Pràctica 2: Accés indexat. Hashing

Bases de Dades — iTIC

Marta Tarrés-Puertas
Sebastià Vila-Marta

26 de febrer de 2015

Índex

1 Organització **1**
1.1 Material necessari 1
1.2 Lliurament 1

2 Introducció **1**

3 Indexat basat en hashing **2**

4 Disseny d'un fitxer indexat per hash **2**

1 Organització

1.1 Material necessari

Simplement us cal un computador amb sistema operatiu GNU/Linux i connexió a xarxa. Pel que fa a la documentació de referència, les pàgines de man i les referències detallades al final del document us poden ser útils.

1.2 Lliurament

Cal lliurar la pràctica en un tarfile a través d'Atenea en la data fixada.

2 Introducció

Sovint l'accés més escaient a les dades d'un fitxer no és en base a la seva posició dins el fitxer, com en el cas dels fitxers d'accés directe, sinó en base al seu contingut. Un exemple clàssic el trobem quan el fitxer conté informació de persones: sovint per accedir a aquesta informació la forma més escaient és fer-ho a través dels noms de les persones. Aquest mode d'accés a les dades s'anomena *accés indexat* i el camp —o camps— de les dades que s'usen per accedir a la informació *claus*.

Aquesta forma d'accés que permet usar un fitxer com si es tractés d'una taula associativa és molt semblant al que s'aconsegueix amb els diccionaris de `Python`. En el cas dels diccionaris les dades s'emmagatzemen a memòria, en canvi, en el cas dels fitxers indexats s'emmagatzemen al disc.

Sovint per implementar l'accés indexat s'usen dos fitxers físics: un conté les dades pròpiament mentre que l'altre conté l'estructura de dades que s'usa per a l'accés indexat. Aquest darrer fitxer físic sovint es coneix simplement com a *índex*. Hi ha moltes estructures de dades factibles per a implementar índexs. Els arbres de cerca en les seves diverses variants (BTrees, B+Trees, etc.) en són una. Les taules de hash en són una altra.

L'objectiu d'aquesta pràctica és implementar una petita aplicació basada en un fitxer d'accés indexat. L'estratègia que s'usarà es fonamentarà en les taules de hash.

3 Indexat basat en hashing

Sigui $D = \{d_0, \ldots, d_m\}$ una col·lecció de dades. Sigui $k_i = K(d_i)$ la clau corresponent a la dada d_i. Assumim que $\forall 0 > i \leq m : mida(d_i) = mida(d_0)$. En aquest context és senzill emmagatzemar les dades del conjunt D en un fitxer d'accés directe de forma que cada dada d_i ocupa una posició coneguda $p(d_i)$ dins del fitxer.

Un índex és una estructura de dades que permet calcular eficientment $p(d_i)$ si es coneix k_i. La implementació hash d'un índex es basa en:

- Una funció d'aleatorització $h(k) : K \to \{0, \ldots, n\}$ que mapeja l'espai de totes les claus possibles a un espai generalment més reduït de valors naturals consecutius. La funció d'aleatorització pot calcular el mateix valor de hash per a dues claus diferents. Els nombres primers són bons candidats a l'hora d'escollir valor per n.

- Un fitxer d'accés directe estructurat en *buckets*, que no són altra cosa que registres de mida fixa que contenen un cert nombre B de claus.

El funcionament és aproximadament el següent. Quan es vol emmagatzemar una clau k a l'índex, se'n calcula el seu valor de hash h_k usant la funció d'aleatorització. Aleshores, la informació associada a aquesta clau s'emmagatzema en el bucket número h_k. Això permet que, quan es cerqui la informació relacionada amb la mateixa clau simplement calgui calcular de nou el valor h_k i cercar dins del bucket corresponent aquesta clau.

Finalment, associat a cada clau hi ha un valor enter que indica a quina posició del fitxer de dades es troben les dades corresponents a aquesta clau.

Si després de moltes insercions que van a parar al mateix bucket aquest s'omple estem en una situació de *col·lisió*. En aquest cas cal aplicar algun tipus d'estratègia que permeti resoldre el cas. Una estratègia habitual consisteix a enllaçar al bucket ple un nou bucket buit i continuar emplenant aquest darrer. Naturalment això implica també tirar enrera aquesta decisió a mida que s'esborren elements i els buckets enllaçats deixen de ser necessaris.

Tasca 1 Estudieu l'estructura d'un índex hash en base a les referències. Preneu notes de la Lecture 7 [DL13]

Tasca 2 Investigueu la funció predefinida de `Python` **hash**().

4 Disseny d'un fitxer indexat per hash

Tasca 3 Dissenyeu i implementeu un script per representar un fitxer indexat de persones. De cada persona se'n té el nom, el DNI i l'any de naixement. L'accés ha de ser a partir del nom. L'estructura ha d'admetre operacions per inserir i consultar les dades per nom. El mecanisme d'indexat s'ha de basar en una estratègia hash. Podeu suposar que no hi haurà *col·lisions*. Afegiu-hi els jocs de proves pertinents.

Tasca 4 Ampliació.

Afegiu-hi la gestió en cas de *col·lisions* i permeteu l'esborrat de dades. Afegiu-hi els jocs de proves pertinents.

Referències

[Wiki15] Contributors of Wikipedia. Hash table. `http://en.wikipedia.org/wiki/Hash_table`. (consultat 26 de febrer de 2015).

[Python15] Python project contributors. The Python Standard Library. `https://docs.python.org/2/library/` (consultat 26 de febrer de 2015).

[Hashing] Problem Solving with Algorithms and Data Structures, `http://interactivepython.org/runestone/static/pythonds/SortSearch/Hashing.html` (consultat 26 de febrer de 2015).

[DL13] Prof. Erik Demaine, Prof. Charles Leiserson, Lecture 7: Hashing, Hash Functions. MITOpenCourseware. http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-046j-introduction-to-algorithms-sma-5503-fall-2005/video-lectures/lecture-7-hashing-hash-functions/ (consultat 26 de febrer de 2015).