****

**TREBALL FINAL DE GRAU**

**TÍTOL:** Estructures de grafs amb equivalències d’arestes aplicades a l’anàlisi de dades relacionals

**AUTORS:** Rodríguez Navas, Laura

**DATA DE PRESENTACIÓ:** Maig, 2017

**COGNOMS:** Rodríguez Navas **NOM:** Laura

**TITULACIÓ:** Grau en Enginyeria Informàtica

**PLA:** 2010

**DIRECTOR:** José Luis Balcázar Navarro

**DEPARTAMENT:** Ciències de la Computació

**QUALIFICACIÓ DEL TFG**

|  |  |  |
| --- | --- | --- |
|  | **TRIBUNAL** |  |
| **PRESIDENT** | **SECRETARI** | **VOCAL** |

J. M. Merenciano Saladrigues Bernardino Casas Fernández Sergio Sánchez Lopez

##### DATA DE LECTURA: 31 de maig de 2017

###### Aquest Projecte té en compte aspectes mediambientals:  Sí  No

**RESUM**

En aquest treball es vol contribuir a la investigació de les 2-estructures mitjançant l’aportació d’una eina que faciliti el desenvolupament de treballs futurs que es fonamentin en la utilització de la teoria de les 2-estructures per a l’anàlisi de dades relacionals.

La teoria de les 2-estructures proporciona una infraestructura matemàtica per a la descomposició de grafs. Permet representar múltiples grafs en una sola estructura algebraica, una 2-estructura, que es deriva en una descomposició única de 2-estructure més simples.

Durant el treball, s’ha dut a terme un estudi amb dues finalitats:

* El disseny i la implementació d’un paquet de software que analitzi i representi visualment les principals estructures involucrades en la teoria de les 2-estructures.
* La investigació i el desenvolupament de possibles aplicacions de les 2-estructures en l’anàlisi de dades relacionals que formen part de les bases de dades relacionals.

**Paraules clau (màxim 10):**

|  |  |  |  |
| --- | --- | --- | --- |
| Python | Grafs | Ciències de la la Computació | Algorítmica |
| NetworkX | 2-estructures | Anàlisi de dades | Pydot |
| Graphviz | SQLite |  |  |

**ABSTRACT**

This project seeks to contribute to the research of the 2-structures by providing a tool to facilitate the development of future work which is based in the utilization of the theory of 2-structures for the relational data analysis.

The theory of 2-structures provides a mathematical infrastructure for the decomposition of graphs. Allows multiple graphs represented a single algebraic structure, a 2-structure, which results in a unique 2-structures simpler decomposition.

During the project, a study it has carried out with two purposes:

* Design and implementation of a software package to analyse and visualize the main structures representations involved in the theory of 2-structures.
* The research and development of possible applications of 2-structures in the relational data analysis that are part of the relational databases.

**Keywords (10 maximum):**

|  |  |  |  |
| --- | --- | --- | --- |
| Python | Graphs | Computer Science | Algorithmic |
| NetworkX | 2-structures | Data Analysis | Pydot |
| Graphviz | SQLite |  |  |

AGRAÏMENTS

Aquest treball representa el final d'una segona etapa de la meva vida. Que va començar per casualitat quan vaig decidir deixar l'esport d'elit. Durant aquesta segona etapa he tingut la gran sort de conviure amb un grup de companys excepcionals, a l’EPSEVG i també a la FIB. La seva companyia ha fet el camí a recórrer més fàcil i divertit.

No intentaré anomenar-los a tots, ja que són molts i no me’n vull deixar cap. Confio amb que ells ja saben qui són i els hi dono mil gràcies per la seva fantàstica companyia durant aquests anys. Tot i així, estic segura que comprendran que faci una excepció i mencioni de manera especial a la meva gran amiga, i companya de pràctiques durant gairebé tota la carrera, Noemí. Durant el temps que vaig passar a la FIB com el que he passat a l’EPSEVG ha estat un pilar tan en l’àmbit acadèmic com en el personal.

Per motius semblants anomenats anteriorment tampoc faré una llistat de professors, dels que en gran part guardo un bon record y amb els que he après molt. No obstant, he d’agrair la confiança, la llibertat, la motivació, la inspiració i el recolzament rebut per en Jose Luís Balcázar alhora de desenvolupar aquest projecte. També m’agradaria agrair, a en Bernardino Casas per ensenyar-me a programar i a la Neus Català per iniciar-me en el món de l’anàlisi de dades, aptituds bàsiques per a aquest projecte.

Per últim, el més important. No tinc paraules per expressar el meu agraïment per a tot el que els hi dec als meus pares, Lluís i Montse, que han estat al meu costat en els bons moments però també en els dolents. Sense ells, l'imminent enginyera informàtica que ha escrit aquestes línies no seria res.

A tots ells els hi dedico aquest treball.

ÍNDEX

ÍNDEX DE FIGURES

ÍNDEX DE TAULES

ÍNDEX DE CODIS

PART I

INTRODUCCIÓ I PLANIFICACIÓ

1. INTRODUCCIÓ

En aquest capítol es descriuen els elements fonamentals en el que s’emmarca el projecte com poden ser l’abast, els objectius, l’estat de l’art, la metodologia emprada i l’avaluació tecnològica. A més s’indica de forma abreviada el contingut i la distribució de la resta dels capítols de la memòria.

* 1. Abast i objectius

Degut a que el temps disponible pel desenvolupament del treball és limitat, l’abast del treball és el d’analitzar i trobar la millor solució per a la representació visual de les 2-estructures que faciliti la comprensió de l’anàlisi de dades relacionals. A més, també incorpora la investigació de possibles noves funcionalitats que es podrien utilitzar en l’anàlisi de dades relacionals a partir de les 2-estructures.

Per tant, segons l’abast, podem agrupar i resumir els objectius en la consecució de dues fites.

La primera fita del treball consisteix en el disseny i la implementació d’un paquet de software que compleix els requisits següents:

* Permet la creació i la consulta de propietats de la estructura principal que cobreix l’estudi fonamental sobre la teoria de les 2-estructures.
* Dibuixa gràficament la representació d’aquesta estructura principal anomenada anteriorment.
* Integra els components necessaris en un programa interactiu extern (que no s’ha implementat sota l’entorn de desenvolupament del treball) que permet la visualització i l’emmagatzematge d’aquesta estructura principal.

La segona fita té com a propòsit la investigació i el desenvolupament de possibles aplicacions de les 2-estructures en l’anàlisi de les dades relacionals que formen part de les bases de dades relacionals.

Per aquest motiu es pren com a punt de partida el següent objectiu:

* Establiment de teoremes i/o resultats que permetin enfortir l’anàlisi de dades relacionals mitjançant l’ús de les 2-estructures.

Algun dels objectius del treball inclouen la utilització d’una aplicació gràfica.

Aquesta aplicació, no està integrada en l’entorn de programació, és a dir, que s’hi accedeix a partir d’una crida al programari interactiu extern corresponent. De manera que l’usuari ha de sortir de l’entorn de treball per a realitzar les visualitzacions gràfiques de les 2-estructures amb el programari interactiu extern.

A més, un dels objectius principals del projecte, tot i que l’usuari no hi interactua de manera directa, es la implementació d’un algoritme que dibuixi correctament el model de visualització gràfica d’una 2-estructura. Es tracta d’un model basat en DOT [[R](http://www.graphviz.org/doc/info/lang.html)] (llenguatge descriptiu de text pla) que proporciona una manera simple de descriure gràfiques i que les fa més entenedores per a l’usuari.

Tots els models basats en DOT seran creats des de cero i l’algoritme encarregat de crear els models, generarà una disposició dels elements que facilita a l’usuari la tasca d’entendre el model molt ràpidament.

* 1. Estat de l’art
     1. *Matemàtiques preliminars*
        1. *Teoria de conjunts*

La teoria de conjunts és la branca de les matemàtiques que estudia els conjunts [[R](https://ca.wikipedia.org/wiki/Teoria_de_conjunts)].

Un conjunt és una col·lecció d'objectes ben definits.

Exemples:

* El conjunt dels nombres naturals és N = {1, 2, 3, . . .}.
* El conjunt dels enters és Z = {. . . , −3, −2, −1, 0, 1, 2, 3, . . .}.
* El conjunt dels nombres racionals és Q = {m/n : m ∈ Z, n ∈ Z, n ≠ 0}.
* El conjunt dels nombres reals s’anomena R.

Si A i B són conjunts, llavors:

A és un subconjunt de B (A ⊆ B) si cada element de A és també un element de B. Per exemple, el conjunt de nombres parells naturals és un subconjunt del conjunt de tots els nombres naturals. Cada conjunt A és un subconjunt de si mateix, és a dir, A ⊆ A. El conjunt buit és un subconjunt de cada conjunt A, 0 ⊆ A. La relació binària entre els conjunts A i B és un subconjunt A x B.

1. La seva unió es defineix com A ∪ B = {x : x ∈ A o x ∈ B}.
2. La seva intersecció es defineix com A ∩ B = {x : x ∈ A i x ∈ B}.
3. La seva diferència es defineix com A \ B = {x : x ∈ A i x ∉ B}.
4. El producte cartesià entre A i B es defineix com A x B = {(x, y) : x ∈ A i y ∈ B}.
5. El complement de A es defineix com Ac = {x : x ∉ A}.

La relació binària R ⊆ A x A és una relació d’equivalència que sempre compleix les tres condicions següents:

1. R és reflexiva: per a cada element de a ∈ A, tenim que (a, a) ∈ R.
2. R és simètrica: per a cada a i b en A, si (a, b) ∈ R, llavors (b, a) ∈ R.
3. R és transitiva: per a cada a, b i c en A, si (a, b) ∈ R i (b, c) ∈ R, llavors (a, c) ∈ R.

*1.2.1.2 Teoria de grafs*

El precursor de la teoria de grafs va ser Leonhard Euler, que la va iniciar tot intentant resoldre el problema dels set ponts de Königsberg (Euler, 1736) [[R](https://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg)].



Figura 1. Solució d’Euler.

La teoria de grafs és una branca de les matemàtiques i la informàtica que es dedica a l'estudi dels grafs, estructures matemàtiques utilitzades per a modelitzar relacions entre parelles d'objectes.

La teoria de grafs es troba estretament relacionada amb la teoria de les 2- estructures. Concretament, una 2-estructura es pot considerar un graf dirigit complet amb les arestes acolorides.

*1.2.1.2.1 Conceptes bàsics*

Els grafs s’utilitzen per a representar relacions (simples o d’ordre) entre objectes del mateix tipus. Els objectes reben el nom de nodes o vèrtex i les relacions entre ells s’anomenen arestes.

Un graf G es pot definir com la parella G = (V, E) on V és un conjunt d’elements anomenats vèrtexs i E és un conjunt d’elements anomenats arestes. Cada element d’E és una parella de vèrtex diferent.

Existeixen els grafs dirigits i els grafs no dirigits depenen de si les arestes estan orientades o no ho estan.

**Exemple 1**. A la figura 2 veiem una representació gràfica d’un graf no dirigit. En un graf no dirigit les relacions són simètriques: E (u, v) = E (v, u).



Figura 2. Graf no dirigit.

**Exemple 2**. A la figura 3 veiem la representació gràfica de l’exemple 1 però en un graf dirigit. En un graf dirigit les relacions no són simètriques i tenen una direcció: E (u, v) va del vèrtex u al vèrtex v, denotat amb una fletxa u → v.



Figura 3. Graf dirigit.

Els grafs (dirigits o no) poden estar etiquetats o no etiquetats en funció de si les arestes tenen o no informació associada.

Donat un domini V de vèrtexs i un domini E d’etiquetes, es defineix un graf dirigit i etiquetat G com una funció que associa etiquetes a parelles de vèrtexs.

*G ∈ { f: V x V = E }*

Per a un graf no etiquetat la definició és similar. El graf G és una funció que associa un booleà a parelles de vèrtexs.

**Exemple 3**. A la figura 4 veiem la representació gràfica de l’exemple 1 però en un graf ponderat. Un graf ponderat és un tipus particular de graf etiquetat on la informació associada a l’aresta és un nombre real positiu que normalment representa un cost.



Figura 4. Graf ponderat.

El grau d’un vèrtex v, definit com a deg(v), és el nombre d’arestes incidents sobre v.

El camí d’un graf és una seqüència de vèrtex connectats per les arestes. Un camí és un cicle, si comença i acaba al mateix vèrtex. Un camí simple és un camí sense cap vèrtex repetit. Un graf està connectat si hi ha camí entre cada parella de vèrtexs.

* + 1. *Teoria de les 2-estructures*

La teoria de les 2-estructures va ser introduïda per Ehrenfeucht i Rozenberg el 1990.

Aquesta teoria proporciona una infraestructura matemàtica per a la descomposició i la transformació dels grafs. Es tracta d’un formalisme molt potent i robust que permet representar múltiples grafs en una sola estructura algebraica, una 2-estructura, i derivar-ne d’ella una descomposició única en 2-estructures més simples.

Per a l’elaboració d’aquest treball s’ha treballat principalment amb l’estudi previ [R] on s’estableixen els fonaments de la teoria de les 2-estructures i que recopila i amplia el contingut sobre la teoria actual fins a la data de la seva publicació.

La part 'estàtica' de la teoria de les 2-estructures (del capítol 3 al 10) es refereix principalment a la descomposició i a la no-descomposició. Del capítol 3 al 7 es considera la 2-estructura no etiquetada que es defineix per una relació d'equivalència a les arestes. Les 2-estructures no etiquetades representen l’estructura principal del paquet de software desenvolupat.

La teoria de les 2-estructures enriqueix l'estudi de les gramàtiques dels grafs posant a disposició les eines de la teoria de grups i la teoria de la commutació.

*1.2.2.1 Conceptes bàsics*

S'utilitza el nom genèric de "2-estructura" per tal d'emfatitzar que investiguem els sistemes de relacions binàries en un entorn abstracte. Particularment, s’investiguen les representacions jeràrquiques de les 2-estructures a través d'arbres on la relació local o recursiva (quan es defineix en termes de si mateixa o del seu tipus) entre descendents directes d'un node interior es dóna a través d'una estructura binària.

La noció d'una 2-estructura és més general que la noció d'un graf i menys general que la noció d'una estructura relacional.

Una 2-estructura es pot considerar un graf dirigit complet amb les arestes acolorides. I cada color diferent de les arestes una classe d’equivalència entre els nodes del graf.

Una característica molt important de les 2-estructures és que poden des composar-se en 2-estructures més simples. Aquesta descomposició és única per a les 2-estructures i consisteix en trobar subconjunts d’un graf, anomenats clans.

Cada 2-estructura es representa com una estructura en forma d’arbre. En aquest arbre cada node intern que es troba en la 2-estructura representa un element del graf o bé un subconjunt d’aquest.

**Exemple 4**. A la figura 5 veiem una representació gràfica d’una 2-estructura. En ella, els colors de les arestes serveixen per a distingir a quina classe d’equivalència pertany cada clan. Cada rectangle representa un clan i conté un element i un subconjunt d’un graf.



Figura 5. Exemple d’una 2-estructura.

1.3 Metodologia i avaluació tecnològica

*1.3.1 Metodologia*

Per a la organització del treball la metodologia triada, si bé és senzilla, ha resultat molt efectiva en el desenvolupament del treball. Com que era un projecte on l’equip de desenvolupament consistia en una única persona (en aquest cas l’estudiant que realitza el treball), es va decidir seguir una metodologia similar a Scrum. Que és una metodologia àgil pensada sobretot pel desenvolupament de software i l’adaptabilitat de qualsevol canvi com a mitjà per augmentar les possibilitats d’èxit d’un projecte. Una de les principals claus de l'èxit de Scrum és que està basat en la premissa que, durant el desenvolupament dels productes, els clients poden canviar les seves opinions sobre què volen i què necessiten. A més, els imprevistos no presenten una gran dificultat per al desenvolupament dels productes.

Per tant, es va decidir realitzar les reunions necessàries amb el director del treball constantment, amb l’objectiu de facilitar la proposta de solucions i la obtenció de feedback.

Els rols es van assignar de la següent manera:

* Scrum Master: és el que s’encarrega de fer el seguiment d’allò que realitza l’equip de desenvolupament, en aquest cas és el director del treball.
* Product Owner: és qui representa els interessats en el projecte, en aquest cas també és el director del treball.
* L’Equip de Desenvolupament: el que s’encarrega de desenvolupar el producte, en aquest cas és l’estudiant que realitza el treball.

També es va decidir utilitzar el model Kanban per a la organització de les tasques del treball mitjançant l’aplicació web Wrike [[R](https://www.wrike.com/)]. Ja que el model Kanban proporciona un sistema de gestió del procés molt visual que ajuda a la presa de dedicions.

* Wrike: és una eina en línia per a la gestió de projectes i la col·laboració en el treball. Que permet als seus usuaris gestionar i realitzar un seguiment dels projectes, terminis, horaris i altres processos d’un flux de treball. També permet als usuaris col·laborar entre si. Wrike també es pot trobar per a dispositius iOS i Android.

Pràcticament totes les versions del programari compten amb un cicle de treball que actualitza els usuaris de qualsevol activitat realitzada per altres usuaris dels grups de treball. Les característiques socials també estan integrades en el programari. A més també s'integra amb un seguit de diferents productes com: Google Apps, Microsoft Outlook, Microsoft Excel, Microsoft Project, Google Drive, Dropbox, Apple Mail Box, IBM Connections, i altres.



Figura 6. Taulell Kanban a Wrike.

Wrike també incorpora la generació automàtica del diagrama de Gantt. Però per a la simplificació del diagrama es va optar per a importar les tasques al format de fitxer XML [[R](https://ca.wikipedia.org/wiki/Extensible_Markup_Language)], triar les tasques més importants que mostrava i crear un nou diagrama de Gantt amb el programari GanttProject [[R](http://www.ganttproject.biz/)].

Perquè GanttProject està dissenyat seguint les bases del principi KISS. Aquest principi afirma que els sistemes que funcionen millor són els que es mantenen simples, i que s'ha d'evitar qualsevol complexitat afegida que no resulti clarament necessària.

* GanttProject és una aplicació d’escriptori multi plataforma per a la programació i gestió de projectes. La seva funcionalitat principal és la creació de diagrames Gantt. Facilita la exportació dels diagrames a formats de fitxers PNG, PDF i HTML. També s'integra amb Microsoft Project a l'hora d'importar i exportar projectes, i es complementa amb aplicacions que fan servir fulls de càlcul en formats de fitxer CSV. De la mateixa manera, permet la compartició dels projectes amb col·legues mitjançant WebDAV [[R](http://www.webdav.org/)].



Figura 7. Logotip del programari GanttProject.

Per a la implementació i les proves del paquet de software desenvolupat, a mesura que s’anava avançant i es van començar a plantejar els primers diagrames de classes, es va triar la metodologia orientada a objectes.

Les bases d’aquest tipus de metodologia són:

* Tot és un objecte i cada objecte té una identitat pròpia.
* Un programa és un conjunt d’objectes que interactuen entre ells.
* Cada objecte pertany a un tipus d’objecte concret: una classe.
* Objectes del mateix tipus tenen un comportament idèntic.

El mètode d’avaluació proposat per a la implementació es divideix en:

1. Periòdicament es posa a prova la implementació mitjançant un conjunt de jocs de prova que es composen de casos senzills i aleatoris per a assegurar-se de la pròpia correctesa.
2. Per comprovar la correcta visualització de la implementació mitjançant el programari interactiu extern es realitzen proves a nivell d’usuari (beta-testing) per a assegurar-se de que és suficientment intuïtiu i es comprova el funcionament en tots els seus àmbits. Òbviament aquestes comprovacions es realitzen en fases avançades del treball.

*1.3.2 Avaluació tecnològica*

*1.3.2.1 Llenguatges de programació*

El llenguatge de programació triat pel desenvolupament del paquet de software és Python [[R](https://www.python.org/)]. Perquè és un llenguatge molt utilitzat en entorns de computació científica, ja que ofereix un entorn de programació obert i flexible, en el que és fàcil iniciar-se, gràcies a la seva filosofia de disseny, que busca llegibilitat en el codi, i la seva sintaxi, que permet expressar conceptes en menys línies de codi del que seria possible en altres llenguatges, com per exemple Java o C.



Figura 8. Logotip del llenguatge de programació Python.

Python està dissenyat per ser un llenguatge molt visual, i, com a característica principal utilitza el [sagnat](https://ca.wikipedia.org/wiki/Sagnat). Això és poc comú en [llenguatges de programació](https://ca.wikipedia.org/wiki/Llenguatge_de_programaci%C3%B3), ja que molts utilitzen delimitadors. A més, té una llibreria estàndard molt gran i aporta eines ja programades que poden crear una gran varietat de funcions.

Una de les característiques de Python que també s’ha utilitzat és la combinació amb altres tipus de llenguatge, C i C++, i els diferents mòduls per a connectar bases de dades relacionals.

El paquet de software implementat és compatible amb les versions de Python 2 i 3.

*1.3.2.2 Llibreries*

Pel tractament de les bases de dades relacionals es va utilitzar el mòdul sqlite3 [[R](https://docs.python.org/3/library/sqlite3.html)] que proporciona una interfície SQL compatible amb l'especificació DB-API 2.0 (PEP 249 [[R](https://www.python.org/dev/peps/pep-0249/)]).

* SQLite és una llibreria de C que proporciona una base de dades que no requereixen el procés d’un servidor independent i permet l'accés a la base de dades utilitzant una variant no estàndard del llenguatge de consulta SQL. També és possible crear prototips d'una aplicació que utilitza SQLite i després exportar el codi a una base de dades més gran, com PostgreSQL o Oracle.

Per a la creació i la manipulació dels grafs es va utilitzar NetworkX [[R](https://networkx.github.io/)].

* NetworkX és un paquet de programari del llenguatge Python per a la creació, manipulació, i l'estudi de l'estructura, la dinàmica i les funcions de les xarxes complexes. L'audiència potencial de NetworkX inclou matemàtics, físics, biòlegs i informàtics. A NetworkX, els nodes poden ser qualsevol objecte, per exemple, una taula hash (una estructura de dades que associa claus o claus amb valors), una cadena de text, una imatge, un objecte XML, un altre graf, etc.

Proporciona:

* Estructures de dades per a grafs, digrafs i multi-grafs.
* Molts algoritmes de grafs estàndards.
* Generadors de grafs simples, grafs aleatoris, etc.
* Llicència BSD de codi obert.
* Una interfície d’algoritmes numèrics existents i codificats en C, C++ i Fortran.
* La capacitat de tractar grans quantitats de dades.
* Fiabilitat amb més de 1800 proves d'unitat.

Per a generar tots els subconjunts possibles d’un graf, que té un alt cost de processament, i altres recorreguts, es va pensar en utilitzar la llibreria itertools de Python [[R](https://docs.python.org/3/library/itertools.html)]. Ja que proporciona funcions per a recórrer molt eficients. La llibreria s’inspira en construccions APL [[R](https://en.wikipedia.org/wiki/APL_(programming_language))], Haskell [[R](https://www.haskell.org/)] i SML [[R](https://en.wikipedia.org/wiki/Standard_ML)].

Una vegada creats els grafs i les 2-estructures es necessitava una eina de visualització per a observar els resultats. La llibreria triada va ser Pydot [[R](https://github.com/erocarrera/pydot)].

* Pydot és una interfície per a Graphviz [[R](https://pypi.python.org/pypi/graphviz)], que pot analitzar i bolcar en llenguatge DOT, llenguatge que utilitza el programari Graphviz [[R](http://www.graphviz.org/)] i està escrit en Python pur. A més, NetworkX pot convertir els seus grafs en aquest format.

El paquet de software implementat és compatible amb diferents sistemes operatius (Windows, OS X i Ubuntu) i amb diferents versions de Python com s’ha comentat anteriorment. Per a aconseguir l’objectiu s’han utilitzat les diferents llibreries següents: os [[R](https://docs.python.org/3/library/os.html)], sys [[R](https://docs.python.org/3/library/sys.html?highlight=sys#module-sys)], subprocess [[R](https://docs.python.org/3/library/subprocess.html?highlight=subprocess#module-subprocess)] i six [[R](https://pythonhosted.org/six/)], amb les seves funcionalitats respectives.

* Mòdul os: llibreria que permet l’ús de funcionalitats pròpies d’un sistema operatiu.
* Mòdul sys: llibreria que permet l'accés a algunes variables que introdueixen els usuaris utilitzades per un intèrpret d’ordres, que té la capacitat de traduir-les, i les funcions que interactuen amb aquest intèrpret d’ordres.
* Mòdul subprocess: llibreria que té la intenció de reemplaçar la llibreria os.
* Mòdul six: llibreria de compatibilitat entre Python 2 i 3.

*1.3.2.3 Programari*

Les particularitats de la representació de la descomposició d’un graf en una 2-estructura i la creació d’una 2-estructura va provocar la cerca d’un programa extern per a la visualització. Aquesta eina és Graphviz.

* Graphviz és un programari de codi obert de visualització gràfica que converteix els programes de disseny en format DOT a diagrames o imatges.



Figura 9. Logotip del programari Graphviz.

**Exemple 5.** A la figura 10 es pot veure el resultat de la conversió que efectua el programari Graphviz. A partir d’un fitxer dissenyat en format DOT (a l’esquerra de la imatge) es crea la visualització corresponent (a la dreta de la imatge).



Figura 10. Exemple de conversió del programari Graphviz.

Durant el desenvolupament es va utilitzar l’entorn de programació integrat i multi plataforma Pycharm [[R](https://www.jetbrains.com/pycharm/)] i es va mantenir una copia actualitzada del paquet de software en un emmagatzematge remot amb GitHub [[R](https://github.com/)] [(https://github.com/lrodrin/TFG.git]((https:/github.com/lrodrin/TFG.git)).

* Pycharm està desenvolupat per la companyia Jetbrains, basat en IntelliJ IDEA [[R](https://www.jetbrains.com/idea/)], l’entorn de programació integrat de la mateixa companyia però enfocat cap al llenguatge de programació Java (que ja havia utilitzat anteriorment durant l’assignatura de Projecte de Programació) i la base d’Android Studio [[R](https://developer.android.com/studio/index.html)].

Principalment permet escriure bon codi (seguint les normes PEP-8 [[R](https://www.python.org/dev/peps/pep-0008/)]), ràpid i eficient, gràcies a les funcions que incorpora. Ofereix compilació de codi intel·ligent, correccions de codi, ressaltat d'error en temps real i completesa de codi, juntament amb la re factorització de codi automatitzat i capacitats de navegació. Proporciona suport de primera classe per a Python, JavaScript, CoffeScript, TypeScript, CSS, i altres. També utilitza la cerca intel·ligent per saltar a qualsevol classe, arxiu o símbol, o fins i tot qualsevol finestra d'acció de l’entorn de desenvolupament o eina.



Figura 11. Logotip de Pycharm.

.

Pycharm pot integrar el sistema de control de versions Git [[R](https://git-scm.com/)], que utilitza l’ emmagatzematge remot GitHub. Aquest fet va afavorir la manipulació alhora d’utilitzar el control de versions.

* Git és un programari de sistema de control de versions dissenyat per Linus Torvalds, pensat per a l'eficiència i la confiabilitat del manteniment de versions d'aplicacions amb una enorme quantitat de fitxers de codi font. Proporciona una gestió eficient de projectes grans, gràcies a la rapidesa de gestió de diferents arxius, entre altres millores d'optimització en la velocitat d'execució.

Entre les característiques més rellevants de Pycharm trobem:

* Forta incidència en la no-linealitat dels canvis.
* Gestió distribuïda. Els canvis s'importen com a ramificacions, i es poden barrejar de la manera en què ho fa una ramificació en l'emmagatzematge local.
* Els magatzems d'informació poden publicar-se per a HTTP [R], FTP [R], SSH [R], rsync [[R](https://rsync.samba.org/)] o mitjançant un protocol natiu, a part de ser possible emular CVS [[R](http://cvs.nongnu.org/)].



Figura 12. Logotip de Git.

Per a accelerar el processament de les dades relacionals es va utilitzar el programari apriori [[R](http://www.borgelt.net/apriori.html)].

* apriori és un programa per a trobar regles d'associació i el conjunt d'elements més freqüents a partir de l'algoritme de mineria influent apriori [[R](https://en.wikipedia.org/wiki/Apriori_algorithm)] que duu a terme una cerca en amplada. La cerca en amplada és un algoritme per recórrer o buscar elements d’un graf (normalment utilitzant un arbre). Intuïtivament, es comença a l'arrel (escollint algun node com a element arrel en el cas d'un graf) i s'exploren tots els veïns d'aquest node. A continuació per a cada un dels veïns s'exploren els seus respectius veïns adjacents, i així fins que es recorri tot l'arbre.

Es va triar el programari apriori perquè és un programa molt ràpid ja que utilitza un arbre definit anteriorment per a organitzar el conjunt d'elements a cercar.

*1.3.2.4 Sistemes Gestors de Base de Dades (SGDB)*

En la última fase del treball es va dur a terme la investigació de les possibles aplicacions de les 2-estructures en l’anàlisi de dades relacionals. I per a l’anàlisi i la gestió d’aquest tipus de dades es van triar les bases de dades SQLite.

* SQLite és una base de dades relacional continguda en una llibreria escrita en C. Incorpora un motor de bases de dades SQL independent. Que a diferència d’altres bases de dades relacionals, no és un sistema que funciona amb el paradigma client-servidor, sinó que s’integra a dins d’altres programes. Les bases de dades SQLite s'emmagatzemen en un fitxer (normalment amb extensió .sqlite o .db) que conté tant la definició de l'estructura de les dades com les dades mateixes.



Figura 13. Logotip de SQLite.

Bàsicament es va triar aquest tipus de base de dades relacional perquè el codi per a SQLite és de domini públic, multi plataforma i és la base de dades de major desplegament al món. I alhora de començar una línia d’investigació és molt important la senzillesa i la generalització que aporta.

1.4 Estructura del document

El treball s’estructura en dues parts: memòria i apèndix. A la memòria es sintetitza les aportacions realitzades per l’estudiant. A l’apèndix es dóna una guia d’instal·lació i ús de l’entorn d’anàlisi i visualització del paquet de software.

El contingut dels capítols de la memòria és el següent. En el capítol 2 es mostra la planificació que s’ha dut a terme durant el desenvolupament del treball. En el capítol 3 i 4 es presenta l’anàlisi econòmic i l’anàlisi de sostenibilitat del treball. En el capítol 5 es mostra l’anàlisi i el disseny del paquet de software desenvolupat.

En els capítols 6, 7, 8 i 9 es documenta la implementació del paquet de software desenvolupat. Concretament, en el capítol 6 es descriuen les dades d’entrada que es necessiten. En el capítol 7 es descriu la generació dels grafs. En el capítol 8 s’explica com funciona la descomposició dels grafs en 2-estructures. I finalment, el capítol 9 s’associa explícitament a les 2-estructures.

En l’últim capítol, el 10, es resumeix el treball realitzat, es dóna una valoració personal d’aquest i es relaciona amb les maneres possibles de continuar-lo.

1. PLANIFICACIÓ TEMPORAL

2.1 Descripció de les tasques

Inicialment el treball es va planificar fins al torn de presentació de febrer. No obstant, per la falta de temps, es va prorrogar fins al torn de presentació de maig.

Per tant, la planificació del treball s’ha estès de la següent manera:

Fase 1 (Fase inicial): des del 11/07/2016 al 31/10/2016 al diagrama de Gantt. Fase d’entrar en contacte amb l’entorn de desenvolupament i valorar les diferents opcions i algoritmes a implementar. L’anàlisi i el disseny de la implementació de la implementació estan inclosos en aquesta fase.

Fase 2 (Fase d’implementació bàsica): des del 2/11/2016 al 31/01/2017 al diagrama de Gantt. Fase on, un cop decidits els algoritmes, les llibreries de visualització i altres detalls, es comencen a implementar els algoritmes bàsics sobre unes dades mínimament realistes en un entorn de visualització senzill.

Fase 3 (Fase d’implementació avançada): des del 1/02/2017 al 31/03/2017 al diagrama de Gantt. Fase on, una vegada tenim la base, podem realitzar el procediment per acabar d’implementar els algoritmes definitius del treball i la integració d’aquests amb les bases de dades relacionals.

Fase 4 (Fase d’adaptació): des del 1/04/2017 al 15/04/2017 al diagrama de Gantt. Fase on, ja desenvolupats els algoritmes, els adaptem per a que siguin més específics i realistes de cara als aspectes de l’anàlisi de dades relacionals. El test i les proves de la implementació estan inclosos en aquesta fase.

Fase 5 (Fase d’optimitzacions i millores): des del 16/04/2017 al 30/04/2017 al diagrama de Gantt. Acabat el programari principal, es busquen tot tipus d’optimitzacions per a millorar el temps i els resultats del paquet de software.

2.2 Diagrama de Gantt

Aquí s’observa el diagrama de Gantt:



Figura 14. Diagrama de Gantt de la planificació temporal.

1. GESTIÓ ECONÒMICA I PRESSUPOST

3.1 Identificació de costos

Podem dividir els costos involucrats d’aquest treball en: costos de recursos humans, de maquinari i de programari.

3.2 Recursos humans

Pel desenvolupament d’aquest treball es necessita com a mínim un/a programador/a que tingui coneixement previ en ciències de la computació i tingui per tant el coneixement sobre la teoria de grafs i sobre la cerca i l’anàlisi d'informació massiva.

El sou brut anual d’un perfil d’aquest estil, amb una titulació superior en enginyeria informàtica, pot ser aproximadament uns 32.000 €. Sense tenir en compte els extres, aproximadament, uns 2.667 € al mes.

Per calcular els costos reals que suposa pel contractant hem d’afegir-hi les cotitzacions socials del 29,70% (23,60% de contingències comuns, 5,50% d'atur i 0,60% de formació professional).

|  |  |
| --- | --- |
| **Costos pel contractant** | **Preus** |
| Sou brut mensual | 2.667 € |
| +23,60% Cotitzacions de contingències comuns | 629,41 € |
| +5,50% Cotitzacions d'atur | 146,68 € |
| +0,60% Cotitzacions de formació professional | 16 € |
| **Total** | 3.459,09 € |

Taula 1. Costos que suposa el treballador pel contractant.

Pel que fa al/a treballador/a se li descompta del sou brut mensual: el 6,35% de cotitzacions socials (4,70% de contingències comuns, 1.55% d'atur, 0,10% de formació professional) i el 20% de IRPF.

|  |  |
| --- | --- |
| **Costos pel treballador** | **Preus** |
| Sou brut mensual | 2.667 € |
| -4,70% Cotitzacions de contingències comuns | -125,35 € |
| -1,55% Cotitzacions d'atur | -41,34 € |
| -0,10% Cotitzacions de formació professional | -2,67 € |
| -20% de IRPF | -533,40 € |
| **Total** | 1.964,24 € |

Taula 2. Sou net que rep el treballador.

3.3 Recursos de maquinari

Els recursos de maquinari necessaris per a aquest treball són molt limitats. Bàsicament es necessita un únic ordinador de gama alta per a la realització de les proves dels algoritmes, el control de versions extern, la gestió i l’administració d’una base de dades i per a l’execució del programari corresponent.

No obstant, com que ja es disposava d’un ordinador de gama alta, els costos de maquinari són mínims i es resumeixen de la següent manera:

|  |  |
| --- | --- |
| **Maquinari** | **Preus** |
| Ordinador de gama alta | (1.200 €) 0 € |
| **Total** | 0 € |

Taula 3. Preu del maquinari.

3.4 Recursos de programari

Tots els recursos de programari que s’utilitzen no comporten cap despesa. Fins i tot, el programari que normalment és de pagament, s’ha pogut demanar la llicència estudiant durant un any i s’ha utilitzat gratuïtament. També es tenen en compte les dades relacionals que s’utilitzen en aquest treball, però que tampoc tenen cap cost degut a que són proporcionades pel director del treball.

|  |  |
| --- | --- |
| **Programari** | **Preus** |
| Llibreries (NetworkX, Pydot, etc.) | 0 € |
| Programari Graphviz | 0 € |
| Programari Pycharm Professional | (199 €/any) 0 € |
| Aplicació Web Wrike Professional | (588 €/any) 0 € |
| Programari Astah Professional | (57 €/any) 0 € |
| Control de versions GitHub | 0 € |
| Data Sources | 0 € |
| **Total** | 0 € |

Taula 4. Preu del programari.

3.5 Resum de costos

Finalment, podem resumir els diferents costos mensuals de recursos humans, de maquinari i de programari de la següent manera:

|  |  |
| --- | --- |
| **Concepte** | **Preus** |
| Salaris | 3.459,09 € |
| Amortitzacions de maquinari | 0 € |
| Manteniment de llicències de programari | 0 € |
| **Total** | 3.459,09 € |

Taula 5. Resum de costos mensuals.

I el cost total d'aquest treball, si tenim en compte que es realitza durant 11 mesos, aproximadament, es resumiria de la següent manera:

|  |  |
| --- | --- |
| **Concepte** | **Preus** |
| Salaris | 38.049,99 € |
| Amortitzacions de maquinari | 0 € |
| Manteniment de llicències de programari | 0 € |
| **Total** | 38.049,99 € |

Taula 6. Resum de costos totals.

S'ha de tenir en compte que el cost real d'aquest projecte, al estar realitzat dins de l’àmbit d'un Treball Final de Grau de modalitat A, seria nul en el concepte de salaris.

1. SOSTENIBILITAT I COMPROMÍS SOCIAL

4.1 Valoració de la sostenibilitat del treball

La viabilitat econòmica d’aquest treball és alta, principalment perquè els recursos a invertir són mínims i si el treball acaba sent utilitzat per alguna empresa, la devolució de la inversió resultant seria elevada. A més els costos de materials són mínims, ja que és un treball purament de programació, i el material que s’ha utilitzat ja s’havia adquirit anteriorment. Per aquest motiu, pel fet de reutilitzar material existent, la valoració ambiental és molt alta. I una vegada s’acabi la vida útil del material d’aquest projecte es pot seguir reutilitzant, per exemple donant-l’hi a la associació de la UPC, Tecnologia per a Tothom (TxT) [[R](https://txt.upc.es/)], que permet reutilitzar ordinadors antics per a donar-los a gent més necessitada.

La viabilitat social d’aquest treball també és bona, ja que amb la seva aplicació es podrà millorar l’eficiència de l’anàlisi de dades relacionals, i per tant suposarà la millora tant com per a les empreses que perden molt de temps amb un munt de dades per a analitzar com pels treballadors que les analitzen.

En els següents apartats es detallen els aspectes de sostenibilitat econòmics, socials i ambientals responent un conjunt de preguntes.

4.2 Aspectes econòmics

* *El cost del treball ho faria viable si hagués de ser competitiu?*

Sí. Perquè només es necessita un sol programador/a amb coneixement de ciències de la computació. I el maquinari i el programari són limitats o directament gratuïts com s’ha comentat a l’anàlisi econòmic. No obstant, s’hauria de comptar amb la col·laboració d’alguna empresa del sector Big Data que tingui accés a més dades. Cosa que faria augmentar el cost del treball. Però, tot i així seria viable.

* *Es podria realitzar un treball similar en molt menys temps o amb molts menys recursos, i per tant menor cost?*

Es segur que amb més recursos, com per exemple amb més programadors i els seus propis ordinadors es podria realitzar aquest treball més ràpidament. Però és difícil reduir el cost en general, ja que, com s’ha comentat anteriorment els costos del treball són mínims.

* *El temps dedicat a cada tasca és proporcional a la seva importància (s'ha dedicat molt de temps a desenvolupar parts del treball que podien haver estat reutilitzades amb tecnologies / projectes / coneixements existents)?*

Totes les tasques desenvolupades en aquests treball no tenen una base ja existent, s’ha portat a terme un treball d’investigació. I s’ha dedicat més temps del normal en general.

4.3 Aspectes socials

* *Hi ha una necessitat real del producte / servei?*

L’anàlisi de les dades relacionals per part de les empreses “no és un treball futur”, sinó que és ja essencial per a la seva estratègia de negoci i cap empresa, independentment de la seva mida, n’hauria de quedar fora, donat que s’ha demostrat que és un element clau que permet prendre millors decisions, tant estratègiques com en el dia a dia, fins al punt de anticipar-se al mercat. Amb aquest treball es vol millorar l’anàlisi d’aquestes dades i l’adaptació a la situació actual de forma molt més eficient.

* *Satisfer aquesta necessitat, millora la qualitat de vida dels consumidors?*

Una major quantitat de dades implica més vulnerabilitat per als consumidors ja que n’augmenta el perill d’ús maliciós. I protegir el consumidor d’això és fonamental.

A més, l’ús de l’anàlisi de dades en camps com en la fixació de preus, una tendència a l’alça, pot servir per equilibrar el preu de venda en temps real en funció de l’oferta i la demanda. Com a conseqüència tindríem preus més adients pels consumidors.

4.4 Aspectes ambientals

* Quins recursos es poden reaprofitar d'altres treballs?

La majoria del maquinari que s’utilitza en aquest projecte no s’ha adquirit per a la realització del mateix.

* S’ha tingut en compte el desmantellament del procés/producte al medi ambient? Si és un producte, s’han tingut en compte en el seu disseny criteris de facilitació de les seves posterior reciclatge?

Una vegada acabi la vida útil dels ordinadors utilitzats durant el treball es poden donar a l‘associació per a la reutilització dels ordinadors de la UPC anomenada TxT (Tecnologia per a Tothom) que readaptaran els ordinadors per a poder distribuir-los a persones que no en tenen.

**PART II**

**IMPLEMENTACIÓ**

1. ANÀLISI I DISSENY DE LES 2-ESTRUCUTRES

L’anàlisi i el disseny realitzat per a la implementació es comenta breument a continuació. Concretament, el model de dades de les 2-estructures1 i les decisions de disseny adoptades.

A la figura 15 es mostra el diagrama de classes amb les representacions de les principals estructures i les seves relacions2.



Figura 15. Diagrama de classes amb el modelat i el disseny de les 2-estructures.

La classe Data gestiona i adequa les dades relacionals d’entrada per a crear el Graph.

La classe Graph crea el graf:Graph amb les dades relacionals generades per la classe Data. El graf resultant s’utilitza per a la construcció de la 2-estructura:Structure.

La creació de la 2-estrcutura: Structure es pot dividir en tres parts:

* Inicialització del graf:Graph.
* Representació de les classes d’equivalències del graf:Graph.
* Exportació del graf:Graph.

Des del primer moment es va pensar en representar les classes d’equivalència entre les arestes del graf:Graph, pintant les arestes de diferents colors. Perquè el resultat final que volem sigui el més visual i intuïtiu possible.

El graf:Graph pot ser de tipus pla (o planar), pla (o planar) amb llindar, lineal o exponencial. El tipus de graf: Graph amb el que es crea la 2-estructur:Structure a indica el tipus de la 2-estructura: Structure. Per tant, el graf:Graph i la 2-estructura:Structure tenen la mateixa tipificació.

La classe Subset genera tots els subconjunts possibles que del graf:Graph a partir d’ell.

I per a cada parella de valors <Graph, Subset> existeix un únic clan:Clan, el qual constitueix la descomposició del graf:Graph i representa la classe Clan. La descomposició del graf:Graph està formada per un número finit de subconjunts d’aquest, segons si tenen la mateixa classe d’equivalència o no.

L’ús dels clans millora l’eficiència de les consultes sobre les propietats de les 2-estructures perquè es divideix el graf:Graph en seccions més petites.

Com que un clan: Clan pot ser trivial o primer, tipus de clans que s’expliquen més endavant, es generalitza amb aquestes dues subclasses a la figura 15, Trivial i Primal. Es va prendre aquesta decisió perquè una 2-estructura:Structure bàsicament està formada de clans trivials i primers.

La classe Trivial representa els clans de longitud u i els clans que contenen tots els elements del graf:Graph. A més, un clan:Clan trivial sempre es considera que també és un clan:Clan primer. I la classe Primal representa els clans que no experimenten superposició. Propietat que compleixen tots els clans primers i que s’explica més detalladament en el capítol 8 d’aquest document.

La classe Structure crea la 2-estructura:Structure que està basada en la descomposició del graf:Graph en clans. Com s’ha comentat anteriorment, el graf:Graph i la 2-estructura:Structure tenen la mateixa tipificació, i per tant la 2-estructura:Structure també pot ser de tipus pla (o planar), pla (o planar) amb llindar, lineal o exponencial.

Finalment la classe Interface genera la visualització del graph:Graph i la de la 2-estructura:Structure amb l’ajuda del programari interactiu extern anomenat Graphviz.

5.1 interacció del sistema per a la visualització

Interacció del sistema per a la visualització dels grafs i les 2-estructures.

Preguntes del main.

Es crida a Graphviz.

A continuació es mostra el diagrama de classes complet amb les representacions de les principals estructures, les seves relacions i operacions:



Figura 16. Diagrama de classes complet amb el modelat i el disseny de les 2-estructures.

Durant la fase d’optimització i millores es va comprovar que la unitat central de processament, durant l’execució del paquet de software amb grans grups de dades anava molt lenta. Per aquest motiu es modificar el disseny anteriorment esmentat.

La classe Subset enlloc de generar tots els subconjunts possibles del graf:Graph, generarà els subconjunts més freqüents possibles del graf:Graph.

A la figura 17 es mostra el nou diagrama de classes amb les representacions de les principals estructures i les seves relacions2.



Figura 17. Nou diagrama de classes amb el modelat i el disseny de les 2-estructures.

En aquest nou model de disseny, la classe Data gestiona i adequa les dades relacionals d’entrada per a la classe Subset que generarà els subconjunts més freqüents possibles del graf:Graph.

1 Veure l’estat de l’art per a una petita introducció a les 2-estructures.

2 S'han omès els atributs i les operacions de les classes per claredat.

A continuació es mostra el nou diagrama de classes complet amb les representacions de les principals estructures, les seves relacions i operacions:



Figura 18. Nou diagrama de classes complet amb el modelat i el disseny de les 2-estructures.

1. Dades d’entrada

Alhora de treballar amb les dades relacionals, que necessiten les bases de dades relacionals amb les que s’implementa el paquet de software, podem dividir-les en tres tipus de fitxers. O sigui que només es permet la càrrega de dades relacionals pel paquet de software desenvolupat dels tipus de fitxers següents:

* Fitxers de tipus ARFF [[R]](http://www.cs.waikato.ac.nz/ml/weka/arff.html)
* Fitxers de tipus TXT
* Fitxers de tipus DB [[R](https://www.sqlite.org/fileformat.html)]

Els fitxers de dades contenen les dades relacionals necessàries per a la creació de les taules SQLite, que s’utilitzen per a la creació dels grafs i també per a la generació de tots o els subconjunts més freqüents de cada graf. I conseqüentment per a cada 2-estructura. Cada fitxer de dades és equivalent a una taula SQLite.

Es van triar els fitxers de tipus ARFF i BD perquè són dos formats molt utilitzats en el camp de la mineria de dades [[R](https://en.wikipedia.org/wiki/Data_mining)]. Els fitxers de tipus TXT també s’utilitzen però la seva elecció va lligada a la simplicitat i els pocs errors que aquest tipus de fitxers provoquen.

* 1. Fitxers ARFF

Els fitxers ARFF van ser desenvolupats pel Projecte d'aprenentatge automàtic, en el Departament de Ciències de la Computació de la Universitat de Waikato (Nova Zelanda), per a utilitzar-los amb el programari d'aprenentatge de màquina Weka [[R](http://www.cs.waikato.ac.nz/~ml/)].

Un fitxer ARFF (Attribute Relation File Format) és un arxiu de text ASCII que descriu una llista d'instàncies que comparteixen un conjunt d'atributs. L'estructura del fitxer és molt senzilla i es divideix en 3 seccions: @relation, @attribute i @data.

* *@relation <relation-name>*

Tots els fitxer ARFF han de començar amb aquesta declaració a la primera línia ja que no es poden deixar línies en blanc al inici del fitxer. *<relation-name>* és una cadena de caràcters que conté el nom que se li vol donar a *@relation*.

* *@attribute <attribute-name> <datatype>*

En aquesta secció s’inclou una línia per a cada atribut (o columna equivalent a la taula SQLite) que es vulgui incloure en el conjunt de dades, indicant el nom i el tipus de dada. Amb *<attribute-name>* es proporciona el nom de l'atribut, que ha de començar per una lletra. I amb *<datatype>* es representa el tipus de dada per a aquest atribut (o columna equivalent a la taula SQLite).

El tipus de dada de *<datatype>* pot ser:

* Numèric. Nombres reals o enters.
* string (text).
* date [*<date-format>]* (data). Format de la data, que és del tipus "yyyy-MM-dd'T'HH: mm: ss" (ISO 8601 [[R](https://tools.ietf.org/html/rfc3339)]).
* *<nominal-specification>*. Llista de possibles valors: *{<nominal-name1>, <nominal-name2>, <nominal-name3>, ...}.*

Els tipus de dades numèric, text i data són sensibles a les majúscules. I el tipus de dada text és molt útil en aplicacions de mineria de dades, ja que permet crear conjunts de dades a partir de diferents cadenes d’atributs.

* *@data*

En l’ultima secció s’inclouen les dades pròpiament dites. Les dades de cada columna de la taula SQLite es troben separades per comes i totes les files (cada línia del fitxer a partir del nom de secció @data és equivalent a una fila de la taula SQLite) han de tenir el mateix nombre de columnes, el número total de declaracions *@attribute* de la secció anterior.

Si no es disposa d'alguna dada, es col·loca un signe d'interrogació (?) en el seu lloc. I el separador de decimals dels valors numèrics ha de ser obligatòriament un punt.

**Exemple 6.** A la figura 19 es pot veure el format d’un fitxer ARFF apte per a l’entorn de proves del paquet de software desenvolupat.



Figura 19. Fitxer ARFF.

El els fitxers ARFF també es poden afegir línies que comencin amb el caràcter % que representen comentaris.

Totes les declaracions (@relation, @attribute i @data) són sensibles a majúscules.

6.2 Fitxers TXT

Els fitxers TXT són documents de text estàndard que contenen text sense format. Això fa que siguin reconeguts per a qualsevol programa de processament d'edició de text.

A causa de la seva simplicitat, els fitxers de text s'utilitzen molt sovint per a l'emmagatzematge d'informació. Ja que eviten alguns dels problemes trobats en altres formats de fitxer, com ara l’ordre de bytes (*endianness* [[R](https://www.google.es/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=0ahUKEwjcw6v1i7vTAhULVxQKHcq-AfMQFggmMAE&url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FEndianness&usg=AFQjCNGSFyerytQ4VYCGNO0RELDc6-lqbg)]), els bytes de farciment, o diferències en el nombre de bytes en una paraula màquina.

Un desavantatge dels fitxers de text és que en general tenen una entropia baixa. Això vol dir que la informació ocupa més espai d'emmagatzematge del necessari.

**Exemple 7.** A la figura 20 es pot veure el fitxer ARFF de l’exemple 6 però en format TXT apte per a l’entorn de proves del paquet de software.



Figura 20. Fitxer TXT.

A la primera línia del fitxer, cada cadena de caràcters representa una columna de la taula SQLite. I la resta de línies contenen les dades per a cada fila de la taula SQlite. Cada fila està formada per parelles de valors <columna:dada (columna, fila)>.

6.3 Fitxers DB

Els fitxers DB tenen un format de fitxers de bases de dades genèriques. La informació que emmagatzema un fitxer d’aquest tipus normalment es composa d’una sèrie de taules, els camps d’aquestes taules i dades per a aquests camps. Una vegada s’han emmagatzemat les dades, la informació és organitzada d'acord amb el model de dades corresponent. El model d'estructura del treball és el model de dades relacional [[R](https://en.wikipedia.org/wiki/Relational_model)].

Una característica molt rellevant és que es poden exportar a un altre format. Al format CSV [[R](http://fileformats.archiveteam.org/wiki/CSV)]. El format CSV és molt senzill i els fitxers amb aquest format bàsicament s’utilitzen per a representar dades en forma de taula. Els camps de la taula de dades es troben separats per comes. I igual que amb els fitxers TXT, són reconeguts per a qualsevol programa de processament d'edició de text.

1. GRAFS

Una 2-estructura es pot considerar un graf dirigit complet amb les arestes acolorides. Per tant, tot graf desenvolupat en el paquet de software és complet.

Un graf és complet quan entre totes les parelles de vèrtexs del graf existeix una aresta.

Suposant que |E| és el nombre total d’arestes d’un graf podem afirmar que:

* |E| = n · (n-1) si el graf és dirigit.
* |E| = n · (n-1) / 2 si el graf no és dirigit.

**Exemple 8.** A la figura 21 es pot observar un graf complet i no dirigit de 5 nodes. L’etiquetatge dels nodes és de 0 a n-1. I el nombre total d’arestes és 10 (5 · (5-1) / 2).



Figura 21. Graf complet.

A més, tots els grafs desenvolupat del paquet de software també compleixen la localitat de Gaifman [R]. A continuació en podem veure un exemple.

**Exemple 9**. Considerant una base de dades que conté la taula T1 següent:



Taula 7. Taula T1.

El graf de Gaifman o també anomenat graf primer per a aquesta taula és el següent graf no dirigit:

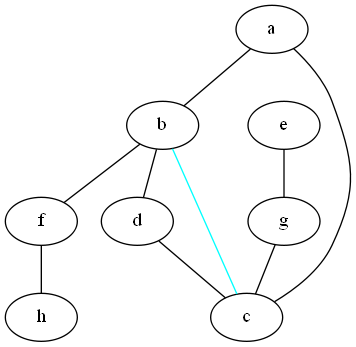


Figura 22. Graf de Gaifman.

En el graf de la figura 22, per a cada parella de valors hi ha una aresta entre ells si es troben junts en una fila de la taula T1. Per exemple, a i b es troben junts a la primera fila, i per tant existeix una aresta entre ells en el graf. D'altra banda, no hi ha cap fila que contingui la relació de nodes a i g, i per tant no hi ha cap aresta entre ells en el graf. L’aresta b-c del graf es dibuixa d’un color diferent ja que el valor b-c es troba repetit a la taula T1.

Cada color de les arestes del graf representa una classe d’equivalència. En el graf de la figura 22 existeixen dues classes d’equivalències, ja que hi ha dos colors diferents.

7.1 creació dels grafs

*7.1.1 Inicialització dels grafs*

Inicialment a partir de la llibreria NetworkX es crea un graf buit sense nodes i arestes. Tots els seran acíclics [R].

I donada una taula SQLite, ja anteriorment creada a partir d’un dels tipus de fitxers que conté les dades d’entrada, es seleccionen les dades per a afegir-les als nodes i a les arestes del graf. Cada element d’una columna de la taula representa un node del graf i per a cada parella de valors que contenen les files es crea una aresta entre els dos valors.

**Exemple 10.** Considerant la taula T2 següent:



Taula 8. Taula T2.

Es seleccionen les files de la taula per a la inicialització d’un graf. Cada element de les files (obviant la repetició dels nodes) representa un node del graf. Per tant, els nodes del graf són: a, c, b, f, d i e.

A la figura 23 s’observa la inicialització del graf resultant. On totes les arestes de cada parella de nodes inicialment són discontínues.



Figura 23. Inicialització del graf de l’exemple 10.

*7.1.2 Classes d’equivalència de les arestes d’un graf*

*7.1.2.1 Definició*

Una classe d’equivalència és el nombre de parelles de valors, de les dades relacionals que conté una taula SQLite, que són repetides. Cada classe d’equivalència es representa mitjançant les arestes d’un graf amb un color diferent. O sigui que, les classes d’equivalències estan representes en un graf per diferents colors. I dues arestes d’un graf tindran el mateix color si pertanyen a la mateixa classe d’equivalència.

Donada una aresta (u, v) es diu que és simètrica si i només sí l’aresta (v, u) pertany a la mateixa classe d’equivalència. Sinó es diu que és antisimètrica. Així doncs, una 2-estructura és simètrica si totes les seves arestes són simètriques. I una 2-estructura és antisimètrica si totes les seves arestes són antisimètriques.

**Exemple 11.** El graf que representa una 2-estructura, que es mostra a la figura 24, és una 2-estructura simètrica. En ella existeixen tres classes d’equivalències: la negra, la blava i la vermella.



Figura 24. Graf que representa una 2-estructura.

Notar que a l’exemple 11, l’element C es relaciona de la mateixa manera amb els elements A, B, D i E (arestes de color negre). Això vol dir que, aquests elements pertanyen a la mateixa classe d’equivalència. També ho podem observar amb l’element B que es relaciona de la mateixa manera amb els elements A, D i E (arestes de color vermell). O amb l’element A que ho fa amb els elements E i D (arestes de color blau).

*7.1.2.2 Algoritme “labeledEdges”*

L’algoritme que acoloreix les arestes dels grafs segons la classe d’equivalència a la que pertanyen s’anomena *“labeledEdges”*. Aquest algoritme compte el número d’equivalències de cada parella de valors d’una taula SQLite. I per a cada aresta del graf les etiqueta amb el nombre total d’equivalències trobades.



Codi 1. Algoritme *“labeledEdges”*.

*7.1.3 Exportació dels grafs*

Una vegada s’han creat els grafs, s’exporten al format DOT del que fa ús el programari interactiu extern (Graphviz) mitjançant la llibreria NetworkX i el seu submòdul Pydot conjuntament.

La funció nx.nx\_pydot.write\_dot(graf:Graph, nom del graf amb extensió .dot), dibuixa els grafs amb el format DOT de Graphviz.

*7.1.3.1 Fitxers DOT*

Els fitxers DOT es composen d’un llenguatge de descripció de grafs en text pla. Normalment compten amb les extensions de fitxer gv o dot. I per defecte, aquests fitxers assumeixen la codificació de caràcters UTF-8.

El llenguatge DOT pot descriure grafs dirigits i grafs no dirigits. Els grafs dirigits normalment s’utilitzen per mostrar diagrames de flux i arbres de dependències. El llenguatge és compatible amb els comentaris d'estil: / \* \* / i //.

**Exemple 12.** A la figura 25 es poden observar dos fitxers en format DOT que representen el graf no dirigit (esquerra) i un graf dirigit (dreta) de la figura 24.

Figura 25. Graf no dirigit i dirigit en format DOT.

En un graf no dirigit s'utilitza un doble guió (-) per a mostrar les relacions entre els nodes. I en un graf dirigit s’utilitza la fletxa (->).

Es poden aplicar diferents atributs en els grafs, als nodes i a les arestes. Aquests atributs poden controlar aspectes com ara el color, la forma i estils de les línies. Pels nodes i les arestes, una o més parelles d’atribut-valor es col·loquen entre claudàtors després d'una declaració i abans del punt i coma (que és opcional).

La figura 26 mostra la gramàtica abstracta que defineix el llenguatge DOT. Els caràcters literals es donen entre cometes simples. Els parèntesis indiquen una agrupació. Els claudàtors tanquen elements opcionals. I les barres verticals separen les alternatives.



Figura 26. Gramàtica del llenguatge DOT.

Les paraules node, edge, graph, digraph, subgraph, i strict no són sensibles a les majúscules. I les cadenes de caràcters entre cometes dobles es poden concatenar utilitzant l’operador '+'.

El nom del graf o també anomenat ID, pot estar buit (“”) o incloure:

* Qualsevol cadena de caràcters [a-zA-Z\200-\377], caràcters connectats per ('\_') o dígits [0-9]. Un dígit però no pot estar mai a l’inici d’una cadena de caràcters.
* Un nombre tal que [-]?(.[0-9]+ | [0-9]+(.[0-9]\*)? ).
* Qualsevol cadena de caràcters entre cometes dobles.
* Una cadena de caràcters en format HTML (<...>).

La propietat strict prohibeix la creació de múltiples arestes, és a dir, no pot haver una aresta amb el mateix node cua i node cap. En els grafs no dirigits, no pot haver-hi una aresta connectada en els mateixos dos nodes.

*7.1.3.1.1 Subgrafs i Clústers*

Els sungrafs i els clústers es processen d’una manera especial en els fitxers DOT. Per exemple, els subgrafs juguen tres papers. En primer lloc, un subgraf pot ser emprat per a representar l'estructura d’un graf, que indica quins nodes i arestes han d’estar agrupats junts. Aquest és el paper habitual dels subgrafs i en general s'especifica la informació semàntica sobre els components del graf. En el segon paper, un subgraf pot proporcionar un context per a l'establiment d'atributs. Per exemple, un subgraf podria especificar que el blau és el color per defecte per a tots els nodes definits en ell mateix. I el tercer paper dels subgrafs involucra directament com serà exposat un graf a certs motors de disseny.

Si el nom d’un subgraf comença amb clúster, Graphviz assenyala el subgraf com subgraf especial (clúster). Si és compatible amb el format DOT, el motor de la disposició farà el disseny de manera que els nodes que pertanyen a l'agrupació es dibuixin junts, amb tot el dibuix de l'agrupació dins d'un rectangle delimitador.

Hi ha certes restriccions amb els subgrafs i els clústers. En primer lloc, el noms d'un graf i el dels subgrafs comparteixen el mateix espai de noms. Per tant, cada subgraf ha de tenir un nom únic. I en segon lloc, tot i que els nodes poden pertànyer a qualsevol dels subgrafs, se suposa que segueixen una jerarquia estricta en forma d’arbre.

*7.1.3.1.2 Exemple*

A l’esquerra de la figura 27 es pot veure un exemple senzill d’un un script que descriu l'estructura d'un graf i el graf resultant a la dreta. És un graf no dirigit i conté els atributs de color i estil a les arestes. No contempla cicles per la propietat strict.

Les línies 2 a 7 representen els nodes del graf i la resta de línies representen les relacions (arestes) entre els vèrtexs d’aquest graf.



Figura 27. Script en format DOT i graf resultant.

7.2 tipus de grafs

Divisió dels grafs en 4 casos. Un graf pot ser planar (o pla), planar (o pla) amb llindar, lineal o exponencial.

* + 1. *Graf planar*

Un graf planar és equivalent a un graf de Gaifman o graf primer simple. Les parelles <atribut: valor>, de cada fila d’una taula SQLite amb les que es representen els nodes i les arestes del graf, s’acoloreixen del mateix color si com a mínim les parelles <atribut: valor> es repeteixen una vegada a la taula. O sigui que s’acoloreixen totes les arestes del graf on el nombre total d’equivalències de cadascuna sigui igual o superior a u.

En aquest tipus de grafs, existeixen dues classes d’equivalències: es repeteix l’aresta o no es repeteix. Que es repeteixi l’aresta vol dir que existeix com a mínim una relació <atribut:valor> a la taula SQLite que uneix els nodes i l’aresta equivalents.

A partir de l’algoritme següent es pot construir un graf planar.



Codi 2. Algoritme planar.

L’algoritme planar acoloreix les arestes del graf amb el nombre total d’equivalències igual o superior a u, de color negre i la línia contínua. I les arestes del graf que no pertanyen a cap classe d’equivalència (el nombre total d’equivalències d’una aresta és igual a zero) no es modifiquen. Aquestes arestes són el resultat del procés anterior d’inicialització del graf, documentat en l’apartat anterior.

**Exemple 13.** A la figura 28 es pot observar un graf planar. Hi ha cinc classes d’equivalències (1, 2, 3, 4 i 6).



Figura 28. Graf planar.

* + 1. *Graf planar amb llindar*

Un graf planar amb llindar és un graf planar on el nombre total d’equivalències de cada aresta del graf s’acoloreix del mateix color segons si aquest valor és més gran o és igual o més petit que el llindar (nombre enter) introduït per l’usuari.

En aquest tipus de grafs, també existeixen dues classes d’equivalències: les classes d’equivalències que estan per sobre el valor del llindar i les classes d’equivalències que tenen el mateix valor del llindar o estan per sota el valor del llindar.

A partir de l’algoritme següent es pot construir un graf planar amb llindar.



Codi 3. Algoritme planar amb llindar.

Si el nombre total d’equivalències d’una aresta del graf és igual o més gran que el valor del llindar anotat per l’usuari, l’algoritme planar amb llindar acoloreix l’aresta del graf de color negre i la línia contínua. En canvi, si el nombre total d’equivalències d’una aresta del graf és més petit que el valor del llindar anotat per l’usuari, l’algoritme planar amb llindar distingeix la línia de l’aresta discontínuament.

**Exemple 14.** A la figura 29 es pot observar un graf planar amb el llindar igual a 3. Es pot comprovar que si el nombre total d’equivalències d’una aresta és igual o superior a 3, l’aresta és de color negre i la línia no és discontínua. Si el nombre total d’equivalències d’una aresta és inferior a 3, l’aresta és discontínua.



Figura 29. Graf planar amb llindar.

* + 1. *Graf lineal*

Un graf lineal és un graf planar on el nombre total d’equivalències de cada aresta del graf s’acoloreix d’un color diferent. Cada color és únic per a la classe d’equivalència corresponent. Concretament, en el treball es processen 10 colors bàsics [[R](http://www.graphviz.org/doc/info/colors.html)] que més endavant s’anomenen.

En aquest tipus de grafs, el número de classes d’equivalències és igual al número de colors que es processen. Per tant, existeixen tantes classes d’equivalències com colors hi hagi en el graf.

A partir de l’algoritme següent es pot construir un graf lineal.



Codi 3. Algoritme lineal.

L’algoritme lineal acoloreix cada aresta segons el seu nombre total d’equivalències. Les arestes del graf que no acoloreix, és perquè el nombre total d’equivalències és igual a 0, i no es modifiquen. Aquestes últimes són el resultat del procés anterior d’inicialització del graf.

A continuació es mostra una taula de les equivalències entre les classes d’equivalències i el color que les representen:

|  |  |
| --- | --- |
| **Classe d’equivalència** | **Color** |
| 1 | Negre |
| 2 | Turquesa |
| 3 | Verd |
| 4 | Magenta |
| 5 | Taronja |
| 6 | Blau |
| 7 | Vermell |
| 8 | Groc |
| 9 | Marró |
| 10 | Gris |

Taula 9. Taula d’equivalències.

**Exemple 15.** A la figura 30 es pot observar un graf lineal. A cada classe d’equivalència li correspon un color diferent. El nombre de classes d’equivalències del graf és cinc (1, 2, 3, 4 i 6). Per tant els colors que s’han utilitzat per a acolorir les arestes també són cinc.



Figura 30. Graf lineal.

* + 1. *Graf exponencial*

Un graf exponencial és un graf planar on diferents classes d’equivalències s’acoloreixen del mateix color. Cada color ja no és únic per a cada classe d’equivalència, sinó que un mateix color comprèn un interval exponencial de classes d’equivalències.

En aquest tipus de grafs, les classes d’equivalències s’agrupen segons la sèrie de potències del dos. L’interval exponencial té el rang de de 20 fins a 210.

Igual que en els grafs lineals, es processen 10 colors bàsics. Més endavant s’anomena la composició dels grups amb les classes d’equivalències i els colors bàsics corresponents.

A partir de l’algoritme següent es pot construir un graf exponencial.



Figura 31. Graf exponencial.

Inicialment l’algoritme exponencial crea un graf lineal. A partir del graf lineal, l’algoritme acoloreix diferents classes d’equivalències segons l’interval exponencial que comprenen. Les arestes del graf que no acoloreix, és perquè el nombre total d’equivalències és igual a 0, i no es modifiquen. Aquestes últimes són el resultat del procés anterior d’inicialització del graf.

A continuació es mostra una taula dels grups amb els quals s’agrupen les classes d’equivalències i el color del grup en qüestió.

|  |  |
| --- | --- |
| **Interval de classes d’equivalències** | **Color** |
| (0, 1) | Negre |
| (2, 3) | Turquesa |
| (4, 7) | Verd |
| (8, 15) | Magenta |
| (16, 31) | Taronja |
| (32, 63) | Blau |
| (64, 127) | Vermell |
| (128, 255) | Groc |
| (256, 511) | Marró |
| (512, 1023) | Gris |

Taula 10. Taula dels grups d’equivalències.

**Exemple 16.** A la figura 32 es pot observar un graf exponencial. El nombre de classes d’equivalències del graf és cinc. Es pot comprovar com les classes d’equivalència, 2 i 3, que pertanyen al mateix grup estan acolorides amb el mateix color. El mateix amb les classes d’equivalència , 4 i 6.



Figura 32. Graf exponencial.

1. DESCOMPOSICIÓ DEL GRAFS

Les descomposicions són d’especial interès dins de les matemàtiques i les seves aplicacions, des de que proporcionen eines per a la divisió de problemes complexos en problemes més petits que són més fàcils de solucionar. Les descomposicions són un cas especial del principi de *“divideix i venceràs”* [[R](https://en.wikipedia.org/wiki/Divide_and_conquer_algorithm)].

La idea amb la que es basa la descomposició dels grafs consisteix en trobar subconjunts per a cada graf, anomenats clans, en els quals els elements continguts es relacionen de la mateixa manera amb tots aquells elements fora del subconjunt. Concretament, en aquesta descomposició els grafs es divideixen en clans primers, un tipus especial de clans, i la construcció de les 2-estructures indica les relacions necessàries entre els clans primers dels grafs.

La descomposició dels grafs en clans, procés que també és conegut com la descomposició modular o la descomposició de substitució, és un exemple d'una descomposició estretament relacionada amb la descomposició per a quocients d’àlgebra.

* 1. clans

La noció tècnica més important en aquest treball és la definició de clan i es defineix a continuació.

Un subconjunt X ⊆ Dg d’un graf g és un clan si cada node y ∉ X “veu” els nodes de X de la mateixa manera i cada dos nodes x1, x2 ∈ X “veuen” cada node y ∉ X de la mateixa manera: per a tot x1, x2 ∈ X i y ∉ X,

(y, x1) Rg (y, x2) i (x1, y) Rg (x2, y)

**Exemple 17.** A la figura 33 s’han encerclat en forma de rectangle els clans X = {A, B} i Y = {C, D} del graf g. El subconjunt {A, C} no és un clan, perquè els nodes F, E i D no distingeixen A de C amb la mateixa classe d’equivalència: (F, A) Rg ≠ (F, C) Rg, (E, A) Rg ≠ (E, C) Rg i (D, A) Rg ≠ (D, C) Rg.



Figura 33. Clans X i Y de g.

Tots els subconjunts que formen un clan a la figura 33 són: {'A'}, {'B'}, {'C'}, {'D'}, {'E'}, {'F'}, {'B', 'A'}, {'C', 'D'} i {'C', 'A', 'F', 'B', 'D', 'E'}.

* 1. clans trivials

Per a qualsevol graf, els subconjunts únics amb el format {x}, x∈ Dg i els que contenen tots els nodes d’un graf sempre són clans. Concretament aquests clans s’anomenen clans trivials. Contràriament, un clan no trivial és un clan X de tal manera que |X| ≥ 2.

**Exemple 18.** Considerant la figura 33 de l’exemple 17 ara podem dir que de fet, els clans X i Y no són clans trivials del graf. Els subconjunts {'A'}, {'B'}, {'C'}, {'D'}, {'E'}, {'F'} i {'C', 'A', 'F', 'B', 'D', 'E'} sí que formen clans trivials.

Els clans trivials també són clans primers.

* 1. clans primers

Les 2-estructures es formen a partir dels clans primers d’un graf.

Conjunts disjunts

1. 2-ESTRUCTURES

Una 2-estructura és una seqüència de valors agrupats, formada per un subconjunt finit D, anomenat domini, i una relació d’equivalència *R ⊆* *E2(D) x E2(D)* en el conjunt *E2(D) = {(u, v) | u ∈ D, v ∈ D, u ≠ v}* de les seves arestes. D’aquesta manera una 2-estructura es pot definir com:

*G = (D, R)*

En aquest treball es desenvolupa la teoria de la 2-estructures, i en particular es demostra que cada 2-estructura pot ser construïda a partir de la descomposició d’un tipus de graf en clans primers. Per tant, cada tipus graf representa un tipus de 2-estructura. Aquest resultat s'obté a través d'una representació jeràrquica en forma d’arbre.

Les 2-estructures es poden diferenciar en tres subclasses bàsiques: completes, lineals i primitives. Una 2-estructura és completa quan només té definida una classe d’equivalència. Cada 2-estructura completa és primitiva si |D| ≤ 2.

Una 2-estructura és lineal quan té definides dues classes d’equivalències i les arestes incloses en cadascuna d'elles estableixen un ordre total sobre els elements del domini D.

Una 2-estructura és primitiva quan no pot des composar-se en altres 2-estructures. També diem que una 2-estructura és primitiva si només conté clans trivials. Les 2-estructures primitives són un cas especial i important en la teoria de les 2-estructures.

* 1. creació de les 2-estructures
  2. tipus de 2-estructures

Divisió de les 2-estructures en 4 casos. Una 2-estructura, igual que els grafs, pot ser planar (o plana), planar (o plana) amb llindar, lineal o exponencial.

* + 1. *2-structura planar*
    2. *2-estructura planar amb llindar*
    3. *2-estructura lineal*

*9.2.4 2-estructura exponencial*

9.3 visualització de les 2-estructures

*9.3.1 Fitxers DOT*

Si un atribut per defecte es defineix utilitzant un node, una aresta o declaració gràfica, o per una assignació d'atribut no connectat a un node o vora, qualsevol objecte del tipus apropiat definit després heretarà aquest valor d'atribut. Això es manté fins que l'atribut per defecte s'estableix a un nou valor, des de quin punt s'utilitza el nou valor. Objectes definits abans que un atribut predeterminat està establert tindrà un valor de cadena buida adjunta a l'atribut d'una vegada feta la definició d'atribut per defecte.

*9.3.1.1 Exemple*

A l’esquerra de la figura 34 es pot veure un exemple senzill d’un un script que descriu l'estructura resultant de la figura 35.

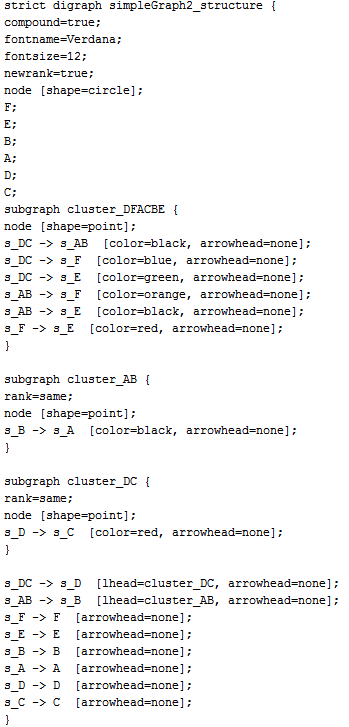


Figura 34. Script en format DOT.



Figura 35. 2-estructura.

*9.3.1.2 Limitacions*

És possible especificar detalls de disseny amb DOT, encara que no totes les eines que s’implementen són òptimes. En funció de les eines utilitzades, els usuaris han de confiar en algoritmes automatitzats de traçat (potencialment amb un resultat inesperat) o amb formes mal col·locades.

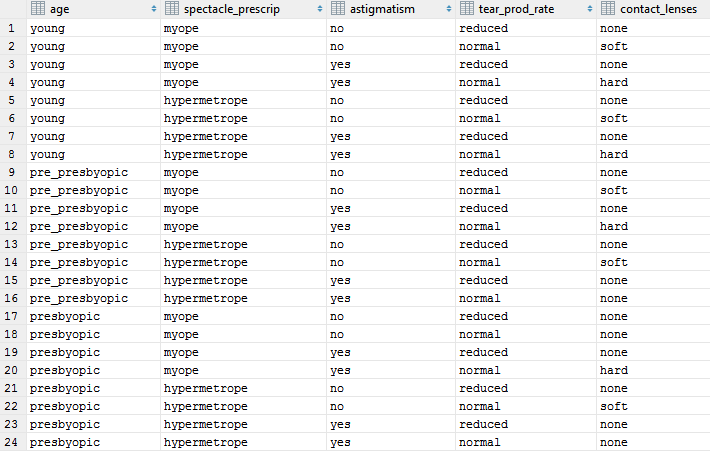
A continuació podem veure la limitació del llenguatge DOT amb la que ens hem trobat durant el treball:



Figura X. 2-structure.

Com es pot observar a la figura X, els rectangles que representen els clans primers, no estan ben alineats seguint la estratègia del tractament de la informació i el coneixement de comandes Top-down [R]. Com a conseqüència que els subgrafs de dispersió o clústers no són part del llenguatge DOT, sinó únicament una convenció sintàctica adherida per alguns dels motors de disseny de Graphviz.

1. exemple
   1. dades relacionals



Taula X . Taula.

* 1. graf
     1. *Graf planar*

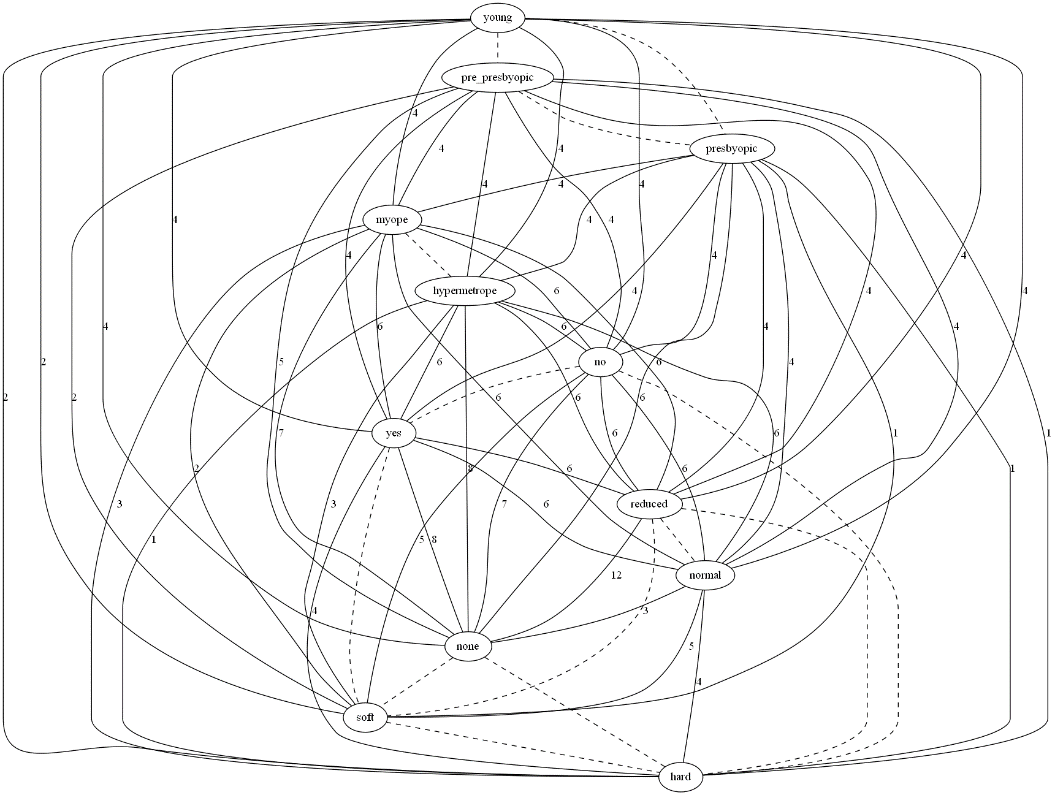


Figura X. Graf planar.

* + 1. *Graf planar amb llindar*

K = 4.

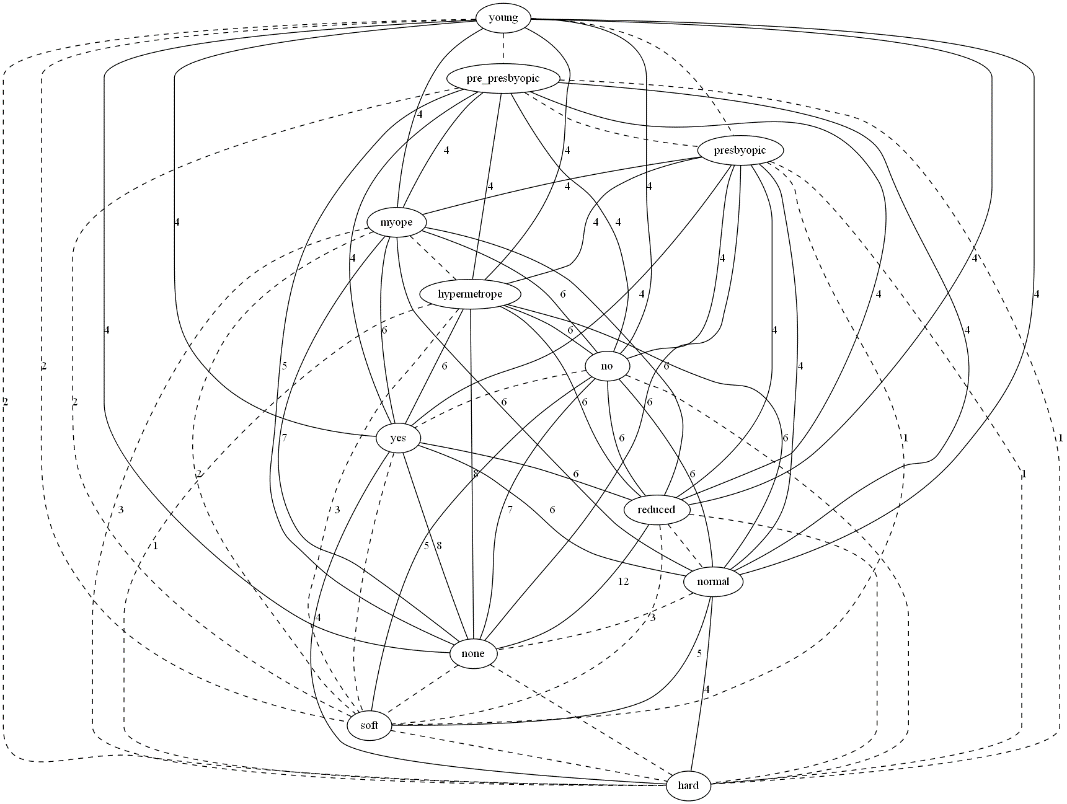


Figura X. Graf planar amb llindar.

* + 1. *Graf lineal*

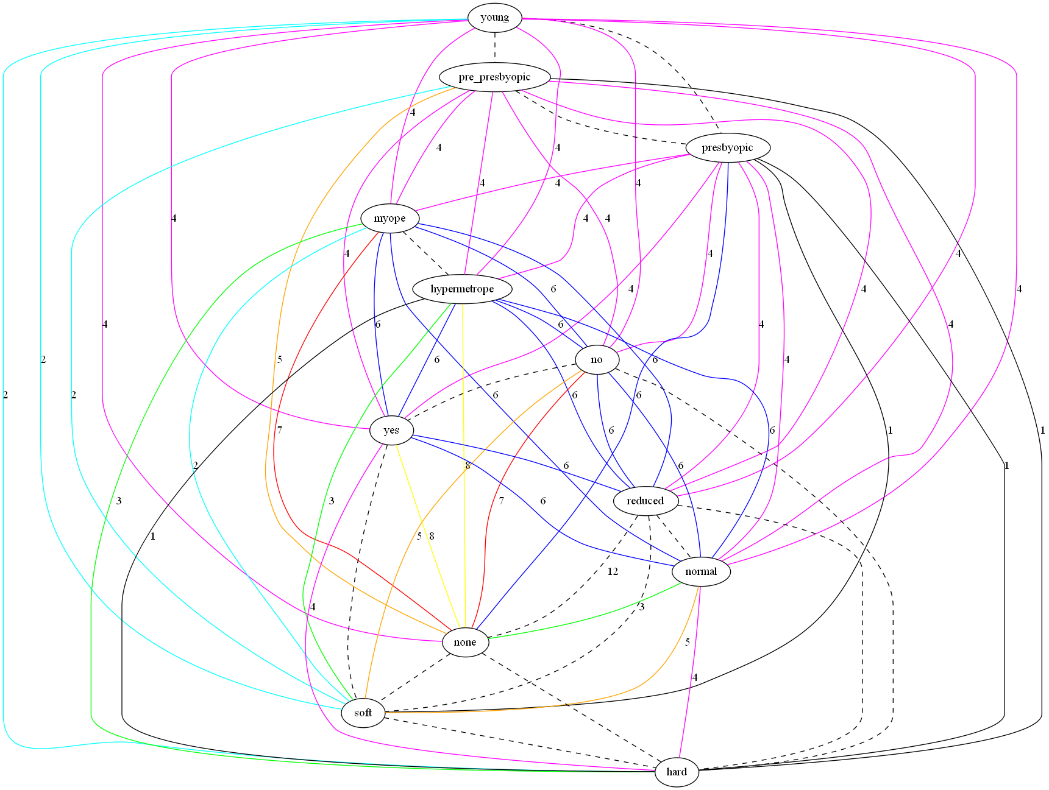
**

Figura X. Graf lineal.

* + 1. *Graf exponencial*

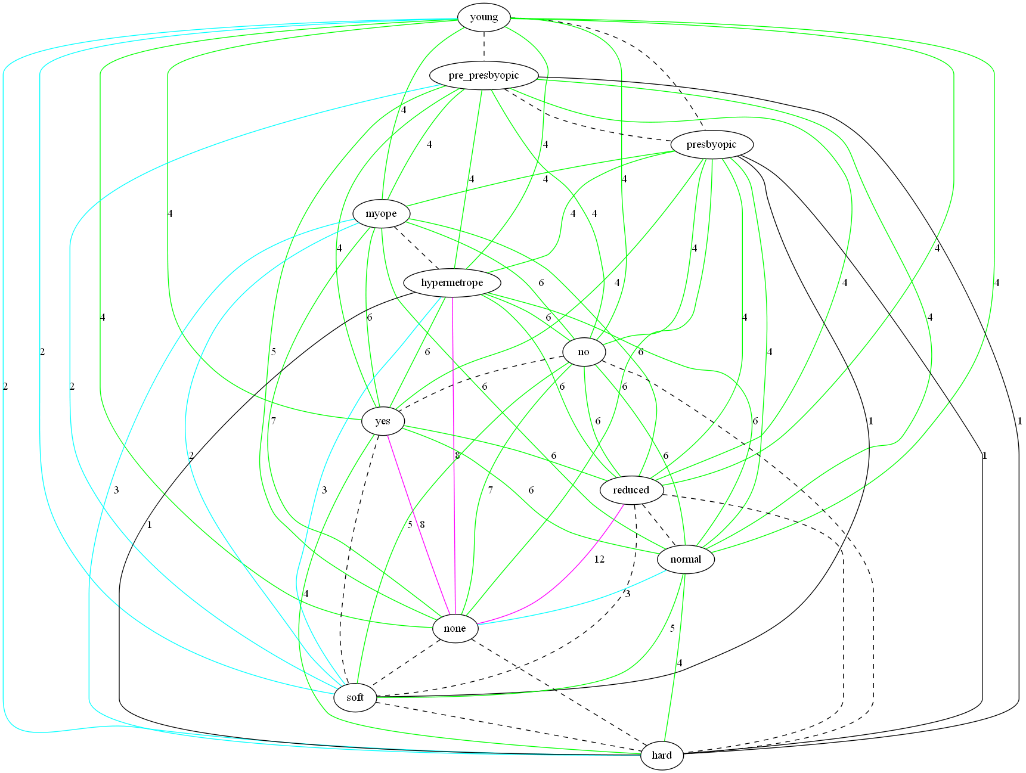


Figura X. Graf exponencial.

* 1. descomposició del graf
  2. 2-estructura

1. CONCLUSIONS

En aquest capítol es presenta un resum del conjunt del treball realitzat i es descriuen les possibilitats de continuació. També s’ofereix una valoració personal sobre el desenvolupament del treball.

* 1. Resultats del treball
  2. treball futur

Ampliar colors.

Més dades.

Millores algoritmes Eclat i FP-growht -> depht-first algorithms

* 1. valoració personal

La realització del treball final de grau ha estat una experiència gratificant i enriquidora.

Sobretot pel repte que suposa realitzar un treball d’investigació. I per la doble naturalesa del treball, que es composa d’una part més teòrica i una part més pràctica.

A la part pràctica s’ha desenvolupat el paquet de software, on he pogut aplicar molts dels coneixements de programació adquirits durant la carrera. I a la part teoria es troba l’estudi de les 2-estructures, tema totalment desconegut, que gràcies a la seva exploració m’ha resultat molt interessant.

Aquest treball també s’ha convertit en la oportunitat d’estudiar alguns dels temes inclosos en la mineria de dades, que per problemes d’horaris no vaig poder cursar a l’assignatura de Mineria de Dades (MIDA).

Un altre dels motius a valorar molt positivament és la llibertat que he tingut i el suport que he rebut per a desenvolupar el treball.

**BIBLIOGRAFIA**

**GLOSSARI DE SIGNES, SÍMBOLS, ABREVIATURES, ACRÒNIMS I TERMES**

APÈNDIXS

A subconjunts més freqüents

En aquest apèndix s’introdueixen les nocions fonamentals de l’algoritme per a trobar els subconjunts més freqüents.

*A.1 Algoritme apriori*

L’algoritme apriori és un algoritme influent en la mineria de dades que s’utilitza per a trobar els conjunts d’elements més freqüents de les bases de dades transaccionals [[R](https://es.wikipedia.org/wiki/Transacci%C3%B3n_(base_de_datos))]. L’algoritme va ser proposat per Agrawal i Srikant el 1994.

Conceptes clau:

* Conjunts d’elements freqüents: els conjunts d'elements que tenen un suport mínim (indicat per Li per a cada ith - conjunt d’elements).
* Propietat apriori: qualsevol subconjunt d’un conjunt d’elements freqüents ha de ser freqüent. Per exemple, si {AB} és un conjunt freqüent, els subconjunts {A} i {B} també ho són.
* Operació d’unió: per a trobar Lk , un conjunt de candidats format per k-conjunts d'elements freqüents, s’ha de crear la unió de Lk-1 amb si mateix.

Apriori utilitza l’enfocament *" bottom up"* [[R](https://ca.wikipedia.org/wiki/Top-down_i_Bottom-up)], la cerca en amplada [R] i l’arbre de Merkle (figura X).

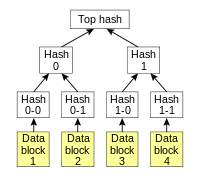


Figura X. Arbre de Merkle.

A continuació es mostra el pseudocodi de l’algoritme que iterativament busca els conjunts d’elements més freqüents a través de la cardinalitat (número total d’elements d’un conjunt), d’1 a k (k-conjunt d’elements):

* Ck es genera mitjançant la unió Lk-1 amb si mateix.
* Qualsevol (k-1)-conjunt d’elements que no és freqüent no pot ser un subconjunt d'un k-conjunt d'elements freqüents.

Ck : candidat a conjunt d'elements de mida k

Lk : conjunt d'elements freqüent de mida k

L1 = {conjunt d’elements freqüents};

**for** (k = 1; Lk !=∅; k++) **do begin**

Ck+1 = candidats generats a partir de Lk;

**for** **each** transacció t a la base de dades **do**

incrementar el recompte de tots els candidats a Ck+1 que estan continguts en t

Lk+1 = candidats a Ck+1 amb min\_support

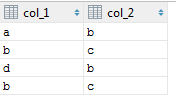
**end**

**return** ∪k Lk;

Cada transacció és vista com un conjunt d’elements. Donat un valor d'un llindar C, l'algoritme a priori identifica tots els conjunts d’elements que són subconjunts d'almenys C transaccions a la base de dades t.

A continuació genera els conjunts d’elements candidats de mida k a partir de conjunts d’elements de mida k-1. Després poda els candidats que tenen un suport no freqüent i el conjunt candidat conté tots els conjunts freqüents de mida k. Finalment, escaneja la base de dades per determinar els conjunts d’elements freqüents entre els candidats.

**Exemple A.1**. A partir de la taula T:



Taula X. Taula T.

S’utilitza Apriori per a determinar els subconjunts més freqüents de la taula T. Definim que un element és freqüent si com a mínim es repeteix una vegada a la taula T (suport igual a 1). S’obtenen els resultats següents:

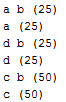


Figura X. Subconjunts més freqüents.

El proper pas és generar la llista de totes les parelles d’elements més freqüents: [{'b', 'a'}, {'a'}, {'d', 'b'}, {'d'}, {'c', 'b'}, {'c'}]