6η Εργασία: Βελτιστοποίηση Ερωτημάτων και Εναύσματα

Προθεσμία: 22/05/2020

Σκοπός:

Η παρούσα εργασία αποτελείται από δύο μέρη. Στο πρώτο μέρος θα προσθέσουμε εναύσματα ώστε να εξασφαλίσουμε ότι συγκεκριμένες ιδιότητες της βάσης διατηρούνται, ενώ στο δεύτερο θα βελτιώσουμε το χρόνο εκτέλεσης ερωτημάτων με την προσθήκη ευρετηρίων.

Προαπαιτούμενα:

Θεωρούμε πως η βάση είναι στην κατάσταση που περιγράφεται στην 5η Εργασία και υπάρχουν τα ερωτήματα συνάθροισης που τρέξατε στην εργασία αυτή.

Ζητούμενα Εργασίας:

Μέρος Α: Εναύσματα

- Γράψτε μια συνάρτηση εναύσματος (trigger) η οποία όταν προστίθεται/διαγράφεται μία εγγραφή στον πίνακα Listing, ενημερώνει το πεδίο host_listings_count στην/στις αντίστοιχη/αντίστοιχες εγγραφή/εγγραφές του πίνακα Host. Ελέγξτε ότι η συνάρτηση σας λειτουργεί σωστά προσθέτοντας/διαγράφοντας μία εγγραφή στον πίνακα Listing και βλέποντας αν το πεδίο(θα μπορούσε να έχει πολλαπλές εγγραφές) host_listings_count του συγκεκριμένου host έχει ενημερωθεί σωστά.
- Σκεφτείτε και υλοποιήστε μία δική σας συνάρτηση εναύσματος για κάποιον/κάποιους από του πίνακες της βάσης.

Γράψτε τον κώδικα για τα εναύσματα σε ένα αρχείο με όνομα **triggers.sql.**
Για το δικό σας έναυσμα προσθέστε σε σχόλιο μια σύντομη περιγραφή για το τι κάνει.

EXPLAIN ANALYZE - Παράδειγμα

Καθώς η SQL είναι μια δηλωτική (declarative) γλώσσα, ένα ερώτημα περιγράφει ποιο θέλουμε να είναι το αποτέλεσμα του, παρά το ποια είναι ακριβώς τα βήματα με τα οποία θα φτάσει η βάση δεδομένων στο αποτέλεσμα αυτό. Συνεπώς, ένα σύστημα διαχείρισης βάσεων δεδομένων (ΣΔΒΣ) μπορεί να εκτελέσει ένα ερώτημα με n διαφορετικούς τρόπους και επιλέγοντας από ένα σύνολο αλγορίθμων για κάθε τελεστή. Η εντολή EXPLAIN ANALYZE μας δίνει τη δυνατότητα να δούμε το πλάνο εκτέλεσης που ακολουθήθηκε από το ΣΔΒΣ για την εκτέλεση ενός ερωτήματος.

Έστω ότι τρέχουμε την EXPLAIN ANALYZE στο παρακάτω ερώτημα.

```
EXPLAIN ANALYZE SELECT Listing.city, Count(Listing.id) AS listings
FROM Listing GROUP BY (Listing.city) ORDER BY listings;
```

Το πλάνο που προκύπτει είναι το εξής:

```
QUERY PLAN
------------------------------------------------------------------------------
------------------------------------------------------------------------------
"Sort          (cost=2992.89..2993.17     rows=109     width=15)     (actual
time=17.315..17.323 rows=109 loops=1)"
"  Sort Key: (count(id))"
"  Sort Method: quicksort  Memory: 32kB"
"   ->   HashAggregate   (cost=2988.11..2989.20 rows=109 width=15) (actual
time=17.226..17.246 rows=109 loops=1)"
"        Group Key: city"
"        ->  Seq Scan on listing  (cost=0.00..2930.41 rows=11541 width=11)
(actual time=0.009..5.396 rows=11541 loops=1)"
"Planning Time: 1.249 ms"
"Execution Time: 17.391 ms"
------------------------------------------------------------------------------
------------------------------------------------------------------------------
```

Το πλάνο απεικονίζεται ως μια δενδρική δομή με τους τελευταίους κόμβους να απεικονίζουν τα φύλλα του δέντρου, τα οποία επιστρέφουν εγγραφές από τους πίνακες. Ξεκινάμε λοιπόν την ανάγνωση του πλάνου από κάτω προς τα πάνω. Κάθε κόμβος, εκτός από τη ρίζα του δέντρου, ξεκινά με -> και αναπαριστά την εκτέλεση μιας ενέργειας. Για κάποιες ενέργειες μπορεί να υπάρχει επιπλέον πληροφορία όπως είναι το πεδίο πάνω στο οποίο έγινε η ταξινόμηση (π.χ. Sort Key: (count(id))). Τέτοιες πληροφορίες, με εξαίρεση τον κόμβο-ρίζα, δεν ξεκινούν με -> και έτσι τις ξεχωρίζουμε από τους κόμβους ενεργειών.

Ας δούμε τις ενέργειες που εκτελούνται στο παραπάνω πλάνο μία πρός μία.

- Seq Scan on "Listing": ένα table scan στον πίνακα Listing.
- HashAggregate: το HashAggregate είναι ένας από τους αλγορίθμους με τους οποίους η Postgres συναθροίζει πίνακες. Ο αλγόριθμος αυτό χρησιμοποιεί ένα hash table για να γκρουπάρει και να συναθροίσει τις εγγραφές. Ο κόμβος αυτός εκτελείται πάνω στις εγγραφές που προέκυψαν από το table scan του πίνακα Listing.
- Sort: ταξινόμηση του αποτελέσματος που προκύπτει από τον κόμβο HashAggregate.

Παρατηρούμε ότι στο πλάνο εκτέλεσης αναφέρονται και κάποιες μετρήσεις. Για παράδειγμα το cost προκύπτει από μία συνάρτηση κόστους που χρησιμοποιεί ο optimizer ερωτημάτων της Postgres για να εκτιμήσει το κόστος εκτέλεσης κάθε κόμβου και βασίζεται σε στατιστικά στοιχεία των πινάκων, όπως ο αριθμός των εγγραφών ή ο αριθμός των blocks στον δίσκο. Το actual time είναι ο πραγματικός χρόνος που χρειάστηκε ένας κόμβος του πλάνου για να εκτελεστεί, ενώ το rows αναφέρεται στις γραμμές που προσπελάστηκαν από έναν κόμβο. Τέλος, τα planning και execution time αναφέρονται στο χρόνο βελτιστοποίησης και εκτέλεσης του συνολικού πλάνου αντίστοιχα.

Μέρος Β: Ευρετήρια

- Έστω ότι έχουμε το παρακάτω ερώτημα:

  ```
  SELECT "Host".host_id, COUNT(*) FROM "Listing", "Host" WHERE
  "Host".host_id="Listing".host_id GROUP BY "Host".host_id;
  ```

  Χρησιμοποιήστε την εντολή EXPLAIN ANALYZE για να τυπώσετε το πλάνο και το χρόνο εκτέλεσης του ερωτήματος. Στη συνέχεια προσθέστε ένα κατάλληλο ευρετήριο που θα επιταχύνει τον χρόνο εκτέλεσης του παραπάνω ερωτήματος και ξανατρέξτε την εντολή EXPLAIN ANALYZE για να δείτε το νέο χρόνο εκτέλεσης.
- Έστω ότι έχουμε το παρακάτω ερώτημα:

  ```
  SELECT id, price FROM "Listing", "Price" WHERE guests_included > 5
  AND price > 40;
  ```

  Χρησιμοποιήστε την εντολή EXPLAIN ANALYZE για να τυπώσετε το πλάνο και το χρόνο εκτέλεσης του ερωτήματος. Στη συνέχεια πειραματιστείτε προσθέτοντας ευρετήρια σε ένα ή περισσότερα πεδία και ελέγξτε αν ο χρόνος εκτέλεσης βελτιώνεται. Η προσθήκη ευρετηρίου στο πεδίο price θα βοηθούσε; Αιτιολογήστε.
- Για κάθε ένα από τα ερωτήματα συνάθροισης που εκτελέσατε στην 5η εργασία, προσθέστε ένα κατάλληλο ευρετήριο που θα επιταχύνει την εκτέλεσή του. Τρέξτε την εντολή EXPLAIN ANALYZE όπως και στα προηγούμενα ερωτήματα για να διαπιστώσετε αν όντως μειώνεται ο χρόνος εκτέλεσης. Εξηγήστε γιατί επιλέξατε το συγκεκριμένο ευρετήριο. Αν για κάποια ερώτηση δε μπορείτε να βρείτε ένα ευρετήριο που να την επιταχύνει, γράψτε ποια ευρετήρια δοκιμάσατε να προσθέσετε και εξηγήστε γιατί θεωρείτε ότι δε μπορεί να επιταχυνθεί το συγκεκριμένο ερώτημα με την προσθήκη ευρετηρίων (η εξήγηση μπορεί να είναι σε υψηλό επίπεδο / διαισθητική, δεν είναι απαραίτητο να αναλύσετε τα ακριβή βήματα του πλάνου εκτέλεσης).

Βάλτε τα πλάνα εκτέλεσης που προκύπτουν για κάθε ερώτημα από την εντολή EXPLAIN ANALYZE πριν και μετά την προσθήκη ευρετηρίων σε ένα αρχείο με όνομα **`plans.txt`**. Πριν από κάθε πλάνο γράψτε την εντολή EXPLAIN ANALYZE από την οποία προέκυψε. Τέλος για κάθε ερώτημα στο οποίο προσθέσατε ευρετήριο γράψτε ένα συγκεντρωτικό σχόλιο της μορφής:

`/* Query 1: w/out index: 110 ms; w/index: 80 ms */`

Στο ίδιο σχόλιο προσθέστε και την αιτιολόγησή σας για την επιτάχυνση ή μη που προκύπτει από την προσθήκη ευρετηρίου.
Σε ένα αρχείο με όνομα **`query_indexes.sql`** βάλτε τις εντολές που τρέξατε για τη δημιουργία των ευρετηρίων.

Εργαλεία:

- AWS RDS Postgres instance
- Postgres psql ή/και PgAdmin

Συμβουλές για την υλοποίηση:

- Τρέξτε την εντολή VACUUM ANALYZE σε όλους τους πίνακες της βάσης σας πριν ξεκινήσετε να τυπώνετε τα πλάνα εκτέλεσης.
- Τρέξτε **`set enable_seqscan=off;`** για να απενεργοποιήσετε το scan των πινάκων (table scan) όταν δοκιμάζετε ένα υποψήφιο ευρετήριο.
- Ειδικά για τα εναύσματα, προτού αρχίσετε να τα εκτελείτε, καλό είναι να φτιάξετε ένα αντίγραφο των πινάκων που επηρεάζει το έναυσμα. Εάν γίνει κάποιο λάθος και σας είναι δύσκολο να επιστρέψετε στην προηγούμενη κατάσταση που ήταν ο πίνακάς σας απλώς ξεκινάτε να δουλεύετε με το αντίγραφό του. Σε περίπτωση που φτιάξετε τέτοια αντίγραφα πινάκων, όμως, όταν ολοκληρώσετε την Εργασία παρακαλούμε σβήστε τα αντίγραφα για να διευκολυνθούμε στην διόρθωση.
- Επιβεβαιώστε ότι ο χρήστης της βάσης του οποίου μας στέλνετε τα credentials όντως έχει πρόσβαση στη βάση σας και μπορεί να τρέχει `SELECT` queries.
- Τρέξτε και ελέγξτε τα αποτελέσματα όλων των εντολών SQL στην Airbnb βάση σας.

Χρήσιμα links:

Εντολή VACUUM:
https://www.postgresql.org/docs/9.6/sql-vacuum.html

Εντολή EXPLAIN:
https://www.postgresql.org/docs/9.6/using-explain.html

Βασική εντολή για δημιουργία ευρετηρίου:
https://www.postgresql.org/docs/9.6/sql-createindex.html

Δημιουργία ευρετηρίου σε περισσότερες από μία κολώνες:
https://www.postgresql.org/docs/9.6/indexes-multicolumn.html

Δημιουργία μερικού (partial) ευρετηρίου
https://www.postgresql.org/docs/9.6/indexes-partial.html

Δημιουργία εναύσματος
https://www.postgresql.org/docs/9.6/plpgsql-trigger.html
http://www.postgresqltutorial.com/creating-first-trigger-postgresql/

Παραδοτέα:

- Δημιουργήστε ένα .txt αρχείο στο οποίο θα αναγράφονται το endpoint του AWS instance σας (μπορείτε να το δείτε στο AWS console, *RDS* > *Databases* > *db_identifier* > *Connectivity section*), το όνομα της βάσης σας και το username και το password ενός χρήστη με read-only δικαιώματα, ώστε να μπορούμε να δούμε τους πίνακες της βάσης σας. Το .txt αρχείο θα πρέπει να έχει την παρακάτω μορφή:

  Endpoint: <name_of_the_endpoint>
  Username: <username>
  Password: <password>
  Database: <name_of_the_database>

- Βάλτε τα αρχεία **`plans.txt`**, **`query_indexes.sql`** και **triggers.sql** σε ένα φάκελο. Το όνομα του φακέλου πρέπει να αποτελείται από τους αριθμούς μητρώου σας χωρισμένους με παύλα, δηλαδή *αριθμός_μητρώου_1-αριθμός_μητρώου_2*. Δημιουργήστε ένα .zip αρχείο αυτού του φακέλου, το οποίο θα έχει το ίδιο όνομα με τον φάκελο.
- Κάντε υποβολή το .zip αρχείο στο eclass στην ενότητα *Εργασίες / 6η Εργασία*.