ΣΥΣΤΗΜΑΤΑ ΔΙΑΧΕΙΡΙΣΗΣ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

MiniDB Εργασία

Task 1.3 Hash Indexes

Μέλη ομάδας

Δημήτρης Δημητρίου Π18036

Αντώνης Εκατομμάτης Π18039

Αλέξανδρος Γκινετσι Π18028

1. Hash indexes με διαφορετικά hashing functions (keys) δική σας επιλογής

Σε αυτό το ερώτημα ασχοληθήκαμε με την δημιουργία της κλάσης HashTable αλλά και με τις διαδικασίες που εκτελούνται για την δημιουργία των hash indexes.

MiniDB.HashTable

Σε αυτήν την ενότητα αναφέρονται όλοι οι μέθοδοι που αφορούν το HashTable για το πρώτο ερώτημα της εργασίας.

HashTable.__init__(self)

Function που τρέχει όταν αρχικοποιούμε το HashTable object.Ουσιαστικά δημιουργούμε τα slots(ή αλλιώς τα buckets) και θέτουμε ίσο τον αριθμό των στοιχείων που έχουν τοποθετηθεί στα buckets ίσο με 0.

HashTable.display_hash(self)

Προβάλλει το τρέχον HashTable δηλαδή τα buckets και τα περιεχόμενα τους.

HashTable._hash_function1(self, string ,size)

Αποτελεί συνάρτηση κατακερματισμού και πιο συγκεκριμένα υλοποίηση της djb2 του Daniel J. Bernstein. Το 5381 με το οποίο αρχικοποιείται η μεταβλητή hash προέκυψε έπειτα από έρευνα απόδοσης και επιλέχθηκε λόγω των καλών ιδιοτήτων όπως ότι είναι μονός και πρώτος αριθμός . Για κάθε χαρακτήρα του string που δίνεται σαν είσοδος υπολογίζουμε το ord του (δηλαδή τον αριθμό που αντιπροσωπεύει στο Unicode) και το προσθέτουμε στο άθροισμα ((hash <<5)+hash) το οποίο είναι ισοδύναμο με hash*33 +ord (c) .Ουσιαστικά υπολογίζουμε το hash(i) = hash(i-1) * 33 + str[i] όπου str[i] ο i χαρακτήρας του string.

Στο τέλος διαιρούμε το hash με το size.Το υπόλοιπο της διαίρεσης αναφέρει σε ποιο bucket θα πρέπει να τοποθετηθεί η τιμή που αντιστοιχεί στο κλειδί-string.

HashTable. _hash_function2(self, string ,size)

Αποτελεί συνάρτηση κατακερματισμού και πιο συγκεκριμένα υλοποίηση της sdbm. Για κάθε χαρακτήρα του string που δίνεται σαν είσοδος

υπολογίζουμε το ord του (δηλαδή τον αριθμό που αντιπροσωπεύει στο Unicode) και το προσθέτουμε στο άθροισμα (hash << 6) + (hash << 16) – hash το οποίο είναι ισοδύναμο με το hash(i) = hash(i - 1) * 65599 + str[i]. Το 65559 επιλέχθηκε και αυτό έπειτα από δοκιμές και λόγω των καλών ιδιοτήτων του (πρώτος αριθμός).

Στο τέλος διαιρούμε το hash με το size .Το υπόλοιπο της διαίρεσης αναφέρει σε ποιο bucket θα πρέπει να τοποθετηθεί η τιμή που αντιστοιχεί στο κλειδί-string.

HashTable._hash_function3(self,key_str, size)

Παίρνουμε κάθε χαρακτήρα c του κλειδιού που θα χρησιμοποιηθεί για κατακερματισμό και αθροίζονται τα ord(c).Έπειτα διαιρούμε το άθροισμα με το size.Επιστρέφουμε το υπόλοιπο από την διαίρεση.

HashTable. _hash_function4 (self,key, size)

Αφορά μόνο integers κλασική μέθοδος ,επιστρέφουμε το υπόλοιπο της διαίρεσης του key με το size.

HashTable._get_hash_index(self, key)

Επιστρέφει το δείκτη-αριθμό του slot που πρέπει να τοποθετηθεί το αντικείμενο με βάση το κλειδί key.

HashTable.get(self, key_search)

Αποτελεί την μέθοδο αναζήτησης.

Αρχικά προσδιορίζουμε με βάση το κλειδί αναζήτησης(key_search) σε ποιο bucket πρέπει να ψάξουμε. Στην συνέχεια ψάχνουμε μέσα στο bucket κλειδι/ά που ταιριάζουν με το κλειδί αναζήτησης και επιστρέφεται λίστα με τα αντίστοιχα values.

HashTable.set (self, key, value)

Με την μέθοδο αυτή προσθέτουμε σε κατάλληλο bucket χρησιμοποιώντας την συνάρτηση κατακερματισμού το key και το value.Η προσθήκη γίνεται με την μορφή tuple (key,value).

HashTable.delete(self,key_search)

Με την μέθοδο αυτή διαγράφουμε από το κατάλληλο bucket όλες τις τιμές που αντιστοιχούν σε κλειδί key_search.Αν δεν υπάρχουν τιμές με το αντίστοιχο κλειδί εκτυπώνεται μήνυμα λάθους.

HashTable_resize(self)

Αναδιοργανώνουμε το HashTable διπλασιάζοντας τον αριθμό slots-buckets και κάνοντας rehasing όλες τις τιμές και τοποθετώντας τις στο νέο HashTable.

Database

Database.create_index(self,table_name,index_name,column='pk_idx',index_type='Btree')

Δημιουργούμε ευρετήριο είτε Btree είτε Hash

Αρχικά ελέγχουμε αν υπάρχει το column που έχει δοθεί από τον χρήστη στον πίνακα που θέλει να φτιάξει ευρετήριο ή αν δεν έχει προσδιορίσει κάποιο column ελέγχουμε αν ο πίνακας έχει primary key.Αν οι παραπάνω συνθήκες ισχύουν , ελέγχουμε αν το index_name που έδωσε υπάρχει στον πίνακα meta_indexes και αν όχι δημιουργείτε είτε Btree ευρετήριο είτε Hash.Σε διαφορετικές περιπτώσεις εμφανίζονται κατάλληλα μηνύματα λάθους.

Database._construct_hash_index(self,table_name, index_name,column)

Δημιουργεί hash index με όνομα index_name πάνω στο column του table_name.Μόλις φτιαχτεί το ευρετήριο αποθηκεύεται σε .pkl αρχείο χρησιμοποιώντας την _save_index

Database. drop_index(self,name)

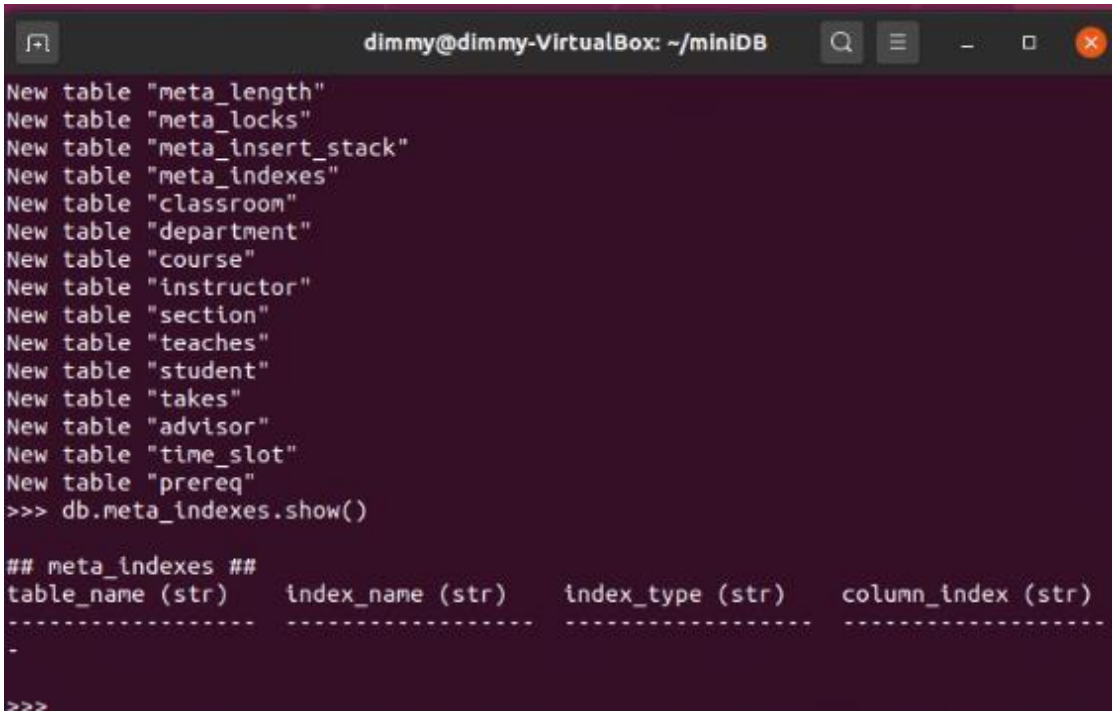
Διαγράφουμε το ευρετήριο με όνομα name.Γίνεται διαγραφή τόσο των .pkl αρχείων με το όνομα του ευρετηρίου όσο και διαγραφή εγγραφής από τον πίνακα meta_indexes.Αν δεν υπάρχει το ευρετήριο με το όνομα αυτό εμφανίζεται κατάλληλο μήνυμα.

Είναι ιδιαίτερα σημαντικό να επισημανθεί ότι η δομή του πίνακα meta_indexes τροποποιήθηκε και πλέον είναι της εξής μορφής:

1. Πεδίο όνομα πίνακα -table_name τύπου string

2. Πεδίο όνομα index -index_name τύπου string
3. Πεδίο τύπου ευρετηρίου -index_type και τύπου string
4. Πεδίο στήλη που έχει γίνει το ευρετήριο -column_index τύπου string.

Screenshots πρώτου ερωτήματος



Δημιουργία hash index στον πίνακα department με όνομα i1 στην στήλη building.

```
>>> db.create_index('teaches','i2','course_id','Btree')
Creating Btree index.
key  CS-101 indexes  0
key  CS-315 indexes  1
key  CS-347 indexes  2
key  FIN-201 indexes  3
key  MU-199 indexes  4
key  PHY-101 indexes  5
key  HIS-351 indexes  6
key  CS-101 indexes  7
key  CS-319 indexes  8
key  BIO-101 indexes  9
key  BIO-301 indexes  10
key  CS-190 indexes  11
key  CS-190 indexes  12
key  CS-319 indexes  13
key  EE-181 indexes  14
>>>
```

Δημιουργία Btree index στον πίνακα teaches με όνομα i2 στην στήλη course_id.

```
>>> db.create_index('classroom','i3','room_number','Hash')
Creating Hash index
key  101 indexes  0
key  514 indexes  1
key  3128 indexes  2
key  100 indexes  3
key  120 indexes  4
>>> db.meta_indexes.show()

## meta_indexes ##
table_name (str)    index_name (str)    index_type (str)    column_index (str)
------------------  ------------------  ------------------  ------------------
--
department          i1                  Hash                building
teaches             i2                  Btree               course_id
classroom           i3                  Hash                room_number

>>>
```

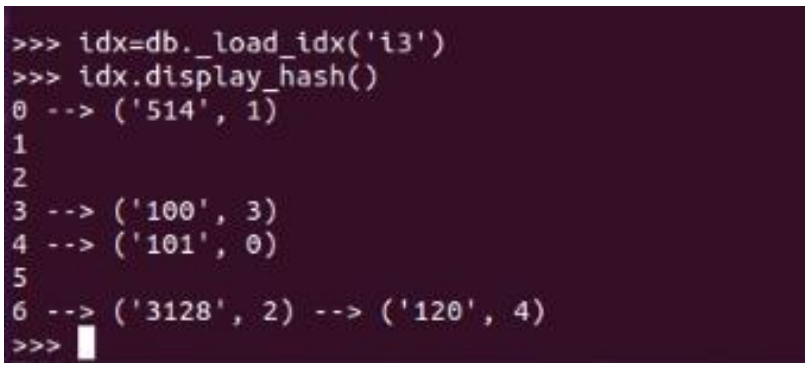
Δημιουργία hash index στον πίνακα classroom με όνομα i3 στην στήλη room_number.Έπειτα εμφάνιση πίνακα meta_indexes.

```
>>> db.drop_index('i2')
Deleted 1 rows
>>> db.meta_indexes.show()

## meta_indexes ##
table_name (str)    index_name (str)    index_type (str)    column_index (str)
------------------  ------------------  ------------------  ------------------
--
department          i1                  Hash                building
classroom           i3                  Hash                room_number

>>>
```

Διαγραφή του index με όνομα i2.



Εμφάνιση hashTable του hash index i3.

2.Hash search

Σε αυτό το ερώτημα γίνεται η χρήση του ευρετήριού hash για αναζήτηση και υλοποιήθηκαν η _select_where_with_hash στο table.py και έγιναν κάποιες αλλαγές στην select στο database.py

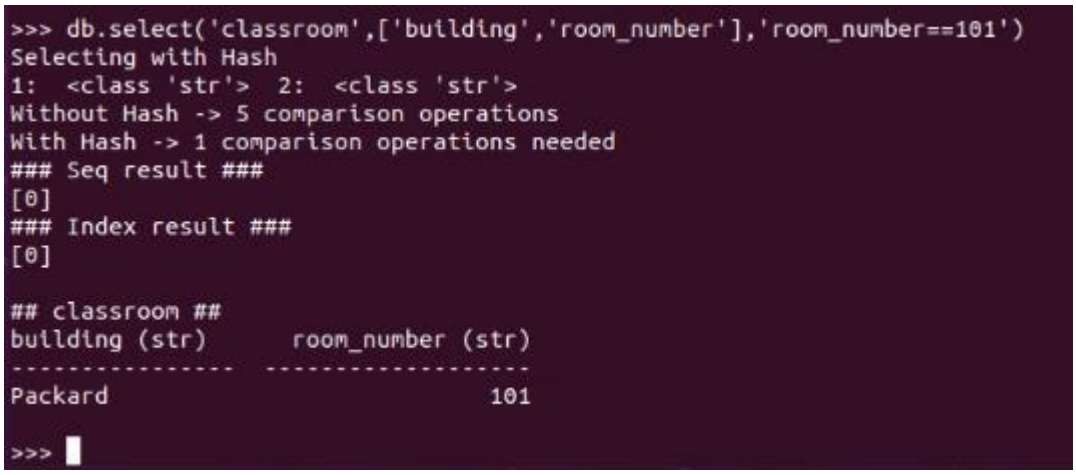
Database. select(self, table_name, columns, condition=None, order_by=None, asc=False,\ top_k=None, save_as=None, return_object=False)

Wrapper function της table._select_where ή της table._select_where_with_hash (αν ο πίνακας table_name έχει index στην στήλη πάνω στην οποία γίνεται το select). Φορτώνει μέσω της database.load το τελευταίο checkpoint της βάσης και ελέγχει αν ο πίνακας με όνομα table_name είναι κλειδωμένος. Αν δεν είναι, τον κλειδώνει και καλεί την κατάλληλη select (με ή χωρίς hash index). Για να γίνει select με βάση τον κατακερματισμό πρέπει να έχει hash index ο πίνακας στην στήλη που είναι στο condition και να είναι αποκλειστικά ερώτηση ταυτότητας(==). Στην συνέχεια επιστρέφει και κάνει show τον πίνακα αν return_object=False, αλλιώς επιστρέφει τον πίνακα σαν αντικείμενο table. Αν save_as!=None, αποθηκεύει τον πίνακα στη βάση με το όνομα save_as. Τέλος, ξεκλειδώνει τον πίνακα table_name και αποθηκεύει ένα νέο checkpoint της βάσης.

Table._select_where_with_hash(self, return_columns, hs, condition, order_by=None, asc=False, top_k=None)

Όμοια με την table._select_where, με την διαφορά ότι κάνει χρήση ενός HashTable object (hs) για να εντοπίσει τα rows που ικανοποιούν τη συνθήκη condition. Γίνεται χρήση της HashTable.get.Εκτυπώνει τα αποτελέσματα (indexes όπου condition is true) και των δύο τεχνικών (με/χωρίς hash index) καθώς και τον αριθμό συνολικών συγκρίσεων μεταξύ στοιχείων, δείχνοντας την διαφορά στην απόδοση.

Κάνοντας αναζήτηση μέσω ευρετηρίου στον πίνακα department στην συνθήκη building==Watson.(όπου είχαμε φτιάξει hash index i1 στο πίνακα department στη στήλη building).



Κάνοντας αναζήτηση μέσω ευρετηρίου στον πίνακα classroom με συνθήκη room_number==101.(όπου είχαμε φτιάξει hash index i3 στο πίνακα classroom στην στήλη room_number).

3.Hash join

Για την hash join υλοποιήθηκε η function Database.hash_join .

Database.hash_join(self, left_table_name, right_table_name, condition, save_as=None, return_object=False)

Αρχικά ελέγχουμε την συνθήκη condition και αν είναι κενή επιστρέφει. Αν η συνθήκη δεν είναι κενή τότε αναλύεται σε τελεστές της μορφής left_condition_column , right_condition_column και ελέγχουμε αν αυτές οι στήλες που έχουν δοθεί ανήκουν όντως στον αριστερό και στον δεξιό πίνακα αντίστοιχα. Αν ναι τότε φορτώνει μέσα από το database.load το τελευταίο checkpoint της βάσης και ελέγχει αν ο αριστερός είτε ο δεξιός πίνακας είναι κλειδωμένος. Αν κανένας από τους δύο δεν είναι κλειδωμένος τότε γίνεται αναζήτηση στο πίνακα meta_indexes έτσι ώστε να βρούμε αν και οι δύο πίνακες έχουν ευρετήριο hash πάνω στο column στο οποίο έχουν ζητήσει να γίνει hash join.Αν τουλάχιστον ένας πίνακας δεν έχει ευρετήριο κατακερματισμού πάνω στο column που έχει ζητήσει να χρησιμοποιηθεί για hash join τότε προβάλλόνται κατάλληλα μηνύματα προς το χρήστη και η συνάρτηση επιστρέφει. Αν και οι δύο πίνακες έχουν ευρετήρια κατακερματισμού στις συγκεκριμένες τους στήλες τότε χρησιμοποιείται η μέθοδος του hash join δηλαδή για κάθε ζεύγος bucket Ri και Si(Ri για τον αριστερό πίνακα και Si για τον δεξιό πίνακα) ψάχνουμε πρώτα το slot Ri και για κάθε εγγραφή r του αναζητούμε εγγραφές του bucket Si που ταιριάζουν με την εγγραφή r του αριστερού πίνακα. Μόλις βρούμε εγγραφές που ταιριάζουν γίνεται σύνδεση τους και εισάγονται σαν μια εγγραφή στο καινούργιο πίνακα join_table.Τέλος η μέθοδος επιστρέφει και κάνει show () τον πίνακα αν το return_object είναι false αλλιώς επιστρέφει τον πίνακα σαν αντικείμενο table.Αν η save_as!=None αποθηκεύει τον πίνακα join_table στην βάση με το όνομα save_as.

```
>>> db.hash_join('department','classroom','building==building')
condition column of left table building
condition column of right table  building
i1
make a hash index for right table  classroom  in column  building
```

Μήνυμα προς τον χρήστη ότι ο πίνακας classroom δεν έχει φτιάξει ευρετήριο κατακερματισμού στην στήλη building και έτσι δεν είναι δυνατόν να γίνει η hash_join.

```
>>> db.create_index('classroom','i4','building','Hash')
Creating Hash index
key  Packard indexes  0
key  Painter indexes  1
key  Taylor indexes  2
key  Watson indexes  3
key  Watson indexes  4
>>> db.meta_indexes.show()

## meta_indexes ##
table_name (str)    index_name (str)    index_type (str)    column_index (str)
------------------  ------------------  ------------------  ------------------
--
department          i1                  Hash                building
classroom           i3                  Hash                room_number
classroom           i4                  Hash                building

>>>
```

Δημιουργία hash index στον πίνακα classroom με όνομα i4 στην στήλη building .Έπειτα εμφάνιση πίνακα meta_indexes.

```
>>> db.inner_join('department','classroom','building==building')
left  1
right  0
## Select ops no. -> 35
# Left table size -> 7
# Right table size -> 5

## department_join_classroom ##
department_dept_name (str)   department_building (str)    department_budget (int)  classroom_building (str)    classroom_room_number (str)   classroom_capacity (int)
---------------------------  -------------------------  -----------------------  --------------------------  ----------------------------  --------------------------
Biology                      Watson                                      90000  Watson                                       100                          30
Biology                      Watson                                      90000  Watson                                       120                          50
Comp. Sci.                   Taylor                                     100000  Taylor                                      3128                          70
Elec. Eng.                   Taylor                                      85000  Taylor                                      3128                          70
Finance                      Painter                                    120000  Painter                                      514                          10
History                      Painter                                     50000  Painter                                      514                          10
Music                        Packard                                     80000  Packard                                      101                         500
Physics                      Watson                                      70000  Watson                                       100                          30
Physics                      Watson                                      70000  Watson                                       120                          50
```

Χρήση inner join των πινάκων department, classroom και με συνθήκη department.building==classroom.building.

```
>>> db.hash_join('department','classroom','building==building')
condition column of left table building
condition column of right table  building
i1
i4
## Select ops no. -> 9
# Left table size -> 7
# Right table size -> 5

## department_join_classroom ##
department_dept_name (str)   department_building (str)    department_budget (int)  classroom_building (str)    classroom_room_number (str)   classroom_capacity (int)
---------------------------  -------------------------  -----------------------  --------------------------  ----------------------------  --------------------------
Biology                      Watson                                      90000  Watson                                       100                          30
Biology                      Watson                                      90000  Watson                                       120                          50
Comp. Sci.                   Taylor                                     100000  Taylor                                      3128                          70
Elec. Eng.                   Taylor                                      85000  Taylor                                      3128                          70
Finance                      Painter                                    120000  Painter                                      514                          10
History                      Painter                                     50000  Painter                                      514                          10
Music                        Packard                                     80000  Packard                                      101                         500
Physics                      Watson                                      70000  Watson                                       100                          30
Physics                      Watson                                      70000  Watson                                       120                          50

>>>
```

Χρήση hash_join των πινάκων department, classroom και με συνθήκη department.building==classroom.building.

4.Hashing visualization

Για αυτό το ερώτημα αναπτύχθηκε η συνάρτηση plot στην κλάση HashTable

HashTable.plot(self)

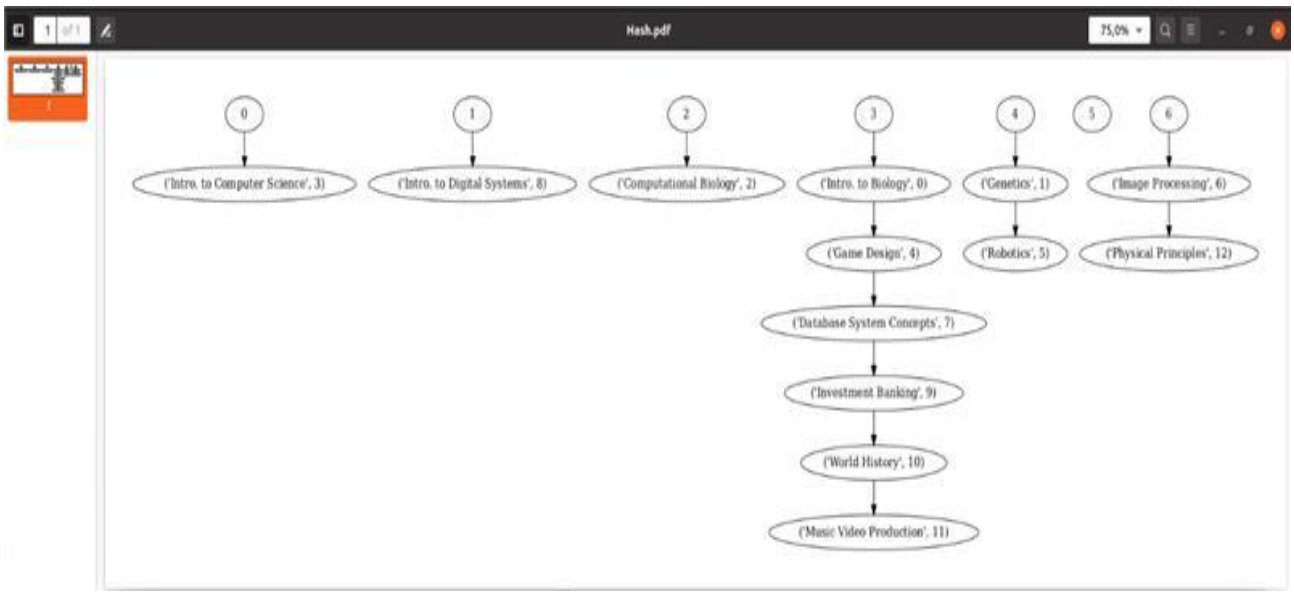
Γιαν την υλοποίηση αυτής της συνάρτησης χρησιμοποιήθηκε το package graphviz για την οπτικοποιήση του hash map.Ουσιαστικά δημιουργούμε ένα γράφημα όπου παρουσιάζονται όλα τα buckets καθώς και τα περιεχόμενα τους. Πιο συγκεκριμένα δημιουργείται για κάθε bucket ένας κόμβος με τον αριθμό του. Αν το bucket δεν είναι κενό δημιουργούνται για τα περιεχόμενα του αντίστοιχοι κόμβοι και δημιουργούνται δεσμοί μεταξύ τους. Στο τέλος κάνουμε save και render το γράφημα και κάνουμε προβολή του τελικού γραφήματος σε pdf μορφή.

```
>>> db.create_index('course','i5','title','Hash')
Creating Hash index
key  Intro. to Biology indexes  0
key  Genetics indexes  1
key  Computational Biology indexes  2
key  Intro. to Computer Science indexes  3
key  Game Design indexes  4
key  Robotics indexes  5
key  Image Processing indexes  6
key  Database System Concepts indexes  7
key  Intro. to Digital Systems indexes  8
key  Investment Banking indexes  9
key  World History indexes  10
key  Music Video Production indexes  11
key  Physical Principles indexes  12
```

Δημιουργία hash index στον πίνακα course με όνομα i5 στην στήλη title .

```
>>> idx=db._load_idx('i5')
>>> idx.plot()
```

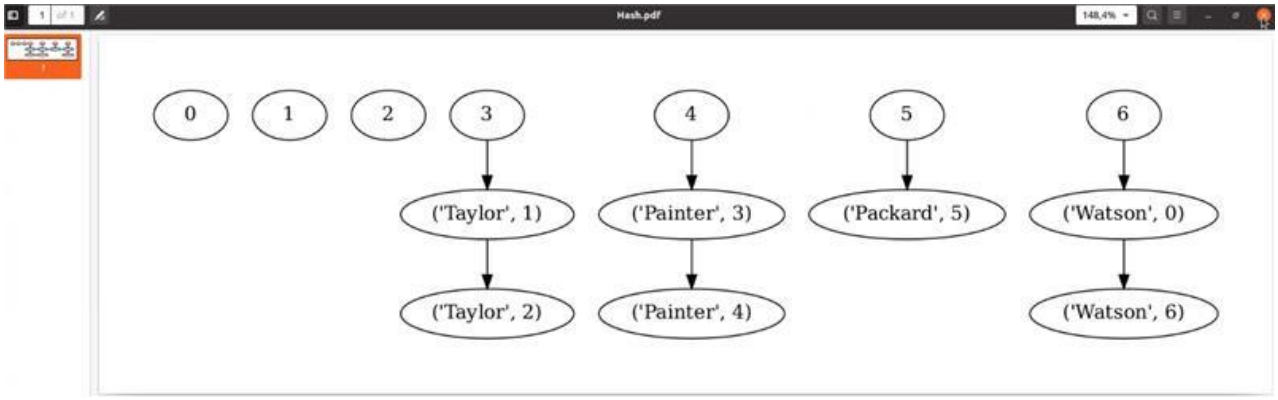
Χρήση του plot για το i5 index.



Χρήση της plot για visualization του hashmap του hash index i5

```
>>> idx=db._load_idx('i1')
>>> idx.plot()
```

Χρήση του plot i1.



Χρήση της plot για visualization του hashmap της hash index i1