Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Datenbanksysteme · Prof. Dr. Grust

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Datenbanksysteme I

WS 2019/20
Torsten Grust, Christian Duta

Assignment #3

Submission Deadline: November 12, 2017 - 10:00

Exercise 1: Declarativity and Query Execution Strategies (10 Points)

Database queries are written in a *declarative manner*: The user specifies *"what"* the desired result is, but not *"how"* to compute it. But before we delve into this declarative world of SQL and let the query processor of PostgreSQL do the job, we are going to explore the effects that different – but semantically equivalent – execution strategies can have on query runtime.

We will query the LEGO dataset (`contains.csv`, `bricks.csv`, `minifigs.csv` and `sets.csv`) which should be familiar from the lecture and are provided to you in the `legosets.zip` archive.

1. The following naive PyQL query returns all minifigs of category `'67'` that are part of sets `'671-1'` or `'377-1'`. However the query does not perform well:

```
from DB1 import Table
contains = Table('contains.csv')
bricks = Table('bricks.csv')
minifigs = Table('minifigs.csv')

for c in contains:
    for m in minifigs:
        if c['piece'] == m['piece']:
            if (c['set'] == '671-1' or c['set'] == '377-1'):
                if m['cat'] == '67':
                    print { 'set': c['set'], 'name': m['name'] }
```

   Implement an alternative version that is *as efficient as possible* by exploiting properties of the data and the query itself, such as:

   > Order of rows, early filtering of data, uniqueness or disjointness of values in columns, query strategy, etc.
   > (Keep in mind that not all of them may apply here).
   >
   > *Note: As in the lecture, you may use the loop control keywords* `break` *(and* `continue`*) or* `for...else`*.*

Exercise 2: Query Optimization through Data Pre-Processing (10 Points)

In the lecture, we discussed multiple variations of pre-processing the data to further improve the efficiency of the "weight of LEGO set 5610" query. Another such strategy was proposed by a fellow student in a previous course.

The idea is simple: A `piece` referenced in `contains.csv` may be found in either `bricks.csv` or `minifigs.csv`. To look up a piece's details, we have to search both files since we can't tell from a piece identifier whether it refers to a brick or a minifig. However, this knowledge could be explicitly represented in the data by extending `contains.csv` with an additional column `type`: For each entry in `contains.csv`, `type` indicates whether the piece is a brick (`'b'`) or minifig (`'m'`).

1. Construct a new file `contains_type.csv`[1], which is a copy of `contains.csv` but with an additional column `type` as described above. Approach this problem in two steps:

   (a) Write a PyQL query to construct the data for `contains_type.csv` from the three original files in the form of a list of dictionaries.

   Hint: A naive query would probably take far too long ($\sim$ several hours). Instead, try to optimize your PyQL query. Consider that `minifigs.csv` is a small table compared to `bricks.csv` ($\sim$ 17% of all pieces). We also know that if a piece is not a minifigure, it is a brick. Further, pieces are unique and `minifigs.csv` is sorted by its column `piece`.

   Note: Even optimized your query may take some minutes to process. So think carefully about possible improvements and be patient while running your query.

   (b) Use code from `DB1.py` to construct a `Table` object from the list and dump it to a file in the usual CSV format.

   Consider the following example:

   ```
   l = [{'a': 23, 'b': 'foo'}, {'a': 42, 'b': 'bar'}]
   Table(l).dump('foo.csv')
   ```

2. Implement a PyQL query leveraging the new `type` column. Compare the query execution times for the lecture's baseline query (slide 12 in slide set 3) and your new optimized variant.

3. Provide a formula that estimates the work performed by the optimized PyQL query. (For reference, see slide 13 and 15 in slide set 3 about *Declarativity*.)

[1] This file has also been provided to you in the `legosets.zip` archive