ΠΑΠΠΑΣ ΑΘΑΝΑΣΙΟΣ | ΑΜ: Π19212
ΤΕΡΤΙΓΚΑΣ ΑΠΟΣΤΟΛΗΣ | ΑΜ: Π20186
ΚΩΣΤΟΡΙΖΟΣ ΗΛΙΑΣ | ΑΜ: Π20111

ISSUE #1

a) Αρχικά πρόσθεσα τον not equal operator ('!=': operator.ne) στο file misc.py, στα λεξικά των functions get_op, split_condition και reverse_op .

Μετά πρόσθεσα τον BETWEEN operator (σύνταξη είναι BETWEEN value,value) στα αντίστοιχα λεξικά. Εδώ ως value αυτού του κλειδιού είναι ένα lambda function το οποίο κάνει συγκρίσεις με το άνω και κάτω όριο που δώθηκε. Και εδώ χρειάστηκαν αλλαγές στα functions του misc.py που αναφέρθηκαν, καθώς και στο _parse_condition του table.py.

Τέλος πρόσθεσα τον γενικό NOT operator . Εδώ χρησιμοποίησα την reverse_op, και έτσι έκανα αλλαγές μόνο στην split_condition (εκτός του ότι πρόσθεσα και τον 'not between' operator στα λεξικά). Ουσιαστικά αν το op είναι 'NOT', τότε απλώς στέλνεται ο επόμενος operator στην reverse op και μετά ακολουθείται η συνηθισμένη διαδικασία. Επίσης έκανα λίγες τροποποίησεις στην reverse_op γιατί δεν ήταν εντελώς σωστή, π.χ το αντίθετο του <= δεν είναι το >= αλλά το >.

b) Δημιούργησα την συνάρτηση relevant_rows(condition) στο table.py. Αρχικά διαχωρίζω το condition που έρχεται βάσει του AND (π.χ αν είναι 'where id=1 AND name='John' OR cred=3' θα διαχωριστεί σε 'id=1' και 'name=John OR cred=3'). Μετά παίρνω όλα τα elements που δεν περιλαμβάνουν OR, και προσθέτω τα rows που μου φέρνουν σε μια προσωρινή λίστα. Τα elements που έχουν OR τα διαχωρίζω βάσει του OR και τα προσθέτω σε μια άλλη προσωρινή λίστα. Μετά αφού αφαιρέσω τα διπλά από την λίστα του OR, κάνω intersect τις προσωρινές λίστες. Όποιο row δηλαδή δεν συναντάται περισσότερες φορές από αριθμό των AND στην αρχική διαχώριση, αφαιρείται από την τελική λίστα.

Οι συναρτήσεις select_where, update_rows και delete_where τροποποιήθηκαν αναλόγως ώστε πλέον να τρέχουν χρησιμοποιώντας την relevant_rows, και συνεπώς να μπορούν να χειρίζονται complex queries με AND και OR

Παραθέτω screenshot που ως πρώτη εντολή δείχνει όλο το table και μετά χρησιμοποιούνται queries με not, between, and, or (είτε μόνα τους είτε σε συνδυασμό με άλλους op και μεταξύ τους)

ISSUE #2

a) Αρχικά τροποποίησα το create_query_plan() στον parser ώστε να μπορεί να αναγνωρίζει την σύνταξη create table(col_name col_type unique, …..). Έτσι όταν το action είναι create_table πλέον δημιουργείται και μια νέα λίστα, το unique_cols, που εμπεριέχει τις λίστες που δηλώνονται unique (όποια δηλωθεί ως primary_key προστίθεται αμέσως). Αυτή η λίστα μπαίνει ως value στο κλειδί dic['unique']. Επίσης στην _insert() του table.py πρόσθεσα και ένα if που τσεκάρει αν το value που θα εισαχθεί στο unique column ήδη υπάρχει, και τότε θα πετάει ValueError, με παρόμοιο τρόπο που γίνεται και το check των primary key στην insert() (πρόσθεσα ένα νέο attr στην table, τη λίστα unique idx, για να με βοηθήσει σε αυτό).

Μετά πρόσθεσα την παράμετρο unique=None στο create_table() της database.py και στον constructor της Table. Τα tables πλέον έχουν μια λίστα unique, που είναι άδεια αν δεν έχει δηλωθεί κάποια στήλη ως unique και αν δεν υπάρχει pk. Επίσης έγινε και update στο meta_indexes table, προστέθηκε ένα νέο column, το col_name τύπου str, για να γνωρίζουμε σε ποια columns του table υπάρχουν indexes.

Αλλαγές έγιναν και στα create_index και construct_index, και για την ενημέρωση των meta_indexes αλλά και για την καθεαυτή κατασκευή του index. Αρχικά διαχώρισα το table name από το column name (από τον parser ερχόταν σε μορφή table_name = 'table_name (col_name)', καθ ότι έτσι γίνεται το create index στο documentation). Μετά το τροποποίησα ώστε να μην πετάει exception αν το column δεν είναι το pk, και το έκανα να πετάει exception μόνο αν το col δεν είναι δεν είναι στην λίστα unique. Μετά έφτιαξα μια νέα παράμετρο για το construct index, την col_name, αφού πλέον για οποιαδήποτε στήλη είναι στην unique λίστα μπορεί να κατασκευαστεί index. Το functionality και των 2 συναρτήσεων τροποποιήθηκε αναλόγως.

Τέλος, αλλαγές έγιναν και στο select() της database.py, αφού πριν τσέκαρε μόνο αν το column του condition ήταν primary key, και δεν προχωρούσε σε select_with_btree αν δεν ήταν. Έφτιαξα μια νέα συνάρτηση, την check_index() η οποία ελέγχει αν υπάρχει το όνομα του condition column στο meta_indexes, και αν ναι, γυρνάει το index του row του, ώστε μετά να βρούμε και το όνομα του index μέσα από το τι index έχει στο meta_indexes (είναι λίγο confusing στην εξήγηση αλλά στον κώδικα γίνεται αντιληπτό πιο εύκολα) για να μπορούμε να τρέξουμε αναζήτηση με btree().

b) Ξεκίνησα αρχικά κατασκευάζοντας το hash.py. Επιγραμματικά, παίρνει κάθε (index, value) tuple από την construct_index, κάνει hash το value και μετά το εισάγει σε κάποιο bucket του directory. Ο αλγόριθμος που έχει υλοποιηθεί είναι αυτός που κάναμε στο μάθημα, είναι η LSB εκδοχή που εισάγει στο bucket αναλόγως τα τελευταία ψηφία, όταν γεμίσει το bucket κάνει split, αν το local depth είναι ίδιο με το global κάνει expansion κτλ. Ο σκοπός του hash index όπως το κατασκεύασα είναι να γυρίσει το row index του value που θα του δωθεί.

Μετά πήγα στην create_index και πρόσθεσα ένα elif για το αν το index type είναι hash, μετά δημιουργείται το record και μετά τρέχει η constuct_index, στην οποία πρόσθεσα μια έξτρα παράμετρο, την Btree που έχει default value True. Αν είναι False, το index κατασκευάζεται με hash και μετά σώζεται.

Στην συνέχεια κατασκεύασα την _select_where_with_hash, η οποία δεν παίρνει τις παραμέτρους distinct, order_by, desc και limit γιατί δεν υποστηρίζει ερωτήσεις διαστήματος όπως η _with_btree. Εδώ απλά χρειαζόμαστε το value που μας γυρνάει η parse_condition ώστε να βρούμε το row idx του μέσω του hash. Μετά απλά γυρνάμε εκείνο το row και τα cols που ζητήθηκαν.

Για να μπορεί να γίνει η επιλογή με ποιο select_where θα τρέξει το col αν έχει index, btree ή hash, δημιούργησα και ένα ακόμα col στο meta_indexes table, το type. Έτσι στην συνάρτηση select() η επιλογή γίνεται βρίσκοντας το type του index. Αν δωθεί type hash, εξασφαλίζεται επίσης πως δεν έχουμε ερώτηση διαστήματος, και έτσι παίρνει μόνο queries με operator '='.

Παραθέτω ss από το ίδιο table, και αρχικά δημιουργώ index σε column που δεν είναι primary key. Μετά κάνω search για να βεβαιωθώ πως η _select_with_btree λειτουργεί κανονικά μετά τις αλλαγές.. Μετά διαγράφω το index δημιουργώ νέο index στο ίδιο column με type hash και κάνω επίσης _select_with_hash για να δω πως λειτουργεί οκ.