Universitat de Girona

# PRÀCTICA 1

## Grau en Enginyeria informàtica

*Laura Galera*

Professor

Miquel Bofill

1 de desembre de 2019

# ORGANITZACIÓ DE LES CLASSES
## UNA QÜESTIÓ DE RESPONSABILITAT

Tot i tractar-se d'una pràctica enfocada al bon ús de les estructures de dades, consider-ho oportú fer una breu pinzellada al disseny del programa. Pot cridar l'atenció el gran -això és relatiu- nombre de classes usades per a la gestió i organització de dades, com també de les posteriors consultes. De fet, no és complex d'entendre que principalment hi haurà cinc classes base (aeroport, aerolínia, ruta, vol i avio[1]), ja que cada línia de fitxer conforma un objecte del tipus esmentat. Ara bé, per què existeixen cinc classes més en plural per cada una de les anteriors, i el més important, per a què?

Aquestes classes (aeroports, aerolínies, rutes, vols i avions) són les encarregades de treballar sobre el conjunt d'objectes apropiats a elles. Així doncs, per exemple, la classe Airports assumirà la responsabilitat de gestionar tot allò que s'associa a un aeroport dins el conjunt. A la part privada de cada classe s'hi aïllen els atributs, com és d'imaginar seran diferents estructures de dades, que permetin satisfer tota operació on les classes inferiors[2] siguin necessàries.

El més interessant és la jerarquització de funcions, que permeten un encapsulament de la interacció entre classes. En altres paraules, per exemple, la classe Airlines s'encarrega d'afegir i verificar cadascuna de les seves aerolínies, entre altres tasques, i per tant, quan la classe Routes necessiti d'ella per decidir si acceptar una ruta o no, simplement li preguntarà si l'aerolínia de la ruta és vàlida, és a dir, usarà un dels mètodes de la interfície d'aerolínies. No obstant això, rutes no té per què conèixer de quina manera guarda aerolínies el seu conjunt, i aquesta és la part que dóna consistència a l'encapsulament. A més, obre la porta a què un altre programador pugui usar alguna d'aquestes classes sense necessitat en absolut de visualitzar com està implementada, únicament amb la descripció. Podem dir, doncs, que cadascú dur a terme petites parts del puzle però que interaccionant amb harmonia entre ells confeccionen un gran programa.

# ESTRUCTURES DE DADES
## AGILITZAR EL PROPÒSIT

Dono pas al pilar fonamental de l'informe: les estructures de dades. En aquesta pràctica les estructures de dades ressalten a tot arreu, per tant voldria distingir tres usos d'elles força diferents. Per una banda, ens trobem amb com emmagatzemar tota la informació d'entrada per tal de preveure una bona eficiència de cara a les consultes. Per l'altra banda, cal usar estructures per a la manipulació de dades durant el processament de les peticions. Per últim, trobo elegant servir únicament allò que és demanat; per tant, hi intervenen noves estructures que fan de pont entre les classes de conjunts i la classe principal.

[1] En el meu cas, totes les classes estan traduïdes a l'anglès.

[2] Es refereix a les classes que conformen un conjunt.

# ESTRUCTURES COM ATRIBUTS

Analitzem primerament les estructures que es troben com a atributs de les classes conjunts. En aquest cas la majoria de les consultes ens guien en cert punt a imaginar que moltes d'elles es basaran a identificar aerolínia, aeroport, avió... a partir d'un dels seus camps. Principalment, queda clar que aquests camps han de ser únics per poder usar-los com a identificadors, en alguns casos s'haurà de recorre a estructures auxiliars pel refinament de les dades. El fet de suposar que a partir d'un camp s'obtindrà l'objecte que conté el valor ens orienta a usar un **map** en molts casos. Com les dades no tenen per què estar ordenades, l'elecció podria haver sigut un **unordered_map**. Tot així, com que la funció de hash ja ve donada pels tipus primitius, no som conscients fins a quin punt és bona i el unordered_map en consultes acostuma a ser O(1), però en els pitjors casos és O(n), en canvi el map sempre és O(log n).

Comentem què s'ha usat per a cada classe i quines idees s'han descartat:

**AIRPORTS**
Començant per **Airports**, aquesta classe consta de tres maps. Cadascun d'ells té com a camp clau ID, IATA o ICAO, i un punter a l'aeroport que ve identificat per aquella clau. S'usen aquests tres maps perquè ens permeten, en primer lloc, validar una ruta -primerament per ID i en cas que es desconegui, per IATA o ICAO- i en segon lloc validar un vol, ja que aquest té el codi IATA de l'aeroport de sortida i el d'arribada. Sempre que sigui possible es consulta el map ID perquè en ell s'hi aïllen tots els aeroports, és un camp que mai és NULL.

En primera instància vaig considerar usar un **vector**, indexant cada aeroport segons ID, ja que fer consultes era en temps constant. El problema és que certs camps quedaven buits, ja que els codis ID dels aeroports no seguien un ordre correlatiu. A més, els espais buits podrien incrementar-se absurdament en un futur si s'entrés un aeroport amb ID, per exemple, 8000000, essent l'últim 5417. També podria arribar a ser perillós pel cas d'accedir a posicions del vector buides.

**AIRLINES**
Amb un criteri similar a Airports, **Airlines** disposa de tres maps on la clau de cadascun és el codi ID, IATA o ICAO. El codi ID és únic i identificatiu, a diferència dels codis IATA i ICAO que poden estar repetits entres vàries aerolínies. Aquesta dificultat suggereix el suport d'estructures auxiliars com són els dos **set** de repetits, un per codi IATA i un altre per ICAO. Simplement l'elecció de sets és deu a què es pot inserir el mateix codi IATA sense haver-se de preocupar si ja estava o no inserit prèviament.
No és gaire satisfactori tenir per atributs d'una classe dos sets que són usats únicament després de la lectura de dades. No obstant això, a part de buidar-se després del refinament de dades[3], és útil fer-ho d'aquesta manera perquè el mètode **copy** també els necessitarà.
Pel cas de les aerolínies, també vaig contemplar la idea d'usar un **vector** indexant cada punter d'aerolínia pel seu codi ID, però per la mateixa raó comentada anteriorment va acabar descartant-se.

[3]No cridar la funció que té aquesta utilitat es podria tractar com una excepció.

**ROUTES**
Per tal de fer una correcta gestió de les dues primeres consultes, les rutes estan pensades per a, a partir d'una aerolínia, obtenir totes les rutes fetes per aquesta. Així és com, la millor opció és tenir un **map**, amb clau ID, i a la seva descripció una **list** amb punters a totes les rutes. La idea d'usar una list és perquè tot el que ens interessa és afegir, per tant fer **push_back**, i recorreguts sobre ella.

El problema de les rutes és que una minoria d'elles no disposa del camp ID, és a dir, és desconegut. En aquesta situació, no es poden ignorar, sinó que es necessita un altre atribut com a identificador de quina aerolínia ha operat aquella ruta. L'única opció és el codi IATA o ICAO de l'aerolínia, podent ser qualsevol dels dos. I així és com sorgeix la necessitat d'afegir dos maps més on la clau és el codi IATA o ICAO de l'aerolínia i una list de punters a totes les seves rutes. Els dos últims maps només s'utilitzen per a les rutes on l'única manera d'identificar l'aerolínia és pel IATA o ICAO. Per tant, sempre es donarà preoritat de consulta al map ID.

**PLANES**
En el cas dels avions, per tal de comprovar que l'equipament de les rutes existeix, és necessari usar un map amb camp clau el codi IATA i a la descripció un punter a l'avió. Únicament cal aquest identificador perquè la classe que requereix de Planes només és Routes. No obstant això, de manera molt similar a Airlines, els codis IATA poden arribar a estar repetits. Per gestionar aquest fet també cal un set que sigui capaç de guardar els codis repetits i efectuar una neteja prominent dels avions impossibles d'identificar.

**FLIGHTS**
Finalment, arriba la classe que destaca entre la resta, ja que 5 de les consultes són principalment gestionades per ella. Això implica ser conscients d'exactament què demana cada petició. Per tal de donar resposta a la tercera, quarta i última consulta, la millor idea és tenir un map on el camp clau sigui la data i, a continuació, a la descripció, una list amb punters a tots els vols fets en aquella data. La segona estructura, que facilita la consulta cinc i sis, és un map amb camp clau el codi IATA de l'aerolínia i a la descripció una llista amb punters a tots els seus vols. La tercera i última estructura, s'usa a la setena consulta per obtenir a partir del codi IATA d'un aeroport, tots els vols efectuats, és a dir, un map amb clau IATA aeroport i a la descripció una llista de punters a vols.

# ESTRUCTURES A LES CONSULTES

No m'allargaré gaire amb les estructures de les consultes, ja que moltes segueixen el mateix esquema. A la majoria, inicialment s'usa un map que permet saber per cada aerolínia o aeroport o data la suma de retards que ha tingut. En el cas de **countRoutes**, no és necessari perquè es pot usar directament un set de parelles, que a la vegada ordena alfabèticament. També he arribat a utilitzar un multimap, com és el cas de la consulta **mostDelayedAirlines**, on també es podria haver usat un map amb una list.
El segon pas és l'ordenació, ja que ens interessa que quedi ordenat pel camp de la descripció del map usat. Llavors, sorgeix el dilema d'usar un map o bé un set, o pel contrari inserir a una list i usar el mètode d'ordenació de la list. La complexitat d'inserir n vegades a un map o set

seria O(NlogN)[4] i ordenar la list també suposaria un cost O(NlogN). Fins i tot, si es tinguès constancia de si habitualment a les consultes es demanen valors relativament petits, és a dir ordenar 22 de 22000, es podria fer una inserció ordenada en una list. Pel meu cas, acostumo a usar un map o bé un set.

# ESTRUCTURES COM A VEHICLE D'INFORMACIÓ

De la manera que he estructurat les classes, el més adequat seria que, feta una petició a un conjunt, aquest retornés exactament allò que espera, interactuant amb altres classes i basant-se evidentment en les dades que té a mà. Per tant, no em sembla bona idea que, per exemple, un conjunt de Rutes, com és **Routes**, mostri l'equipament més usat d'una companyia, ja que no és una propietat inherent d'un conjunt. Tanmateix, sí que té sentit que retorni una estructura amb l'equipament més usat i qui ho ha demanat que s'encarregui de mostrar-ho, en aquest cas l'acció **manageQueries**. Això suposa la necessitat de crear estructures per retornar aquesta informació.

Per cada consulta he usat diferents estructures[5]:

Predominen els vectors, usats sempre que es coneix la mida que tindrà, és a dir, si es demanen les 5 aerolínies amb més retard, el vector tindrà mida 5 o, si es demanen totes, el vector tindrà la mateixa mida que el map d'ordenació. Amb això faig referència a les consultes tres, quatre, cinc i sis, sempre acompanyades d'un vector de parelles.

En el cas de la segona consulta, el més adient seria una parella amb el nom de la companyia i una list dels avions més usats. Ara bé, el problema és que no es podria comprovar si aquella aerolínia existeix però no ha operat cap ruta, ja que una parella és un objecte i no es pot saber si està buit per tal d'indicar que mai ha usat cap equipament.

Per acabar, la setena consulta rep una list de parelles perquè existeix la possibilitat d'haver-hi més d'una data amb la mateixa quantitat de retards màxim. En aquest cas també es podria haver usat un map amb una list on la clau seria el total de retards i a la list les dates. Com es veu hi ha diferents maneres d'enfocar el problema. Personalment, com que la qüestió és fer un recorregut he preferit guardar-ho en una list.

[4]N és la mida del map, set o list.

[5]La consulta número 1 ja s'ha comentat a l'apartat anterior.