КПІ 4

**Створення стовпчикової БД.**

База даних була створена засобами PostgreSQL та його розширенням Citus.

Скрипт створення БД:

CREATE DATABASE columnstore\_bts;

\c columnstore\_bts;

CREATE EXTENSION IF NOT EXISTS citus;

CREATE TABLE airlines (

iata\_code varchar(2) DEFAULT NULL,

airline varchar(30) DEFAULT NULL

) USING columnar;

CREATE TABLE airports (

iata\_code varchar(3) DEFAULT NULL,

airport varchar(80) DEFAULT NULL,

city varchar(30) DEFAULT NULL,

state varchar(2) DEFAULT NULL,

country varchar(30) DEFAULT NULL,

latitude decimal(11,4) DEFAULT NULL,

longitude decimal(11,4) DEFAULT NULL

) USING columnar;

CREATE TABLE flights (

year smallint DEFAULT NULL,

month smallint DEFAULT NULL,

day smallint DEFAULT NULL,

day\_of\_week smallint DEFAULT NULL,

fl\_date date DEFAULT NULL,

carrier varchar(2) DEFAULT NULL,

tail\_num varchar(6) DEFAULT NULL,

fl\_num smallint DEFAULT NULL,

origin varchar(5) DEFAULT NULL,

dest varchar(5) DEFAULT NULL,

crs\_dep\_time varchar(4) DEFAULT NULL,

dep\_time varchar(4) DEFAULT NULL,

dep\_delay decimal(13,2) DEFAULT NULL,

taxi\_out decimal(13,2) DEFAULT NULL,

wheels\_off varchar(4) DEFAULT NULL,

wheels\_on varchar(4) DEFAULT NULL,

taxi\_in decimal(13,2) DEFAULT NULL,

crs\_arr\_time varchar(4) DEFAULT NULL,

arr\_time varchar(4) DEFAULT NULL,

arr\_delay decimal(13,2) DEFAULT NULL,

cancelled decimal(13,2) DEFAULT NULL,

cancellation\_code varchar(20) DEFAULT NULL,

diverted decimal(13,2) DEFAULT NULL,

crs\_elapsed\_time decimal(13,2) DEFAULT NULL,

actual\_elapsed\_time decimal(13,2) DEFAULT NULL,

air\_time decimal(13,2) DEFAULT NULL,

distance decimal(13,2) DEFAULT NULL,

carrier\_delay decimal(13,2) DEFAULT NULL,

weather\_delay decimal(13,2) DEFAULT NULL,

nas\_delay decimal(13,2) DEFAULT NULL,

security\_delay decimal(13,2) DEFAULT NULL,

late\_aircraft\_delay decimal(13,2) DEFAULT NULL

) USING columnar;

**Створення стандартної БД.**

База даних була створена засобами PostgreSQL

Скрипт створення БД:

CREATE DATABASE rowstore\_flights;

\c rowstore\_flights;

CREATE TABLE airlines

(

iata\_code varchar(2) CONSTRAINT PK\_airlines PRIMARY KEY,

airline varchar(30)

);

CREATE TABLE airports

(

iata\_code varchar(3) CONSTRAINT PK\_airports PRIMARY KEY,

airport varchar(80),

city varchar(30),

state varchar(2),

country varchar(30),

latitude decimal(11, 4),

longitude decimal(11, 4)

);

CREATE TABLE flights

(

year smallint,

month smallint,

day smallint,

day\_of\_week smallint,

fl\_date date,

carrier varchar(2),

tail\_num varchar(6),

fl\_num smallint,

origin varchar(5),

dest varchar(5),

crs\_dep\_time varchar(4),

dep\_time varchar(4),

dep\_delay decimal(13, 2),

taxi\_out decimal(13, 2),

wheels\_off varchar(4),

wheels\_on varchar(4),

taxi\_in decimal(13, 2),

crs\_arr\_time varchar(4),

arr\_time varchar(4),

arr\_delay decimal(13, 2),

cancelled decimal(13, 2),

cancellation\_code varchar(20),

diverted decimal(13, 2),

crs\_elapsed\_time decimal(13, 2),

actual\_elapsed\_time decimal(13, 2),

air\_time decimal(13, 2),

distance decimal(13, 2),

carrier\_delay decimal(13, 2),

weather\_delay decimal(13, 2),

nas\_delay decimal(13, 2),

security\_delay decimal(13, 2),

late\_aircraft\_delay decimal(13, 2)

);

CREATE INDEX idx\_flights\_year ON flights (year);

CREATE INDEX idx\_flights\_carrier ON flights (carrier);

CREATE INDEX idx\_flights\_carrier\_delay ON flights (carrier\_delay);

CREATE INDEX idx\_flights\_weather\_delay ON flights (weather\_delay);

CREATE INDEX idx\_flights\_nas\_delay ON flights (nas\_delay);

CREATE INDEX idx\_flights\_security\_delay ON flights (security\_delay);

CREATE INDEX idx\_flights\_late\_aircraft\_delay ON flights (late\_aircraft\_delay);

CREATE INDEX idx\_flights\_arr\_delay ON flights (arr\_delay);

CREATE INDEX idx\_flights\_month ON flights (month);

CREATE INDEX idx\_flights\_dest ON flights (dest);

**Скрипт заповнення обох БД:**

COPY airlines

FROM '/airlines.csv'

WITH (FORMAT CSV, HEADER true, DELIMITER ',');

COPY airports

FROM '/airports.csv'

WITH (FORMAT CSV, HEADER true, DELIMITER ',');

COPY flights

FROM '/flights\_3.csv'

WITH (FORMAT CSV, HEADER true, DELIMITER ',');

**Порівняння баз даних.**

Для виконання поставленої задачі було створено та відтворено наступні запити:

Розрахувати сумарну затримку по містах

SELECT C.city, A.dep\_delay, B.arr\_delay, A.dep\_delay + B.arr\_delay as total\_delay FROM

(SELECT flights.origin as city, *sum*(flights.dep\_delay) as dep\_delay FROM flights GROUP BY city) AS A

FULL OUTER JOIN

(SELECT flights.dest as city, *sum*(flights.arr\_delay) as arr\_delay FROM flights GROUP BY city) AS B

ON A.city = B.city INNER JOIN airports AS C ON A.city = C.iata\_code OR B.city = C.iata\_code;

Швидкість виконання запиту:

-- columnar: execution: 15 s 808 ms,

-- row based: execution: 28 s 806 ms

Порахувати кількість польотів по містах

SELECT C.city, origin\_count, dest\_count, origin\_count + dest\_count FROM

(SELECT *count*(\*) as origin\_count, origin as city FROM flights GROUP BY flights.origin) AS A

FULL OUTER JOIN

(SELECT *count*(\*) as dest\_count, dest as city FROM flights GROUP BY flights.dest) AS B

ON A.city = B.city INNER JOIN airports AS C ON A.city = C.iata\_code OR B.city = C.iata\_code;

Швидкість виконання запиту:

-- columnar: execution: 12 s 608 ms

-- row based: execution: 13 s 606 ms

Знайти місто з найменшою і найбільшою затримкою

SELECT *MIN*(delay) as min\_delay, B.city FROM

(SELECT *MIN*(arr\_delay) as delay, dest as city FROM flights GROUP BY dest

UNION ALL

SELECT *MIN*(dep\_delay) as delay, origin as city FROM flights GROUP BY origin) as A

INNER JOIN airports as B

ON A.city = B.iata\_code

GROUP BY B.city ORDER BY min\_delay LIMIT 1;

Швидкість виконання запиту:

-- columnar: execution: 16 s 596 ms

-- row based: execution: 23 s 334 ms

SELECT *MAX*(delay) as max\_delay, B.city FROM

(SELECT *MAX*(arr\_delay) as delay, dest as city FROM flights GROUP BY dest

UNION ALL

SELECT *MAX*(dep\_delay) as delay, origin as city FROM flights GROUP BY origin) as A

INNER JOIN airports as B

ON A.city = B.iata\_code

GROUP BY B.city ORDER BY max\_delay DESC LIMIT 1;

Швидкість виконання запиту:

-- columnar: execution: 15 s 772 ms

-- row based: execution: 21 s 168 ms

Знайти всі польоти з затримкою більше за середній час затримки

SELECT \* FROM flights

WHERE arr\_delay > (SELECT *AVG*(delay) FROM

(SELECT *AVG*(dep\_delay) as delay FROM flights

UNION ALL

SELECT *AVG*(arr\_delay) as delay FROM flights) AS A)

OR dep\_delay > (SELECT *AVG*(delay) FROM

(SELECT *AVG*(dep\_delay) as delay FROM flights

UNION ALL

SELECT *AVG*(arr\_delay) as delay FROM flights) AS A);

Швидкість виконання запиту:

-- columnar: execution: 13 s 573 ms

-- row based: execution: 24 s 345 ms

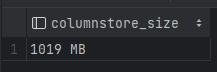
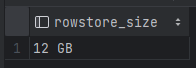
Порівняння об’єму баз даних.

Запит, щоб дізнатись та порівняти об’єми БД.

SELECT *pg\_size\_pretty*(*pg\_database\_size*('columnstore\_bts')) AS columnstore\_size;

SELECT *pg\_size\_pretty*(*pg\_database\_size*('rowstore\_flights')) as rowstore\_size;

Результати:



**Висновок.**

Всі sql запити були відчутно швидше відтворені у стовпчиковій БД. У випадку з наданими даними різниця становила декілька секунд, проте в реальних БД з терабайтами даних ця різниця буде критичною. Під час порівняння об’ємів побачив їхню різницю в понад 10 разів, що показує наскільки краще використовувати стовпчикові БД.