Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи №1
з дисципліни «Програмування інтелектуальних інформаційних систем»

Виконав студент ІП-13, Дейнега Владислав Миколайович

(шифр, прізвище, ім'я, по батькові)

Перевірив Баришич Лука Маріянович

(прізвище, ім'я, по батькові)

Київ 2023

Лабораторна робота 1

Хід роботи:

1. Скрипти для створення бази даних: https://github.com/mariadb-corporation/mariadb-columnstore-sample-data

2. Розрахувати сумарну затримку по містах

```
select airports.city, SUM(groupe_tab.sum_delay) from airports
INNER JOIN (
SELECT union_tab.airport, sum(union_tab.delay) as sum_delay from (
    SELECT flights.origin as airport, flights.dep_delay as delay from `flights`
    UNION ALL
    SELECT flights.dest as airport, flights.arr_delay as delay from `flights`
  ) as union_tab
GROUP BY union_tab.airport
) as groupe_tab
on groupe_tab.airport = airports.iata_code
GROUP BY airports.city
ORDER BY airports.city;
```

city	SUM(groupe_tab.sum_delay
Aberdeen	1775,00
Abilene	8458,00
Adak	-340,00
Agana	2044,00
Aguadilla	6261,00
Akron	15166,00
Albany	26060,00
Albuquerque	48599,00
Alexandria	7221,00
Allentown	7167,00
Alpena	3617,00
Amarillo	18638,00
Anchorage	-2393,00
Appleton	10458,00
Arcata/Eureka	7654,00
Arlington	205589,00
Asheville	8908,00
Aspen	43927,00

3. Порахувати кількість польотів по містах

```
SELECT airports.city, SUM(group_tab.count_group) as count_city FROM airports
INNER JOIN
(SELECT union_tab.airport, SUM(count) as count_group FROM
  (SELECT flights.origin as airport, COUNT(flights.origin) as count from flights GROUP BY airport
  UNION ALL
  SELECT flights.dest as airport, COUNT(flights.dest) as count from flights GROUP BY airport
  ) as union_tab
GROUP BY union_tab.airport) AS group_tab
ON airports.iata_code = group_tab.airport
GROUP BY airports.city
ORDER BY airports.city
```

city	count_city
Aberdeen	270
Abilene	1056
Adak	42
Agana	130
Aguadilla	446
Akron	2540
Albany	3012
Albuquerque	6768
Alexandria	1230
Allentown	777
Alpena	202
Amarillo	2122
Anchorage	5179
Appleton	1073
Arcata/Eureka	581
Arlington	30022
Asheville	923
Aspen	2757
Atlanta	133911
Atlantic City	1658
Augusta	816
Austin	15015
Bakersfield	876

4. Знайти місто з найменшою і найбільшою затримкою

```
select airports.city, SUM(groupe_tab.sum_delay) as summ from airports
INNER JOIN (
SELECT union_tab.airport, sum(union_tab.delay) as sum_delay from (
    SELECT flights.origin as airport, flights.dep_delay as delay from `flights`
    UNION ALL
    SELECT flights.dest as airport, flights.arr_delay as delay from `flights`
  ) as union_tab
GROUP BY union_tab.airport
) as groupe_tab
on groupe_tab.airport = airports.iata_code
GROUP BY airports.city
ORDER BY summ DESC
LIMIT 1;
```

city	summ
Chicago	1990719,00

```
select airports.city, SUM(groupe_tab.sum_delay) as summ from airports
INNER JOIN (
SELECT union_tab.airport, sum(union_tab.delay) as sum_delay from (
    SELECT flights.origin as airport, flights.dep_delay as delay from `flights`
    UNION ALL
    SELECT flights.dest as airport, flights.arr_delay as delay from `flights`
  ) as union_tab
GROUP BY union_tab.airport
) as groupe_tab
on groupe_tab.airport = airports.iata_code
GROUP BY airports.city
ORDER BY summ
LIMIT 1;
```

city	summ
Moab	-2485,00

5. Знайти всі польоти з затримкою більше за середній час затримки

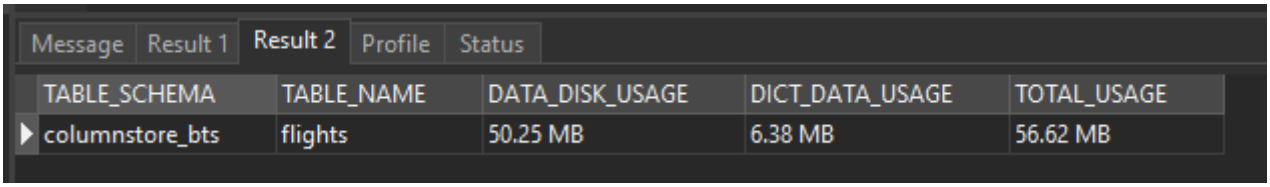
```
SELECT * FROM flights
WHERE (dep_delay + arr_delay) > (
  SELECT AVG(dep_delay +arr_delay) FROM flights);
```

year	month	day	day_of_week	fl_date	carrier	tail_num	fl_num	origin	dest	crs_dep_time	dep_time	dep_delay	taxi_out	wheels_off	wheels_on	taxi_in	crs_arr_time	arr_ti
2015	1	15	4	2015-01-15	AA	N4YTAA	362	DFW	MKE	2000	2032	32,00	13,00	2045	2236	5,00	2217	2241
2015	1	16	5	2015-01-16	AA	N502AA	362	DFW	MKE	2000	2044	44,00	12,00	2056	2252	8,00	2217	2300
2015	1	22	4	2015-01-22	AA	N467AA	362	DFW	MKE	2000	2023	23,00	17,00	2040	2224	8,00	2217	2232
2015	1	25	7	2015-01-25	AA	N4XGAA	362	DFW	MKE	2000	2029	29,00	9,00	2038	2245	4,00	2217	2249
2015	1	3	6	2015-01-03	AA	N3CAAA	363	LGA	ORD	1950	2025	35,00	20,00	2045	2146	9,00	2130	2155
2015	1	4	7	2015-01-04	AA	N3DTAA	363	LGA	ORD	1950	2114	84,00	14,00	2128	2240	9,00	2130	2249
2015	1	5	1	2015-01-05	AA	N3AWAA	363	LGA	ORD	1950	2039	49,00	25,00	2104	2217	135,00	2130	32
2015	1	7	3	2015-01-07	AA	N3CTAA	363	LGA	ORD	1955	2219	144,00	16,00	2235	2328	7,00	2134	2335
2015	1	16	5	2015-01-16	AA	N3LKAA	363	LGA	ORD	1955	2000	5,00	49,00	2049	2151	5,00	2134	2156
2015	1	19	1	2015-01-19	AA	N3JKAA	363	LGA	ORD	1955	2024	29,00	45,00	2109	2208	8,00	2134	2216
2015	1	20	2	2015-01-20	AA	N3LUAA	363	LGA	ORD	1955	2015	20,00	19,00	2034	2128	7,00	2134	2135
2015	1	2	5	2015-01-02	AA	N3HHAA	364	ORD	LGA	1500	1518	18,00	9,00	1527	1754	9,00	1800	1803
2015	1	4	7	2015-01-04	AA	N3JFAA	364	ORD	LGA	1500	1518	18,00	15,00	1533	1807	5,00	1800	1812
2015	1	6	2	2015-01-06	AA	N3CTAA	364	ORD	LGA	1500	1521	21,00	22,00	1543	1812	12,00	1802	1824
2015	1	8	4	2015-01-08	AA	N3LJAA	364	ORD	LGA	1500	1821	201,00	23,00	1844	2117	4,00	1802	2121
2015	1	10	6	2015-01-10	AA	N3GWAA	364	ORD	LGA	1500	1546	46,00	13,00	1559	1832	5,00	1802	1837
2015	1	13	2	2015-01-13	AA	N3HLAA	364	ORD	LGA	1500	1534	34,00	12,00	1546	1823	15,00	1802	1838
2015	1	15	4	2015-01-15	AA	N3FUAA	364	ORD	LGA	1500	1513	13,00	18,00	1531	1808	6,00	1802	1814
2015	1	16	5	2015-01-16	AA	N3DHAA	364	ORD	LGA	1500	1603	63,00	19,00	1622	1853	11,00	1802	1904
2015	1	29	4	2015-01-29	AA	N3CXAA	364	ORD	LGA	1500	1512	12,00	17,00	1529	1811	3,00	1802	1814
2015	1	1	4	2015-01-01	AA	N3JPAA	365	DCA	DFW	1005	1029	24,00	11,00	1040	1253	19,00	1235	1312
2015	1	2	5	2015-01-02	AA	N3GVAA	365	DCA	DFW	1005	1021	16,00	12,00	1033	1238	11,00	1235	1249
2015	1	4	7	2015-01-04	AA	N3HEAA	365	DCA	DFW	1005	1012	7,00	13,00	1025	1246	14,00	1235	1300

6. Заміряти вбудованими методами об”єм БД та швидкість виконання запитів. Порівняти звичайну і стовпчикову

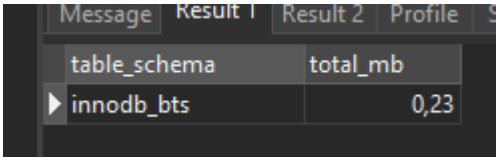
```
SELECT
  table_schema,
  ROUND(SUM(data_length + index_length) / POWER(2,20), 2) as total_mb
FROM
  information_schema.`TABLES`
WHERE
  table_schema = 'innodb_bts';

call columnstore_info.table_usage('columnstore_bts', 'flights')
```



Message | Result 1 | Result 2 | Profile | Status

TABLE_SCHEMA	TABLE_NAME	DATA_DISK_USAGE	DICT_DATA_USAGE	TOTAL_USAGE
columnstore_bts	flights	50.25 MB	6.38 MB	56.62 MB



Message | Result 1 | Result 2 | Profile

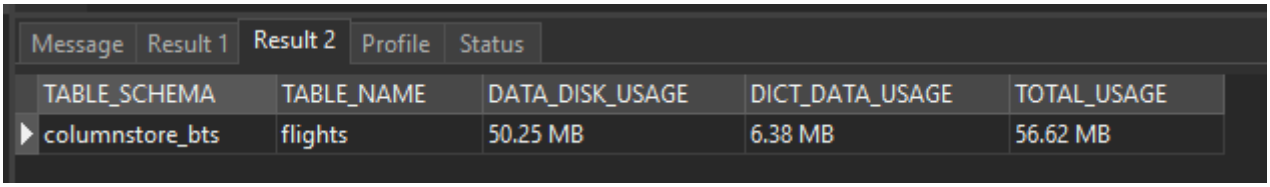
table_schema	total_mb
innodb_bts	0,23

Результат:

Час виконання скриптів

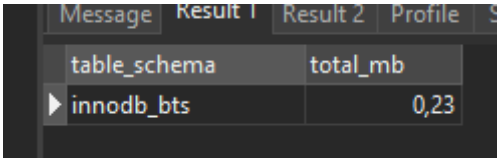
№	innodb	Columstore
1	10.14	0.35
2	3.89	0.22
3	10.2	0.35
4	10.24	0.4
5	8.55	4.07

Об’єм баз даних



Message | Result 1 | Result 2 | Profile | Status

TABLE_SCHEMA	TABLE_NAME	DATA_DISK_USAGE	DICT_DATA_USAGE	TOTAL_USAGE
columnstore_bts	flights	50.25 MB	6.38 MB	56.62 MB



Message | Result 1 | Result 2 | Profile

table_schema	total_mb
innodb_bts	0,23

Висновок:

Після виконання всіх завдань і досліджень, можемо зробити такі висновки:

Створення баз даних: Ми успішно створили стовпчикову та звичайну (реляційну) бази даних за допомогою наданих даних. Розгляньте приклад застосунку flights-app на GitHub, який може служити відмінним шаблоном для подібних проектів.

Розрахунок сумарної затримки по містах: Ми використали SQL-запит для підрахунку суми затримок для кожного міста, що дозволило нам зрозуміти, які міста мають найбільше загальної затримки.

Порахування кількості польотів по містах: Ми використали SQL-запит для підрахунку кількості польотів для кожного міста, що дозволило нам з'ясувати, які міста є найбільш активними з точки зору польотів.

Пошук міста з найменшою і найбільшою затримкою: Ми використали SQL-запит для знаходження міста з найменшою і найбільшою затримкою, що дозволило нам ідентифікувати крайні значення затримок в містах.

Знаходження всіх польотів з затримкою більше за середній час затримки: Ми використали SQL-запит для виділення всіх польотів, які мали затримку більше за середнє значення затримки, що дозволило нам виявити винятки в затримках.

Вимірювання обсягу баз даних та швидкості виконання запитів: Ми провели вимірювання обсягу стовпчикової та реляційної баз даних, а також порівняли швидкість виконання запитів. За результатами дослідження ми можемо зробити висновок щодо того, яка база даних краще підходить для конкретних завдань і потреб нашого проекту.

В цілому, наше дослідження дозволило нам ефективно виконати аналіз даних, знайти важливу інформацію та визначити оптимальну базу даних для нашого проекту. Важливо продовжувати вивчати і вдосконалювати навички роботи з базами даних, оскільки це важливий аспект сучасної розробки програмного забезпечення.