Πανεπιστήμιο Πειραιώς
Σχολή Τεχνολογιών Πληροφορικής & Επικοινωνιών
Τμήμα Πληροφορικής

Ακαδ. έτος 2022-23 (χειμ. εξάμηνο)

Data Science Lab. @ University of Piraeus

Συστηματα Διαχειρισης Βασεων Δεδομενων

5ο εξ., υποχρ. κατευθ. ΠΣΥ / ΤΛΕΣ

Διδάσκων: καθ. Γιάννης Θεοδωρίδης

Εργαστηριακοί βοηθοί: Γιώργος Θεοδωρόπουλος, Γιάννης Κοντούλης (ΚΕΚΤ εργ. 205)

ΕΡΓΑΣΙΑ ΜΑΘΗΜΑΤΟΣ

(σε ομάδες έως 3 ατόμων)

Εισαγωγή

Θα εργαστείτε με το miniDB project, το οποίο αποτελεί ένα RDBMS φτιαγμένο εξ ολοκλήρου σε Python. Στο GitHub repository του project (https://github.com/DataStories-UniPi/miniDB) θα βρείτε αφενός το σχετικό documentation (στο αρχείο README) και αφετέρου τα διάφορα tasks με τα οποία μπορείτε να ασχοληθείτε (στο issues tab). Στον τίτλο του κάθε task αναφέρεται και ο μέγιστος βαθμός που αυτό μπορεί να δώσει (X/50). Βαθμολογία που υπερβαίνει το 50 θεωρείται bonus (στην εργασία και μόνο, δεν συμψηφίζεται με την γραπτή εξέταση), που σημαίνει ότι τυχόν λάθη ή ελλείψεις θα αντιμετωπιστούν επιεικώς.

Παραδοτέο

Η κάθε ομάδα θα παραδώσει τις υλοποιήσεις της μέσω ενός pull request στο GitHub. Το workflow που θα πρέπει να ακολουθηθεί είναι το εξής:

- Fork της miniDB στον λογαριασμό ενός ατόμου της ομάδας.
- Αλλαγές στο συγκεκριμένο fork από **όλα** τα μέλη της ομάδας. Η χρήση του upload files μέσω του GitHub website δεν συστήνεται διότι χάνονται τα ιστορικά των αρχείων. Το σωστό workflow αποτελείται από πολλά μικρά commits τα οποία συνοδεύονται από περιγραφές που βοηθούν στην κατανόηση του τι έγινε και από ποιον.

- Όταν οι υλοποιήσεις ολοκληρωθούν, δημιουργία pull request. Για σχετικές οδηγίες μπορείτε να συμβουλευτείτε το link. Κάθε ομάδα θα κάνει **μόνο** ένα pull request.

Δεν χρειάζεται να επισυνάψετε κάποιο ξεχωριστό αρχείο με επεξήγηση του κώδικα σας. Ωστόσο, θα πρέπει να συμπεριλάβετε τα παρακάτω:

- Στον **κώδικα** θα πρέπει να υπάρχουν comments που να εξηγούν τις διάφορες σχεδιαστικές αποφάσεις που έχουν παρθεί, καθώς και Docstrings (‘’’ info ‘’’) στην αρχή των διαφόρων functions/classes που ενδέχεται να υλοποιήσετε τα οποία ακολουθούν το πρότυπο που φαίνεται στα ήδη υπάρχοντα Docstrings.
- Στο **pull request**, θα πρέπει να γράψετε ένα **αναλυτικό description** για το τι έχει υλοποιηθεί, καθώς και να συμπεριλάβετε παραδείγματα εκτέλεσης τόσο με την μορφή εντολών, όσο και με screenshots. Θα πρέπει επίσης αναγράφονται τα ονόματα και τα ΑΜ των μελών της ομάδας σας.

Ημερομηνία παράδοσης - παρουσίασης

Η εργασία θα πρέπει να έχει παραδοθεί (**θα πρέπει να υπάρχει pull request**) μέχρι την τελευταία μέρα της εξεταστική Ιανουαρίου 2023. Αλλαγές που έχουν γίνει μετά την ημερομηνία αυτή δεν θα ληφθούν υπόψη. Απαραίτητη διευκρίνιση: **εργασίες δεν γίνονται δεκτές κατά την εξεταστική Σεπτεμβρίου**.

Ενδέχεται κάποιες ομάδες να χρειαστεί να παρουσιάσουν την δουλειά τους. Τυχόν παρουσιάσεις θα πραγματοποιηθούν μετά το τέλος της εξεταστικής και θα είναι σύντομες. Θα πληροφορηθείτε για την ακριβή ημερομηνία μέσω ανακοίνωσης στο Gunet2.

Απορίες σχετικά με την εργασία

Για οποιαδήποτε απορία σχετική την εργασία, μπορείτε να απευθυνθείτε στον Γιώργο Θεοδωρόπουλο μέσω email (gstheo@unipi.gr).

Ζητήματα δεοντολογίας

Είναι προφανές ότι η βαθμολογία πρέπει να αντικατοπτρίζει το επίπεδο της γνώσης που αποκόμισε ο φοιτητής μέσα από το μάθημα και κατάφερε να μεταφέρει αυτή τη γνώση στην εργασία. Για να εξασφαλιστεί όσο είναι δυνατό η παραπάνω αρχή, σε περίπτωση αντιγραφής οι εμπλεκόμενες εργασίες **μηδενίζονται**.

Issues

#1 - **Enrich WHERE statement** by supporting (a) NOT and BETWEEN operators (**5/50**) and (b) AND and OR operators (**10/50**)

Currently, miniDB does not support the use of (a) the simple NOT and BETWEEN operators as well as (b) complex where conditions combined with AND or OR operators. The target of this issue is to add this functionality, both at mSQL level and internally, i.e., in table.py and database.py.

#2 - **Enrich indexing functionality** by supporting (a) BTree index over unique (non-PK) columns (**10/50**) and (b) Hash index over PK or unique columns (**10/50**)

Currently, miniDB supports BTree indexing over the primary key of a table. The target of this issue is to (a) add support for BTree index over other columns as well, if they are declared as 'unique', and (b) add support for Hash index over the PK or a unique column of a table (implement extendible hashing, either MSB or LSB variant, by using a hash function based on the modulo operator "%" – dict is a hash struct in python that can be used in your implementation). The required changes affect the SQL parser mdb.py as well as the maintenance of the meta_table "meta_indexes". In either case, as a preparatory action, the CREATE TABLE statement should be enriched to support the declaration of a column as 'unique'.

#3 - **Implement miniDB's query optimiser** by (a) building equivalent query plans based on respective RA expressions (**10/50**) and (b) choosing the optimal query execution plan using a cost-based evaluation (**20/50**)

Currently, miniDB generates a single query plan (dictionary) that corresponds to the 'obvious' translation of a user's query. The target of this issue is to support cost-based query optimization. In detail: (a) build a module that is responsible for creating multiple equivalent query plans using relational algebraic equivalence transformation rules and (b) implement a simple query evaluator that uses per column statistics (number of distinct values, min-max values, histograms, etc.) to evaluate the cost of each query plan; the best plan should then be fed into miniDB for execution. Important note: the query optimizer should, on the one hand, maintain statistics metadata and, on the other hand, make **the least necessary** changes to the internal miniDB files (table.py and database.py). Use PostgreSQL and/or SQLite as a frame of reference when necessary.

Καλή Επιτυχία!