Externe Datastructuren

Taak 3 Algoritmen en Datastructuren 2

Abdullah Sabaa Allil - 0575278

abdullah.sabaa.allil@vub.be

2021-2022

Motivatie van keuzes

In mijn implementatie heb ik gekozen om een nieuwe tabel te maken, met een nieuw schema dat het type en size van het te verwijderen attribuut niet bevat, en dan alle tuples uit de oude tabel inserten naar de nieuwe tabel zonder het te verwijderen attribuut. De reden waarom er voor deze manier gekozen wordt is om ervoor te zorgen dat de blokken van de datafile zo optimaal mogelijk benut worden na het verwijderen van een attribuut.

De manier waarop de implementatie werkt is als volgt:

- Maak een nieuw schema aan met de attributen van de oude tabel behalve het te verwijderen attribuut.
- Maak een nieuwe tabel aan met dat nieuwe schema.
- Ga alle nodes van de oude tabel af en insert alle tupels naar de nieuwe tabel zonder dat attribuut. Delete telkens de oude nodes van zodra je alle tupels daarvan hebt beozcht.
- Eens je alle nodes hebt bezocht, “kopieer” de oude tabel naar de nieuwe tabel (door de header en het schema van de oude tabel naar de nieuwe tabel te kopieren) en delete het oude schema en de oude header.
- Verwijder alle indexes die op deze tabel werken, en voeg ze opnieuw toe, behalve de indexes op het verwijderde attribuut.

Een alternatieve implementatie had kunnen zijn: pas het schema aan en zet een sentinel value in plaats van het te verwijderen attribuut om aan te geven dat het verwijderd is, en in de rest van de implementatie zorg je ervoor dat je skipt telkens je die sentinel tegenkomt. Het voordeel hiervan is dat het computationeel sneller gaat zijn (omdat je niets moet afgaan en de index files niet hoeft aan te passen). Het nadeel is dat je veel garbage gaat hebben in de datafile (dat is ook de reden waarom ik voor mijn implementatie heb gekozen).

De grootste troef van mijn implementatie is dus het feit dat de blokken van de datafile optimaal gebruikt worden omdat het attribuut effectief uit de blokken van de datafile verwijderd wordt. Dit is ook duidelijk te zien aan het voorbeeld van de *drinks* database gegeven in de testfile:

- Initieel heeft de tabel 85 nodes.
- Na het droppen van het *country* attribuut heeft de tabel 29 nodes.
- Na het droppen van het *wine_servings* attribuut heeft de tabel 25 nodes.
- Na het droppen van het *total_litres_of_pure_alcohol* attribuut heeft de tabel 8 nodes.
- Na het droppen van het *spirit_servings* attribuut heeft de tabel 4 nodes.
- Na het droppen van het *beer_servings* attribuut wordt de tabel gedropt omdat er geen andere attributen zijn.

In de alternatieve implementatie zou de tabel altijd 85 nodes hebben!

Aanpassingen aan de a-d folder

Volgende bestanden worden toegevoegd:

- *test-alter.rkt:* dit bestand zal de implementatie testen op twee datasets:
    - een database met de tabel *olieprijs.* Dit is het gegeven voorbeeld in de opgave.
    - een database met de tabel *drinks*: Dit is een online database van *fivethirtyeight*[1]*.* De database werd door mij een beetje aangepast. De tabel bevat 170 tupels en 5 attributen, en zit in het bestand *drinks.csv*. in *test-alter.rkt* wordt de procedure *insert-data-from-file!* geimplementeerd die bestanden inleest en de inhoud daarvan naar een tabel insert.

Volgende bestanden worden aangepast:

- *a-d/d-b/table/fixed-size-slots/schema.rkt*: De procedure *copy-schema-w/o-attribute* wordt geimplementeerd. Deze procedure zorgt voor het kopieren van het schema op de disk zonder een bepaald attribuut, en daarna het nieuwe schema in central memory terug te geven.
  De procedure *print-schema* wordt ook geimplementeerd. Deze procedure zorgt voor het weergeven van de types en sizes van de attributen in het schema.
- *a-d/d-b/table/fixed-size-slots/node.rkt:* De procedure *record-without* wordt geimplementeerd. Deze procedure geeft de record van een gegeven slot terug zonder een bepaald attribuut dat meegeegven wordt als argument.
- *a-d/d-b/table/fixed-size-slots/table.rkt*: *delete-attribute!* wordt geimplementeerd. Deze procedure zorgt voor het verwijderen van een gegeven attribuut in een gegeven tabel.
  De procedure *drop!* wordt lichtjes aangepast om ervoor te zorgen dat de tabel in central memory ook weet dat hij gedropt wordt. Dit wordt daarna gebruikt in de procedure *print* die dan zal aangeven dat een tabel gedropt is.
- *a-d/d-b/database.rkt*: de procedure *alter-table-drop-column!* wordt geimplementeerd. Deze procedure zal een gegeven attribuut van een gegeven tabel droppen en de index files hermaken voor die tabel. Deze procedure zal ook zorgen dat de tabel gedropt wordt als er geen attributen meer zijn.

[1] https://github.com/fivethirtyeight/data/blob/master/alcohol-consumption/drinks.csv