Διαχείρηση Σύνθετων Δεδομένων
Εργασία 1

Όθωνας Γκαβαρδίνας
ΑΜ: 2620

Μέρος 1
Πρόβλημα: Υλοποίηση merge-join μαζί με grouping
Αρχεία εισόδου: title.basics.tsv, title.akas.tsv
Γλώσσα: Python 3.6.7

Το αρχείο αποτελείται από μία γραμμή για κάθε εγγραφή, και κάθε attribute της εγγραφής είναι χωρισμένο με TAB.
Για το λόγο αυτό διαβάζω κάθε φορά μία γραμμή τη διαχωρίζω στον χαρακτήρα '\t' και αποθηκεύω το αποτέλεσμα σε μία λίστα.
Η διαδικασία αυτή γίνεται και για τα δύο αρχεία εισόδου.

Υπάρχει ένας βρόχος, που σταματά όταν κάποιο από τα δύο αρχεία φτάσει στο τέλος του. Τότε θα έχει τη μορφή [" "], και συνεπώς μήκος 1. (αυτό προέκυψε πειραματικά)

Στο βρόχο, με κοινό τη στήλη tconst, γίνεται συνένωση.

- Εάν τα δύο αλφαριθμητικά είναι ίδια προθέτοντε αρχικά τα στοιχεία για το αρχείο title.basics.tsv, και στη συνέχεια εκείνα για το title.akas.tsv. Στη διαδικασία αυτή εμπεριέχεται και ομαδοποίηση που θα περιγραφεί πιο κάτω.
- Εάν το πρώτο είναι μεγαλύτερο, μετακινείται το δεύτερο κατά μία γραμμή.
- Εάν μικρότερο, μετακινείται το πρώτο κατά μία γραμμή.

Για την ομαδοποίηση, χρησιμοποιήθηκε ένα λεξικό regions_dict. Το λεξικό αυτό έχει για κλειδί αλφαριθμητικό (εναλλακτικός τίτλος), και για τιμή λίστα (περιοχές).
Η διαδικασία έχει ως εξής:
Βρίσκω κάποιες γραμμές με ίδιο attribute tconst, τότε:

- Διατρέχω το δεύτερο αρχείο μέχρι να πάψουν να είναι ίδια τα tconst.
- Όταν πάψουν να είναι ίδια τυπώνω στο αρχείο τα δεδομένα μαζί με αλλαγή γραμμής.
- Κάθε φορά καθαρίζω το λεξικό μου.

-Χρησιμοποιήθηκαν οι open_files(), close_files(), read_first_lines(), για να φαίνεται πιο καθαρός ο κώδικας.

Μέρος 2
Πρόβλημα: Υλοποίηση merge-join με pipelining
Αρχεία εισόδου: title.basics.tsv, title.ratings.tsv, title.principals.tsv
Γλώσσα: Python 3.6.7

Για την υλοποίηση της άσκησης αυτής χρησιμοποίησα generators. Συγκεκριμένα, δημιούργησα τη συνάρτηση **generator()**, η οποία ανοίγει ένα αρχείο και κάθε φορά επιστρέφει την επόμενη γραμμή του αρχείου. (επίσης διαβάζει τη γραμμή με τις κεφαλίδες των στηλών)

Η συνάρτηση **scan()** διαχειρίζεται την παραπάνω συνάρτηση. Χρησιμοποίησα τη συνάρτηση **tee** από τη βιβλιοθήκη **itertools**, η οποία επιτρέπει την κλωνοποίηση ενός generator. Επίσης χρησιμοποίησα ένα global λεξικό **d**, στο οποίο αποθηκεύω κάθε generator που δημιουργώ. Η scan(), κάθε φορά επιστρέφει το ζητούμενο iterator, (ο οποίος αποτελεί το στοιχείο 0, στο αποτέλεσμα του tee).

Για την υλοποίηση του merge-join ως iterator, η συνάρτηση mj() λειτουργεί ως εξής:

-Διαβάζει τις δύο πρώτες γραμμές, και δημιουργεί από αυτές λίστες.
-Κρατά δύο λίστες prev, που χρησιμεύουν στη διαδικασία.

-Σε ένα βρόχο while, συγκρίνονται τα αποτελέσματα του πρώτου και δεύτερου iterator με βάση το στοιχείο 0, δηλαδή το tconst.
-Εάν είναι ίσα:
    Προχωρά η διαδικασία δημιουργίας της γραμμής εξόδου.
    Αποθηκεύονται οι δύο λίστες, στις λίστες prev.
    Διαβάζει τις δύο επόμενες γραμμές, και αποθηκεύει σε λίστες.
-Εάν το πρώτο είναι μεγαλύτερο:
    Μόνο εάν το tconst του είναι ίσο με το tconst του perv2, δημιουργείται η γραμμή εξόδου.
    Μόνο για τη γραμμή δεύτερη, αποκτά την τιμή της η λίστα prev, και διαβάζεται η επόμενη γραμμή.
-Εάν το δεύτερο είναι μεγαλύτερο:
    Γίνεται το αντίστροφο.

Η συνάρτηση **create_output()**, ξεχωρίζει εάν αυτό που της δόθηκε ως στήλες, είναι πλειάδα, ή αριθμός και δημιουργεί την έξοδο ως string, με τελικό χαρακτήρα την αλλαγή γραμμής.

Η ορθότητα της άσκησης ελέγχεται βγάζοντας κάθε φορά τα σχόλια σε ένα από τα τρία κομμάτια της **test_main()** συνάρτησης.
(Επίσης έχω δημιουργήσει μία συνάρτηση main που παίρνει ως είσοδο ονόματα αρχείων και αριθμούς στηλών και δημιουργεί το επιθυμητό αποτέλεσμα ως “γενικότερη υλοποίηση”)

Μέρος 3

(α)
title.basics: tconst, titleType, primaryTitle, originalTitle, isAdult, startYear, endYear, runtimeMinutes, genres
title.episode: tconst, parentTconst, seasonNumber, episodeNumber

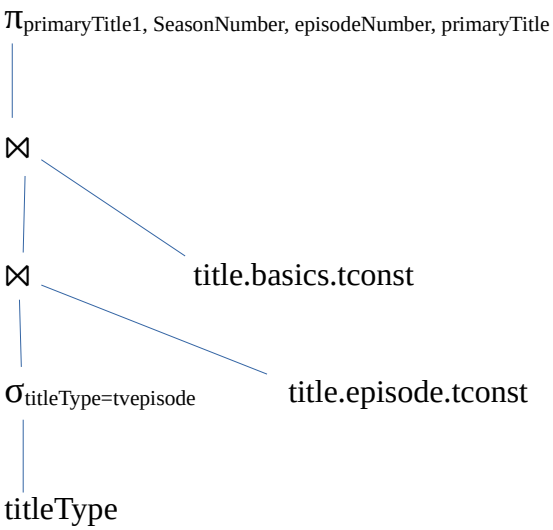
A(primaryTitle1, SeasonNimber, episodeNumber, parentTconst)
← $\pi_{primaryTitle,\ SeasonNumber,\ episodeNumber,\ parentTconst}((\sigma_{titleType=tvepisode}(title.basics))$
$\bowtie_{title.basics.tconst=title.episode.tconst}$ (title.episode))

$\pi_{primaryTitle1,\ SeasonNumber,\ episodeNumber,\ primaryTitle}$(A $\bowtie_{parentTconst=title.basics.tconst}$ (title.basics))

(β)
Σε συντομογραφία (άτυπο):
$((\sigma_{type=tv}(basics)) \bowtie_{tconst} episode) \bowtie_{tconst} (basics)$

Δηλαδή: η σειρά των πράξεων είναι:
1.σ 2.⋈ 3.⋈

Η πράξη join, έχει κόστος το οποίο εξαρτάται από την είσοδό του. Επομένως με την παραπάνω πράξη το πλάνο εκτέλεσης θα είναι, το παρακάτω, το οποίο είναι καλό, επειδή, το selection που γίνεται στην αρχή μικραίνει την είσοδο που θα περάσει στο join.



Η χρήση ενός B+-tree, για τον πίνακα title.basics, για το πρώτο πεδίο του πίνακα δεν θα βοηθούσε ιδιαίτερα. Αυτό επειδή η πράξη της επιλογής που υπάρχει αφορά το πεδίο titleType, ενώ το πρώτο πεδίο (tconst) χρειάζεται σε συνένωση, όπου γίνεται απλό σειριακό πέρασμα (merge join).

(γ)
Στην έξοδο, με την πράξη της προβολής (PROJECTION), κρατώ μόνο διακριτές εγγραφές. Αυτό έχει αρκετά μεγάλο κόστος, όμως παράγει το αποτέλεσμα που ζητώ.