DataBase Implementation Project: Phase 3

***Abstract*—This document is a report for Database Implementation Project, regarding the optimization of SQL queries.**

***Keywords—experiment, analysis***

* Introduction

The data set is a record of all parking tickets that have been issued in Chicago from January 1996 to May 2018. It contains different types of violations, and the locations where violations occurred. The data set is large enough to act as a population sample for different statistical computations, such as analysing trends.

 The queries are:

* Finding how the number of parking tickets produced changes over each year.
* Finding which make of vehicles has been receiving the most tickets over the years.
* Finding how the number of parking tickets given in different community areas changes over the years.
* How many tickets were given each year for prohibited parking due to blocking access, alleys, driveways, or fire lanes?
* In which years, after 1999, was the most amount of money spent on paying parking tickets?
* What is the average amount of level 1 fines issued per month?
* How many tickets were issued for parking violations in each community area?
* What is the most common type of violation and in which areas is it most frequently issued?
* Which vehicle received the least number of tickets?
* What is the average number of tickets per parking enforcement officer?
* How many tickets were issued between 1998-2000 and what was the total amount of level 2 fines collected?
* What month has the highest number of tickets given?
* What officer has given the most tickets?

From phase 2, query 3 was changed. It is now ordered by “community\_area\_name”, then by year.

* Data
* The database management system is SQLite DB browser.
* The machine has a RAM capacity of 16.00 GB.
* The database consists of a single table.
* It contains 36 columns and 54430546 rows of data.
* The database has a total size of 19.7 GB on disk.
* The database was initially a “.csv” file and was converted to “.db” using the DBMS.
* Experiments

1. Finding how the number of parking tickets produced changes over each year.

“SELECT year, count(year) AS num\_tickets

FROM parking\_tickets

GROUP BY year

ORDER BY year ASC;“

Attempted to create an index on “year”.

Attempted to rewrite query to be more efficient.

1. Finding which make of vehicles has been receiving the most tickets over the years.

“SELECT vehicle\_make, count(vehicle\_make) AS num\_tickets

FROM parking\_tickets

GROUP BY vehicle\_make

ORDER BY vehicle\_make ASC;”

Attempted to create an index on “vehicle\_make”.

Attempted to rewrite query to be more efficient.

1. Finding how the number of parking tickets given in different community areas changes over the years.

“SELECT community\_area\_name, year, count(community\_area\_name) AS num\_tickets

FROM parking\_tickets

GROUP BY community\_area\_name, year

ORDER BY community\_area\_name, year ASC;”

Attempted to create an index on “community\_area\_name AND year”.

Attempted to rewrite query to be more efficient.

1. How many tickets were given each year for prohibited parking due to blocking access, alleys, driveways, or fire lanes.

“SELECT year, count(violation\_code) AS num\_tickets

FROM parking\_tickets

WHERE violation\_code = ‘0964100C’

GROUP  BY year

ORDER BY year ASC;”

Attempted to create an index on “violation\_code AND year”.

Attempted to rewrite query to be more efficient.

1. In which years, after 1999, was the most amount of money spent on paying parking tickets?

“SELECT year, sum(total\_payments) AS yearly\_sum

FROM parking\_tickets

WHERE year > 1999

GROUP BY year

ORDER BY yearly\_sum DESC;”

Attempted to create an index on “year AND total\_payments”.

Attempted to rewrite query to be more efficient.

1. What is the average amount of level 1 fines issued per month?

“SELECT month, AVG(fine\_level1\_amount) AS avg\_level\_1\_fine

FROM parking\_tickets

GROUP BY month;”

Attempted to create an index on “month AND fine\_level1\_amount”.

Attempted to rewrite query to be more efficient.

1. How many tickets were issued for parking violations in each community area?

“SELECT community\_area\_name, COUNT(\*) AS num\_tickets

FROM parking\_tickets

GROUP BY community\_area\_name;”

Attempted to create an index on “community\_area\_name”.

Attempted to create an index on “year” and “num\_tickets”.

Attempted to rewrite query to be more efficient.

1. What is the most common type of violation and in which areas is it most frequently issued?

“SELECT violation\_description, community\_area\_name, COUNT(\*) AS num\_tickets

FROM parking\_tickets

GROUP BY violation\_description, community\_area\_name

ORDER BY num\_tickets DESC

LIMIT 1;”

Attempted to create an index on “violation\_description AND community\_area\_name”.

Attempted to create an index on year “num\_tickets”.

Attempted to rewrite query to be more efficient.

1. Which vehicle received the least number of tickets?

“SELECT vehicle\_make, COUNT(\*) AS num\_tickets

FROM parking\_tickets

GROUP BY vehicle\_make

ORDER BY num\_tickets ASC

LIMIT 1;”

Attempted to create an index on “vehicle\_make”

Attempted to create an index on year “num\_tickets”.

Attempted to rewrite query to be more efficient

1. What is the average number of tickets per parking enforcement officer?

“SELECT AVG(num\_tickets) AS avg\_num\_tickets\_per\_officer

FROM (SELECT officer, COUNT(\*) AS num\_tickets

      FROM parking\_tickets

      GROUP BY officer) AS t;”

Attempted to create an index on “officer”.

Attempted to create an index on year “num\_tickets”.

Attempted to rewrite query to be more efficient.

1. How many tickets were issued between 1998-2000 and what was the total amount of level 2 fines collected?

“SELECT COUNT(\*) AS num\_tickets, SUM(fine\_level2\_amount) AS total\_fines

FROM parking\_tickets

WHERE year BETWEEN '1998' AND '2000';”

Attempted to create and index on “fine\_level2\_amount AND year”.

Attempted to create an index on “year AND fine\_level2\_amount”.

Attempted to create an index on year “num\_tickets”.

Attempted to rewrite query to be more efficient.

1. What month has the highest number of tickets given?

“SELECT month, COUNT(\*) AS num\_tickets

FROM parking\_tickets

GROUP BY month

ORDER BY num\_tickets DESC

LIMIT 1;”

Attempted to create an index on “month”.

Attempted to create an index on year “num\_tickets”.

Attempted to rewrite query to be more efficient.

1. What officer has given the most tickets?

“SELECT officer, COUNT(\*) AS num\_tickets

FROM parking\_tickets

GROUP BY officer

ORDER BY num\_tickets DESC

LIMIT 1;”

Attempted to create an index on “officer”.

Attempted to create an index on year “num\_tickets”.

Attempted to rewrite query to be more efficient.

* Analysis

From the above experiments, a general trend was that rewriting the queries in a more efficient way was unsuccessful because all our queries were already as efficient as they could be.

For each query, the respective experiment resulted in the following:

1. After adding an index on “year”, the runtime was improved from 237601 ms to 11632 ms.

1. Adding an index on “vehicle\_make”, caused the runtime to improve from 138878 ms to 13107 ms.

1. Adding an index on “community\_area\_name AND year”, decreased the runtime from 587078 ms to 16460 ms.

1. After adding an index on “violation\_code AND year”, the runtime was improved from 94538 ms to 33 ms.

1. Adding an index on “year AND total\_payments”, the runtime was improved from 185007 ms to 13966 ms.

1. After adding an index on “month AND fine\_level1\_amount”, the runtime was improved from 149669 ms to 8077 ms.

1. Creating an index on the num\_tickets attribute would also be expensive, as it would require us to create a new table with a num\_tickets column. However, after adding an index on “community\_area\_name”, the runtime was improved from 386510 ms to 7237 ms.

1. Creating an index on the num\_tickets attribute would also be expensive, as it would require us to create a new table with a num\_tickets column. Adding an index on “violation\_description AND community\_area\_name”, caused the runtime decrease from 1005306 ms to 29153 ms.

1. Creating an index on the num\_tickets attribute would also be expensive, as it would require us to create a new table with a num\_tickets column.After adding an index on “vehicle\_make”, the runtime was improved from 432868 ms to 17881 ms.

1. Creating an index on the num\_tickets attribute would also be expensive, as it would require us to create a new table with a num\_tickets column.After adding an index on “officer”, the runtime was improved from 526285 ms to 16148 ms.

1. Creating an index on the num\_tickets attribute would also be expensive, as it would require us to create a new table with a num\_tickets column. Adding an index on “fine\_level2\_amount AND year” causes the runtime to be reduced from 248967 ms to only 28963 ms. But adding an index on “year AND fine\_level2\_amount”, reduces the runtime to 3938 ms.

1. Creating an index on the num\_tickets attribute would also be expensive, as it would require us to create a new table with a num\_tickets column.After adding an index on “month”, the runtime was improved from 401162 ms to 15746 ms.
2. Creating an index on the num\_tickets attribute would also be expensive, as it would require us to create a new table with a num\_tickets column.After adding an index on “officer”, the runtime was improved from 543243 ms to 16025 ms

* Discussions

Breaking down that one big table into multiple smaller ones, each with attributes with different focus might have provided more statements with different types and a lot more options when it comes to the experiment and analysis parts.

For example, we could have one table “Violation” with the attributes “violation\_code”, and “violation\_description”.

References

1. <https://www.propublica.org/datastore/dataset/chicago-parking-ticket-data>
2. SQLite DB browser.
3. IEEE templates
4. **IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove template text from your paper may result in your paper not being published.**