Advent of Code 2020 Day 1 in SQL

William John Holden

2 December 2020

Advent of Code

Advent of Code is a Christmas-themed series of computer programming puzzles created by Eric Wastl. You can complete these puzzles in any programming language you wish, and on day 1 of 2020 I gave SQL a try.

SQL might not be an obvious language for puzzles more commonly solved with general-purpose languages like Java, JavaScript, C++, C#, Python, etc. The reason I wanted to use SQL for this particular problem is that the problem is so easily formulated in a *declarative* language.

Declarative programming is yet another programming paradigm, just like procedural (imperative) programming, functional programming, object-oriented programming. Unlike imperative programming, where you tell the machine what to *do*, in declarative programming you tell the machine what you *want*.

Part 1 of day 1 asks us to find, in a set of integers, the product xy from a pair x and y where $x + y = 2020$.

Let's look at that sentence again.

```
Find the product
from a set
where x + y = 2020
```

The sentence itself is is so close to real SQL that the solution is very clear! Let's dive in.

(A note to the reader: some of the code listings are given in extremely small fonts to fit wide output to on the page. Please use the zoom on your PDF reading software.)

Setting things up

OK, first let's set a few things up to do this.

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 32
Server version: 8.0.22-0ubuntu0.20.04.2 (Ubuntu)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE AOC;
Query OK, 1 row affected (0.03 sec)
```

```
mysql> USE AOC;
Database changed
mysql> CREATE TABLE Day1 (x INTEGER);
Query OK, 0 rows affected (0.07 sec)

mysql> SHOW COLUMNS IN Day1;
+-------+------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+-------+------+------+-----+---------+-------+
| x     | int  | YES  |     | NULL    |       |
+-------+------+------+-----+---------+-------+
1 row in set (0.01 sec)
```

Here we have created a database named `AOC` and a table named `Day1`. The table contains a single integer value named `x`.

Now we need the puzzle data. Your data will be different from mine but it should not matter. I used this PowerShell command to format my input as SQL queries.

```
PS> Get-Content -Path .\input.txt | ForEach-Object { "INSERT INTO Day1 VALUES ( $($_) );" }
INSERT INTO Day1 VALUES ( 1974 );
INSERT INTO Day1 VALUES ( 1902 );
INSERT INTO Day1 VALUES ( 1356 );
(Output truncated)
```

Paste those `INSERT` commands into MySQL and we are good to go.

```
mysql> SELECT COUNT(*) FROM Day1;
+----------+
| COUNT(*) |
+----------+
|      200 |
+----------+
1 row in set (0.00 sec)
```

Day 1 Part 1

Now that we have the values in the database we can solve this problem declaratively. To do so we will *self-join* the `Day1` table to itself. We won't specify a join condition, so this is actually just the Cartesian Product `Day1` $\times$ `Day1`.

```
mysql> SELECT COUNT(*) FROM Day1 AS X, Day1 AS Y;
+----------+
| COUNT(*) |
+----------+
|    40000 |
+----------+
1 row in set (0.01 sec)
```

The join contains $200^2 = 40000$ tuples. The `AS` keyword aliases `Day1` since we will need to distinguish values from the left and right relations.

```
mysql> SELECT X.x, Y.x, X.x + Y.x, X.x * Y.x
    -> FROM Day1 AS X, Day1 AS Y
    -> WHERE X.x + Y.x = 2020;
+------+------+-----------+-----------+
| x    | x    | X.x + Y.x | X.x * Y.x |
```

```
+------+------+-----------+-----------+
|  196 | 1824 |      2020 |    357504 |
| 1824 |  196 |      2020 |    357504 |
+------+------+-----------+-----------+
2 rows in set (0.00 sec)
```

And there's your answer for part 1.

Day 1 Part 2

Part 2 twists the problem by asking for the product xyz from a triple x, y, and z such that $x + y + z = 2020$. For this, we will need two self-joins.

```
mysql> SELECT COUNT(*) FROM Day1 AS X, Day1 AS Y, Day1 AS Z;
+----------+
| COUNT(*) |
+----------+
|  8000000 |
+----------+
1 row in set (0.00 sec)
```

Now, $200^3 = 8000000$ is a lot of possible combinations, but it is still small enough that we can compute it directly on a modern computer.

```
mysql> SELECT X.x, Y.x, Z.x, X.x + Y.x + Z.x, X.x * Y.x * Z.x
    -> FROM Day1 AS X, Day1 AS Y, Day1 AS Z
    -> WHERE X.x + Y.x + Z.x = 2020;
+------+------+------+-----------------+-----------------+
| x    | x    | x    | X.x + Y.x + Z.x | X.x * Y.x * Z.x |
+------+------+------+-----------------+-----------------+
|  694 |   14 | 1312 |            2020 |        12747392 |
| 1312 |   14 |  694 |            2020 |        12747392 |
|  694 | 1312 |   14 |            2020 |        12747392 |
|   14 | 1312 |  694 |            2020 |        12747392 |
| 1312 |  694 |   14 |            2020 |        12747392 |
|   14 |  694 | 1312 |            2020 |        12747392 |
+------+------+------+-----------------+-----------------+
6 rows in set (0.41 sec)
```

A faster query

The part 2 solution completed in 0.41 seconds. Can we do better? Yes, we can! If we give the `WHERE` clause a little bit more information we can reduce the size of the join buffers along the way.

```
mysql> SELECT X.x, Y.x, Z.x, X.x + Y.x + Z.x, X.x * Y.x * Z.x
    -> FROM Day1 AS X, Day1 AS Y, Day1 AS Z
    -> WHERE X.x + Y.x + Z.x = 2020
    -> AND X.x + Y.x < 2020;
+------+------+------+-----------------+-----------------+
| x    | x    | x    | X.x + Y.x + Z.x | X.x * Y.x * Z.x |
+------+------+------+-----------------+-----------------+
| 1312 |  694 |   14 |            2020 |        12747392 |
|  694 | 1312 |   14 |            2020 |        12747392 |
|   14 |  694 | 1312 |            2020 |        12747392 |
|  694 |   14 | 1312 |            2020 |        12747392 |
```

```
|   14 | 1312 |  694 |             2020 |         12747392 |
| 1312 |   14 |  694 |             2020 |         12747392 |
+------+------+------+------------------+------------------+
6 rows in set (0.02 sec)
```

The additional clause specifies that $x + y < 2020$. The machine does not assume that z is a nonnegative integer, but we know this from the problem statement and can specify this invariant. Now the query completes in only 0.02 seconds!

Join Plans

Let's take a closer look at how those joins will work.

```
mysql> EXPLAIN SELECT X.x, Y.x, Z.x, X.x + Y.x + Z.x, X.x * Y.x * Z.x FROM Day1 AS X, Day1 AS Y, Day1 AS Z WHERE X.x + Y.x + Z.x = 2020;
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+----------------------------------------------------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra                                              |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+----------------------------------------------------+
|  1 | SIMPLE      | X     | NULL       | ALL  | NULL          | NULL | NULL    | NULL |  200 |   100.00 | NULL                                               |
|  1 | SIMPLE      | Y     | NULL       | ALL  | NULL          | NULL | NULL    | NULL |  200 |   100.00 | Using join buffer (hash join)                      |
|  1 | SIMPLE      | Z     | NULL       | ALL  | NULL          | NULL | NULL    | NULL |  200 |   100.00 | Using where; Using join buffer (hash join)         |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+----------------------------------------------------+
3 rows in set, 1 warning (0.00 sec)

mysql> EXPLAIN SELECT X.x, Y.x, Z.x, X.x + Y.x + Z.x, X.x * Y.x * Z.x FROM Day1 AS X, Day1 AS Y, Day1 AS Z WHERE X.x + Y.x + Z.x = 2020 AND X.x + Y.x < 2020;
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+----------------------------------------------------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra                                              |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+----------------------------------------------------+
|  1 | SIMPLE      | X     | NULL       | ALL  | NULL          | NULL | NULL    | NULL |  200 |   100.00 | NULL                                               |
|  1 | SIMPLE      | Y     | NULL       | ALL  | NULL          | NULL | NULL    | NULL |  200 |   100.00 | Using where; Using join buffer (hash join)         |
|  1 | SIMPLE      | Z     | NULL       | ALL  | NULL          | NULL | NULL    | NULL |  200 |   100.00 | Using where; Using join buffer (hash join)         |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+----------------------------------------------------+
3 rows in set, 1 warning (0.00 sec)
```

Another idea to improve performance is to set `Day1.x` as an indexed column.

```
mysql> SHOW COLUMNS IN Day1;
+-------+------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+-------+------+------+-----+---------+-------+
| x     | int  | YES  |     | NULL    |       |
+-------+------+------+-----+---------+-------+
1 row in set (0.00 sec)

mysql> SHOW INDEXES FROM Day1;
Empty set (0.00 sec)

mysql> ALTER TABLE Day1 ADD PRIMARY KEY (x);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> SHOW COLUMNS IN Day1;
+-------+------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+-------+------+------+-----+---------+-------+
| x     | int  | NO   | PRI | NULL    |       |
+-------+------+------+-----+---------+-------+
1 row in set (0.00 sec)

mysql> SHOW INDEXES FROM Day1;
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Day1  |          0 | PRIMARY  |            1 | x           | A         |         200 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
1 row in set (0.01 sec)
```

Now the query plans can use indexed joins throughout.

```
mysql> EXPLAIN SELECT X.x, Y.x, Z.x, X.x + Y.x + Z.x, X.x * Y.x * Z.x
    -> FROM Day1 AS X, Day1 AS Y, Day1 AS Z
    -> WHERE X.x + Y.x + Z.x = 2020;
+----+-------------+-------+------------+-------+---------------+---------+---------+------+------+----------+-------------------------------------------------------+
| id | select_type | table | partitions | type  | possible_keys | key     | key_len | ref  | rows | filtered | Extra                                                 |
+----+-------------+-------+------------+-------+---------------+---------+---------+------+------+----------+-------------------------------------------------------+
|  1 | SIMPLE      | X     | NULL       | index | NULL          | PRIMARY | 4       | NULL |  200 |   100.00 | Using index                                           |
|  1 | SIMPLE      | Y     | NULL       | index | NULL          | PRIMARY | 4       | NULL |  200 |   100.00 | Using index; Using join buffer (hash join)            |
|  1 | SIMPLE      | Z     | NULL       | index | NULL          | PRIMARY | 4       | NULL |  200 |   100.00 | Using where; Using index; Using join buffer (hash join) |
+----+-------------+-------+------------+-------+---------------+---------+---------+------+------+----------+-------------------------------------------------------+
3 rows in set, 1 warning (0.00 sec)

mysql> EXPLAIN SELECT X.x, Y.x, Z.x, X.x + Y.x + Z.x, X.x * Y.x * Z.x
    -> FROM Day1 AS X, Day1 AS Y, Day1 AS Z
    -> WHERE X.x + Y.x + Z.x = 2020
    -> AND X.x + Y.x < 2020;
+----+-------------+-------+------------+-------+---------------+---------+---------+------+------+----------+-------------------------------------------------------+
| id | select_type | table | partitions | type  | possible_keys | key     | key_len | ref  | rows | filtered | Extra                                                 |
+----+-------------+-------+------------+-------+---------------+---------+---------+------+------+----------+-------------------------------------------------------+
|  1 | SIMPLE      | X     | NULL       | index | NULL          | PRIMARY | 4       | NULL |  200 |   100.00 | Using index                                           |
|  1 | SIMPLE      | Y     | NULL       | index | NULL          | PRIMARY | 4       | NULL |  200 |   100.00 | Using where; Using index; Using join buffer (hash join) |
|  1 | SIMPLE      | Z     | NULL       | index | NULL          | PRIMARY | 4       | NULL |  200 |   100.00 | Using where; Using index; Using join buffer (hash join) |
+----+-------------+-------+------------+-------+---------------+---------+---------+------+------+----------+-------------------------------------------------------+
3 rows in set, 1 warning (0.01 sec)
```

Performance is *slightly* improved.

```
mysql> SELECT X.x, Y.x, Z.x, X.x + Y.x + Z.x, X.x * Y.x * Z.x
    -> FROM Day1 AS X, Day1 AS Y, Day1 AS Z
    -> WHERE X.x + Y.x + Z.x = 2020;
+------+------+------+-----------------+-----------------+
| x    | x    | x    | X.x + Y.x + Z.x | X.x * Y.x * Z.x |
+------+------+------+-----------------+-----------------+
| 1312 |  694 |   14 |            2020 |        12747392 |
| 1312 |   14 |  694 |            2020 |        12747392 |
|   14 |  694 | 1312 |            2020 |        12747392 |
|  694 |   14 | 1312 |            2020 |        12747392 |
|  694 | 1312 |   14 |            2020 |        12747392 |
|   14 | 1312 |  694 |            2020 |        12747392 |
+------+------+------+-----------------+-----------------+
6 rows in set (0.37 sec)

mysql> SELECT X.x, Y.x, Z.x, X.x + Y.x + Z.x, X.x * Y.x * Z.x
    -> FROM Day1 AS X, Day1 AS Y, Day1 AS Z
    -> WHERE X.x + Y.x + Z.x = 2020
    -> AND X.x + Y.x < 2020;
+------+------+------+-----------------+-----------------+
| x    | x    | x    | X.x + Y.x + Z.x | X.x * Y.x * Z.x |
+------+------+------+-----------------+-----------------+
|  694 | 1312 |   14 |            2020 |        12747392 |
| 1312 |  694 |   14 |            2020 |        12747392 |
|   14 | 1312 |  694 |            2020 |        12747392 |
| 1312 |   14 |  694 |            2020 |        12747392 |
|   14 |  694 | 1312 |            2020 |        12747392 |
|  694 |   14 | 1312 |            2020 |        12747392 |
+------+------+------+-----------------+-----------------+
6 rows in set (0.01 sec)
```

The fastest query: `LIMIT 1`

The puzzle can be solved even faster! For this particular problem we do not care about *all* solutions x, y, and z. We only need *one* solution. So, if we use `LIMIT 1` we allow the machine to immediately stop computing once it finds a single satisfying triple.

This was not my idea, but rather came from a helpful comment from Reddit. One of the best reasons to participate in Advent of Code is its positive and helpful community.

```
mysql> SELECT X.x, Y.x, Z.x, X.x + Y.x + Z.x, X.x * Y.x * Z.x
    -> FROM Day1 AS X, Day1 AS Y, Day1 AS Z
    -> WHERE X.x + Y.x + Z.x = 2020
    -> LIMIT 1;
+------+-----+----+-----------------+-----------------+
| x    | x   | x  | X.x + Y.x + Z.x | X.x * Y.x * Z.x |
+------+-----+----+-----------------+-----------------+
| 1312 | 694 | 14 |            2020 |        12747392 |
+------+-----+----+-----------------+-----------------+
1 row in set (0.01 sec)

mysql> SELECT X.x, Y.x, Z.x, X.x + Y.x + Z.x, X.x * Y.x * Z.x
    -> FROM Day1 AS X, Day1 AS Y, Day1 AS Z
    -> WHERE X.x + Y.x + Z.x = 2020
    -> AND X.x + Y.x < 2020
    -> LIMIT 1;
+-----+------+----+-----------------+-----------------+
| x   | x    | x  | X.x + Y.x + Z.x | X.x * Y.x * Z.x |
+-----+------+----+-----------------+-----------------+
| 694 | 1312 | 14 |            2020 |        12747392 |
+-----+------+----+-----------------+-----------------+
1 row in set (0.00 sec)
```

Happy holidays!