int4 NULL,
                                               REFERENCES airlines(carrier)
    "day" int4 NULL,
                                               ON DELETE SET NULL
    dep_time int4 NULL,
                                               ON UPDATE CASCADE;
    sched_dep_time int4 NULL,
                                          ALTER TABLE flights ADD
    dep_delay float8 NULL,
                                               CONSTRAINT flights_fk_dest
    arr time int4 NULL,
                                               FOREIGN KEY (dest)
    sched_arr_time int4 NULL,
                                               REFERENCES airports(faa)
    arr_delay float8 NULL,
                                               ON DELETE SET NULL
    carrier text NULL,
                                               ON UPDATE CASCADE;
    flight int4 NULL,
    tailnum text NOT NULL,
                                           ALTER TABLE flights ADD
    origin text NOT NULL,
                                               CONSTRAINT flights_fk_origin
    dest text NULL,
                                               FOREIGN KEY (origin)
    air time float8 NULL,
                                               REFERENCES airports(faa)
    distance float8 NULL,
                                               ON DELETE SET NULL
    "hour" float8 NULL,
                                               ON UPDATE CASCADE;
    "minute" float8 NULL,
    time_hour timestamptz NOT NULL,
                                          ALTER TABLE flights ADD
    CONSTRAINT flights_pk
                                               CONSTRAINT flights_fk_planes
                                               FOREIGN KEY (tailnum)
        PRIMARY KEY (
                                               REFERENCES planes(tailnum)
            tailnum, origin, time_hour)
);
                                               ON DELETE SET NULL
                                               ON UPDATE CASCADE;
```

- time_hour → sched_dep_time
- sched_dep_time, dep_time → dep_delay
- sched_arr_time, arr_time → arr_delay
- origin, dest, dep_time, arr_time \rightarrow airtime
- time_hour → year, month, day, hour, minute
- year, month, day, hour, minute → time_hour
- origin, dest → distance
- (tailnum, origin, time_hour) est une clé
- (flight, dest, origin, year, month, day) est une clé
- Solution
- Solution
- Solution

Exercice: Requêtes (schéma nycflights)

Requête 1

Pour chaque couple origine/destination, lister les caractéristiques de l'avion le plus rapide sur la liaison.

Solution

```
WITH R As (
  SELECT f.origin, f.dest, f.tailnum, RANK() OVER w AS rnk
  FROM flights AS f
  WHERE f.airtime IS NOT NULL
  WINDOW w AS (PARTITION BY f.origin, f.dest ORDER by f.airtime DESC)
SELECT R.origin, R.dest, p.*
FROM (SELECT * FROM R WHERE R.rnk=1) AS S
  JOIN planes as p
  ON (R.tailnum=p.tailnum);
WITH R As (
  SELECT f.origin, f.dest, min(f.airtime) as min_time
  FROM flights AS f
  WHERE f.airtime IS NOT NULL
  GROUP BY f.origin, f.dest
), S AS (
  SELECT f.origin, f.dest, f.tailnum
  FROM flights AS f NATURAL JOIN R
  WHERE f.airtime = R.min_time
SELECT S.origin, S.dest, p.*
FROM S
  JOIN planes as p
  ON (S.tailnum=p.tailnum);
```

Requête 2

Pour chaque aéroport d'origine, déterminer pour chaque heure de la journée, les températures maximales et minimales

Solution

```
WITH o AS (
    SELECT DISTINCT f.origin
    FROM flights as f
)
SELECT w.origin, w.hour, MAX(w.temp), MIN(w.temp)
FROM o NATURAL JOIN weather w
GROUP BY w.hour;
```

Requête 3

Pour chaque aéroport d'origine, pour chaque température enregistrée en début d'heure, arrondie à l'entier le plus proche, indiquer la proportion de vols avec un retard supérieur à 30 mn au décollage.

Solution

```
WITH R AS (
    SELECT f.origin, f.year. f.month, f.day, f.hour, f.flight, f.dep_delay, ROUND(w.temp, 0) as
FROM flights f JOIN weater w ON
    (f.origin=w.origin AND
    f.year=w.year AND
    f.month=w.month AND
    f.day=w.day AND
    f.hour=w.hour)
)

SELECT f.origin, f.t, SUM(f.dep_delay > 30)/COUNT(*) AS p
FROM R AS f
GROUP BY f.origin, f.t;
```

Requête 4

Pour chaque aéroport de destination, lister les modèles d'avion qui ont atterri au moins une fois dans cet aéroport.

Solution

```
SELECT DISTINCT(f.dest, p.models)
FROM flights f JOIN planes p ON (f.tailnum=p.tailnum);
```

Requête 5

Pour chaque modèle d'avion, lister pour chaque semaine, le nombre de vols efectivement réalisés.

Solution

```
SELECT p.model, DATE_PART('week', f.time_hour) AS semaine, COUNT(*) AS n FROM flights f JOIN planes p ON (f.tailnum=p.tailnum)
WHERE f.dep_time IS NOT NULL
GROUP BY p.model, DATE_PART('week', f.time_hour) AS semaine;
```

Requête 6

Quelles sont les destinations qui ne sont pas desservies le jeudi?

Solution

```
WITH R AS (
    SELECT DISTINCT f.dest
    FROM flights f
    WHERE DATE_PART('week', f.time_hour) = 'Thursday'
), S AS (
    SELECT DISTINCT f.dest
    FROM flights f
)

SELECT * FROM S

EXCEPT

SELECT * FROM R;
```

Requête 7

Quelles sont les villes desservies par une seule compagnie le dimanche?

Solution

```
SELECT f.dest
FROM flights f
WHERE DATE_PART('week', f.time_hour) = 'Sunday'
GROUP BY f.dest
HAVING COUNT(DISTINCT f.carrier) = 1;
```

- L'utilisation d'une clause WITH (Common Table Expression) plutôt que d'une requête imbriquée rend le code plus lisible.
- Dans la réponse, nous donnons plus que ce qui était demandé. On aurait pu se contenter de ;

. . .

Requête 8

Quelles sont les compagnies pour les quelles le retard médian au décollage est supérieur à $15\,$ minutes ?

Solution

```
SELECT f.carrier
FROM flights f
WHERE f.dep_delay IS NOT NULL
GROUP BY f.carrier
HAVING MEDIAN(f.dep_delay) > 15;
```

Requête 9

Quelles sont les destinations qui sont desservies quotidiennement par une compagnie ?

Solution

Requête 10

Quelles sont les compagnies qui exploitent des avions de tous les constructeurs?

Solution

Quelques conseils

- Préférez les clauses WITH et les jointures aux requêtes imbriquées sauf si la requête imbriquée est très simple. C'est une question de lisibilité et donc souvent de correction.
- Ne mélangez pas les fonctions fenêtres et les clauses ${\tt GROUP \ BY \ \dots}$

```
SELECT ..., FOO() OVER W
FROM R
WINDOW w AS (PARTITION BY ... ORDER BY ...)
GROUP BY ...;
```

est tout simplement incorrect.

- Lorsque vous effectuez un partitionnement par GROUP BY \dots , la clause SELECT \dots est sévèrement contrainte, vous n'y trouverez que
 - -les colonnes qui ont servi dans la clause ${\tt GROUP}\;$ BY $\;\ldots,$ normalement elles devraient toutes y figurer
 - des fonctions d'aggrégation, comme COUNT(...), SUM(...), VAR(...)