Αναφορά μέρους 3 της Β φάσης

Γεωργακάς Ιωάννης Ιάσων
ΑΜ: 2017030021

Περάκης Εμμανουήλ
ΑΜ:2017030099

Σκοπός του μέρους αυτού της δεύτερης φάσης του πρότζεκτ ήταν η χρήση διαφόρων ευρετηρίων προς την επιτάχυνση του χρόνου εκτέλεσης των αιτημάτων που υλοποιούνται στα μέρη Α και Β της φάσης αυτής. Τα ευρετήρια (indexes) είναι μία καλή μέθοδος βελτίωσης της απόδοσης ερωτημάτων, καθώς η postgresql by default ελέγχει κάθε γραμμή ενός πίνακα σειριακά πράγμα που για μεγάλους πίνακες μπορεί να αποβεί χρονοβόρο.

Α.

Στο ερώτημα αυτό μελετάται το αίτημα “Βρείτε τους φοιτητές που έχουν ένα ορισμένο πατρώνυμο” χρησιμοποιώντας την EXPLAIN ANALYSE και τους χρόνους εύρεσης και εκτέλεσης που αυτή εξάγει. Ξεκινώντας, το αίτημα εκτελείται για τους φοιτητές που προυπήρχαν στη βάση και οι οποίοι δεν ξεπερνολυν τις μερικές εκατοντάδες. Ύστερα, με χρήση μίας βοηθητικής συνάρτησης, εισάγονται επιπλέον 100000 φοιτητές και το αίτημα επαναλαμβάνεται.

Αρχικά εκτελείται το αίτημα χωρίς κάποιο ευρετήριο. Στη συνέχεια τα indexes που χρησιμοποιήθηκαν ήταν διαδοχικά:

- bTree
- Hash
- bTree with clustering

Το ευρετήριο που χρησιμοποιεί binary tree (bTree) παρουσιάζει βέλτιστη απόδοση όταν απαιτείται η εύρεση τιμών οι οποίες ανήκουν σε κάποιο εύρος (Range query). Αντιθέτως ο κατακερματισμός (Hashing) είναι βέλτιστος κατά την αναζήτηση μίας συγκεκριμένης τιμής, με χρήση του τελεστή “=”. Εφόσον στο ερώτημα ψάχνουμε τους φοιτητές με συγκεκριμένο πατρώνυμο, το Hashing αναμένεται να είναι πιο γρήγορο για πολλά δεδομένα. Τέλος, η εφαρμογή clustering με index bTree πάνω στο πατρώνυμο βοηθάει στη μείωση του χρόνου εύρεσης όλων των μαθητών με το συγκεκριμένο πατρώνυμο, καθώς αυτοί θα είναι πιο πιθανό να βρίσκονται στην ίδια σελίδα δίσκου.

Ο πίνακας απόδοσης των διαφόρων μεθόδων, για περίπου 150 φοιτητές, παρατίθεται παρακάτω στον Πίνακα 1. Για κάθε διφορετικό ευρετήριο, το αίτημα εκτελείται τρεις φορές.

index / **Execution time**	**1η εκτέλεση**	**2η εκτέλεση**	**3η εκτέλεση**
κανένα	0.088 ms	0.273 ms	0.060 ms
bTree	0.090 ms	0.061 ms	0.132 ms
Hashing	0.024 ms	0.030 ms	0.039 ms
btree with clustering	0.082 ms	0.084 ms	0.078 ms

Πίνακας 1

Τα ίδια ευρετήρια ακριβώς επαναχρησιμοποιούνται και για εκτέλεση του αιτήματος για 100000 φοιτητές.

index Execution time	1η εκτέλεση	2η εκτέλεση	3η εκτέλεση
κανένα	1.272 ms	0.479 ms	0.707 ms
bTree	0.649 ms	0.603 ms	0.744 ms
Hashing	0.409 ms	0.610 ms	0.518 ms
btree with clustering	0.354 ms	0.364 ms	0.375 ms

Πίνακας 2

Όπως φαίνεται από τους δύο παραπάνω πίνακες το indexing σχεδόν σε κάθε περίπτωση μειώνει το χρόνο εκτέλεσης, παρότι οι διαφορές μεταξύ των χρόνων μπορεί να θεωρηθούν και αμελητέες, ειδικά στην περίπτωση του μικρού αριθμού φοιτητών. Αυτό μπορεί να οφείλεται στο γεγονός ότι ακόμα και 100000 φοιτητές είναι ένα μικρό νούμερο αν λάβουμε υπόψιν το μέγεθος της μνήμης RAM στους σύγχρονους Η/Υ η οποία μπορεί να κυμένεται από 4-16 GB (ή και παραπάνω). Αυτό σημαίνει ότι δεν απαιτούνται προσβάσεις στο δίσκο πράγμα που μείωνει σημαντικά το χρόνο εκτέλεσης.

Παρατηρείται ότι για λίγους φοιτητές ο κατακερματισμός παρουσιάζει το μικρότερο χρόνο εκτέλεσης, όπως αναμενόταν καθώς γίνεται αναζήτηση μίας συγκεκριμένης τιμής-κλειδιού. Για πολλούς φοιτητές παρότι το Hashing υπερτερεί του bTree και του noIndex, η χρήση clustering σε συνδυασμό με ευρετήριο bTree είναι βέλτιστη. Αυτό συμβαίνει καθώς στο clustering μόλις βρεθεί η πρώτη τιμή οι υπόλοιπες λαμβάνονται σχεδόν άμεσα.

Β.

Στο ερώτημα αυτό μελετάται το αίτημα “Βρες φοιτήτριες που έχουν περάσει μάθημα με συγκεκριμένο κωδικό μαθήματος και συγκεκριμένο τελικό βαθμό”. Το αίτημα αυτό εκτελείται πάλι αρχικά με λίγους φοιτητές και ύστερα ο αριθμός τους αυξάνεται. Τα ευρετήρια που χρησιμοποιούνται είναι όμοια με αυτά του ερωτήματος Α. Επιπροσθέτως, μετά από κάθε εκτέλεση των αιτημάτων για τα διάφορα indexes, χρησιμοποιούνται οι εντολές set enable_hashjoin=off και set enable_mergejoin=off; ώστε να απενεργοποιηθούν οι αλγόριθμοι που εφαρμόζονται κατά το join.

Στους Πίνακες 3 και 4 παρουσιάζονται τα αποτελέσματα για λίγους και πολλούς φοιτητές αντίστοιχα.

index Execution time	1η εκτέλεση	2η εκτέλεση	3η εκτέλεση	1η εκτέλεση (hashjoin off)	2η εκτέλεση (hashjoin off)	3η εκτέλεση (hashjoin off)	1η εκτέλεση (mergejoin off)	2η εκτέλεση (mergejoin off)	3η εκτέλεση (mergejoin off)
κανένα	0.527 ms	0.274 ms	0.251 ms	0.673 ms	0.508 ms	0.253 ms	0.299 ms	0.322 ms	0.194 ms
bTree	0.405 ms	0.344 ms	0.341 ms	0.336 ms	0.317 ms	0.161 ms	0.294 ms	0.166 ms	0.198 ms
Hashing	0.304 ms	0.357 ms	0.394 ms	0.295 ms	0.246 ms	0.298 ms	0.171 ms	0.154 ms	0.205 ms

Πίνακας 3

index Execution time	1η εκτέλεση	2η εκτέλεση	3η εκτέλεση	1η εκτέλεση (hashjoin off)	2η εκτέλεση (hashjoin off)	3η εκτέλεση (hashjoin off)	1η εκτέλεση (mergejoin off)	2η εκτέλεση (mergejoin off)	3η εκτέλεση (mergejoin off)
κανένα	1.174 ms	0.371 ms	0.392 ms	0.452 ms	0.320 ms	0.352 ms	0.238 ms	0.312 ms	0.340 ms
bTree	0.409 ms	0.175 ms	0.292 ms	0.331 ms	0.515 ms	0.356 ms	0.365 ms	0.231 ms	0.146 ms
Hashing	0.687 ms	0.327 ms	0.147 ms	1.120 ms	0.235 ms	0.298 ms	0.455 ms	0.210 ms	0.142 ms

Πίνακας 4