Títol: Aplicació per fer consultes de cubs de dades usant MapReduce

Volum: 1

Alumne: Jaume Ferrarons Llagostera

Director/Ponent: Albert Abelló Gamazo

Departament: ESSI

Data: 28 de juny del 2011

DADES DEL PROJECTE

Títol del Projecte: Aplicació per fer consultes de cubs de dades usant MapReduce.

Nom de l'estudiant: Jaume Ferrarons Llagostera
Titulació: Enginyeria en informàtica
Crèdits: 37,5
Director/Ponent: Albert Abelló Gamazo
Departament: ESSI

MEMBRES DEL TRIBUNAL

President: Antoni Urpí Tubella

Vocal: Oscar Romero Moral

Secretari: Juan Aranda López

QUALIFICACIÓ

Qualificació numèrica:
Qualificació descriptiva:

Data:

Índex de continguts

1 Introducció....9
2 Objectius....11
2.1 Objectius de les proves....12
2.2 Decisió de la tecnologia....12
3 Conceptes bàsics....13
3.1 Data Warehouse (DW)....13
3.2 Extraction Transformation and Load (ETL)....13
3.3 Desnormalització....13
3.4 OnLine Analytical Processing (OLAP)....15
3.4.1 Cub de dades....15
3.4.1.1 Dimensió....16
3.4.1.2 Nivell d'agregació....17
4 Requeriments del sistema....19
4.1 Requeriments funcionals....19
4.1.1 Creació dels índexs....19
4.1.2 Realització de consultes....19
4.2 Requeriments no funcionals....21
4.3 Requeriments no tècnics....22
5 Model de desenvolupament de software....23
6 Tecnologies emprades....27
6.1 Apache HDFS....27
6.1.1 Arquitectura....27
6.2 Apache Hadoop MapReduce....28
6.2.1 Operacions....28
6.2.2 Arquitectura....29
6.3 Apache HBase....31
6.3.1 Model de dades....31
6.3.2 Arquitectura....32
7 Algorismes....37
7.1 Creació d'índexs....38
7.1.1 Índex 1-nivell....39
7.1.2 Índex multinivell....42
7.2 Consulta del cub de dades....44
7.2.1 Obtenció de les claus....45
7.2.1.1 Accés a l'índex d'un sol nivell....45
7.2.1.2 Accés a l'índex multinivell....48
7.2.1.3 Versió anterior....50
7.2.2 Obtenció dels valors....52
7.2.2.1 Index Random Access (IRA)....52
7.2.2.2 Index Filtered Scan (IFS)....56
7.2.2.3 Filtered Source Scan (FSS)....59
7.3 Desnormalització....60
7.3.1 Primera versió....64
7.3.1.1 Pseudocodi....64

8 Planificació i estudi econòmic....67
8.1 Descripció de les etapes....67
8.1.1 Estudiant el problema....68
8.1.2 Aprenentatge....68
8.1.3 Configuració del sistema....68
8.1.3.1 Configuració de l'entorn de desenvolupament....68
8.1.3.2 Configuració del clúster per fer proves....69
8.1.4 Implementació....71
8.1.5 Jocs de proves....71
8.1.5.1 Modificacions del TPC-H....71
8.1.5.2 Desnormalitzador....72
8.1.5.3 Traducció de les consultes....72
8.1.6 Primeres proves....73
8.1.7 Test dels algorismes....73
8.1.8 Proves de rendiment....73
8.1.9 Millores de rendiment....74
8.1.10 Memòria del projecte....74
8.1.11 Presentació del projecte....74
8.2 Planificació inicial....75
8.3 Planificació realitzada....76
8.4 Pressupost del projecte....77
8.5 Cost de la planificació realitzada....78
9 Especificació....79
9.1 Casos d'ús....79
9.1.1 Configuració de l'execució....80
9.1.2 Interfície textual....80
9.1.3 Interfície gràfica....81
9.1.4 Creació d'índexs....81
9.1.5 Realitzar consulta....82
9.1.6 Visualitzar resultats....84
9.1.7 Executar codi....84
9.1.8 Modificar opcions....85
9.1.9 Codi generat....86
9.2 El model conceptual....86
9.2.1 Attribute....86
9.2.2 Dimension....87
9.2.3 Condition....87
9.2.4 Selection....87
9.3 El model de navegabilitat....88
10 Disseny....89
10.1 Esquema de dades de l'índex....89
10.2 Diagrama de classes....90
10.2.1 Condition....90
10.2.2 ConditionAND....90
10.2.3 ConditionNode....90
10.2.4 ConditionOR....91

10.2.5 Dimension....91
10.2.6 HBaseUtils....91
10.2.7 HumanReadableException....91
10.2.8 Interpreter....92
10.2.9 MapReduce0....92
10.2.10 MapReduce1....92
10.2.11 MapReduce2....93
10.2.12 MapReduceScanner....94
10.2.13 Parameters....94
10.2.14 RowCounter....94
10.2.15 Selection....94
10.2.16 Utilities....94
10.2.17 CodeExecutorPanel....94
10.2.18 ConfigurationDialog....95
10.2.19 DimensionPanel....95
10.2.20 GeneratedCodePanel....95
10.2.21 Main....95
10.2.22 MainWindow....95
10.2.23 QueryExecutorPanel....95
10.2.24 ResultPanel....95
10.3 Diagrames de seqüència....97
10.3.1 Diagrama de seqüència de la creació d'índexs....97
10.3.2 Diagrama de seqüència d'una consulta (IRA)....100
11 Implementació....107
11.1 Programa principal....107
11.1.1 Problemes trobats....108
11.2 Generació dels jocs de proves....111
11.2.1 Generador de les taules del TPC-H....111
11.2.2 Carregador de dades....113
11.2.2.1 Rendiment....113
11.2.3 Desnormalitzador....114
12 TPC-H....117
12.1 Procés de desnormalització....118
12.1.1 Temps requerit....120
12.2 Consultes del TPC-H....121
12.2.1 Factor de selecció....122
12.2.2 Mesures seleccionades....123
12.2.3 Mida de l'agrupació....124
12.2.4 Nombre de clàusules....124
12.3 Índexs....125
13 Proves....127
13.1 Proves de funcionament....127
13.1.1 Creació de dimensions....127
13.1.1.1 Dimensions simples....127
13.1.1.2 Dimensions compostes....128
13.1.1.3 Ordre en les dimensions compostes....130

13.1.2 Consultes....131
13.1.2.1 Consultes bàsiques....131
13.1.2.2 Múltiples clàusules....132
13.1.2.3 Agrupació de valors....132
13.1.3 Desnormalitzador....134
13.2 Proves de rendiment....135
13.2.1 Espai requerit de disc....135
13.2.2 Desnormalitzador....137
13.2.3 Comparació d'índexs d'un nivell amb índexs multinivell....140
13.2.3.1 Mida de la taula dels índexs....140
13.2.3.2 Temps de creació dels índexs....143
13.2.3.3 Creació dels índexs amb dos servidors....146
13.2.3.4 Conclusions....148
13.2.4 Comparació dels algorismes de consulta....149
13.2.4.1 Factor de selecció....149
13.2.4.2 Mida de l'agrupació....159
13.2.4.3 Mesures seleccionades....164
13.2.4.4 Nombre de clàusules....169
13.2.4.5 Conclusions....175
14 Conclusions....177
14.1 Línies de futur....178
15 Glossari....181
16 Bibliografia....183
16.1 Articles....183
16.2 Llibres....183
16.3 Manuals....183
16.4 Pàgines web....183
16.5 Altres recursos....184
17 Annex I: Taula de proves....185
18 Annex II: Manual d'usuari....187
18.1 Instal·lació dels serveis....187
18.1.1 Hadoop MapReduce i HDFS....187
18.1.1.1 Arrancada del servei....189
18.1.2 HBase....190
18.1.2.1 Arrancada del servei....191
18.2 Funcionament....192
18.2.1 Programa principal....192
18.2.1.1 Paràmetres....192
18.2.1.2 Interfície textual....194
18.2.1.3 Interfície gràfica....196
Opcions generals....201
Opcions d'execució....202
Opcions d'estadístiques....203
18.2.2 Generador de taules del TPC-H....203
18.2.3 Carregador....204
18.2.4 Desnormalitzador....204

1 Introducció

Avui dia tota la informació és digitalitzada i processada amb ordinadors. Per aquest motiu, es fan necessàries les eines per a manipular grans quantitats de dades. Fins fa pocs anys s'han fet servir gestors de bases de dades relacionals per a gestionar tota la informació. Però els temps canvien i les bases de dades relacionals resulten massa lentes i ineficaces a l'hora de processar tantes dades com les que es manegen actualment. És per això que s'estan desenvolupant altres tipus de gestors de bases de dades que no siguin relacionals i que es puguin executar en arquitectures distribuïdes en *cloud*.

En aquest sentit, l'any 2004, Google va presentar la tecnologia MapReduce com a solució per a processar grans quantitats de dades de forma concurrent en grans clústers. Des d'aquest moment MapReduce ha estat objecte d'investigació per a propòsits molt diversos. L'única restricció que presenta aquest model és que les dades s'han de poder expressar en parelles de clau i valor.

D'altra banda, les tecnologies distribuïdes en *cloud* resulten molt atractives a les petites i mitjanes empreses per la seva política de servei, que només es paga per ús i que, per tant, els permet transformar costs fixos en variables. El mercat de les tecnologies *cloud* és un mercat emergent i en el qual s'està invertint un esforç molt gran.

Les empreses tenen bases de dades plenes d'informació, com ara sobre els seus clients, i en volen extreure informació útil, per exemple, detectar quin és el perfil dels seus millors clients. Per a poder-ho fer necessiten aplicar un procés ETL (*Extraction, Transformation and Load*). L'inconvenient és que els processos ETL solen ser molt lents i, no obstant això, molt presents en la vida diària d'una empresa.

L'objectiu d'aquest projecte és implementar un procés ETL utilitzant MapReduce i avaluar-ne el rendiment. Aquest procés ETL està pensat perquè s'executi de forma concurrent i distribuïda dins d'un clúster. Una empresa, per exemple, carregaria les dades a processar en el *cloud* i executaria tot el procés ETL de forma distribuïda i concurrent.

Les proves de rendiment són un factor clau per a aquest projecte. Tenen per objectiu mesurar els temps dels diferents algorismes i poder estimar quines condicions són favorables per a cadascun d'ells. Per fer-ho, he adaptat les proves del TPC-H que ens proporcionen una

estructura de base de dades realista. Simultàniament, aquestes proves també porten una bateria de consultes per executar sobre la base de dades i extreure'n informació. Les proves del TPC-H estan ideades per a fer tests de rendiment en entorns de suport de decisions com els que hi poden haver a les empreses.

Com es pot veure, la importància de les proves que s'han fet d'aquest projecte és molt gran, ja que en depenen les diferents conclusions que se n'extreuen. Amb les conclusions obtingudes, s'ha volgut donar resposta a la principal qüestió: quines són les condicions en què seria millor utilitzar un algorisme en lloc d'un altre? Aquesta no és una pregunta fàcil de respondre i per aquest motiu s'han tractat amb rigor tots els resultats obtinguts i se n'ha de fer una valoració completament objectiva.

Personalment, la possibilitat de realitzar aquest projecte m'ha proporcionat l'oportunitat de participar en un projecte d'investigació i posar en pràctica els coneixements adquirits al llarg de la carrera. A més, m'ha donat un motiu per a estudiar tecnologies molt innovadores i que tot just s'estan començant a implantar en moltes empreses. He après com funciona el paradigma de programació del MapReduce i el gestor de bases de dades no relacionals HBase. També m'ha permès aprofundir els meus coneixements en el llenguatge de programació Java, conèixer noves biblioteques... En resum, m'ha aportat moltes experiències que espero que em serveixin en el meu futur professional.

2 Objectius

En aquest apartat presento els diferents objectius del projecte, que defineixen en què consisteix aquest projecte i quines funcionalitats se n'esperen. He identificat els següents objectius:

- El sistema ha d'implementar tres algorismes de consulta de cubs de dades utilitzant MapReduce.
- S'ha de permetre la definició de les dimensions del cub de dades i crear-ne els seus índexs.
- S'han de poder especificar les clàusules de predicat de les consultes per tal de restringir quina informació volem obtenir.
- El sistema ha de permetre especificar els valors a partir dels quals s'agruparan les dades obtingudes del cub de dades. Per a cada agrupació que es faci de les dades seleccionades se n'ha de calcular el valor de la seva agregació.
- S'han de poder realitzar consultes amb comoditat. Per a cada consulta s'han de poder especificar tots els seus paràmetres.
- Cal desenvolupar una interfície gràfica que permeti realitzar les consultes d'una forma visual i intuïtiva.
- El sistema ha de permetre prendre mesures de temps dels diferents procediments que s'executen per tal de facilitar l'estudi del rendiment.
- El sistema ha de ser fàcil de configurar.
- S'ha de poder executar la mateixa seqüència de comandes repetidament i de forma automàtica.
- El sistema a desenvolupar s'ha de poder executar de forma concurrent i distribuïda en diversos ordinadors.
- El sistema ha de permetre la preparació de proves amb facilitat.

2.1 Objectius de les proves

Les proves són un factor molt rellevant per a aquest projecte. Per això s'han definit uns objectius específics per a les proves. En aquest apartat recullo els objectius de les proves realitzades del projecte:

- L'estructura de la base de dades de les proves ha de ser realista.
- Les proves han de permetre treballar amb diferents volums de dades.
- El procés de proves s'ha de poder automatitzar.
- Els resultats han de permetre extreure conclusions sobre els algorismes.
- S'ha de fer un tractament objectiu de les dades.

2.2 Decisió de la tecnologia

La decisió tecnològica ja em va venir donada: els algorismes a implementar estaven pensats per a ser programats amb MapReduce. Les biblioteques de Hadoop MapReduce estan implementades en Java. No obstant això, s'han fet adaptacions per a poder invocar MapReduce des de C o Python, tot i així, per evitar problemes de compatibilitat, el projecte s'ha implementat en Java.

Com a gestor de bases de dades s'ha fer servir HBase. És un gestor de bases de dades no relacional que funciona sobre Hadoop MapReduce i que permet treballar amb grans quantitats de dades. L'altra opció era Hive, que també ofereix un sistema de gestió de bases de dades sobre Hadoop, però presenta l'inconvenient que el seu principal objectiu és oferir una interfície semblant al SQL per a treballar amb grans volums de dades sobre Hadoop. Per aquest motiu es va optar per HBase, que s'apropa més a les necessitats del projecte.

3 Conceptes bàsics

En aquest capítol presento diversos conceptes fonamentals per a poder comprendre aquesta memòria i el projecte en general.

3.1 Data Warehouse (DW)

Un Data Warehouse o Magatzem de Dades és una base de dades que conté la informació històrica d'una organització i que està pensada per a extreure'n informació. Les dades que conté s'han obtingut de diverses fonts de l'organització, per exemple, de bases de dades operacionals de l'organització.

Els Data Warehouse estan pensats per a dur-hi a terme activitats d'explotació de dades com: mineria de dades, OLAP, generació d'informes... Cal dir, a més, que cap d'aquestes activitats no requereix la modificació de les dades contingudes en el Data Warehouse.

3.2 Extraction Transformation and Load (ETL)

Un procés ETL és aquell procés que permet extreure informació d'una o més bases de dades operacionals i carregar-la en un Data Warehouse. Consisteix en tres etapes diferents:

- **Extracció**: és el procés d'obtenció de la informació de les fonts externes que es volen carregar a la base de dades.
- **Transformació**: és el conjunt d'operacions que fem a les dades obtingudes de l'extracció abans de guardar-les a la base de dades. Per exemple, es poden netejar les dades, tractar valors que falten, calcular mitjanes, filtrar els valors que ens interessin...
- **Càrrega**: consisteix a guardar les dades processades en el Data Warehouse.

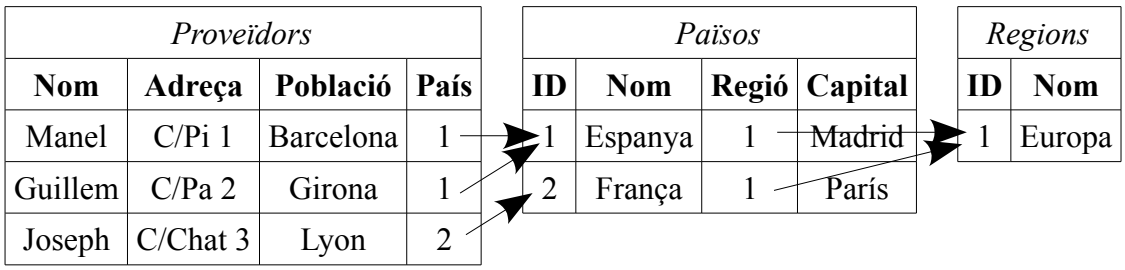
3.3 Desnormalització

La desnormalització és el procés que pretén eliminar les claus foranes d'una base de dades amb una estructura normalitzada per tal d'incrementar-ne el rendiment augmentant-ne la redundància de les dades[1]. Les claus foranes són referències entre taules, és a dir, són els elements d'una taula que conformen una referència inequívoca a una altra taula perquè contenen

1 Definició extreta de: Candace C. Fleming, Barbara Von Halle. *Handbook of Relational Database Design*. 1989

els elements identificadors imprescindibles per poder referir-se a una tupla en concret de l'altra taula.

Vegem a continuació dos exemples de bases de dades. Totes dues contenen les mateixes dades però la primera està normalitzada i la segona no.



Proveïdors			
Nom	**Adreça**	**Població**	**País**
Manel	C/Pi 1	Barcelona	1
Guillem	C/Pa 2	Girona	1
Joseph	C/Chat 3	Lyon	2

Països			
ID	**Nom**	**Regió**	**Capital**
1	Espanya	1	Madrid
2	França	1	París

Regions	
ID	**Nom**
1	Europa

Taula 1: Esquema normalitzat

Podem veure que, a l'esquema normalitzat anterior, la columna de país de la taula de proveïdors conté un identificador de país que es correspon al de la taula de països. El mateix succeeix entre la taula de països i la de regions.

Nom	**Adreça**	**Població**	**Nom país**	**Capital**	**Regió**
Manel	C/Pi 1	Barcelona	Espanya	Madrid	Europa
Guillem	C/Pa 2	Girona	Espanya	Madrid	Europa
Joseph	C/Chat 3	Lyon	França	París	Europa

Taula 2: Esquema desnormalitzat

Per a passar d'un esquema a l'altre s'ha de fer una join entre els valors de les taules igualant la clau forana d'una taula amb la clau primària de l'altra. En el cas de l'exemple hauríem de fer un parell de joins, que són les següents:

- Entre la taula de proveïdors i la de països: hem d'igualar el valor de la columna de país de la taula de Proveïdors amb el valor de l'identificador de la taula de països.

- Entre la taula resultant de la join anterior i la taula regions: s'ha d'igualar el valor de la columna de regió amb l'identificador de la taula de regions.

Podem veure que una de les principals diferències d'un esquema normalitzat amb un de desnormalitzat és la quantitat d'informació repetida que hi trobem.

Les redundàncies poden introduir anomalies; n'hi ha de diferents tipus, però bàsicament vol dir que una modificació que només suposaria un canvi utilitzant un esquema de taules normalitzat comportaria més canvis en un esquema desnormalitzat. En el cas d'exemple, si volem canviar la capital d'un país a l'esquema normalitzat, només hem de canviar el valor a l'entrada que correspongui de la taula país, però en canvi, a l'esquema desnormalitzat s'han de modificar totes les entrades de tots els clients del país que la població del qual estem modificant.

Per exemple, si volem canviar la capital d'Espanya per Sevilla, en el cas de la base de dades normalitzada només hem de modificar la capital de la fila 1 de la taula de països. En canvi, en el cas de l'esquema desnormalitzat hauríem de modificar la columna de capital de la primera i de la segona fila.

3.4 OnLine Analytical Processing (OLAP)

Els sistemes OLAP permeten solucionar ràpidament consultes d'anàlisi multidimensionals. Aquests sistemes serveixen, en general, per a generar informes, fer previsions, aplicar tècniques de mineria de dades... ja que permeten obtenir ràpidament la informació que compleix certs criteris. El nucli d'un sistema OLAP és un cub de dades.

3.4.1 Cub de dades

Un cub de dades és una estructura que permet una anàlisi ràpida de les dades que conté. La propietat que caracteritza els cub de dades és la capacitat de manipular i analitzar les dades des de múltiples perspectives.

Un cub de dades seria, per exemple, el que presento a continuació:

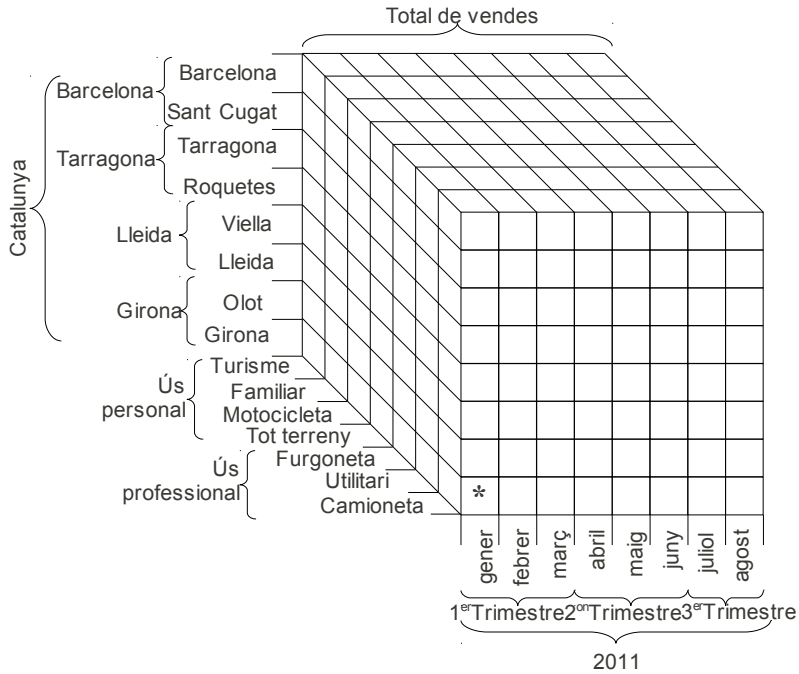


Figura 1: Exemple de cub de dades

A la figura 1 es mostra un cub de dades en el qual es presenten les vendes de vehicles d'un conjunt de concessionaris. La casella de baix a l'esquerra de la cara frontal del cub (marcada amb un asterisc), conté el nombre de camionetes venudes durant el mes de gener del 2011 en els concessionaris de la ciutat de Girona.

3.4.1.1 Dimensió

Una dimensió és un conjunt de columnes o atributs que tenen una relació jeràrquica en el seu significat. Per exemple, a la dimensió *Lloc*, de la Figura 1, la comunitat autònoma engloba la província i les províncies, al seu torn, les ciutats.

El cub de dades de la figura 1 consta de tres dimensions: Lloc, Categoria del Vehicle i Data. Podem veure que la dimensió Data consta de tres nivells d'agregació: Any, Trimestre i Mes, un Any engloba Trimestre i Trimestre engloba Mes. La dimensió Lloc consta de la Comunitat Autònoma, Província i Ciutat. La dimensió Categoria de vehicles consta de Tipus d'Ús i Tipus de Vehicle.

3.4.1.2 Nivell d'agregació

El nivell o grau d'agregació d'un atribut dins d'una dimensió bé determinat pel seu significat. Direm que un atribut té un nivell d'agregació més gran que un altre en el cas que el segon estigui inclòs en el significat del primer. Un exemple molt clar poden ser les dates: una data bé determinada per un any, un mes i un dia. Podem dir que un dia pertany o està inclòs dins d'un mes i al mateix temps dins d'un any. Al seu torn, un mes està inclòs dins d'un any però no dins d'un dia. Així, tindríem que l'atribut any de la data és el que té el nivell d'agregació més gran. A continuació, el mes tindria un nivell d'agregació intermedi i, finalment, el dia tindria el nivell d'agregació més baix de tots.

Els nivells d'agregació d'una dimensió són els possibles conjunts que podem fer dels atributs tals que per a tot element del conjunt i hagin tots els atributs amb un nivell d'agregació més gran que aquest dins la dimensió. En el cas de l'exemple, el de la dimensió de la data, els nivells d'agregació possibles serien els següents:

any

any, mes

any, mes, dia

4 Requeriments del sistema

L'objectiu d'aquest apartat és recollir, definir i analitzar les característiques que ha de tenir aquest projecte.

4.1 Requeriments funcionals

Els requeriments funcionals són aquelles operacions i serveis que ha de realitzar el software resultant del projecte. Tenint en compte que l'objectiu principal del projecte és la implementació del sistema ETL (veure els algorismes al capítol 7) per a cubs de dades multidimensionals utilitzant MapReduce i l'avaluació del seu rendiment, les funcionalitats que ha de tenir el projecte.

4.1.1 Creació dels índexs

S'ha de permetre la creació dels índexs per a les dimensions del cub de dades. Aquest procés consisteix a extreure informació de la taula de fets i guardar-la de tal manera que sigui possible obtenir la llista d'entrades de la taula que contenen un valor donat en una columna determinada.

Per a la creació de les dimensions, la comanda que es farà servir seguirà la següent gramàtica:

CREATE DIMENSION *nom_dimensió* ***ATTRIBUTES*** $nom\_columna^{+}$

On *nom_dimensió* és el nom de la dimensió per a la qual estem creant l'índex i $nom\_columna^{+}$ és el nom de les columnes que inclou la dimensió ordenades pel nivell d'agregació de gran a petit.

4.1.2 Realització de consultes

El sistema ha de permetre la realització de consultes sobre el cub de dades. Aquest procés inclou, la consulta a l'índex(depenent de l'algorisme), l'obtenció de les dades, l'agrupació i l'agregació. La comanda que es farà servir per a aquest tipus d'operació ha de complir la gramàtica següent:

SELECT $nom\_columna^{+}$ ***WHERE*** *nom_dimensió = valors* $[$***::*** *nom_dimensió = valors*$]^{*}$

GROUP BY $nom\_columna^{*}$

On *nom_columna*$^{+}$ és el nom dels diferents valors que volem obtenir, *nom_dimensió = valor* és una clàusula del predicat de la consulta i *nom_columna** és el conjunt de columnes respecte al valor de les quals es farà l'agrupació dels valors seleccionats.

Les clàusules del predicat de les consultes s'interpreten de la forma següent:

- Les clàusules sobre la mateixa dimensió han de tenir un efecte d'unió de conjunts.
- Les clàusules sobre dimensions diferents han de tenir entre elles un efecte d'intersecció de conjunts.

Els *valors* de les les clàusules del predicat han de seguir la gramàtica: *[valor%]*$^{+}$

Per exemple, si tenim una taula que conté informació sobre sabates com podria ser la següent:

model	**color**	**talla**	**preu**
s1	groc	33	30€
s2	vermell	33	40€
s3	verd	33	50€
s4	blau	34	60€

Taula 3: Taula d'exemple

Si fem servir la clàusula *talla = 33%* obtindríem els models de sabata *s1, s2, s3*. D'altra banda, si les clàusules del predicat fossin *talla = 33% :: color = vermell%*, el resultat seria el model *s2*, perquè les clàusules en dimensions diferents es comporten com la intersecció de conjunts. Finalment, si fem servir el predicat *talla = 33% :: color = vermell% :: color = groc %*, obtindrem els identificadors *s1* i *s2*, perquè les clàusules sobre la mateixa dimensió, en aquest cas el color, es comporten com la unió dels conjunts.

A les clàusules del predicat, a més, s'ha de poder especificar que a partir d'un nivell d'agregació donat volem totes les mesures la dimensió de les quals té es valors especificat fins al moment. Això ho farem acabant la restricció de la clàusula amb: [All%]. Per exemple, si fem una consulta amb el predicat *color = All%* obtindrem totes les entrades de la taula.

Així a les consultes hi hem de poder especificar quins valors volem obtenir de la taula de fets, quines restriccions volem que compleixin les dades obtingudes i finalment respecte a quins valors volem que s'agrupin les dades obtingudes.

Finalment, cal dir que l'operació d'agregació dels valors obtinguts ha de ser la suma. D'aquesta manera per a cadascuna les mesures que seleccionem, ens retornarà el valor d'aquest per a cada agrupació. Per exemple, si de la taula 3 obtenim el preu de les sabates, de les talles 33 i 34, agrupats per la talla utilitzant una consulta com la següent:

SELECT *preu* ***WHERE*** *talla = 33 :: talla = 34* **GROUP BY** *talla*

Obtindríem per resultat el preu de *120€* (*30€ + 40€ + 50€*) per a les sabates de la talla *33* i en canvi obtindríem *60€* per a les sabates de la talla *34*.

4.2 Requeriments no funcionals

Els requeriments no funcionals són els criteris que ens serviran per a saber en quina mesura s'han assolit les qualitats que ha de tenir el projecte. Aquest projecte té diversos requeriments no funcionals que són clau per al seu èxit. Els descric a continuació:

- S'ha d'implementar el projecte fent servir els algorismes explicats en el capítol Algorismes.
- L'eficiència és fonamental en tots els aspectes del projecte. S'ha de tenir en compte que estem treballant amb grans quantitats de dades i que les volem processar. S'ha de fer de manera eficient tant pel que fa a temps de CPU i espai ocupat a memòria com pel que fa a aprofitar al màxim l'oportunitat que ens dóna el clúster per a fer paral·lelisme.
- La usabilitat és la facilitat en què una persona pot usar el projecte. El programa executa les comandes introduïdes per l'usuari; per a aquesta finalitat amb una interfície textual en tindríem suficient, però no seria gaire usable ni amigable. És molt fàcil equivocar-se escrivint el nom dels atributs que interessen i escrivint el nom les taules que vols utilitzar, i molt frustrant per a l'usuari equivocar-se una vegada i una altra fins que aconsegueix escriure correctament tots els noms i es conegui la sintaxi de les comandes que he creat. És per aquesta raó que es fa necessària una interfície gràfica, per a facilitar la vida als usuaris i poder preparar les proves més fàcilment.
- És un projecte orientat a la computació distribuïda i concurrent, per tant, requereix un clúster per a la seva execució.

- El sistema ha de ser a prova de fallades. Com que es tracta d'un procés executat en diferents ordinadors concurrentment, és molt fàcil que algun d'ells falli o tingui algun problema. Per tant, el sistema ha de ser capaç de recuperar-se i seguir endavant.
- El sistema ha de ser fàcilment modificable i ampliable per tal de poder fer front als possibles imprevistos que puguin sorgir en el transcurs del desenvolupament del projecte.
- El projecte ha de ser programat en Java, el llenguatge natiu de les llibreries utilitzades en aquest projecte per tal de maximitzar-ne la compatibilitat.
- Es faran servir les llibreries Hadoop MapReduce 0.20.0 i Hadoop HBase[2] 0.20.6 pel desenvolupament del projecte.

4.3 Requeriments no tècnics

Els requeriments no tècnics són aquells que no estan inclosos ni ens els requeriments funcionals ni en els no funcionals, per exemple, són els requeriments que tracten temes del negoci, costs del projecte o terminis d'entrega. Per a aquest projecte, el requeriment no tècnic més important és el següent:

Els resultats obtinguts s'han de presentar al DaWaK 2011 (un congrés de Data Warehousing) abans del dimarts 12 d'abril del 2011. Inicialment aquest termini era el dimarts 29 de març, però fou aplaçat.

Aquest requeriment implica que per a poder fer les proves cal que el projecte estigui implementat i funcionant correctament abans de la data límit.

2 Inicialment la versió era 0.20.6 i el vam actualitzar a la versió 0.90.1.

5 Model de desenvolupament de software

Tot el procés de desenvolupament del projecte s'ha fet seguint la metodologia Agile[16]. Aquesta metodologia està ideada per a la realització de projectes petits i mitjans dels quals es necessiten resultats ràpids. Agile consisteix a fer petites iteracions i fer les proves al més aviat possible per tal de detectar els problemes abans i poder reaccionar amb més promptitud enfront dels imprevistos. El cicle de desenvolupament de software fent servir Agile és el següent:

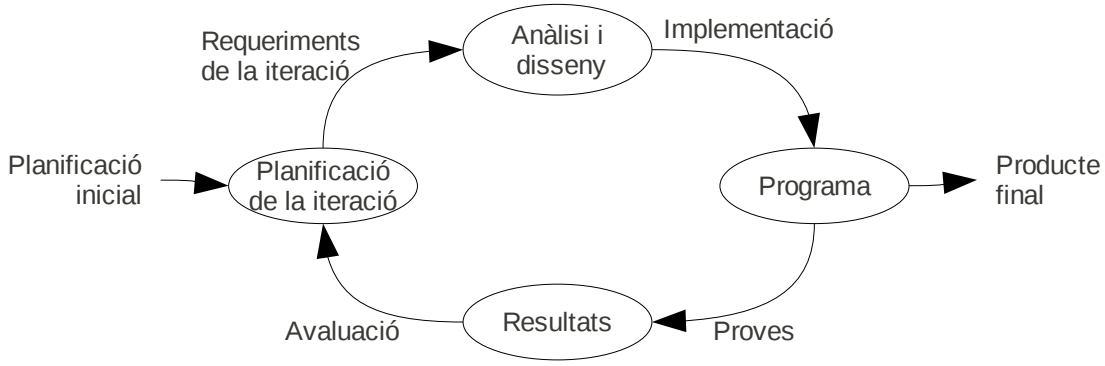


Figura 2: Cicle de desenvolupament d'Agile

Podem veure que els elements que apareixen en el cicle de desenvolupament són els següents:

- **Planificació inicial**: És el conjunt de requeriments funcionals, no funcionals i requeriments no tècnics.
- **Planificació de la iteració**: Consisteix a prioritzar els requeriments que tenim per tal d'assolir tots els objectius i escollir quins requeriments volem assolir en aquesta iteració.
- **Requeriments de la iteració**: És el conjunt de funcionalitats i característiques que volem tenir al final de la iteració.
- **Anàlisi i disseny**: És el procés en el qual es planifica com s'han de realitzar les tasques de la iteració per tal d'assolir els requeriments seleccionats.
- **Implementació**: Consisteix a programar tot el que s'ha especificat a l'etapa d'anàlisi i disseny.

- **Programa**: És l'executable que ha de complir els requeriments de la iteració actual.
- **Proves**: Consisteix a comprovar que el software obtingut satisfà els requeriments que ens hem proposat fins el moment.
- **Resultats de les proves**: És un conjunt de valors o informació que ens permet validar els objectius de la iteració.
- **Avaluació dels resultats**: Per saber en quina mesura es compleixen els requeriments objectius. També inclou una revisió dels requeriments del sistema, se'n poden afegir, modificar o eliminar.
- **Producte final**: és el software obtingut que satisfà els requeriments del sistema i que ha de satisfer les necessitats del client.

El model de desenvolupament Agile és un conjunt de metodologies de desenvolupament de software basat en un desenvolupament iteratiu i incremental. Els 12 principis del manifest d'Agile són els següents i veiem que s'adeqüen a les necessitats reals del projecte:

1. *La satisfacció del client ve donada per la ràpida entrega de software útil*: En el nostre cas la implementació havia de ser ràpida per tal de poder presentar els resultats al congrés.
2. *Ha de permetre canvis en els requeriments fins i tot en fases tardanes del desenvolupament*: S'han donat certes modificacions d'última hora, com per exemple, actualitzar les llibreries de l'HBase, calcular el factor de selecció per a poder fer les estadístiques...
3. *El software s'ha d'entregar en poques setmanes o mesos*: Aquest factor és clau per tal de poder tenir temps per fer les proves de rendiment per a l'article.
4. *El software funcional és la principal mesura del progrés*: És un punt clau per al desenvolupament del software, que sigui funcional perquè volem resultats ràpids.
5. *El desenvolupament del software s'ha de poder fer a un bon ritme*: Aquest aspecte ja es pot apreciar en el diagrama de Gantt de la planificació inicial del projecte (a l'apartat 8.2).

6. *Comunicació pròxima entre el negoci i els desenvolupadors*: Com que el desenvolupament l'he fet jo sol, no era un factor rellevant, però sí la comunicació que he mantingut amb els directors del projecte, que ha estat fluïda i fructífera.

7. *La conversa cara a cara és el millor mètode de comunicació*: La comunicació que he mantingut amb els directors del projecte ha estat majoritàriament en correus electrònics, però quan s'havien de tractar problemes o qüestions importants ens hem trobat per tal de tenir una millor comunicació.

8. *El projecte és desenvolupat per gent motivada i de confiança*: A més a més de la motivació de realitzar aquest Projecte Final de Carrera per tal de poder obtenir el títol, el tema d'aquest projecte m'ha atret i ha captat el meu interès i les meves ganes per tal de realitzar un bon treball.

9. *Cerca constant de l'excel·lència tècnica i del bon disseny*: Un requisit d'aquest projecte és la correcta implementació i fer-ho al millor possible per tal d'obtenir uns bons resultats i un bon rendiment.

10. *Simplicitat*: Vol dir que el projecte ha de ser un sol element connex, sense parts independents que podrien perjudicar la comunicació entre els desenvolupadors. El projecte només té un objectiu principal: provar i avaluar l'eficiència dels algorismes.

11. *Equips que s'autoorganitzen*: He posat a prova la meva capacitat d'organització enfront de les diferents dificultats que m'he anat trobant al llarg del desenvolupament de tot el projecte.

12. *Adaptació periòdica als canvis de circumstàncies*: Era necessària una capacitat d'adaptació molt gran per a fer front als possibles inconvenients i imprevistos que podien sorgir al llarg del desenvolupament. En aquest aspecte, ens ajuda molt el desenvolupament en iteracions que proporciona Agile.

6 Tecnologies emprades

En aquest apartat faig un petit resum de les diferents tecnologies que he fet servir en aquest projecte:

6.1 Apache HDFS

Apache Hadoop Distributed File System és el sistema de fitxers distribuït usat per les aplicacions de Hadoop. Distribueix les dades del sistema de fitxers entre els diferents nodes del clúster i fa rèpliques dels blocs de dades.

HDFS està pensat per a poder treballar amb arxius molt grans i ser al mateix temps tolerant a les fallades del maquinari dels nodes del clúster. Ofereix, també, una gran velocitat de lectura/escriptura de dades en detriment del temps d'accés (p.e. el temps d'obrir arxius). Està pensat, a més, per aplicacions en què s'escriuen les dades un cop i es consulten vàries vegades.

6.1.1 Arquitectura

HDFS ens ofereix una arquitectura d'un sistema de fitxers multinivell tradicional. És a dir, amb arxius, carpetes i subcarpetes. Actualment no permet la creació d'enllaços a fitxers ni a carpetes, però s'hi està treballant.

El sistema de fitxers està distribuït en els diferents nodes del clúster i es fa de forma transparent a l'usuari. Per fer-ho, HDFS fa servir una arquitectura de màsters i esclaus. El màster s'anomena NodeName i els esclaus DataNodes, que els explico a continuació:

- **NodeName**: són els nodes que gestionen les metadades del sistema d'arxius. Conté aquelles dades que informen sobre propietats dels arxius o carpetes com per exemple: el nom, la data de creació, el propietari, la mida... i en quin node es troben els diferents blocs de l'arxiu. El NodeName, a més, gestiona la replicació de blocs a dins del clúster.

  Un client es connectarà en un NodeName quan vulgui realitzar operacions de l'estil de crear, moure o eliminar un arxiu, canviar el nom d'un arxiu o bé per obtenir l'adreça del DataNode en el que ha de fer l'escriptura o lectura.

- **DataNode**: són els nodes que contenen els blocs d'informació dels arxius. Un client es connectarà en aquests nodes quan vulgui llegir o escriure el contingut d'un arxiu, les dades es transferiran de forma directa, sense passar per cap NodeName.

6.2 Apache Hadoop MapReduce

MapReduce és un model de programació (o framework) que ens permet generar i processar grans quantitats de dades. Permet a usuaris que no siguin experts en computació distribuïda executar processos que tractin grans quantitats dades en clústers.

Apache Hadoop MapReduce és una implementació d'Apache del model de programació utilitzat per Google[2].

6.2.1 Operacions

El model de programació MapReduce consisteix a fer un tractament de les dades en forma de clau/valor. El model consta, com a mínim, de dues funcions: una funció Map, una funció Reduce i opcionalment pot tenir l'operació Combine. L'usuari ha d'implementar aquestes operacions, configurar l'entrada d'on s'extrauran les dades per executar el procés i, també, especificar el lloc on es desaran els resultats obtinguts. Es pot modificar, però, qualsevol part del procés d'execució però com a mínim s'han d'implementar les operacions i configurar l'execució.

Per a les tres operacions que s'han d'implementar, les seves entrades i sortides seran parelles clau/valor. Per representar les parelles, al llarg de la memòria faré servir la notació:

<clau; valor>

Per cada parella d'entrada s'invocarà l'operació Map. Aquesta parella serà processada i l'operació Map escriurà o no una o diverses parelles clau/valor de sortida que anomenarem parelles intermèdies. A continuació, s'agruparan aquelles parelles intermèdies que tinguin la mateixa clau. Per cadascuna de les agrupacions fetes s'invocarà l'operació Reduce que rebrà, per cada clau, el conjunt de valors agrupats i emetrà al seu torn un conjunt de parelles clau valor.

Opcionalment es pot fer servir una operació Combine entre l'operació Map i l'operació Reduce. En aquest cas, l'operació Combine tindrà per entrada la sortida de l'operació Map i la

seva sortida serà l'entrada de l'operació Reduce. L'operació Combine també rebrà les parelles clau/valor, emeses a les operacions Map, agrupades pel valor de les seves claus però només d'aquelles operacions Map que s'han executat en el mateix ordinador què s'executa el Combine.

6.2.2 Arquitectura

El clúster d'ordinadors està compost per un màster (*JobTracker*) i treballadors. El màster, manté una llista de tots els ordinadors del clúster, els treballadors, i és l'encarregat d'assignar-lis les tasques que realitzaran.

Per executar una tasca de MapReduce el primer que fa el màster és dividir l'entrada fent servir el partidor que li hem proporcionat per a la tasca i separar-la en diferents parts. El partidor, és una classe que implementa un algorisme que separa les dades d'entrada del MapReduce segons un criteri determinat. Per exemple, si l'entrada del MapReduce és un arxiu de l'HDFS, el partidor per defecte farà una part per cada bloc de l'arxiu (els blocs són d'uns 250MB). Per cadascuna de les parts generades s'executarà una tasca Map.

A continuació, el màster transfereix les operacions Map, Reduce i Combine als diferents treballadors del clúster a més de indicar-los les dades que han de fer servir.

Cada treballador pot executar més d'una tasca Map, Combine o Reduce simultàniament, però es recomana que un ordinador, com a màxim, executi simultàniament tantes tasques com processadors tingui.

El màster inicialitza i configura la nova tasca i distribueix el codi que s'ha d'executar en cadascun dels treballadors. Un cop fet, cada treballador executarà les tasques Map que se li han assignat. Per executar cadascun dels Maps, primer haurà d'obtenir la part de les dades d'entrada que ha de fer servir (normalment es troben al HDFS). Un cop fet, processarà cadascuna de les parelles clau/valor llegides i generarà un conjunt de parelles intermèdies. Seguidament ordenarà les parelles intermèdies obtingudes, les agruparà per clau i les guardarà al disc. Si s'ha definit una operació Combine l'executarà per processar les parelles de sortida obtingudes de totes les operacions Map que s'han executat el mateix ordinador que el Combine.

En cada treballador, les sortides obtingudes, ja siguin de l'operació Map o Combine, s'agruparan ordenadament per clau. Finalment, si s'ha definit l'operació Combine es processarà el resultat obtingut amb aquesta operació i es tornarà a desar al disc. Cada treballador anirà informant al màster a mesura que vagi realitzant les diferents tasques per tal que el màster pugui portar un control sobre el progrés de la tasca.

A mesura que les diferents operacions de Map (o Combine si aquesta s'ha definit) van acabant, els diferents nodes del clúster distribueixen les dades cap els ordinadors que n'han de fer l'operació Reduce. Fan servir una funció de hash per decidir, segons el valor de clau de cadascuna de les parelles a, quin dels treballadors s'ha d'enviar cada parella.

Quan una tasca Reduce hagi rebut tota l'entrada que espera, ordenarà totes les parelles rebudes segons la seva clau i després les agruparà pel valor que tinguin a la clau. Després per cadascuna de les parelles executarà l'operació Reduce. Un cop fet, desarà el resultat on se li hagi indicat.

En cas que el màster perdi la connexió amb algun dels treballadors durant un cert temps, reassignarà les seves tasques a un altre treballador.

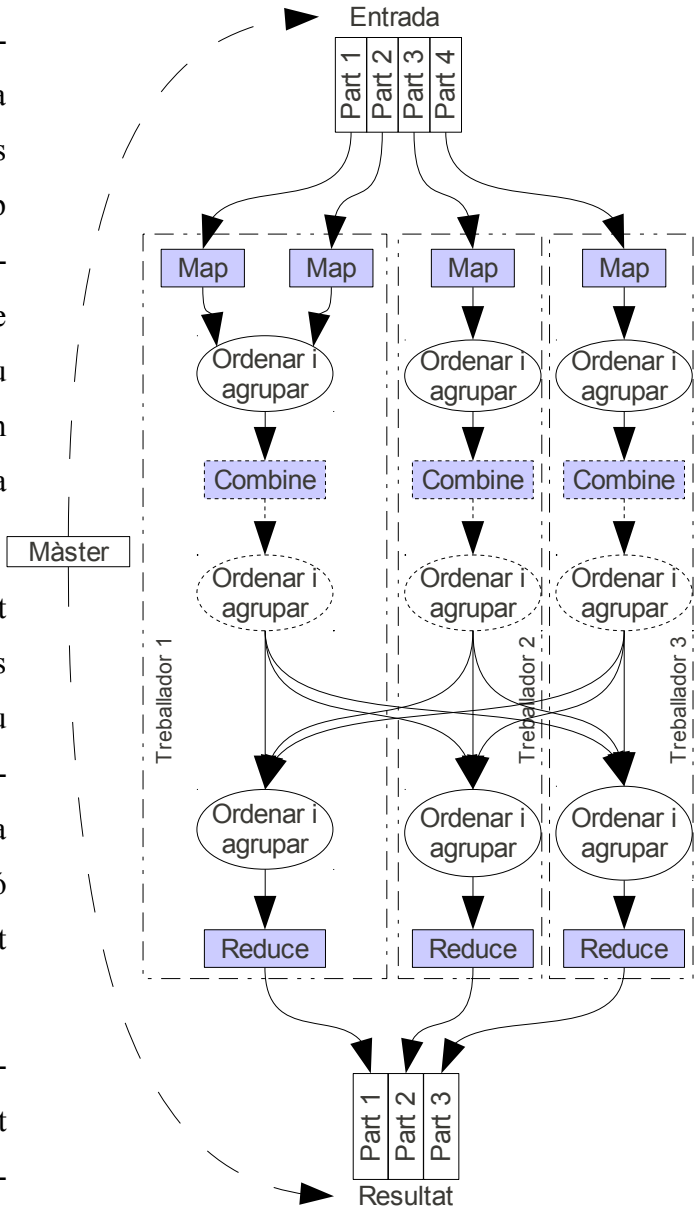


Figura 3: Procés d'execució d'un MapReduce

A la figura 3 es mostra un exemple en què tenim tres treballadors. Els elements marcats amb línies discontínues són les modificacions que apareixien en el procés en cas de què es

defineixi una operació Combine. Els elements que he marcat amb el fons de color gris, són les parts que ha d'implementar l'usuari.

A l'exemple que es mostra a la figura hi tenim quatre tasques Map de les quals se n'assignen dues al treballador u, una al treballador dos i l'última al treballador tres. Podem veure com cadascun d'ells obté les dades que necessita, les processa amb l'operació Map. A continuació, els resultats de l'operació Map són ordenats i agrupats segons valors de les claus de les parelles. Si s'ha definit l'operació Combine, aquesta agrupa les parelles d'entrada, les processa i els resultats s'ordenen i s'agrupen segons valor de la seva clau. Veiem que, a continuació, tenim tres tasques Reduce, una en cada treballador, les quals reben la informació que els correspon (segons la funció de hash) de les parelles intermèdies emeses per l'operació Map o bé la Combine si aquesta s'ha definit. Un cop executats els diferents Reduce, es desa el seu resultat.

El motiu de fer tantes ordenacions i agrupaments, és que després la primera ordenació, les altres ordenacions són per fusió i, a més a més, es poden fusionar les agrupacions fetes en cas de què tinguin la mateixa clau.

6.3 Apache HBase

Apache HBase és un gestor de bases de dades no relacionals distribuït que funciona sobre HDFS. Està pensat per a treballar amb taules de grans dimensions, de milers de milions de files amb centenars de famílies de columnes.

6.3.1 Model de dades

El model de dades usat per l'Apache HBase és semblant al usat pel Bigtable de Google [1]. La base dades està formada per taules i cada taula té files i columnes. Cada fila està identificada per un valor alfanumèric que anomenarem clau de la fila. Una taula pot tenir múltiples columnes, i aquestes es poden agrupar en famílies. Les famílies de columnes serveixen per a indicar que volem que els valors d'aquestes columnes es guardin junts per tal de millorar-ne el temps d'accés.

Les cel·les de la taula poden contenir qualsevol tipus d'informació, no tenen cap tipus de restricció de format ni de mida. El valor de cadascuna de les cel·les estarà determinat per la

clau de la fila, la columna i l'instant (*timestamp*) de la seva inserció. Vegem un exemple de taula d'HBase:

Clau	*Instant*	**Estadístiques**			**Errors**
		Altes	**Visites**	**Baixes**	
url_web1	t_1[3]	30	50	0	0
	t_2	20	90	2	5
	t_3	15	70	15	60
url_web2	t_2	12	30	15	11
	t_3	20	20	0	0

Taula 4: Aspecte d'una taula en HBase

Aquesta taula d'exemple, podria servir per a desar les estadístiques de diferents llocs web. Conté informació sobre les altes dels usuaris, nombre de visites, el nombre de baixes i el nombre d'errors que han tingut les pàgines web.

En primera instància podem veure que hi ha dues famílies de columnes: *Estadístiques* i *Errors*. La primera família conté 3 columnes: *Altes*, *Visites* i *Baixes*. Per contra, la família *Errors* només consta d'una columna.

En aquesta taula he disposat que hi hagi una fila per cada pàgina web; i que en cada fila es guardin les estadístiques de la pàgina web en diferents instants. S'ha de tenir en compte, a més, que la clau de la fila ha de ser única per evitar sobreescriure les estadístiques d'una pàgina amb les d'una altra perquè dues pàgines web poden tenir el mateix nom. Per això he fet que la seva clau sigui la URL de la pàgina web, perquè la URL és única i, per tant, s'eviten aquests tipus de conflictes.

6.3.2 Arquitectura

HBase ens ofereix una interfície per treballar amb el model de dades que he explicat a l'apartat anterior de forma distribuïda en un clúster, és a dir, la informació de les taules està repartida en diferents ordinadors. No obstant, accedint a HBase a través de l'aplicació o de la API ens apareixen les tuples ordenades per clau, en ordre descendent i de forma consecutiva. Físicament, però, les taules en trenquen per rangs de tuples que s'anomenen regions. Cada node del clúster pot contenir diferents regions de diferents taules.

3 Nomenclatura: t_i vol dir l'instant de temps i, t_i és abans de t_j si $i < j$.

El sistema d'HBase té tres tipus de nodes diferents: HBaseMaster, HRegionServer i HBaseClient. Els explico a continuació:

- **HBaseMaster**: S'encarrega d'assignar regions als HRegionServer. També és el responsable de dirigir operacions com canvis a l'esquema d'una taula, activar o desactivar taules o bé esborrar-les. Desactivar una taula serveix per deixar-la fora de línia i que no se li puguin fer consultes ni modificacions de les dades i per a poder afegir-hi o eliminar-hi famílies de columnes.
- **HRegionServer**: és el responsable de servir les peticions lectura i escriptura. Es comunica amb l'HBaseMaster periòdicament per indicar-li que encara segueix viu. S'aprofita aquesta comunicació per a què el màster li passi la llista de regions que ha de servir i altres informacions. Un HRegionServer s'encarrega de fer les següents operacions:
  - *Peticions d'escriptura*: quan rep una petició d'escriptura primer l'enregistra en el log i després guarda l'operació en una *cache* a memòria. Quan aquesta *cache* arriba a una mida preconfigurada es buida i es fan totes les escriptures en el MapFile corresponent (explico què és un MapFile més endavant). Un cop s'ha buidat la *cache* es marquen les escriptures com a fetes en el log.
  - Peticions de lectura: aquestes peticions es resolen mirant primer a la *cache* de memòria i si la informació buscada no es troba allà, llavors es buscarà en els MapFiles.
- **HBaseClient**: el servei que executa el client per interactuar amb HBase. L'HBaseClient s'encarrega de trobar quin és el HRegionServer que serveix el rang de tuples que ens interessa. Quan es crea un HBaseClient, aquest es comunica amb l'HBaseMaster perquè li digui la localització de la ROOT region. Un cop té localitzada la ROOT region el client contacta amb el RegionServer que conté la informació de la META Region. Un cop fet això, ja sap on ha de buscar la region que li interessa i l'obté.

Aquests components del sistema d'HBase fan servir un gran ventall d'arxius. A continuació explico els més rellevants:

- **META Table**: conté la llista de regions d'una determinada taula que hi ha en el sistema. Per cadascuna de les regions sap quins HRegion les gestiona i quines són la primera i la última clau de les tuples d'aquesta regió. També guarda per cada regió si està o no activada. La informació de la META Table es guarda en regions que es diuen META Region.
- **ROOT Table**: conté la direcció de totes les META Table que hi ha a la base de dades. Aquesta informació és guarda també en una regió anomenada ROOT Region.
- **MapFile**: és el tipus del fitxer que serveix per desar les regions. Ofereix un sistema persistent i ordenat per mapejar de claus a valors (on les claus i valors són cadenes arbitràries de bytes). Ofereix operacions per iterar sobre les diverses claus i accedir a una clau en concret o un rang concret.

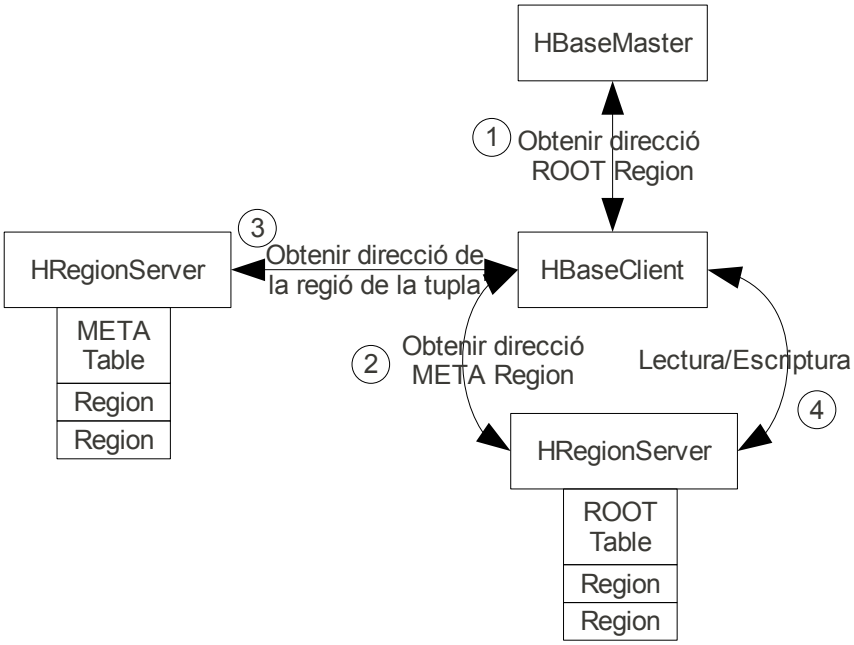


Figura 4: Lectura / Escriptura d'una tupla en HBase

A la figura 4 hi podem veure una configuració d'exemple en la que tenim un HBaseMaster, dos HRegionServer i un HBaseClient. Suposem que l'HBaseClient vol fer una lectura/escriptura d'una tupla. Els passos que seguirà són els següents:

1. El primer que farà serà connectar-se al HBaseMaster per descobrir quin HRegionServer té la ROOT Table. Aquest pas només es farà una vegada, cada vegada que s'encengui el client.

2. Llavors, demanarà al HRegionServer, que té la ROOT Table que li digui quin HRegionServer conté la META Table que té la informació de la taula que li interessa.
3. Accedim al servidor que té la META Table i li demanem quin HRegionServer té la tupla de la taula que ens interessa.
4. Finalment accedeix al HRegionServer i obté la tupla.

Cal fer notar que els HBaseClients tenen una *cache* per tal d'anar-se guardant les direccions i fer les consultes més eficientment.

7 Algorismes

L'objectiu principal del projecte és la implementació de consultes en cubs de dades multidimensionals. Hi ha tres tres algorismes de consulta: un d'ells realitza un escaneig del cub de dades (FSS) i els altres dos, l'IRA i IFS, fan servir índexs que s'han creat prèviament. A continuació mostro un diagrama en què apareixien tots els algorismes que he desenvolupat en el projecte i com es relacionen entre ells:

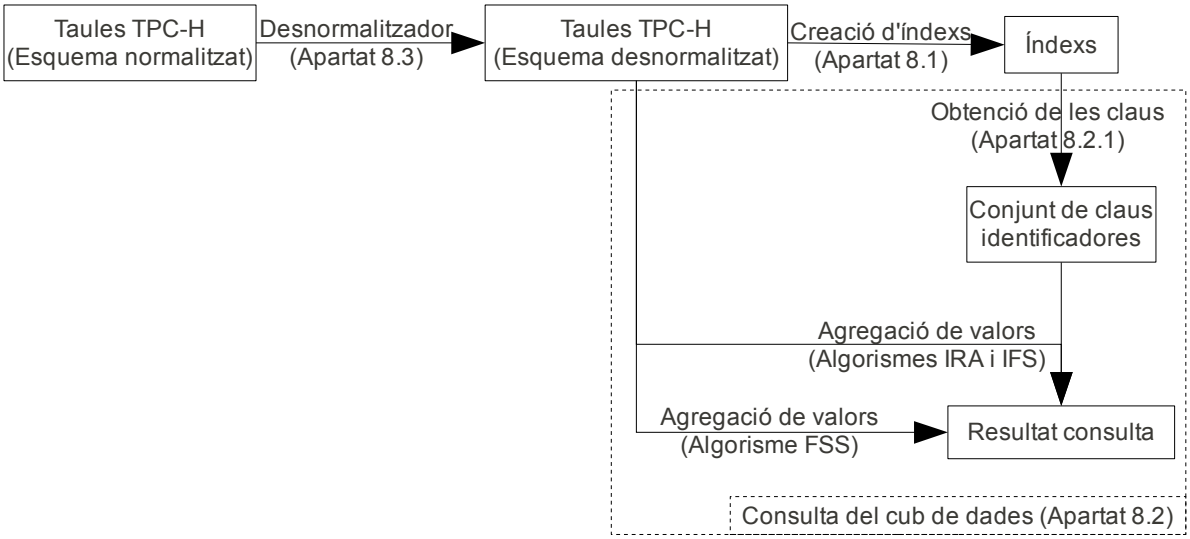


Figura 5: Flux del projecte

A la figura hi podem veure que partim de l'esquema de taules del TPC-H que inicialment està normalitzat. Com que els algorismes de consulta estan pensats per a esquemes desnormalitzats aplicarem un algorisme de desnormalització, explicat a l'apartat 7.3, per a obtenir-lo. S'ha optat fer els algorismes exclusius per esquemes desnormalitzats pels motius següents:

- Per eficiència: com que tens l'esquema desnormalitzat, t'estalvies de fer les joins a l'hora d'obtenir les dades (el cost temporal de les joins és elevat). D'aquesta manera fas el procés de desnormalització només un cop abans de començar a resoldre les consultes.
- Els algorismes estan pensats per a treballar amb cubs de dades que normalment usen esquemes desnormalitzats per motius d'eficiència (punt anterior).
- Treballar amb esquemes normalitzats suposaria incrementar la complexitat dels algorismes. Per exemple s'hauria de fer la join per poder trobar els valors que necessitessis.

- Els algorismes estan pensats per treballar en un entorn d'anàlisi de dades. Els entorns d'anàlisi de dades tenen l'avantatge que les dades no són modificades i, per tant, no es produiran anomalies.
- L'increment d'espai entre l'esquema normalitzat i desnormalitzat no és gaire elevat si utilitzem compressió de les taules (veure apartat 13.2.1).

Un cop tenim l'esquema desnormalitzat, podem procedir a la creació d'índexs per a les diferents dimensions del cub de dades. He implementat dos tipus d'índexs diferents que explico més endavant.

Finalment ja podem fer consultes sobre el cub de dades, bé, si fem servir l'algorisme de l'escaneig (FSS) no fa falta crear cap índexs. Per als dos altres algorismes, l'IRA i l'IFS, sí que es requereixen els índexs. Explico tots els algorismes amb més detall al llarg d'aquest capítol.

Abans, però, aclareixo uns temes de notació i expressió per tal de fer més fàcil i entenedora l'explicació dels algorismes:

- L'operació de concatenació que es fa servir, és sempre separant les diferents cadenes concatenades amb un caràcter que no aparegui en cap d'elles.
- En el pseudocodi he fet servir el símbol *&* per simbolitzar la concatenació. Serveix tant per concatenar Strings com per concatenar llistes.
- Quan un llista es converteix a String, el resultat és la concatenació de tots els seus elements en forma de String.

A continuació explico els algorismes utilitzats i n'incloc, també, el seu pseudocodi:

7.1 Creació d'índexs

La creació dels índexs és molt important per a poder localitzar ràpidament les dades que ens interessen i poder resoldre eficientment les consultes. Una bona forma de crear índexs per a un cub de dades és fer-ne un per a cadascuna de les seves dimensions.

Així, crearem un índex per a cada dimensió del cub de dades. Per fer aquests índexs, per a cada tupla de la taula de fets s'agafen els valors que conformen la dimensió i el seu identifica-

dor. Seguidament, s'agrupen els identificadors de les tuples que tenen els mateixos valors en la dimensió i es desen en la taula d'índexs.

Suposem que la dimensió *d* està composta pels valors de les columnes $c_1, c_2, ..., c_n$ de la taula de fets. Cal dir, també, que les columnes $c_1, c_2, ..., c_n$ estan ordenades segons el seu nivell d'agregació de gran a petit, és a dir, que per una columna c_i i una columna c_j, c_i té un nivell d'agregació més gran que c_j si $i < j$. Les claus de les files són $k_1, k_2, ...$. Tenint en compte aquestes consideracions, podem dir que l'índex ha de contenir informació amb l'estructura següent:

$$d\%c_1\%c_2\%...\%c_n => k_1,k_2,k_3$$

El nom de la dimensió *d* l'he posat al principi de la clau de la fila per tal de poder distingir quines entrades de la taula d'índexs corresponen a cada dimensió, i a més a més, per a què les entrades de la mateixa dimensió quedin en posicions consecutives[4] i aprofitar la localitat espacial en el moment d'accedir-hi.

Cal fer notar també, que les entrades de l'índex quedaran agrupades, a més, pels valors de la dimensió. Les entrades d'una dimensió que tenen el mateix valor a la columna c_1 quedaran consecutives en l'índex; totes en les que coincideixin c_1 i c_2 també, i així per a totes les columnes.

He implementat dos tipus diferents d'índexs: l'un conté per cada combinació de valors les claus de la taula de fets que li corresponen i l'altre conté totes les claus per cada nivell d'agregació. Els explico a continuació:

7.1.1 Índex 1-nivell

Aquest tipus d'índex guarda només les agrupacions de claus que es corresponen al nivell més baix d'agregació de la dimensió. D'aquesta manera, es pretén desar només la informació necessària i com que les tuples estan ordenades en ordre-alfanumèric, totes les dimensions queden agrupades i, a més a més, queden agrupades per cada nivell d'agregació. En contrapartida, si s'han d'obtenir les claus per a un nivell d'agregació alt és possible haver de fer un recorregut llarg.

4 Cal tenir present que HBase manté les tuples ordenades per ordre lexicogràfic creixent del valor de les seves claus. Veure Apache HBase, apartat 6.3.

Aquest tipus d'índexs està pensat per resoldre consultes en les clàusules de les quals s'especifiquin tots els nivells d'agregació de les dimensions. Perquè en cas que no s'hagin especificat tots els nivells, s'hauran d'obtenir múltiples entrades de l'índex que resultarà més lent que obtenir una única entrada. S'obtindran totes aquelles files que tinguin tots els valors especificats en una de les clàusules. Aquest fet però, com veurem a en el capítol de proves no influeix notablement en el rendiment (veure l'apartat 13.2). D'altra banda, si que es pot apreciar una reducció significativa de l'espai necessitat per l'índex respecte els índexs multinivell.

Mostro i explico el pseudocodi de l'algorisme de la creació d'índexs d'un sol nivell:

Pseudocodi

```
MAPREDUCE
input: <key; value>: fact table
    dimension_components: column names from the fact table that
compose the dimension ordered by semantic level of aggregation
    dimension_name: name of the dimension to be created

output: <key; value>: index entries

function Map
input <Row_ID, List Row_Values>
output <Dimension_Key, Row_ID>
    List dimension_values := createNewList()
    for each component in dimension_components do
        dimension_values.append(Row_Values.getValue(component))
    done
    EmitIntermediatePair(<dimension_name&AsString(dimension_values),
Row_ID>)
end function

function Combine
input <Dimension_Key, List Row_ID>
output <Dimension_Key, Row_IDs>
    EmitPair(<Dimension_Key, AsString(Row_ID)>)
end function

function Reduce
input <Dimension_Key, List Row_ID>
output <Dimension_Key, Row_IDs>
    EmitPair(<Dimension_Key, AsString(Row_ID)>)
end function
```

El pseudocodi de l'operació Combine és el mateix que el de la operació Reduce.

Explicació

Com podem veure, he implementat l'algorisme utilitzant un MapReduce. Aquest té per entrada la taula que conté el cub de dades d'origen. També té per entrada el valor *dimension_components* que és el conjunt d'atributs que conformen la dimensió per la qual es crea l'índex i el seu ordre relatiu (ordenats pel nivell d'agregació). S'ha definit, també, com a valor d'entrada *dimension_name* que conté el nom de la dimensió que estem creant.

L'operació Map rep les tuples de la taula de fets d'una en una i per a cadascuna d'elles concatena el nom de la dimensió amb els valors de cada nivell d'agregació. Ho fa de la següent forma:

$$dimensió\%valor_1\%valor_2\%valor_3$$

On *dimensió* és el nom de la dimensió que s'està creant i els diferents valors estan ordenats per ordre de nivell d'agregació, de més agregació a menys agregació. L'ordre és el mateix que el nom de les columnes especificades la variable d'entrada *dimension_components*. Aquest ordre vol dir que primer es posarà el valor de l'atribut té un significat més ampli i cada cop seran més concrets. Vegem-ne un exemple clarificador:

$$data\%any\%mes\%dia$$

En aquest cas, per a la dimensió *data*, l'*any* seria l'element amb el nivell d'agregació més alt perquè inclou diversos *mesos* i *dies*. En canvi, *dia* seria l'element de menor nivell d'agregació perquè a un *dia* concret només li correspon un *any* i un *mes*.

Un cop feta la concatenació de valors, l'operació Map escriu com a parella intermèdia la concatenació realitzada i com a parella la clau de la tupla corresponent. De forma genèrica ho expressaríem de la forma següent:

$$<dimensió\%valor_1\%valor_2\%valor_3;\ clau>$$

Ara és el moment de l'operació Combine. Aquesta operació rep per entrada les parelles intermèdies agrupades pel valor de la seva clau. L'únic que ha de fer l'operació Combine és, per cada agrupació que rebi, emetre una parella que tingui per clau la clau de la parella rebuda i per valor la concatenació dels valors del grup. L'operació Reduce rebrà agrupades les parelles emeses per les operacions Combine i realitzarà sobre aquestes el mateix procés. Així obtindrem parelles amb el següent estil:

$<dimensió\%valor_1\%valor_2\%valor_3;\ clau_1\ clau_2\ clau_3\ clau_4>$

Com podem veure, queden agrupades les claus que tenen els mateixos valors en les dimensions.

El conjunt de parelles emeses pel MapReduce es guarda a la taula dels índexs de l'HBase perquè posteriorment pugui ser consultat.

7.1.2 Índex multinivell

L'índex multinivell és un índex que conté tots els nivells d'agregació per a cadascuna de les dimensions definides, de la més gran a la més petita. Per exemple, suposem una dimensió anomenada *data* que té els atributs *any*, *mes* i *dia*. Llavors, per a l'índex multinivell s'afegiria una entrada per cada *any* que aparegués a la taula de fets amb totes les claus de les files de la taula de fets que es corresponen amb aquest *any*. També, afegiríem, per cada *any* i *mes*, una nova entrada identificada per l'*any* i el *mes* i per valor totes les claus associades a cada *mes* dins de cada *any.* Finalment, també afegiríem a l'índex una entrada per cada *dia* de cada *mes* de cada *any* identificada per aquesta data i per valor el conjunt de claus de fila de la taula de fets que es corresponen amb aquesta data.

Aquest tipus d'índex està pensat per a poder trobar totes les files de la taula de fets que compleixen una certa restricció llegint únicament una entrada de l'índex. Aquest tipus d'índex té un cost espacial més gran que l'índex d'un sol nivell i no aporta un augment de rendiment perceptible segons els resultats obtinguts a les proves.

A continuació presento el pseudocodi de l'algorisme per a la generació d'aquests tipus d'índexs:

Pseudocodi

```
MAPREDUCE
input: <key; value>: fact table
    dimension_components: contains the column names of the fact table
that compose the dimension ordered by the semantic level of
aggregation
    dimension_name: name of the dimension to be created

output: <key; value>: index entries

function Map
input <Row_ID, List Row_Values>
```

```
output <Dimension_Key, Row_ID>
    List dimension_values = createNewList()
    for each component in dimension_components do
        EmitIntermediatePair(<dimension_name&AsString(dimension_values
), Row_ID>)
        dimension_values.append(Row_Values.getValue(component))
    done
    EmitIntermediatePair(<dimension_name&AsString(dimension_values),
Row_ID>)
end function

function Combine
input <Dimension_Key, List Row_ID>
output <Dimension_Key, Row_IDs>
    Emit<Dimension_Key , AsString(Row_ID)>
end function

function Reduce
input <Dimension_Key, List Row_ID>
output <Dimension_Key, Row_IDs>
    Emit<Dimension_Key, AsString(Row_ID)>
end function
```

El codi de les operacions Combine i Reduce és el mateix que el de l'algorisme per a la creació l'índex d'un sol nivell (apartat 7.1.1).

Explicació

Podem veure que els valors d'entrada del MapReduce són els mateixos que en l'algorisme anterior: la taula de fets, el valor *dimension_components* és el conjunt d'atributs que conformen la dimensió i el valor *dimension_name* conté el nom de la dimensió que estem creant.

L'operació Map rep per entrada cadascuna de les files de la taula de fets, però de cada fila només rep les columnes que conformen la dimensió, és a dir, només les columnes llistades a la variable *dimension_components*. Per cada fila que rep emet una parella intermèdia per cadascun dels diferents nivells d'agregació de la dimensió. Aquesta parella intermèdia tindrà per clau el nom de la dimensió concatenat amb els valors de les columnes del nivell d'agregació i per valor la clau de la fila de la taula de fets que es correspon. És a dir, que per cada fila que rebem, si aquesta té una clau k i un conjunt de valors en cadascuna de les columnes de les dimensions $v_1, v_2, ..., v_d$ (ordenats per nivell d'agregació) i on d és el nombre d'atributs que té la dimensió, emetrem com a parelles intermèdies:

$<nom\_dimensió\%v_1;\ k>$
$<nom\_dimensió\%v_1\%v_2;\ k>$

...
$<nom\_dimensió:v_1\%v_2\%...\%v_d;\ k>$

A continuació, toca l'execució de l'operació Combine i de l'operació Reduce. Les explico de cop perquè tenen el mateix codi. Aquestes operacions, reben les parelles emeses per l'operació Map (o per la Combine en cas de ser el Reduce) agrupades pel valor de la seva clau i el que fan és emetre una nova parella amb la mateixa clau i per valor la concatenació de les claus de tuples rebudes. És a dir, que després de l'execució de l'operació Combine i Reduce tindrem totes les claus de les files agrupades respecte els valor de les dimensions per cadascuna de les possibles agregacions. Un exemple de parella amb aquestes característiques seria la següent:

$<nom\_dimensió:v_1\%v_2\%...\%v_n;\ k_1,k_2,k_3>$

Aquesta parella indicaria que el conjunt de files de la taula de fets que tenen per claus k_1,k_2,k_3 tenen en comú els valors v_1, v_2, ..., v_n de les columnes de les dimensions amb $n \leq d$, és a dir, les claus del nivell n-èssim de l'agregació.

El conjunt de parelles emeses pel MapReduce es guarda a la taula dels índexs per a què posteriorment puguin ser consultats.

7.2 Consulta del cub de dades

Donada una consulta amb un conjunt de clàusules, el primer que s'ha de fer és accedir a l'índex i obtenir les entrades que satisfan les restriccions de la consulta. Quan tinguem les entrades de l'índex ja sabrem quines són les files de la taula de fets que ens interessen, concretament, tindrem les claus de les files que hem d'agafar de la taula de fets.

Un cop tenim els identificadors a buscar, accedim a la taula de fets i en prenem les files que ens interessen. D'aquestes files, se n'agafen els valors de les columnes que hem de cercar i s'agrupen segons el valor de les columnes que hem de fer servir per fer l'agrupació. Totes les columnes estan indicades en la consulta.

Aquesta funcionalitat es fa en dues parts: una que obté els identificadors de les files que ens interessen i l'altre que accedeix a la taula de fets per obtenir la informació que realment

ens interessa i l'agrupa. També explico l'algorisme FSS que no consulta els índexs. A continuació explico els algorismes per a dur a terme aquests processos:

7.2.1 Obtenció de les claus

Aquesta part de l'algorisme accedeix a l'índex i n'obté les claus de les files que s'han de cercar de la taula de fets.

D'aquest algorisme n'he fet dues implementacions diferents. Les seves funcionalitats són molt semblant, però se n'esperen rendiments diferents. El seu objectiu és, donada una consulta, obtenir de l'índex els identificadors de les files de la taula de fets que ens interessen. Les dues implementacions varien en la forma en què s'obtenen els identificadors, ja que una està pensada per a fer servir índexs multinivell i l'altra per índexs d'un sol nivell.

A continuació explico aquestes implementacions amb més detall:

7.2.1.1 Accés a l'índex d'un sol nivell

Aquest algorisme està pensat per a treballar amb índexs d'un sol nivell. Per resoldre una consulta aquest algorisme fa el següent: obté les entrades de l'índex que contenen els identificadors de la taula de fets que satisfan les clàusules. A continuació, es prenen només els identificadors de les files que satisfan com a mínim una clàusula de cadascuna de les dimensions que apareixen dins de les clàusules de la consulta.

Com que l'índex és d'un sol nivell, per a les clàusules que no s'han restringit valors pels nivells més baixos d'agregació s'ha de fer un accés per rang. Per exemple, si treballem amb la dimensió *data* composta per l'*any*, el *mes* i el *dia*. Per exemple, si tenim les següents entrades a l'índex:

$$<data\%2010\%01\%04;\ clau_5,\ clau_6>$$
$$<data\%2011\%01\%04;\ clau_1,\ clau_2>$$
$$<data\%2011\%02\%05;\ clau_3,\ clau_4>$$

Si volem obtenir les claus d'aquelles files que la seva data és de l'any 2011 haurem de fer un accés per rang. Podem fer un accés per rang eficientment perquè a l'HBase, tal com ja he explicat, es guarden les files ordenades lexicogràficament pel seu identificador, per tant, les entrades de l'índex referents a la mateixa dimensió i amb els mateixos valors en els *n* primers

nivells d'agregació es guarden consecutivament. Així, agafant totes les files que comencin per *data%2011* ja en tenim prou, i com que són consecutives ho podem fer amb un sol accés de forma eficient.

D'altra banda, per obtenir les claus que satisfan una restricció en la qual s'han especificat valors per a tots els nivells de la dimensió ho farem amb un sol accés a la taula. En l'exemple proposat, si volem trobar les claus de l'índex les quals la seva data és el 4 de gener del 2011, agafant la segona fila n'hi ha prou.

Per a implementar aquest algorisme he fet servir MapReduce. El seu pseudocodi és el següent:

Pseudocodi

```
MAPREDUCE1
input: <key; value>: file with the index entries to fetch
    condition_dimensions: number of distinct dimensions to fulfill the
restrictions
output: <key; value>: intermediate table/file

function Mapper::Run
input <File>
output -
    cIndex := HBase.connect(IndexTable)
    for each line in File do
        Map(line)
    done
end function

function Mapper::Map
input <Line#, Line>
output <Row_ID, Dimension_Name>
    if hasValueForEachDimensionLevel(Line) then
        IDs := cIndex.getRow(Line)
    else
        IDs := cIndex.getRange(Line, Line & "~")
    end if
    for each id in IDs do
        EmitIntermediatePair(id, Line->dimensionName)
    done
end function

function Reduce
input <Row_ID, List Dimension_Names>
output <Row_ID>
    if count_distinct(Dimension_Names) = condition_dimensions then
        EmitPair(Row_ID)
    endif
end function
```

Explicació

El valor d'entrada *condition_dimensions* conté el nombre de dimensions diferents que apareixen a les clàusules de la consulta.

En el pseudocodi veiem que de la classe Mapper n'implementem dues funcions: Run i Map. L'operació Run conté el codi on s'executa, dins d'un bucle, l'operació Map. L'operació Map és la funció que s'executa per cada de línia de l'arxiu de clàusules que té el MapReduce per entrada.

L'operació Run, és una operació que s'executa en cada node del clúster que ha de processar les parelles d'entrada. Aquesta rep d'entrada les parelles amb què s'han d'invocar les operacions Map. La seva funció principal és invocar l'operació Map per a processar cadascuna d'aquestes parelles.

En aquest MapReduce he reimplementat l'operació Run de tal forma que: la connexió a la taula que conté l'índex a l'HBase s'estableixi només un cop i, així, fer servir aquesta connexió per processar totes les parelles. Si no ho féssim, hauríem d'establir la connexió en cada crida de l'operació Map cosa que seria molt més costosa. D'aquesta manera es veu clara la necessitat de reimplementar l'operació Run. En els casos que hem vist fins ara no ha fet falta reimplementar-la perquè per defecte, el que fa es invocar l'operació Map per cada parella que té d'entrada, cosa que és el seu funcionament normal.

Cada cop que s'invoca l'operació Map, aquesta rep una línia de l'arxiu de clàusules. Aquesta línia conté una restricció formada per: el nom de la dimensió i valors especificats pels *n* primers nivells d'agregació. Si en la clàusula s'ha especificat un valor per a cadascun les components de la dimensió, farem un accés directament a la taula per prendre l'entrada que ens interessa aprofitant la connexió feta a l'operació Run. D'altra banda, si només s'ha especificat un valor pels atributs amb un nivell d'agregació més elevat dins de la dimensió, haurem de fer un accés per rang. Per fer aquest accés, hem d'indicar la primera fila que volem (inclosa) i fins a quina fila ens interessa (no inclosa). Per aquest motiu, s'ha indicat en el pseudocodi que l'última fila a buscar és la *Line & "~"* ja que el caràcter '~' és posterior als caràcters que apareixen en les dimensions i, per tant, estarem indicant una fila que ens garanteix que agafem totes les tuples del rang que ens interessa i cap altra.

A continuació, per totes les claus de fila llegides de l'índex emetem una parella intermèdia que té per clau l'identificador de la tupla i per valor el nom de la dimensió sobre la qual s'aplica la clàusula.

Les parelles intermèdies que emet l'operació Map tenen el format següent:

$$<clau_1; dimensió>$$

L'operació Reduce rebrà per entrada les parelles emeses per l'operació Map agrupades per clau. En el nostre cas seria:

$$\begin{cases} <c1; dimensió_1> \\ <c1; dimensió_2> \end{cases}$$
$$\{<c2; dimensió_1>$$
$$\{<c3; dimensió_2>$$

Així, per cada clau de la taula de fets obtinguda a través de l'índex es podrà comprovar que satisfaci com a mínim una de les clàusules de cada dimensió. Ho farem comprovant que el nombre de dimensions diferents és igual al valor de la variable *condition_dimensions*. Finalment, l'operació Reduce emetrà les claus que compleixin com a mínim una clàusula sobre cada dimensió.

La sortida de l'operació Reduce es desarà en un arxiu si després s'executa l'algorisme IFS o bé en una taula si després s'executa l'algorisme IRA. Per referir-me a l'arxiu ho faré amb el nom d'arxiu intermedi i per referir-me a la taula ho faré amb el nom de taula intermèdia.

7.2.1.2 Accés a l'índex multinivell

Per resoldre una consulta fent servir índexs multinivell hem de fer el següent: fer un accés a l'índex per cadascuna de les clàusules que tingui la consulta i agafar-ne l'entrada que hi correspongui. D'aquesta manera obtindrem per cada clàusula quines entrades de la taula de fets la satisfan. Finalment, només ens haurem de quedar els identificador que satisfacin com a mínim una clàusula de cadascuna de les dimensions que apareixen amb clàusules a la consulta.

S'ha fet la implementació d'aquest algorisme fent servir MapReduce. L'entrada de l'operació Map són les línies d'un arxiu que conté les clàusules que s'han de resoldre. L'operació Map anirà rebent les línies de l'arxiu una per una i les anirà processant. La sortida de l'operació Reduce és el conjunt de claus de les files de la taula de fets que satisfan les clàusules.

El pseudocodi de la implementació que n'he fet és el següent:

Pseudocodi

```
MAPREDUCE – Multilevel index access
input: <key; value>: file with the index entries to fetch
    condition_dimensions: number of distinct dimensions to fulfill the
restrictions
output: <key; value>: intermediate table/file


function Mapper::Run
input <File>
output -
    cIndex := HBase.connect(IndexTable)
    for each line in File do
        Map(line)
    done
end function

function Mapper::Map
input <Line#, Line>
output <Row_ID, Dimension_Name>
    IDs := cIndex.getRow(Line)
    for each id in IDs do
        EmitIntermediatePair(id, Line->dimensionName)
    done
end function

function Reduce
input <Row_ID, List Dimension_Names>
output <Row_ID>
    if count_distinct(Dimension_Names) = condition_dimensions then
        EmitPair(Row_ID)
    endif
end function
```

Explicació

La variable d'entrada *condition_dimensions*, com en els casos anteriors, conté el nombre de dimensions diferents que apareixen en les clàusules de la consulta.

Podem veure que implementem dues funcions de la classe Mapper: la funció Run i la funció Map.

Com en el cas anterior, la reimplementació de l'operació Run és necessària. En aquesta operació establim la connexió a la taula que conté l'índex a l'HBase. Primer s'estableix la connexió i després s'invoca la funció Map per cada de línia de l'arxiu que hem de processar.

Cadascuna d'aquestes línies són les diferents clàusules que apareixen a la consulta que s'està executant.

Dins de l'operació Map, obtenim l'entrada de l'índex que es correspon amb la clàusula rebuda per paràmetre. Aquesta entrada conté els identificadors de les tuples de la taula de fets que tenen els valors que estem buscant. Seguidament, per totes les claus de fila obtingudes emetem una parella intermèdia que té per clau l'identificador de la tupla i per valor el nom de la dimensió sobre la qual s'aplica la clàusula.

L'operació Reduce és la mateixa que en el cas anterior.

7.2.1.3 Versió anterior

Aquesta versió és la implementació inicial de l'algorisme per a la obtenció de les entrades de l'índex. Estava pensada per a treballar amb índexs d'un sol nivell però és fàcilment modificable per a què pugui treballar amb índexs multinivell.

Per a aquesta implementació, es creava un filtre per escanejar tota la taula de l'índex i que obtenia les entrades que ens interessen per a resoldre la consulta. El resultat de l'execució d'aquest filtre era l'entrada del MapReduce. Aquest MapReduce, com en els casos anteriors, per a cada entrada obtinguda de l'índex emetia les parelles intermèdies amb clau l'identificador d'una fila de la taula de fets i per valor el nom de la dimensió.

Aquesta implementació és clarament menys eficient que les anteriors en els casos que el factor de selecció sobre l'índex és suficientment petit, ja que fa un recorregut de la taula dels índexs enlloc d'agafar només la informació que ens interessa mitjançant accessos aleatoris.

A continuació mostro el pseudocodi del MapReduce:

Pseudocodi

```
MAPREDUCE 1
input: <key; value>: dimension table (filtered values)
    condition_dimensions: number of distinct dimensions to fulfill the
restrictions
output: <key; value>: intermediate table/file

function Map
input <Dimension_Values, Row_IDs>
output <Row_ID, Dimension_Name>
    for each id in Row_IDs do
        EmitIntermediatePair(id, Dimension_Values->dimensionName)
```

```
    done
end function

function Reduce
input <Row_ID, List Dimension_Names>
output <Row_ID>
    if count_distinct(Dimension_Names) = condition_dimensions then
        EmitPair(Row_ID)
    endif
end function
```

Explicació

El MapReduce té per entrada les tuples de l'índex que necessita per a resoldre les condicions. Per obtenir les tuples que ens interessen de taula d'índexs fem servir un filtre de tuples. Aquest simplement ha d'agafar les tuples que compleixin una determinada condició. La condició que imposem nosaltres és que tinguin una clau de fila determinada. La restricció sobre la clau de fila ha d'anar d'acord amb les clàusules que hem posat a la consulta. Si restringim per a una determinada dimensió uns valors concrets, per exemple:

$$dimensió\%valor_1\%valor_2\%valor_3$$

Així, buscarem a la taula de l'índex les tuples que tinguin una clau de fila que comenci per aquest patró. Així, obtindrem totes les entrades de la taula de condicions que compleixen la clàusula que ens interessa.

Com que ja rebem d'entrada només les tuples que en interessen ja no cal que fem càlculs complexes ni operacions costoses per escollir quines són les bones. Per tant, no cal que reimplementi l'operació Run per agilitzar el procés tal com he fet en els casos de més amunt.

També tenim com a valor d'entrada *condition_dimensions* que conté el nombre de dimensions que tenen clàusules en la consulta.

L'operació Map rebrà les entrades de la taula de l'índex que ens interessen. Per identificador de la taula de fets obtingut emetem una parella intermèdia que té per clau l'identificador de la tupla i per valor el nom de la dimensió sobre la qual s'aplica la clàusula.

L'únic que ha de fer el Reduce, com en els casos anteriors, és comptar el nombre de dimensions diferents que rep per a cadascuna de les claus i si aquest nombre és el mateix que el nombre de dimensions diferents que apareixen a les clàusules de la consulta (contingut a *condition_dimensions*) voldrà dir que aquesta clau compleix les clàusules de la consulta. Les

claus que compleixin les condicions seran emeses com a sortida de l'operació Reduce i s'escriuran a la taula intermèdia o bé a l'arxiu intermedi.

7.2.2 Obtenció dels valors

Aquesta part del procés té per objectiu accedir als valors que ens interessen de la taula de fets, agrupar-los i obtenir la suma per cadascun de les agrupacions, tal com se'ns indica a la consulta. Per a fer aquesta funcionalitat he implementat tres algorismes diferents: l'Index Random Access i l'Index Filtered Scan i el Filtered Source Scan.

7.2.2.1 Index Random Access (IRA)

Aquesta implementació consisteix a fer un accés aleatori als diferents valors que ens interessen de la taula de fets partint dels identificadors de tuples obtinguts de l'etapa anterior. L'entrada de l'operació Map és la taula intermèdia que conté els identificadors de les files de la taula de fets que hem de buscar. Abans d'iniciar el MapReduce, però, es fa un recorregut sobre la taula intermèdia que fusiona els identificadors de fila consecutius en una sola fila. Suposem que conté els identificadors[5]:

1, 2, 5, 8, 9, 10

Després de fer l'agrupació quedarien de la forma següent:

1:2, 5, 8:10

En el cas de l'exemple, podríem obtenir totes les entrades que ens interessen de la base de dades amb només 3 accessos enlloc de 6 tal com haguéssim fet si no agrupéssim els valors consecutius. Fer aquesta agrupació té un cost lineal perquè les tuples es guarden ordenades dins la taula pel seu identificador i, per tant, amb un sol recorregut en tenim prou. Les modificacions que s'han de realitzar a sobre la taula també tenen un cost reduït, perquè HBase ens ofereix accés aleatori a totes les tuples de les taula.

Un cop acabat el recorregut de la taula intermèdia s'inicia el MapReduce. Dins de l'operació Map, per cada identificador que es rebi o per cada rang d'identificadors, es farà un accés aleatori a la taula de fets per tal d'obtenir els valors que estem buscant. Després, s'agruparan els valors obtinguts segons el valor de les columnes per les quals estem fent l'agrupació.

5 En aquest exemple les files estan separades per comes.

Finalment, amb les operacions Combine i Reduce, es durà a terme l'agregació de valors per a cadascuna de les agrupacions resultants.

Aquest algorisme, tal com comprovarem al capítol de proves, està pensat a resoldre ràpidament consultes amb un factor de selecció baix. També està pensat per aquelles consultes en què les files de la taula de fets a cercar quedin properes i es puguin agrupar en un nombre reduït de grans grups, és a dir, en els casos que tinguem localitat espacial. Val a dir, que aquest segon cas també està condicionat a què el factor de selecció sigui baix, perquè si és gaire alt equivaldrà a un escaneig de la taula i en aquests casos són millors els algorismes IFS i FSS.

El pseudocodi d'aquest algorisme és el que mostro a continuació:

Pseudocodi

```
MAPREDUCE2 – Index Random Access
input: <key; value>: intermediate table
    selected_columns: column names from the fact table to aggregate
    group_by_columns: column names from the fact table to group the
results by theirs values
output: <key; value>: aggregated values

function Mapper::Run
input <IntermediateTable>
output -
    cFacts := HBase.connect(FactsTable)
    for each row in IntermediateTable do
        Map(row)
    done
end function

function Mapper::Map
input <Row_ID>
output <Group&Column, Value>
    if Row_ID is a range then
        first := Row_ID.asRange().getFirst()
        last := Row_ID.asRange().getLast() + 1
        row_values := cFacts.getRange(first, last)
    else
        row_values := cFacts.getRow(Row_ID)
    end if
    for each row in row_values do
        List group_by_values := createNewList()
        for each column in group_by_columns do
            group_by_values.append(row.getColValue(column))
        done
        for each column in selected_columns do
            EmitIntermediatePair(AsString(group_by_values)&column,
row.getColValue(column))
        done
```

```
    done
end function

function Combine
input <Group&Column key, List Values>
output <Group&Column, Value>
    for each value in values do
        sum := sum + value
    done
    EmitPair(<key, sum>)
end function

function Reduce
input <Group&Column key, List Values>
output <Group&Column, Value>
    for each value in values do
        sum := sum + value
    done
    EmitPair(<key, sum>)
end function
```

Explicació

Aquest MapReduce té per entrada la taula intermèdia. Aquesta taula conté tots els identificadors de totes les entrades de la taula de fets que s'han de buscar. També té per entrada el valor *selected_columns* que conté el conjunt de columnes de la taula de fets que es seleccionen en la consulta. L'altre valor d'entrada que té definit és *group_by_columns* que conté el nom de les columnes de la taula de fets que s'han de fer servir per l'agrupació dels valors.

En el pseudocodi veiem que hem implementat dues operacions de la classe Mapper: Run i Map. L'operació Run conté el codi on s'executa, dins d'un bucle, l'operació Map. L'operació Map és la funció que s'executa per cada entrada de la taula de les parelles intermèdies. Cadascuna d'aquestes entrades conté un identificador o bé un rang d'identificadors.

A l'operació Run establim la connexió amb la taula de fets que necessitem per a l'operació Map. Així només establim un cop la connexió enlloc de crear-la dins de l'operació Map cada vegada que volguéssim obtenir informació de la taula de fets. Dins de l'operació Run, un cop establerta la connexió, s'invoca l'operació Map per cadascuna de les files de la taula que ha de processar l'operació Map del node.

L'operació Map mira per cada valor que rep si és un identificador o bé un rang d'identificadors. Si és un identificador de fila, obté la tupla amb aquest identificador de la taula de fets (amb la connexió establerta a l'operació Run) i si és un rang, fa una petició de rang per obtenir

totes les tuples amb una sola petició. En la petició de rang, s'incrementa en un l'identificador de l'última fila que s'ha de buscar, perquè a les cerques per rang l'últim valor que indiques no està inclòs.

Per a cada tupla llegida, l'operació Map agafa el valor de les columnes que ens serviran per fer l'agrupació de valors amb l'ordre que s'han especificat i els concatena. A continuació, per cada columna que s'ha de seleccionar (*selected_columns*) emetem una parella intermèdia donant-li per clau els elements de l'agrupació concatenats amb el nom de la columna seleccionada i com a valor, el valor que té la tupla a la columna corresponent.

Per exemple, per a una determinada fila suposem que els valors de les columnes respecte les quals estem agrupant són a_1, a_2, ..., a_n, que les columnes que estem seleccionant són s_1, s_2, ..., s_m i que els valors per a cadascuna de les columnes seleccionades són v_1, v_2, ..., v_m on v_i correspon al valor de la columna seleccionada s_i per $\forall i, 1 \leq i \leq m$. Llavors, per cada fila de la taula de fets que s'obtingui s'emetran les següents parelles intermèdies:

$$<a_1\%a_2\%...\%a_n\%s_1;\ v_1>$$
$$<a_1\%a_2\%...\%a_n\%s_2;\ v_2>$$
$$...$$
$$<a_1\%a_2\%...\%a_n\%s_m;\ v_m>$$

A continuació s'executen les operacions Combine i Reduce. Aquestes dues operacions tenen el mateix pseudocodi. L'operació Combine rep les parelles emeses per l'operació Map agrupades per clau i l'operació Reduce les emeses per les operacions Combine. Així, aquestes operacions reben junts tots els valors de la variable seleccionada s_i que tenen els mateixos valors en les columnes per les quals fem l'agrupació. Així, l'únic que han de fer aquestes operacions és sumar els valors d'entrada i emetre una parella amb la mateixa clau que la que ha rebut i que el seu valor sigui la suma de tots els valors rebuts. D'aquesta manera, sent *valors* el conjunt de valors rebuts per l'operació Combine o Reduce, emetrem la següent clau:

$$<a_1\%a_2\%...\%a_n\%s_m;\quad \sum_{\forall i \in valors} v_{m_i}\ >$$

Així, per cada combinació de valors de les columnes d'agrupació tindrem els valors de les columnes seleccionades.

La sortida de l'operació Reduce es guardarà en una taula de l'HBase on cada fila contindrà els valors per a una de les agregacions i en diferents columnes hi haurà els valors seleccionats. D'aquesta manera la taula contindrà el resultat de la consulta.

Cal notar, però, que depenent de l'operació d'agregació que fem servir, l'operació Combine pot haver de ser diferent de l'operació Reduce. Per exemple, si considerem el cas en què l'operació d'agregació fos la mitjana, llavors a l'operació Combine s'haurien de sumar els valors d'entrada i en canvi a l'operació Reduce sumar els valors d'entrada i dividir el resultat pel nombre de tuples de cada agrupació.

7.2.2.2 Index Filtered Scan (IFS)

Aquest algorisme consisteix a obtenir les tuples que ens interessen de la taula de fets en un sol escaneig. L'escaneig, però, el farem aprofitant el coneixement que obtenim a través dels índexs per fer-lo més eficientment. Aquest algorisme rep per entrada l'arxiu intermedi el qual recorrerem per trobar els identificadors de les files de la taula de fets que hem de cercar. S'ha fet servir un arxiu intermedi enlloc d'una taula, perquè en aquest algorisme només fem una lectura seqüencial de les claus i s'espera que sigui més ràpida la lectura seqüencial d'un arxiu que d'una taula.

Dels identificadors llegits de l'arxiu intermedi n'extreu el valor mínim i màxim per tal de limitar-ne el recorregut sobre la taula de fets. A més a més, es guarda tots els identificador per tal de crear un Bitmap i filtrar les entrades que ens interessen dins de l'operació Run.

L'operació Map rebrà les tuples de la taula de fets que necessita. Per cada tupla rebuda s'agruparan els valors a seleccionar per resoldre la consulta segons el valor de les columnes per les quals estem fent l'agrupació.

Finalment, a les operacions Combine i Reduce es durà a terme l'agregació de valors per a cadascuna de les agrupacions resultants.

Aquest algorisme està pensat per aquelles seleccions en què les tuples que s'han de cercar siguin molt properes dins de la taula de fets. També, per aquelles consultes amb un factor de selecció elevat i que comprovar totes les clàusules fila per fila resulti més lent que fer els accessos a l'índex i crear el Bitmap.

A continuació presento el pseudocodi d'aquest algorisme:

Pseudocodi

```
MAPREDUCE2 – Index Filterd Scan
input: <key; value>: range of rows from the facts table containing all
the needed rows
    selected_columns: column names from the fact table to aggregate
    group_by_columns: column names from the fact table to group the
aggregations by theirs values
    bitmap_info: contains the list of all the entries to be selected
    facts_table_size: number of rows stored in the facts table
output: <key; value>: aggregated values


function Mapper::Run
input <FactsTable>
output -
    bitmap := createBitMap(facts_table_size)
    for each entry in bitmap_info do
        bitmap[entry] := 1
    done
    for each row in FactsTable do
        if bitmap[row.id] = 1 then
            Map(row)
        end if
    done
end function

function Mapper::Map
input <Row_ID, List Row_Values>
output <Group&Column key, Value>
    List group_by_values := createNewList()
    for each column in group_by_columns do
        group_by_values.append(row_values.getColValue(column))
    done
    for each column in selected_columns do
        EmitIntermediatePair(AsString(group_by_values)&column,
row_values.getColValue(column))
    done
end function

function Combine
input <Group&Column key, List Values>
output <Group&Column, Value>
    for each value in values do
        sum := sum + value
    done
    EmitPair(<key, sum>)
end function

function Reduce
input <Group&Column key, List Values>
output <Group&Column, Value>
```

```
        for each value in values do
            sum := sum + value
        done
        EmitPair(<key, sum>)
    end function
```

El codi de les operacions Combine i Reduce és el mateix que el de l'algorisme IRA.

Explicació

Aquest MapReduce té per entrada el rang mínim de tuples consecutives de la taula de fets que contenen tota la informació que necessitem. S'han definit com a valors d'entrada:

- *selected_columns*: conté el conjunt de columnes de la taula de fets que es seleccionen en la consulta.
- *group_by_columns*: conté el nom de les columnes de la taula de fets que s'han de fer servir per a l'agrupació dels valors.
- *bitmap_info*: conté el conjunt d'identificadors de les tuples que s'han d'agafar de la taula de fets.
- *facts_table_size*: conté el nombre d'entrades de la taula de fets.

En el pseudocodi veiem que implementem dues operacions de la classe Mapper: Run i Map. Dins de l'operació Run, creem el bitmap amb tantes posicions com entrades té la taula de fets. Cada posició del bitmap pot estar a 0 o a 1. Inicialment té tots els valors a 0.

A continuació, per cada identificador contingut a la llista *bitmap_info* marquem la seva posició corresponent del bitmap amb un 1. Un cop fet això, per cadascuna de les files rebudes de la taula de fets es mira si el seu identificador està marcat amb un 1 en el bitmap. Si és així invocarem l'operació Map per aquesta fila i sinó l'ignorarem. D'aquesta manera l'operació Map rep les entrades de la taula de fets que s'han de seleccionar i cap altra. Així, com a la implementació de l'IRA, per cada tupla agafa el valor de les columnes que ens serviran per fer l'agrupació de valors amb l'ordre que s'han especificat i els concatena. A continuació, per cada columna que s'ha indicat per seleccionar emetem una parella intermèdia donant-li per clau els elements de l'agrupació concatenats amb el nom de la columna seleccionada i com a valor el valor que té la tupla a la columna corresponent.

El comportament de les operacions Combine i Reduce són exactament les mateixes que en el cas anterior.

7.2.2.3 Filtered Source Scan (FSS)

L'algorisme d'escaneig de la taula consisteix a obtenir la informació que ens interessa per resoldre una consulta de la taula de fets fent-ne un escaneig, és a dir, llegint-la de principi a fi. En aquest cas, no farem servir els índexs. Primer de tot determinarem quina és la informació que ens interessa i, per tant, quines columnes s'han de llegir. S'ha de determinar, també, quines columnes necessitem per poder aplicar les condicions de la consulta. Finalment, crearem un filtre per fer un escaneig de la taula i obtenir només aquelles entrades que compleixin les clàusules de la consulta.

L'operació Map rebrà per entrada cadascuna de les files de la taula de fets que hagin passat els filtres i que, per tant, compleixen les clàusules de la consulta. Només llegirem les columnes que ens interessin per a resoldre la consulta i comprovar les clàusules. Obtindrem com a resultat del MapReduce un conjunt de tuples, que seran el resultat de la nostra consulta.

Aquest algorisme s'implementa amb un sol MapReduce el pseudocodi del qual és el següent:

Pseudocodi

```
MAPREDUCE – Filtered Source Scan
input: <key; value>: filtered facts table
    selected_columns: column names to aggregate from the fact table
    group_by_columns: column names from the fact table to group the
aggregations by theirs values
output: <key; value>: aggregated values

function Map
input <Row_ID, List Row_Values>
output < Group&Column key, Value>
    List group_by_values := createNewList()
    for each column in group_by_columns do
        group_by_values.append(row_values.getColValue(column))
    done
    for each column in selected_columns do
        EmitIntermediatePair(AsString(group_by_values)&column,
row_values.getColValue(column))
    done
end function

function Combine
```

```
input <Group&Column key, List Values>
output <Group&Column, Value>
    for each value in values do
        sum := sum + value
    done
    EmitPair(<key, sum>)
end function

function Reduce
input <Group&Column key, List Values>
output <Group&Column, Value>
    for each value in values do
        sum := sum + value
    done
    EmitPair(<key, sum>)
end function
```

Explicació

El codi del MapReduce, com podem veure és idèntic al del l'algorisme IFS. L'única diferència és que no cal crear el bitmap perquè només ens arriben les files que compleixen totes les clàusules. Per tant, tampoc cal reimplementar l'operació Run de la classe Map. A continuació explico el funcionament d'aquest MapReduce:

El MapReduce rep per entrada les files de la taula de fets que compleixen les clàusules de les consultes. Així, només ens cal fer l'agrupació dels valors d'agregació i sumar-los.

L'operació Map, s'encarrega de generar, a partir de cadascuna de les tuples obtingudes de la taula de fets, les parelles intermèdies que tindran per clau els elements de l'agrupació concatenats amb el nom de la columna seleccionada i com a valor, el valor que té la tupla a la columna corresponent.

D'aquesta manera les operacions Combine i Reduce (que tenen el mateix codi) rebran agrupats els valors per a cada columna seleccionada respecte els valors de les columnes per les quals s'agrupa. Així, l'únic que han de fer, és emetre una nova parella que tingui per clau, la clau rebuda i per valor, la suma dels valors que ha rebut.

7.3 Desnormalització

El procés de desnormalització d'una base de dades ha estat molt important per a poder preparar les proves de rendiment dels algorismes anteriors. Per a poder desnormalitzar el conjunt de proves del TPC-H, he fet servir el següent algorisme:

Començant per les taules més petites per tal de generar els resultats intermedis el més petits possibles, he fet la join entre dues taules A i B, on A té una clau forana a B. L'operació join entre les dues taules, resulta en una taula C que conté totes les tuples d'A i que ha substituït les columnes que contenen les claus foranes a B pels valors de la fila de B que es correspon.

Com que la llibreria d'HBase només permet una taula com a entrada del MapReduce, el que he fet, ha estat posar en una gran taula totes les taules del TPC-H. Cada taula està en una família de columnes diferents. Així, podem tenir dues taules diferents com a entrada del MapReduce.

L'operació de join entre dues taules l'he implementat amb un MapReduce. El seu pseudocodi és el següent:

Pseudocodi

```
MAPREDUCE – join
input: <key; value>: tables to join
    inner_table_name: Name of the table pointed by a foreign key
    outer_table_name: Name of the table with the foreign key
    outer_table_primary_key: Name of the primary key columns from the
outer table of the join
    join_attributes: Name of the columns to join
output: <key; value>: join of the tables

function Map
input <row>
output <join_attributes_key, column_name&column_value[]>
    key := createNewList()
    for each column in join_attributes do
        key.append(row.getValue(column))
    done
    value := createNewList()
    for each column in row do
        if column.getName() not in join_attributes then
            if row.getTableName() = inner_table_name then
                value.append(inner_table_name + column.getName())
            else
                value.append(column.getName())
            end if
            value.append(column.getValue())
        end if
    done
    EmitIntermediatePair(AsString(key), AsString(value))
end function

function Reduce
```

```
input <join_attributes_key, Values>
output <Row_ID, values>
    for each value in values do
        if value.isFromTheInnerTable() then
            inner_value := value
            break
        end if
    done
    for each value in values do
        if not value.isFromTheInnerTable() then
            key := createNewList()
            for each column in outer_table_primary_key do
                key.append(values.getValue(column))
            done
            EmitPair(<key, value&inner_value>)
        end if
    done
end function
```

Explicació

Per simplicitat del pseudocodi anterior, s'ha suposat que la/les columna/es de clau forana de la taula exterior tenen exactament el mateix nom que les de la clau primària que fan referència de la taula interior.

L'operació Map rebrà per entrada les tuples de les taules de les quals en volem fer la join. Per cada una de les tuples que rebi emetrà una parella intermèdia que tindrà per clau el valor dels atributs concatenats que s'han d'igualar per a la join i com a valor de la parella, el valor de totes les columnes de la fila que no estiguin incloses dins de les columnes de la join.

L'operació Reduce rebrà les parelles intermèdies agrupades per clau. Rebrà l'identificador de la clau primària (de la taula interior) i per valor, el conjunt de tuples de la taula externa que la seva clau forana apunta en aquesta clau primària. Enmig d'aquestes tuples hi ha la tupla amb aquesta clau primària de la taula interna. D'aquesta manera, el primer que ha de fer és trobar la tupla de la taula interna que hi ha dins la llista de valors. Un cop fet això, per cadascuna de les tuples de la taula externa se n'ha de trobar la clau que ha d'identificar la nova fila (la clau de la taula exterior) que tindrà per columnes el conjunt de valors de la tupla de la taula externa i de la taula interna.

La sortida de l'operació Reduce es desarà en una família de columnes nova per tal de poder ser l'entrada d'una altra join.

Exemple

Vegem ara un exemple per il·lustrar el funcionament del MapReduce que duu a terme la join entre dues taules diferents. Suposem les següents taules:

Països

ID	Nom	Regió	Capital
A	Espanya	1	Madrid
B	França	1	París
C	Xina	2	Taipei

Regions

Regió	RNom
1	Europa
2	Àsia

Veiem que la taula Països conté 3 països diferents amb les seves respectives capitals i una clau forana a cadascuna de les seves regions. Llavors, l'operació Map rebrà per entrada les parelles clau/valor següents, que es corresponen al contingut de les dues taules[6]:

<A; ID:A,Nom:Espanya,Regió:1,Capital:Madrid>

<B; ID:B,Nom:França,Regió:1,Capital:París>

<C; ID:C,Nom:Xina,Regió:2,Capital:Taipei>

<1; Regió:1,RNom:Europa>

<2; Regió:2,RNom:Àsia>

L'operació Map, un cop hagi processat la informació d'entrada, haurà generat les següents parelles intermèdies:

<1; ID:A,Nom:Espanya,Regió:1,Capital:Madrid>

<1; ID:B,Nom:França,Regió:1,Capital:París>

<2; ID:C,Nom:Xina,Regió:2,Capital:Taipei>

<1; Regió:1,RNom:Europa>

<2; Regió:2,RNom:Àsia>

Que queden agrupades de la forma següent:

<1; ID:A,Nom:Espanya,Regió:1,Capital:Madrid>
<1; ID:B,Nom:França,Regió:1,Capital:París
<1; Regió:1,RNom:Europa>

<2; ID:C,Nom:Xina,Regió:2,Capital:Taipei
<2; Regió:2,RNom:Àsia>

6 El punt i coma separa la clau de la parella del seu valor. Els diferents valors de cada parella estan reparats per una coma.

Finalment l'operació Reduce generarà les següents tuples per a la taula resultant:

<A; ID:A,Nom:Espanya,Regió:1,Capital:Madrid,RNom:Europa>
<B; ID:B,Nom:França,Regió:1,Capital:París,RNom:Europa>
<C; ID:C,Nom:Xina,Regió:2,Capital:Taipei,RNom:Àsia>

7.3.1 Primera versió

En aquest apartat explicaré la primera versió que vaig fer del desnormalitzador i que va resultar molt lenta i tenir certs problemes. El principal problema que tenia era que saturava la xarxa amb massa peticions, el servidor no era prou ràpid i acabava caient la connexió.

Aquest algorisme de desnormalització fa bàsicament el mateix que l'anterior, l'únic que està pensat perquè les taules per desnormalitzar estiguin en taules diferents de l'HBase. L'entrada de l'operació Map era la taula que contenia les claus foranes. Llavors, a l'operació Map s'emetien les tuples de la taula tal com les havíem rebut canviant, però, la clau, que ara passava a ser el valor dels atributs respecte els quals es feia la join. D'aquesta manera, des del Reduce es podia obtenir la tupla referenciada de la taula interior i fer la join d'aquesta amb les tuples rebudes.

A continuació mostro el seu pseudocodi i tot seguit una petita explicació:

7.3.1.1 Pseudocodi

```
MAPREDUCE – join 1st version
input: <key; value>: outer table
    inner_table_name: Name of the table pointed by the foreign key.
    outer_table_name: Name of the table with the foreign key.
    outer_table_primary_key: Name of the primary key columns from the
outer table of the join
    join_attributes: Name of the columns to join
output: <key; value>: join of the tables

function Map
input <row_key, row>
output <join_attributes_key, row>
    key := createNewList()
    for each column in join_attributes do
        key.append(row.getValue(column))
    done
    EmitIntermediatePair(AsString(key), row)
end function
```

```
function Reduce
input <key, rows>
output <Row_ID, values>
    cInner := HBase.connect(inner_table_name)
    inner_row := cInner.getRow(Row_ID)
    for each row in rows do
        key := createNewList()
        for each column in  outer_table_primary_key do
            key.append(values.getValue(column))
        done
        EmitPair(<key, row.getValues()&inner_row.getValues()>)
    done
end function
```

Explicació

Com ja he comentat, l'operació Map rep per entrada la taula que conté les claus foranes (taula exterior). L'operació Map agafa els valors de les columnes respecte les quals s'ha de fer join i emet una parella intermèdia que té per clau la concatenació de valors de la join i per valor el contingut de la pròpia fila. D'aquesta manera, l'operació Reduce rebrà agrupades totes les files les quals se'ls ha de fer la join amb la mateixa fila de la taula interior.

El principal problema d'aquesta implementació, és que com podem veure, la connexió s'establia a dins de l'operació Reduce. És a dir, que per cada entrada de la taula interior s'establia una connexió per obtenir-la. Això suposava un cost molt elevat. Més endavant, després d'implementar la versió que he explicat en l'apartat anterior, vaig descobrir que es podien establir connexions dins de l'operació Run que tant hi és per a la classe Mapper com la classe Reducer. Així, s'hagués pogut aplicar una solució semblant a la dels algorismes IRA i IFS. Tot i així, com que acabes agafant totes les entrades de la taula interior pot no sortir gaire a compte fer un accés aleatori (a l'apartat de proves es comprova que fer accessos aleatoris per obtenir una taula complerta és l'opció menys eficient).

L'operació Reduce, per a cadascuna de les agrupacions que rebi, es connectarà a la taula referenciada (taula interior) i n'obtindrà la fila amb la qual s'ha de fer la join de les files rebudes. Per a cada tupla rebuda de l'operació Map se n'ha de crear la seva clau. Aquesta serà la mateixa clau que tenia la taula exterior. Finalment, emetre'm aquesta clau amb el conjunt de valors de les dues files (una fila de la taula interior i l'altra de l'exterior).

8 Planificació i estudi econòmic

Durant el mes de desembre del 2010 el director del projecte em va proposar fer aquest projecte final de carrera. Per poder entendre de què tractava, vaig començar a mirar-me en el mateix mes de desembre per a què servien el MapReduce i l'HBase. A principis del mes de gener el director em va explicar exactament en què consistia el projecte, quins algorismes s'havien d'implementar i que la data límit per a presentar els resultats en el congrés del DaWaK 2011 (*International Conference on Data Warehousing and Knowledge Discovery 2011*) era el dia 29 de març. En aquest moment ja vaig poder fer una planificació inicial de totes les etapes per a poder fer el projecte.

Un cop acabat el projecte, he refet la planificació inicial del projecte amb els canvis que han patit les diferents etapes, aquesta és la planificació realitzada. Hi ha certes diferències entre la planificació inicial i la planificació realitzada, però s'ha de tenir en compte que hi ha hagut diferents imprevists. Per començar, la data límit del congrés es va allargar fins al 12 d'abril, cosa que va permetre que el desenvolupament del projecte i la durada de les proves s'allarguessin. Aquest allargament del termini va ser molt convenient ja que havien apareguts certs imprevists al llarg del desenvolupament (canvis en els algorismes, actualització de la API,...) i en la configuració del clúster que havien fet enrederir-me en la planificació.

No obstant, cal dir que la planificació inicial ha estat força acurada i que no s'ha vist gaire modificada. Les principals diferències han estat un allargament del temps de desenvolupament, configuració i proves.

Ara faré la descripció de cadascuna de les etapes de què ha constat el projecte i explicaré què ha causat les diferències entre la planificació inicial i la realitzada. Després de les descripcions, incloc els diagrames de Gantt de la planificació inicial i el de la final i finalment el pressupost del projecte.

8.1 Descripció de les etapes

En aquest apartat explicaré en què ha consistit cadascuna de les etapes del projecte:

8.1.1 Estudiant el problema

De cara a preparar el projecte, durant el desembre del 2010 vaig estar mirant en què consistien el MapReduce i l'HBase i, també, una mica com es podien fer servir en el projecte.

8.1.2 Aprenentatge

Aquesta etapa ha consistit a aprendre què són i com funcionen les diferents tecnologies que he fet servir al llarg del projecte: MapReduce, HBase, HDFS i ZooKeeper. Primer de tot, vaig consultar les pàgines web oficials [6][7][8] per comprendre per a què servia cadascuna d'elles i quins valors podien aportar al meu projecte.

També em vaig llegir els papers que va publicar Google sobre MapReduce [2] i BigTable [1], que va inspirar la creació d'HBase, i també un paper [3] sobre el rendiment del MapReduce i quins factors es podien mirar de treballar per tal de fer el projecte més eficient.

A més a més, de forma paral·lela a la lectura d'aquests papers, vaig estar estudiant l'arquitectura dels sistemes a utilitzar per tal de saber què oferien i com s'havien de fer servir en el projecte per a poder fer una bona implementació. Per aquest propòsit m'han estat molt útils les pàgines *wiki* mantingudes pels propis desenvolupadors dels projectes.

Aquesta etapa del projecte no ha patit gaires canvis de la planificació inicial a la final, es pot veure que em vaig avançar una mica en la lectura de la documentació per tal de disposar de més temps per poder estudiar pels exàmens finals del mes de Gener.

8.1.3 Configuració del sistema

La fase de configuració del sistema està dividida en dues parts: la configuració de l'entorn de desenvolupament i la configuració del clúster per fer proves:

8.1.3.1 Configuració de l'entorn de desenvolupament

La instal·lació de l'entorn de desenvolupament la vaig fer en el meu ordinador portàtil. Vaig fer la instal·lació en Linux sense gaires entrebancs. Gairebé va ser descomprimir, configurar i executar. L'únic problema que vaig trobar, va ser que cada cop que reiniciava l'ordinador se'm perdien les dades de l'HDFS i em donava un error de connexió. Vaig estar investigant una mica i vaig descobrir que per defecte escriu la informació al directori temporal

(*/tmp/*) que s'esborra cada cop que es reinicia l'ordinador. Vaig buscar la forma de configurar-lo correctament i ja el vaig tenir plenament funcional.

8.1.3.2 Configuració del clúster per fer proves

Vaig fer una instal·lació distribuïda del sistema a casa. Vaig fer la configuració del clúster en el meu portàtil i a l'ordinador de sobre taula de casa meva. El primer pas va ser fer la instal·lació de l'HDFS de forma distribuïda. En aquesta instal·lació ja vaig començar a trobar problemes. Passava que en cap lloc es deia que la ruta absoluta als arxius de Hadoop havia de ser la mateixa en els diferents ordinadors i jo l'havia fet en rutes diferents. Vaig estar-hi donant voltes fins que vaig trobar un arxiu de log que donava un error i una mica d'explicació i ho vaig poder corregir.

Un cop vaig tenir acabada la instal·lació, vaig poder comprovar que l'HDFS funcionava correctament perquè la capacitat de l'HDFS era la suma de la mida dels discs durs dels dos ordinadors. La configuració inicial que vaig fer pel MapReduce va ser la mateixa que feia servir en local però hi vaig haver d'anar fent modificacions per solucionar els errors que m'anaven apareguen. Finalment vaig arribar a la configuració que explico a l'apartat 18.1.1 (Annex II).

Amb la instal·lació de Hadoop feta, vaig procedir a instal·lar l'HBase com a servidor en un sol node però que fes servir l'HDFS distribuït. Aquesta instal·lació va resultar més complicada del que semblava en un bon principi; en teoria havia de ser com la instal·lació en local però van aparèixer problemes que van alentir molt el procés.

Comparant els diagrames de Gantt es pot veure que l'etapa de configuració va durar més del previst degut a la quantitat de problemes que he trobat configurant el clúster. A continuació explico alguns altres problemes que m'he trobat i que han fet allargar aquesta etapa:

- Per configurar l'HBase en el clúster vaig trobar el problema de configuració següent: A l'hora d'especificar la direcció de l'HDFS s'havia de posar 'portatil2.local'[7] enlloc de 'portatil2'. Aquesta característica no l'havia trobat comentada, totes les guies indicaven que s'havia de posar la direcció IP o el nom de xarxa i no em funcionava. Vaig resoldre

7 El nom de xarxa del meu portàtil és Portàtil2.

aquest problema gràcies a què vaig trobar en un log un missatge d'error que explicava que hi havia aquest problema i com s'havia de resoldre.

- Vaig tenir un problema amb la configuració de les direccions IP dels servidors. En els arxius de configuració d'HBase posava la direcció IP del màster i dels esclaus. Però, quan intentava encendre el clúster, l'esclau enlloc de connectar-se al màster s'intentava connectar a ell mateix.

  Em vaig passar més d'una setmana fent proves i buscant per Internet manuals, explicacions i fòrums que parlessin d'aquest tema, però no vaig trobar res. Fins que un dia se'm va acudir fer escanejos de ports i vaig trobar que segons quina IP provava, si la de xarxa, *localhost* o *loopback* em deixava connectar o no al port del servidor.

  El problema estava en el fitxer de configuració de Linux per a la traducció de noms de xarxa a adreces IP (l'arxiu /etc/hosts). HBase obtenia la direcció IP configurada i la convertia en el nom de xarxa; com que la resolució inversa es feia des del servidor, la direcció IP del servidor la resolia com a *localhost*. A continuació, enviava el nom de xarxa que havia resolt al esclau com la direcció a la que s'havia de connectar per trobar el servidor però el que en realitat estava enviant era *localhost*. D'aquesta manera, cada vegada que l'esclau s'intentava connectar al servidor, es connectava a ell mateix perquè cada ordinador es considera a ell mateix com a *localhost*.

  La solució va ser modificar l'arxiu /etc/hosts de tal forma que primer resolgués la direcció IP amb el nom de l'ordinador i no amb localhost, si originalment l'arxiu era:

  ```
  192.168.2.3     localhost
  192.168.2.3     portatil2
  ```

  el vaig haver de canviar a:

  ```
  192.168.2.3     portatil2
  192.168.2.3     localhost
  ```

  D'aquesta manera, es va resoldre completament el problema.

8.1.4 Implementació

Amb els coneixements bàsics adquirits i l'entorn de desenvolupament configurat, ja podia començar el desenvolupament del projecte. Com ja he explicat, la metodologia que he fet servir és Agile. Aquesta metodologia ha permès un desenvolupament ràpid i eficaç del projecte.

Vaig començar la implementació del projecte a principis de gener del 2011 i amb unes tres setmanes ja tenia el nucli del projecte programat i funcionant. Vaig anar fent les proves en cada iteració per assegurar-me que, a mesura que anava programant, les operacions implementades fins al moment funcionessin correctament. Com veurem a l'apartat 8.1.6, en aquest moment ja vaig poder començar les primeres proves.

Un factor que ha contribuït en gran mesura al ràpid desenvolupament d'aquest projecte ha estat la comunicació fluida que he tingut amb els directors del projecte quan apareixien dubtes o imprevists.

En aquesta etapa he respectat la planificació inicial pel que fa el nombre d'hores requerides. Tal com es pot apreciar, però, he dut a terme la implementació del projecte en un període més dilatat de temps. Tot i així, es pot considerar que s'ha respectat la estimació inicial.

8.1.5 Jocs de proves

Aquesta fase ha consistit en l'elaboració dels jocs de proves. Com ja he explicat, al llarg del desenvolupament del projecte he anat fent diferents proves, algunes per comprovar que els algorismes que funcionessin correctament i d'altres per avaluar-ne el rendiment. Les proves dels jocs de proves, en canvi, són les que he preparat per simular situacions realistes per a poder fer les proves de rendiment. Per preparar aquestes proves ho he fet en diferents parts:

8.1.5.1 Modificacions del TPC-H

Aquesta part del projecte ha consistit a adaptar el generador de jocs de proves del TPC-H. El codi d'aquest generador és obert i se'l pot descarregar qualsevol, l'únic és que no hi ha o no he sabut trobar cap tipus de documentació que expliqui la implementació que n'han fet. Sí que m'he estudiat, però, el manual de referència del TPC-H[5] per tal de comprendre millor la seva utilitat i el seu funcionament general.

També m'he hagut d'estudiar el funcionament intern del programa (23 arxius de codi C entre arxius font i capçaleres) per tal de poder fer-hi les modificacions necessàries. Bé, en primer moment volíem modificar-lo de tal manera que fos el propi programa que ens donés tota la informació desnormalitzada, però vaig adonar-me que no era possible (veure l'apartat 11.2). En aquest moment es va decidir fer la implementació del desnormalitzador que explico a l'apartat següent:

8.1.5.2 Desnormalitzador

Com he explicat a l'apartat anterior, la creació d'aquest desnormalitzador no estava a la previsió inicial perquè no ens podíem imaginar que resultés tant complexa la desnormalització de les taules. La implementació d'aquest programa ha canviat unes quantes vegades degut a problemes de rendiment (per més detalls veure el capítol d'Implementació).

Aquesta etapa s'ha realitzat amb temps que inicialment havia destinat a la tasca de l'apartat anterior. Aquest canvi és justificable, perquè en aquesta etapa he fet una funcionalitat que s'havia d'incloure en l'apartat anterior.

8.1.5.3 Traducció de les consultes

Aquest etapa consistia, inicialment, en prendre totes les consultes del test del TPC-H (22 consultes) i traduir-les, en la mesura possible, per tal que es poguessin executar amb aquest projecte. Vaig començar per les més senzilles i que es podien fer més o menys semblants a les originals. Però la majoria eren força complexes i tenien comparacions entre els valors de les columnes, cosa que el meu projecte no permet fer. Per exemple hi ha consultes en què es busquen les entregues que s'han entregat més tard de la data prevista. L'objectiu d'aquesta consulta és trobar una forma eficient de fer aquesta comparació i trobar els resultats que la satisfan, però el sistema desenvolupat en aquest projecte no està pensat per resoldre aquest tipus de qüestions. Per aquest motiu, es va decidir fer les consultes seguint altres criteris enlloc de la traducció literal.

Així, aquesta etapa realment ha consistit en fer un estudi de les característiques de les consultes del TPC-H i crear un nou conjunt de consultes. Aquestes noves consultes s'han intentat fer que s'adaptessin al patró detectat en l'estudi de les consultes originals. Explico aquesta tasca amb més detall en el capítol 12.

Tot i el canvi de la tasca a realitzar en aquesta etapa, m'he pogut ajustar a la planificació inicial.

8.1.6 Primeres proves

Aquesta etapa del projecte, va consistir a fer les proves de funcionalitats i del rendiment del que tenia implementat del projecte fins al moment. Explico algunes de les proves que vaig fer amb més detall en l'apartat 13.1. Aquestes tenien per objectiu comprovar que el projecte funcionava correctament i que els temps eren els esperats.

Les proves em van permetre detectar amb anterioritat problemes de rendiment com per exemple un cost exponencial a l'hora de crear els índexs, veure l'apartat 11.1.1.

Aquesta etapa no ha patit modificacions respecte la planificació que vaig fer inicialment del projecte tal i com podem comprovar en els diagrames que hi ha més endavant en aquest del capítol.

8.1.7 Test dels algorismes

Aquesta fase ha consistit a fer les proves de funcionament dels diferents algorismes del projecte per tal de poder garantir la qualitat de les mesures de les proves de rendiment de l'etapa que ve a continuació. Les proves que he realitzat les explico en el capítol de proves.

Com es pot veure en les planificacions que hi ha més endavant aquesta etapa no s'ha vist alterada i s'ha pogut realitzar tal com estava prevista.

8.1.8 Proves de rendiment

Aquesta fase del projecte ha consistit a mesurar l'espai que necessita i el temps que tarda el projecte a executar les diferents consultes dels jocs de proves. Les proves s'han dut a terme amb diferents volums de dades i configuracions.

A més de l'execució de les consultes sobre la base de dades, vaig fer uns quants *scripts* per a automatitzar l'execució dels tests i poder-los fer més ràpidament.

Més endavant podem veure que la desviació que hi ha hagut respecte la planificació inicial és baixa. La durada de les proves s'ha allargat una mica degut als problemes de

configuració del clúster que em vaig trobar. Tot i així, com que és va allargar el termini d'entrega del paper, es van poder fer totes les proves.

8.1.9 Millores de rendiment

Aquesta etapa ha consistit a fer millores de diferents tipus, en la configuració o en la implementació dels programes, per tal d'aconseguir una execució més ràpida.

Mirant a la planificació inicial podem veure que aquestes millores comencen abans que les proves de rendiment. Ho vaig planificar d'aquesta manera perquè vaig intuir que, molt possiblement, abans de començar amb les proves rendiment, ja hauria trobat problemes de rendiment que s'haurien d'arreglar. En el capítol d'implementació comento els problemes més importants que m'he trobat i les solucions que he aplicat.

Podem veure que per aquesta fase la planificació inicial i la realitzada són iguals i que m'he cenyit a les previsions.

8.1.10 Memòria del projecte

Aquesta fase del projecte ha consistit a escriure aquesta memòria.

Podem veure a les planificacions que ja des de bon principi vaig començar a fer la memòria. Aquesta, inicialment eren notes o apunts que prenia i que més endavant he convertit en un redactat.

En la planificació final, també hi podem apreciar que el treball en la memòria no ha estat tant constant com havia planificat inicialment tot i que les diferències que hi ha hagut no han estat gaire grans.

8.1.11 Presentació del projecte

Aquesta última fase ha consistit en la preparació de la presentació i defensa d'aquest projecte enfront del tribunal.

8.2 Planificació inicial

Aquest és el diagrama de Gantt de la planificació inicial:

Activitat	Hores	13-19 des	20-26 des	27-2 gen	3-9 gen	10-16 gen	17-23 gen	24-30 gen	31-6 feb	7-13 feb	14-20 feb	21-27 feb	28-6 mar	7-13 mar	14-20 mar	21-27 mar	28-3 abr	4-10 abr	11-17 abr	18-24 abr	25-1 mai	2-8 mai	9-15 mai	16-22 mai	23-29 mai	30-5 jun	6-12 jun	13-19 jun	20-26 jun
1. Aprenentatge	110	10	20	20	20	20	20																						
2. Configuració del sistema	60		10	10					10	10	10	10																	
3. Implementació	100					20	20		20	20	20																		
4.1. Modificar TPC-H	45								10	15	10	10																	
4.3. Traducció de consultes	25									5	10	10																	
5. Primeres proves	30					10	5		5	5	5																		
6. Test dels algorismes	60												15	20	25														
7. Proves de rendiment	140															20	20	20	20	30	30								
8. Millores de rendiment	55												10	15	10	15	5												
9. Memòria del projecte	225						5		10			10	5	5	5	5	5	5	10	5	5	20	30	30	30	40			
10. Presentació del projecte	40																									15	15	10
Total d'hores:	890																												

DiagramaGantt 1: Planificació inicial del projecte

8.3 Planificació realitzada

A continuació mostro el diagrama de Gantt de la planificació realitzada:

Activitat	Hores	13-19 des	20-26 des	27-2 gen	3-9 gen	10-16 gen	17-23 gen	24-30 gen	31-6 feb	7-13 feb	14-20 feb	21-27 feb	28-6 mar	7-13 mar	14-20 mar	21-27 mar	28-3 abr	4-10 abr	11-17 abr	18-24 abr	25-1 mai	2-8 mai	9-15 mai	16-22 mai	23-29 mai	30-5 jun	6-12 jun	13-19 jun	20-26 jun
1. Aprenentatge	90	5	20	30	25		10																						
2. Configuració del sistema	85		10	5					10	15	10	10		5			10	5	5										
3. Implementació	105					20	20		10	10	10	10	10	10	5														
4.1. Modificar TPC-H	23								10	8		5																	
4.2. Desnormalitzador	40									20	10	5	5																
4.3. Traducció de consultes	20						5				5		10																
5. Primeres proves	30					10	5		5	5	5																		
6. Test dels algorismes	60												15	20	25														
7. Proves de rendiment	110															20	20	20	20	30									
8. Millores de rendiment	60												5	5	10	15	10	10	5										
9. Memòria del projecte	275						5		10		5	10	5	5	5	5	5	5	10	20	25	30	30	30	30	40			
10. Presentació del projecte	40																										15	15	10
Total d'hores:	938																												

DiagramaGantt 2: Planificació real del projecte

8.4 Pressupost del projecte

Aquest és el pressupost de la planificació inicial del projecte. Per a cada etapa del projecte he indicat els recursos que he necessitat. Per a fer-lo he comptat un sou de 50€/hora per a les hores de treball normals i un sou de 25€/hora per les hores d'aprenentatge. La documentació i la memòria les comptabilitzo també a 25€/hora perquè part de la memòria no es faria per un projecte real, per exemple: la introducció, conclusions, línies de futur... Aquestes tarifes inclouen els impostos i les taxes pertinents (IRPF, seguretat social,...). El pressupost és el següent:

Recurs	Unitats	Preu unitat(€)	Informació	Total (€)	Observacions
Estudi del problema					
Informàtic	110 h.	25,00 €	Hores de treball	2.750,00 €	Com que són hores d'aprenentatge les compto a meitat de preu
Configuració del sistema					
Informàtic	60 h.	50,00 €	Hores de treball	3.000,00 €	
Implementació					
Informàtic	100 h.	50,00 €	Hores de treball	5.000,00 €	
Jocs de proves					
Informàtic	70 h.	50,00 €	Hores de treball	3.500,00 €	
Primeres proves					
Informàtic	30 h.	50,00 €	Hores de treball	1.500,00 €	
Test dels algorismes					
Informàtic	60 h.	50,00 €	Hores de treball	3.000,00 €	
Proves de rendiment					
Informàtic	140 h.	50,00 €	Hores de treball	7.000,00 €	
Clúster	1	0,00 €	Lloguer del clúster	0,00 €	Com que el clúster l'he fet a casa no ha tingut cost addicional.
Millores de rendiment					
Informàtic	55 h.	50,00 €	Hores de treball	2.750,00 €	
Documentació					
Memòria i documentació	225 h.	25,00 €	Hores de treball	5.625,00 €	Com que són hores de documentació les comptabilitzo a 25€
Presentació	40 h.	25,00 €	Hores de treball	1.000,00 €	Com que són hores de documentació les comptabilitzo a 25€
Subtotal					
				35.125,00 €	
Contingència					
	10.260,00 €	10,00%		3.512,50 €	S'estima en un 10% ja que el pressupost és força detallat però té una probabilitat molt alta d'imprevists.
Total				38.637,50 €	

Aquest és el pressupost de la planificació inicial i té un valor de 38,637,50€. Com podrem comprovar en el següent apartat aquest valor és lleugerament superior al cost real del projecte. D'aquesta manera podem concloure que la contingència ha estat suficient.

Els pressuposts realitzats s'han fet pensant en què faria les proves a casa. De fet, les proves fetes a casa han estat suficients per acomplir l'objectiu del projecte i poder obtenir els resultats dels algorismes. No obstant això, si s'haguessin volgut trobar uns resultats més realistes, s'hauria pogut llogar algun clúster per tal de dur-hi a terme les proves i s'hauria encarit encara més el projecte.

8.5 Cost de la planificació realitzada

En aquest apartat presento el cost que ha tingut en realitat aquest projecte. L'he calculat seguint les tarifes definides a l'apartat anterior. Cal tenir en compte, que en aquest cas, no cal definir marc de contingència ja que les dades definitives. El càlcul del cost és el següent:

Recurs	Unitats	Preu unitat(€)	Informació	Total (€)	Observacions
Estudi del problema					
Informàtic	90 h.	25,00 €	Hores de treball	2.250,00 €	Com que són hores d'aprenentatge les compto a meitat de preu
Configuració del sistema					
Informàtic	85 h.	50,00 €	Hores de treball	4.250,00 €	
Implementació					
Informàtic	105 h.	50,00 €	Hores de treball	5.250,00 €	
Jocs de proves					
Informàtic	83 h.	50,00 €	Hores de treball	4.150,00 €	
Primeres proves					
Informàtic	30 h.	50,00 €	Hores de treball	1.500,00 €	
Test dels algorismes					
Informàtic	60 h.	50,00 €	Hores de treball	3.000,00 €	
Proves de rendiment					
Informàtic	110 h.	50,00 €	Hores de treball	5.500,00 €	
Clúster	1	0,00 €	Lloguer del clúster	0,00 €	Com que el clúster l'he fet a casa no ha tingut cost addicional.
Millores de rendiment					
Informàtic	60 h.	50,00 €	Hores de treball	3.000,00 €	
Documentació					
Memòria i documentació	275 h.	25,00 €	Hores de treball	6.875,00 €	Com que són hores de documentació les comptabilitzo a 25€
Presentació	40 h.	25,00 €	Hores de treball	1.000,00 €	Com que són hores de documentació les comptabilitzo a 25€
Total				36.775,00 €	

El cost de la planificació realitzada és de: 36.775,00€. Podem comprovar doncs el seu cost surt inferior al preu estimat en el pressupost realitzat inicialment i que la desviació que hi ha és de només 1.862,50€.

9 Especificació

En aquest capítol recullo els diferents elements del model d'anàlisi d'aquest projecte. Aquests elements inclouen el model de casos d'ús, el model conceptual i el model de navegabilitat:

9.1 Casos d'ús

Un usuari del meu projecte pot realitzar tres tipus d'interaccions bàsiques amb el sistema: la creació de dimensions, la realització de consultes i la configuració de l'execució.

Aquest sistema només té un tipus d'usuari, que anomenaré *usuari* que pot realitzar els següents casos d'ús:

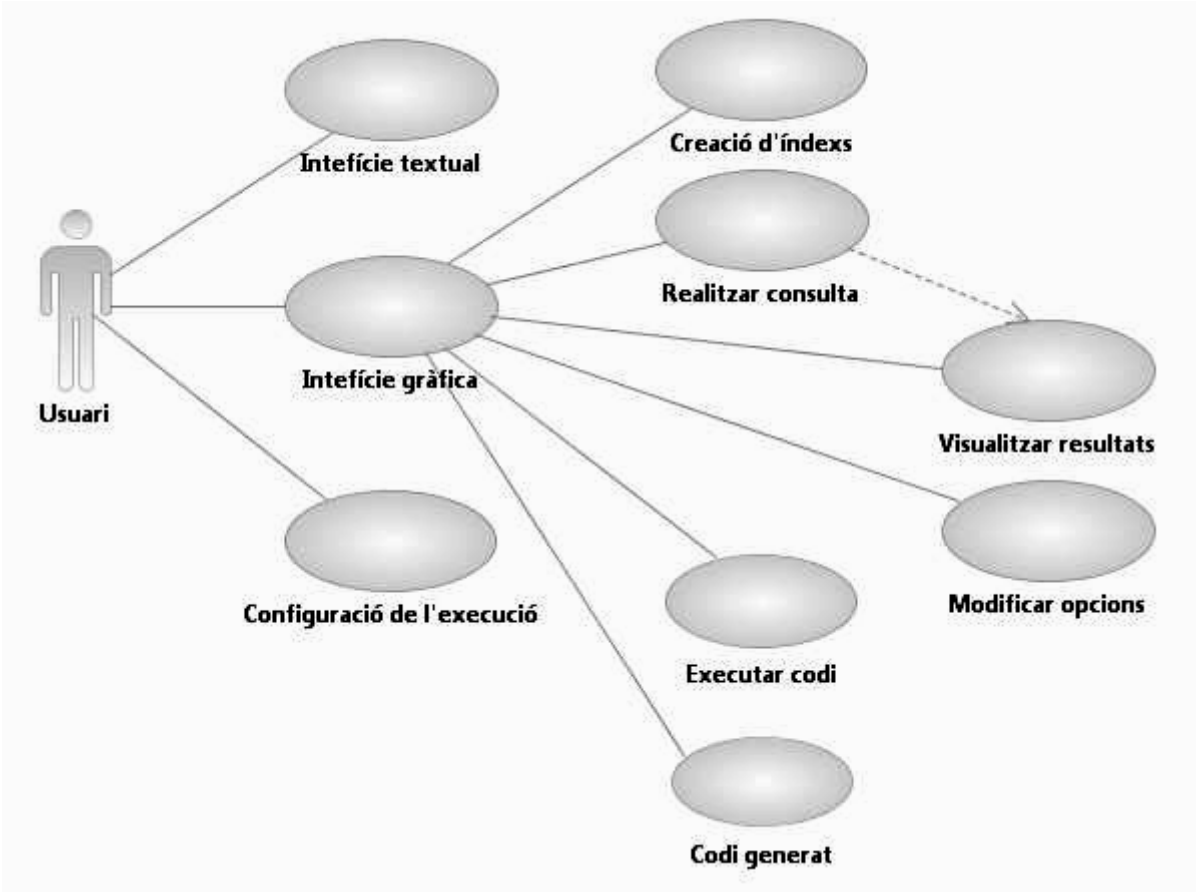


Figura 6: Diagrama de casos d'ús

Aquests són els diferents casos d'ús que un usuari pot executar en aquest projecte. Els explico a continuació:

9.1.1 Configuració de l'execució

Escenari principal d'èxit

1. L'usuari inicia el programa amb un conjunt de paràmetres de configuració.
2. El sistema reconeix la informació, configura l'execució i es procedeix a l'execució del programa.

Escenari alternatiu

1. L'usuari inicia el programa amb un conjunt de paràmetres de configuració però hi ha un paràmetre que no existeix o falta informació per algun dels paràmetres.
2. El sistema informa de l'error a l'usuari i s'acaba l'execució.

9.1.2 Interfície textual

Escenari principal d'èxit

1. L'usuari invoca el programa posant-li per primer paràmetre *shell* per tal d'executar-lo en mode text.
2. L'usuari introdueix la comanda que vol executar i prem Enter per indicar que ja ha escrit tota la comanda.
    - Si és una comanda de creació d'índexs:
        - El sistema crea l'índex indicat utilitzant l'algorisme de creació d'índex d'un sol nivell (apartat 7.1.1) o bé de multinivell (apartat 7.1.2) segons la configuració utilitzada.
        - El sistema mostra el temps emprat per crear l'índex.
        - Es torna al pas 2.
    - Si és una comanda de consulta:
        - El sistema realitza la consulta executant l'algorisme de consulta de cub de dades (apartat 7.2) que es correspon amb la configuració utilitzada.
        - El sistema mostra el temps emprat per realitzar la consulta.

- Es torna al pas 2.

- Si és la comanda per sortir:
  - S'acaba l'execució del programa.

Escenari alternatiu 1

A partir del pas 2 de l'escenari principal d'èxit:

1. Si la comanda donada no existeix:
   - El sistema indica que la comanda no existeix.
   - Es torna al pas 2 de l'escenari principal d'èxit.

Escenari alternatiu 2

A partir del pas 2 de l'escenari principal d'èxit:

1. Si la comanda donada té un format incorrecte:
   - El sistema indica que la comanda té un format incorrecte.
   - Es torna al pas 2 de l'escenari principal d'èxit.

9.1.3 Interfície gràfica

Escenari principal d'èxit

1. L'usuari ha executat el programa normalment, sense que el primer paràmetre sigui *shell*.
2. El sistema mostra la interfície gràfica del programa.

9.1.4 Creació d'índexs

Escenari principal d'èxit

1. L'usuari indica que vol crear un nou índex fent clic a la pestanya del gestor de dimensions.
2. El sistema mostra la pestanya del gestor de dimensions.
3. L'usuari introdueix el nom de la dimensió de la qual vol crear-ne l'índex.

4. L'usuari escull les columnes de la taula de fets que conformen la dimensió.
5. L'usuari ordena, si cal, les columnes segons el seu nivell d'agregació.
6. L'usuari inicia el procés de creació de l'índex prement el botó de crear la dimensió.
7. El sistema crea la dimensió i afegeix la informació del nou índex a la llista d'índexs creats en aquesta execució.

Escenari alternatiu 1

A partir del pas 6 de l'escenari principal d'èxit:

1. Si l'usuari no ha introduït el nom de la dimensió.
    - El sistema indica que no s'ha introduït cap nom per a la dimensió.
        - Es va al pas 3 de l'escenari principal d'èxit i el sistema mostra tota la informació especificada per l'usuari.

Escenari alternatiu 2

A partir del pas 6 de l'escenari principal d'èxit.

1. Si l'usuari no ha escollit cap columna per a la dimensió.
    - El sistema indica que no s'ha escollit cap columna per la dimensió.
        - Es va al pas 4 de l'escenari principal d'èxit i el sistema mostra tota la informació especificada per l'usuari.

9.1.5 Realitzar consulta

Escenari principal d'èxit

1. L'usuari indica que vol realitzar una consulta fent clic a la pestanya de l'executor de consultes.
2. El sistema mostra la pestanya de l'executor de consultes.
3. L'usuari escull el conjunt de mesures que vol obtenir de la taula de fets seleccionant-los de la llista de columnes disponibles.

4. L'usuari escull el conjunt de columnes respecte les quals vol que s'agrupin els resultats de les consultes.
5. L'usuari estableix les restriccions de la consulta.
6. L'usuari escull l'algorisme que vol utilitzar a l'hora de realitzar la consulta.
7. L'usuari indica que vol executar la consulta prement el botó d'executar consulta.
8. El sistema realitza la consulta especificada per l'usuari.
9. Si no existeix cap pestanya per a mostrar els resultats:
    - El sistema crea una nova pestanya per a la visualització de resultats.
10. Es va al pas 2 de l'Escenari principal d'èxit de Visualitzar resultats.

Escenari alternatiu 1

A partir del pas 7 de l'escenari principal d'èxit.

1. Si l'usuari no ha escollit cap columna de valors de la taula de fets per seleccionar:
    - El sistema indica que s'ha d'escollir algun valor per tal de poder realitzar una consulta.
        - Es va al pas 3 de l'escenari principal d'èxit i el sistema mostra tota la informació especificada per l'usuari.

Escenari alternatiu 2

A partir del pas 7 de l'escenari principal d'èxit.

1. Si l'usuari no ha especificat cap condició per a la consulta a realitzar:
    - El sistema indica que s'ha d'entrar com a mínim una restricció per a poder realitzar una consulta.
        - Es va al pas 5 de l'escenari principal d'èxit i el sistema mostra tota la informació especificada per l'usuari.

9.1.6 Visualitzar resultats

Escenari principal d'èxit

1. L'usuari indica que vol veure els resultats de l'última consulta executada fent clic a la pestanya de visualitzar resultats.
2. El sistema mostra el contingut de la taula que conté els resultats de l'última consulta dins d'una taula.

9.1.7 Executar codi

Escenari principal d'èxit

1. L'usuari indica que vol executar comandes escrites fent clic a la pestanya de l'executor de codi.
2. El sistema mostra la pestanya de l'executor de codi.
3. L'usuari introdueix el codi que vol executar.
4. L'usuari escull l'algorisme que vol fer servir per executar les consultes si n'ha especificat.
5. L'usuari inicia l'execució prement el botó Executar.
6. Per a cada comanda especificada per l'usuari:
   - Si és una comanda de creació d'índexs:
     1. El sistema crea l'índex indicat.
   - Si és una comanda de consulta:
     1. El sistema realitza la consulta.

Escenari alternatiu 1

1. A partir del pas 2 de l'escenari principal d'èxit:
2. L'usuari indica que vol esborrar les comandes introduïdes fins el moment prement el botó d'esborrar.
3. El sistema esborra les comandes introduïdes i mostra el quadre de text buit.

Escenari alternatiu 2

A partir del pas 6 de l'escenari principal d'èxit:

1. Si la comanda donada no existeix:
    - El sistema indica que la comanda no existeix.
    - Es torna al pas 3 de l'escenari principal d'èxit preservant el codi introduït per l'usuari.

Escenari alternatiu 3

A partir del pas 6 de l'escenari principal d'èxit:

1. Si la comanda donada té un format incorrecte:
    - El sistema indica que la comanda té un format incorrecte.
    - Es torna al pas 3 de l'escenari principal d'èxit preservant el codi introduït per l'usuari.

9.1.8 Modificar opcions

Escenari principal d'èxit

1. L'usuari indica que vol canviar les opcions d'execució escollint l'opció *configuració* del menú *edició*.
2. El sistema mostra la finestra d'opcions i bloca la finestra principal.
3. L'usuari modifica les opcions que creu convenients.
4. L'usuari confirma els canvis realitzats prement el botó d'aplicar.
5. El sistema modifica la configuració d'acord amb les opcions modificades, tanca la finestra d'opcions i desbloca la finestra principal.

Escenari alternatiu

A partir del pas 2 de l'escenari principal d'èxit:

1. L'usuari indica que no vol aplicar els canvis realitzats fins al moment prement el botó de cancel·lar.

2. El sistema tanca la finestra d'opcions i desbloca la finestra principal.

9.1.9 Codi generat

Escenari principal d'èxit

1. L'usuari indica que vol visualitzar els codi generat equivalent a les operacions fetes a través de la interfície gràfica. Ho fa seleccionant la pestanya de codi generat.
2. El sistema mostra el conjunt d'instruccions equivalent a les operacions fetes des de l'inici de l'execució del programa o bé des de l'últim cop que s'ha netejat la llista.
3. L'usuari indica que vol desar les instruccions generades en un arxiu.
4. El sistema mostra una finestra de selecció d'arxius perquè l'usuari esculli l'arxiu on vol que es desin les comandes.
5. L'usuari escull l'arxiu on vol que es desi el codi generat.
6. El sistema desa les comandes en l'arxiu indicat per l'usuari i tanca la finestra per a la selecció d'arxius.

Escenari alternatiu 1

A partir del pas 2 de l'escenari principal d'èxit:

1. L'usuari indica que vol netejar la llista d'instruccions generades.
2. El sistema neteja el contingut de la llista d'instruccions.

9.2 El model conceptual

El model conceptual pretén obtenir les relacions que hi ha entre els diferents conceptes que apareixen en la descripció del problema feta en l'anàlisi dels requeriments del sistema. Veiem que apareixen els següents conceptes claus:

9.2.1 Attribute

Representa una columna de la taula de fets.

- name: és el nom de la columna dins de la taula de fets.

Aquest concepte no apareix en el diagrama de classe perquè he optat per representar-lo amb un sol String.

9.2.2 Dimension

Representa una dimensió del cub de dades.

- name: nom de la dimensió.
- attributes: és la llista de d'atributs que conformen la dimensió. Estan ordenats per ordre d'agregació.

9.2.3 Condition

Representa una restricció sobre una dimensió d'una consulta.

Conté la informació:

- dimension: dimensió sobre la qual s'aplica la restricció.
- values: valors que es cerquen en la dimensió ordenats per ordre d'agregació.

9.2.4 Selection

Representa una consulta sobre el cub de dades.

Conté la informació:

- selected_attributes: és la llista d'atributs seleccionats en la consulta.
- condition: és el conjunt de condicions de la consulta.
- group_by: és el conjunt d'atributs respecte el valor dels quals s'han d'agrupar els valors seleccionats en la consulta.

El diagrama següent és el diagrama conceptual. Mostra gràficament les relacions que hi ha entre els diferents conceptes:

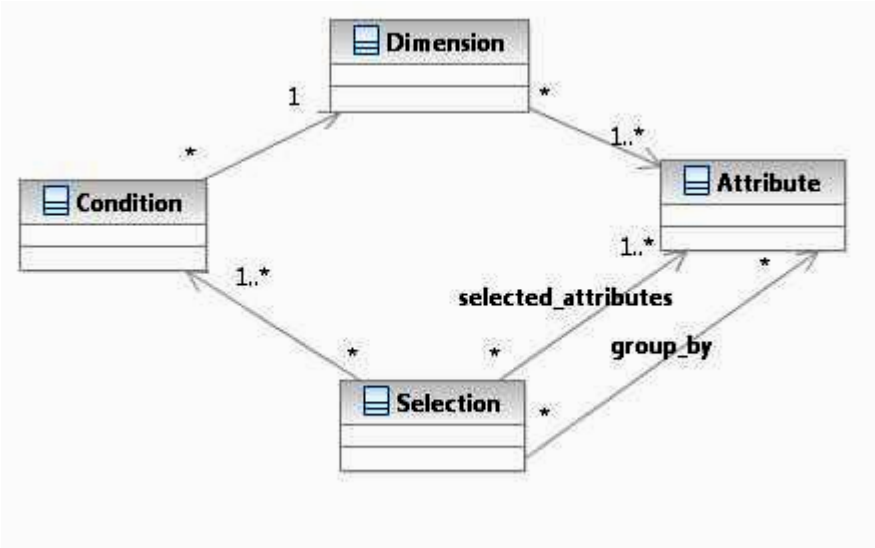


Figura 7: Model conceptual

9.3 El model de navegabilitat

El model de navegabilitat estableix la relació que hi ha entre les diferents finestres del programa.

Tal com podem veure en la figura 8, la interfície gràfica consta de dues finestres diferents. La primera és la finestra principal. Serveix per a dur a terme les principals funcionalitats del projecte: la creació d'índexs de dimensions, l'execució de consultes sobre la taula de fets, l'obtenció del codi generat i la visualització dels resultats de les consultes.

Dins de la finestra principal hi ha cinc pestanyes diferents per realitzar les diferents funcionalitats.

L'altra finestra, la finestra d'opcions, serveix per canviar la configuració de l'execució del programa d'una forma més còmode i gràfica.

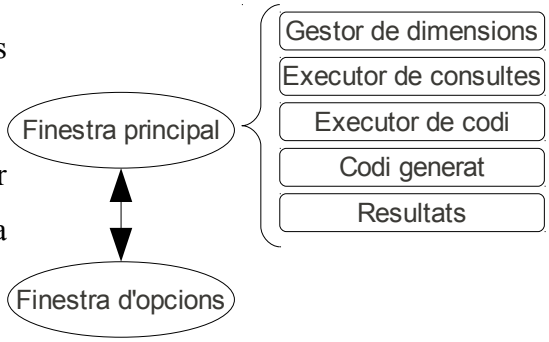


Figura 8: Model de navegabilitat

El funcionament de la interfície gràfica l'explico amb més detall a l'apartat 18.2.1.3 de l'Annex II: Manual d'usuari.

10 Disseny

En aquest capítol veurem els elements de disseny del projecte. Primer explicaré l'esquema de dades utilitzat en els índexs, el diagrama de classes del programa i els diagrames seqüència de les principals funcionalitats del projecte.

10.1 Esquema de dades de l'índex

Les úniques dades que es mantenen entre diferents execucions són el contingut de l'índex. Els índexs estan pensats per a ser creats una vegada i ser utilitzats múltiples cops.

La representació que s'ha optat pels índexs és la de una taula d'HBase amb una sola columna. Aquesta taula conté tots els índexs de les dimensions definides en el cub de dades. Cada fila de la taula representa una entrada diferent d'un índex i a la columna corresponent s'hi desen les claus de fila de la taula de fets que corresponen als valors que representa la fila de l'índex.

Cada entrada de l'índex estarà identificada per una clau composta pel nom de la dimensió i el conjunt de valors de les columnes d'aquesta dimensió ordenats pel seu nivell d'agregació. Els diferents valors de la clau estaran separats per un separador que no apareixerà ni en el nom de la dimensió ni en els valors que puguin prendre les seves columnes. L'esquema de la taula dels índexs seria el següent:

	identificadors
*nom_dimensió%(valors%)**	$clau_1$,$clau_2$,$clau_3$...

Taula 5: Esquema de la taula de l'índex

On $clau_i$ representa cadascun dels identificadors de la taula de fets que tenen els valors corresponents a aquesta entrada de l'índex.

Un exemple d'entrada d'índexs podria ser el següent: Per a la dimensió *dataNaixement*, tenim les columnes *any*, *mes* i *dia*. Llavors un identificador d'una entrada de l'índex per aquesta dimensió tindria la forma:

dataNaixement%1988%03%11%

10.2 Diagrama de classes

El diagrama de classes serveix per especificar el model que es farà servir per mantenir les dades en el sistema i quines parts en realitzaran les funcionalitats. Les classes que formen el projecte són les següents:

10.2.1 Condition

Condition és una interfície implementada pels diferents tipus de condicions que podem fer servir en una consulta. S'ha optat per un disseny de classes amb estructura recursiva per tal de poder implementar més fàcilment condicions complexes.

Un disseny de classes amb estructura recursiva és aquell disseny en què un objecte pot contenir objectes del seu propi tipus.

10.2.2 ConditionAND

Aquesta classe implementa la interfície Condition. Conté un conjunt de restriccions de consultes. A partir d'aquestes restriccions és capaç de generar un filtre per a l'escaneig d'una taula d'HBase tal que només agafarà les entrades de l'índex que compleixin totes les condicions.

Conté el camp:

- conditions: conjunt de Condition a partir de les quals es crearà un filtre que comprovarà que es compleixin totes.

10.2.3 ConditionNode

Aquesta classe implementa la interfície Condition. Representa una restricció d'una consulta. Conté una dimensió i el conjunt de valors els quals s'han fixat per aquesta. Pot generar un filtre per a un escaneig de la taula dels índexs tal que obtingui les entrades de l'índex que siguin iguals o bé comencin iguals als valors especificats en la dimensió.

Conté els camps:

- dimension: és una referència a una Dimensió. Aquesta és la dimensió sobre la qual s'aplicarà la restricció.

- values: és el conjunt de valors que s'han fixat per a la restricció sobre la dimensió.

10.2.4 ConditionOR

Aquesta classe implementa la interfície Condition. Conté un conjunt de condicions a partir de les quals serà capaç de generar un filtre per l'HBase tal que permeti obtenir les entrades de l'índex que compleixin alguna d'aquestes condicions.

Conté el camp:

- conditions: és el conjunt de condicions a partir de les quals es farà un filtre per a què es compleixi alguna d'elles.

10.2.5 Dimension

Aquesta classe conté la informació d'una dimensió del sistema.

Conté la següent informació:

- name: nom de la dimensió.
- attributes: nom de les columnes de la taula de fets que conformen la dimensió. Les columnes estan ordenades pel seu nivell d'agregació.

10.2.6 HBaseUtils

Aquesta classe implementa diferents funcionalitats per a treballar amb les taules de l'HBase. Conté operacions per obtenir el nom de totes les columnes d'una taula i per llegir-ne tot el seu contingut.

10.2.7 HumanReadableException

Aquesta classe hereta de la classe Exception de Java i està pensada per a què contingui dos missatges d'error. Serveix per a poder capturar una Excepció i poder-li donar un missatge d'error alternatiu més entenedor. El nou missatge se li assigna en el moment de la seva creació.

Conté:

- original_exception: conté la informació de l'excepció original.

- message: conté un missatge explicatiu i més entenedor per l'usuari.

10.2.8 Interpreter

Aquesta és la classe encarregada d'executar les diferents operacions invocades a través de la interfície textual.

Conté la informació:

- dimensions: és un hash que conté per cada nom de dimensió l'objecte Dimension corresponent.

10.2.9 MapReduce0

Aquesta és la classe encarregada de configurar i executar el MapReduce per tal de dur a terme la creació d'índexs.

Té les subclasses:

Mapper0: Aquesta classe implementa l'operació Map de l'algorisme Índex 1-nivell (apartat 7.1.1).

Mapper0Multiple: Aquesta classe implementa l'operació Map de l'algorisme Índex multinivell (apartat 7.1.2).

Combiner0: Aquesta classe implementa l'operació Combine dels algorismes Índex 1-nivell i Índex multinivell (dels apartats 7.1.1 i 7.1.2 respectivament).

Reducer0: Aquesta classe implementa l'operació Reduce dels algorismes Índex 1-nivell i Índex multinivell (dels apartats 7.1.1 i 7.1.2 respectivament).

10.2.10 MapReduce1

Aquesta és la classe encarregada de configurar i executar el MapReduce que durà a terme l'obtenció de les claus de fila de la taula de fets que satisfan les restriccions de la consulta.

Té les subclasses:

Mapper1: Aquesta classe implementa l'operació Map de l'algorisme Accés a l'índex d'un sol nivell

Mapper1Multiple: Aquesta classe implementa l'operació Map de l'algorisme Accés a l'índex multinivell.

FileReducer1: Aquesta classe implementa l'operació Reduce dels algorismes Accés a l'índex d'un sol nivell i Accés a l'índex multinivell quan el resultat s'escriu a l'arxiu intermedi.

IntermediateTable: Aquesta classe implementa l'operació Reduce dels algorismes Accés a l'índex d'un sol nivell i Accés a l'índex multinivell quan el resultat s'escriu a la taula intermèdia. S'implementa amb una classe diferents de la anterior, perquè per escriure el resultat d'un MapReduce en una taula s'ha d'heretar la classe TableReducer i, en canvi, per escriure'l en un arxiu s'ha d'heretar la classe Reducer.

10.2.11 MapReduce2

Aquesta classe s'encarrega de configurar i executar el MapReduce que a partir d'un conjunt de claus de fila de la taula de fets. Accedeix a la taula de fets i n'obté els valors de les columnes especificades, n'agrupa els valors segons les columnes d'agrupació que també s'han especificat. Finalment, agrega els valors de cada columna seleccionada per a cada grup i en desa la informació obtinguda dins la taula de resultats.

Té les subclasses:

ScanMapper: Aquesta classe implementa l'operació Map de l'algorisme Filtered Source Scan (FSS).

KeyFilterMapper2: Aquesta classe implementa l'operació Map de l'algorisme Index Filtered Scan (IFS).

IntermediateTableMapper2: Aquesta classe implementa l'operació Map de l'algorisme Index Random Access (IRA).

Combiner2: Aquesta classe implementa l'operació Combine pels algorismes: Filtered Source Scan (FSS), Index Filtered Scan (IFS) i Index Random Access (IRA).

Reducer2: Aquesta classe implementa l'operació Reduce pels algorismes: Filtered Source Scan (FSS), Index Filtered Scan (IFS) i Index Random Access (IRA).

10.2.12 MapReduceScanner

Aquesta classe s'encarrega d'executar i configurar l'execució de l'escaneig de la taula de fets (FSS) per tal de resoldre una consulta.

10.2.13 Parameters

Aquesta classe és l'encarregada de gestionar els paràmetres de configuració de l'execució actual i de processar el paràmetres rebuts inicialment pel programa en la seva crida.

10.2.14 RowCounter

Aquesta classe implementa la funcionalitat de comptar el nombre d'entrades que té una taula de l'HBase

10.2.15 Selection

Aquesta classe conté la informació d'una consulta de selecció de valors feta per l'usuari.

Conté la informació:

- attribute_selection: és la llista de columnes seleccionades de la consulta.
- condition: és una referència a la classe condició que conté les restriccions de la selecció.
- group_by: és el conjunt de columnes respecte el valor de les quals s'ha d'agrupar els valors obtinguts de la taula de fets.

10.2.16 Utilities

Aquesta classe conté un conjunt d'operacions que es requereixen en diferents parts del projecte i que s'ha de garantir que el seu funcionament sigui el mateix. Per aquest motiu se n'ha fet la implementació en aquesta classe, per a augmentar la canviabilitat del projecte. Són operacions per la concatenació d'elements.

10.2.17 CodeExecutorPanel

Aquesta és la classe encarregada de mostrar i gestionar la pestanya per executar codi de la finestra principal de la interfície gràfica.

10.2.18 ConfigurationDialog

Aquesta classe és la responsable de mostrar la finestra d'opcions i gestionar la interacció amb l'usuari.

10.2.19 DimensionPanel

Aquesta classe és la responsable de gestionar la interacció de l'usuari amb la pestanya de gestió de les dimensions que es troba a la finestra principal.

10.2.20 GeneratedCodePanel

Aquesta classe té la responsabilitat de gestionar i mostrar la pantalla de codi generat de la finestra principal.

10.2.21 Main

És la classe encarregada de preparar i iniciar l'execució del programa.

10.2.22 MainWindow

Aquesta classe és l'encarregada de mostrar la finestra principal i gestionar-ne les diferents pestanyes.

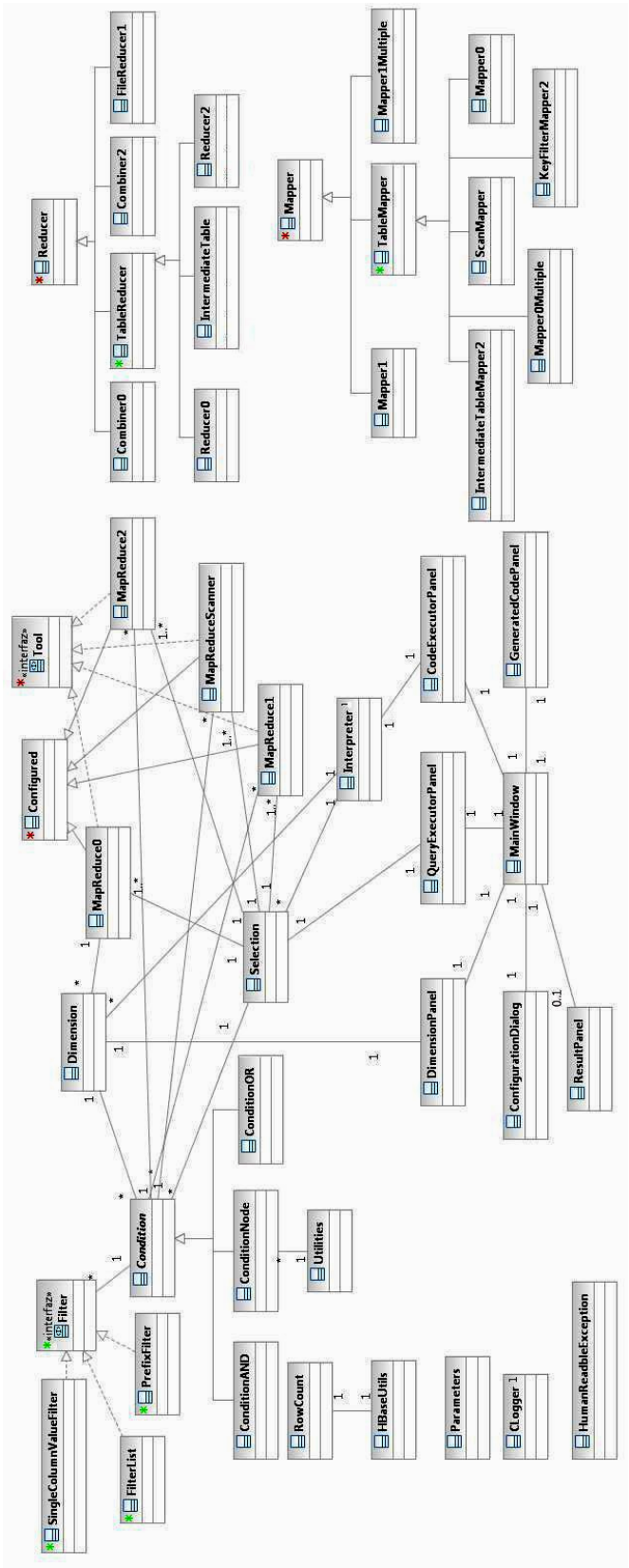
10.2.23 QueryExecutorPanel

Aquest classe s'encarrega de mostrar i gestionar la pestanya de la finestra principal que permet realitzar consultes.

10.2.24 ResultPanel

Aquesta és la classe que gestiona i mostra la pestanya que mostra el resultat de l'última consulta executada.

A continuació mostro el diagrama de classes del projecte i les relacions d'herència amb les classes de les APIs d'HBase i Hadoop:



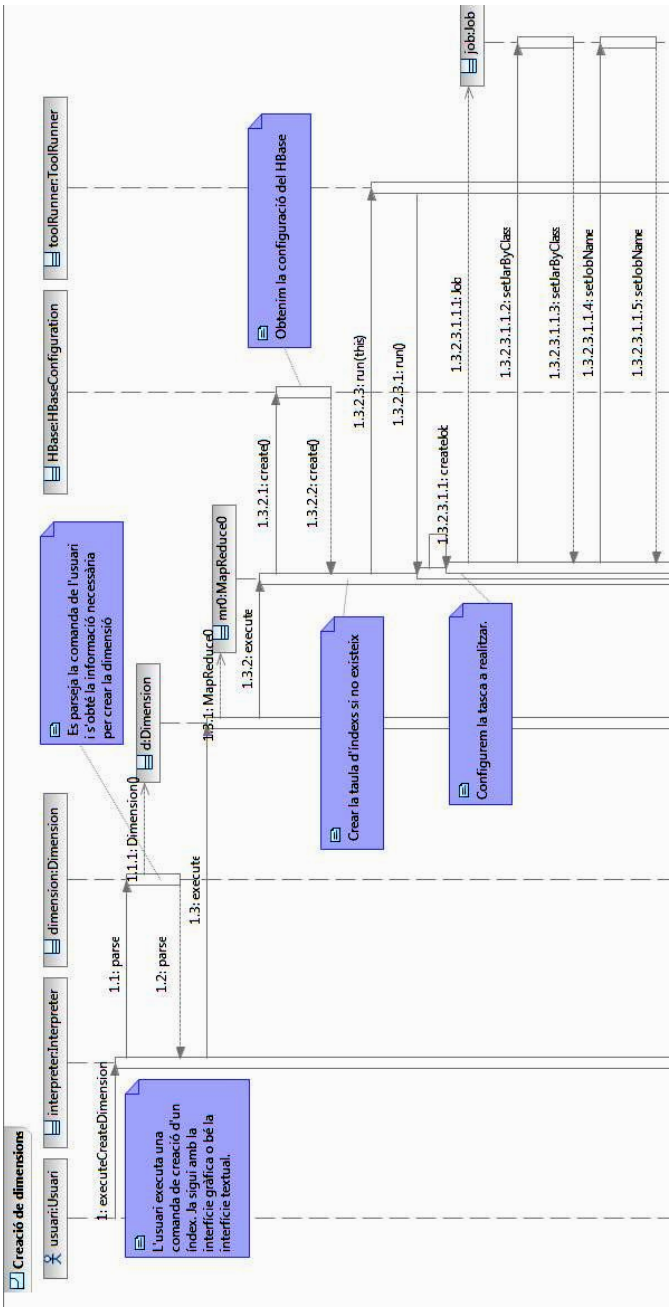
Llegenda
-He marcat amb un asterisc vermell les classes que són de l'API de Hadoop.
-He marcat amb un cercle verd les classes que són de l'API de HBase.

Figura 9: Diagrama de classes

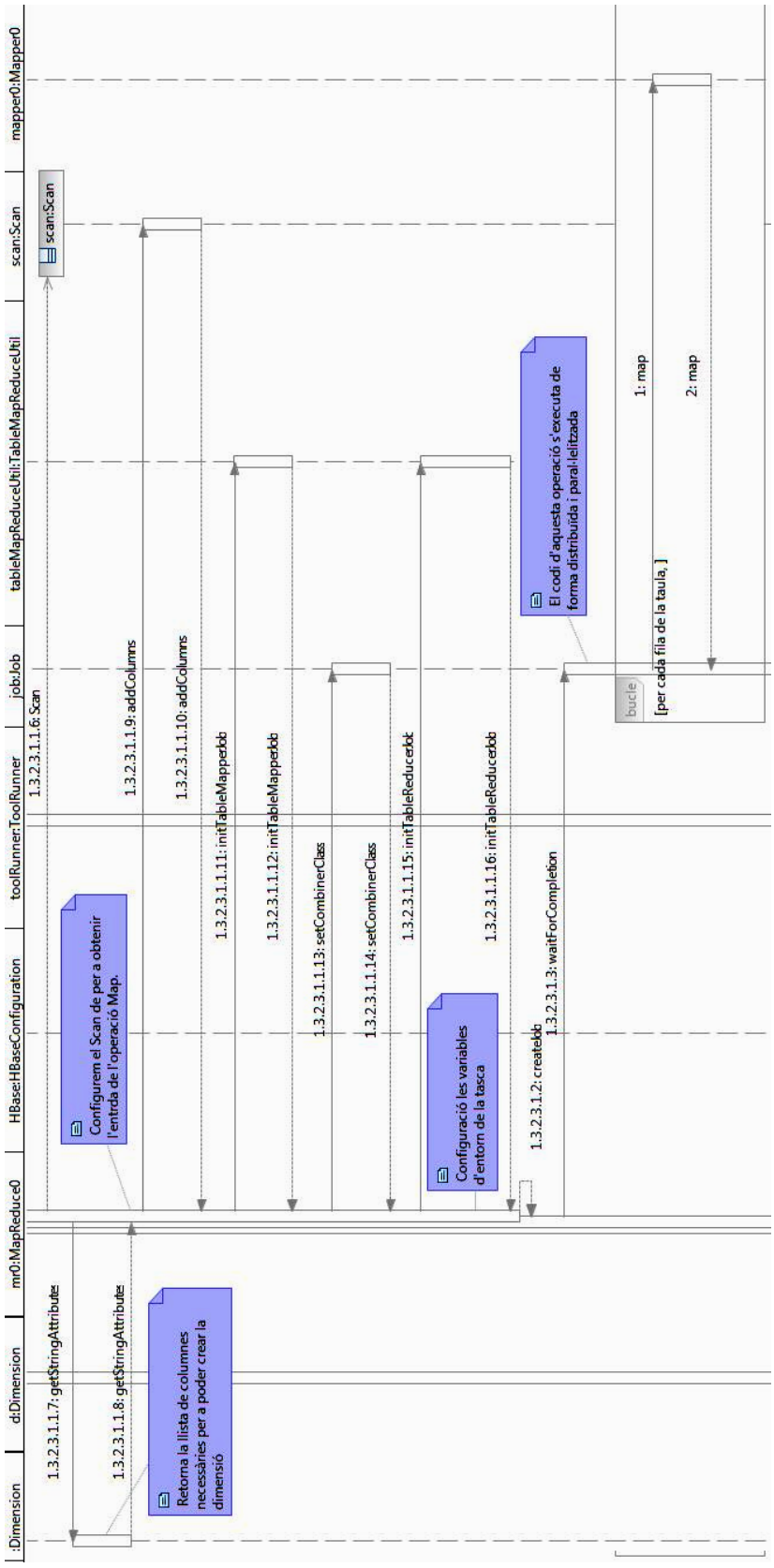
10.3 Diagrames de seqüència

A continuació presento els diagrames de seqüència de les diferents operacions que pot realitzar el projecte: la creació d'índexs i la realització de consultes utilitzant diferents algorismes.

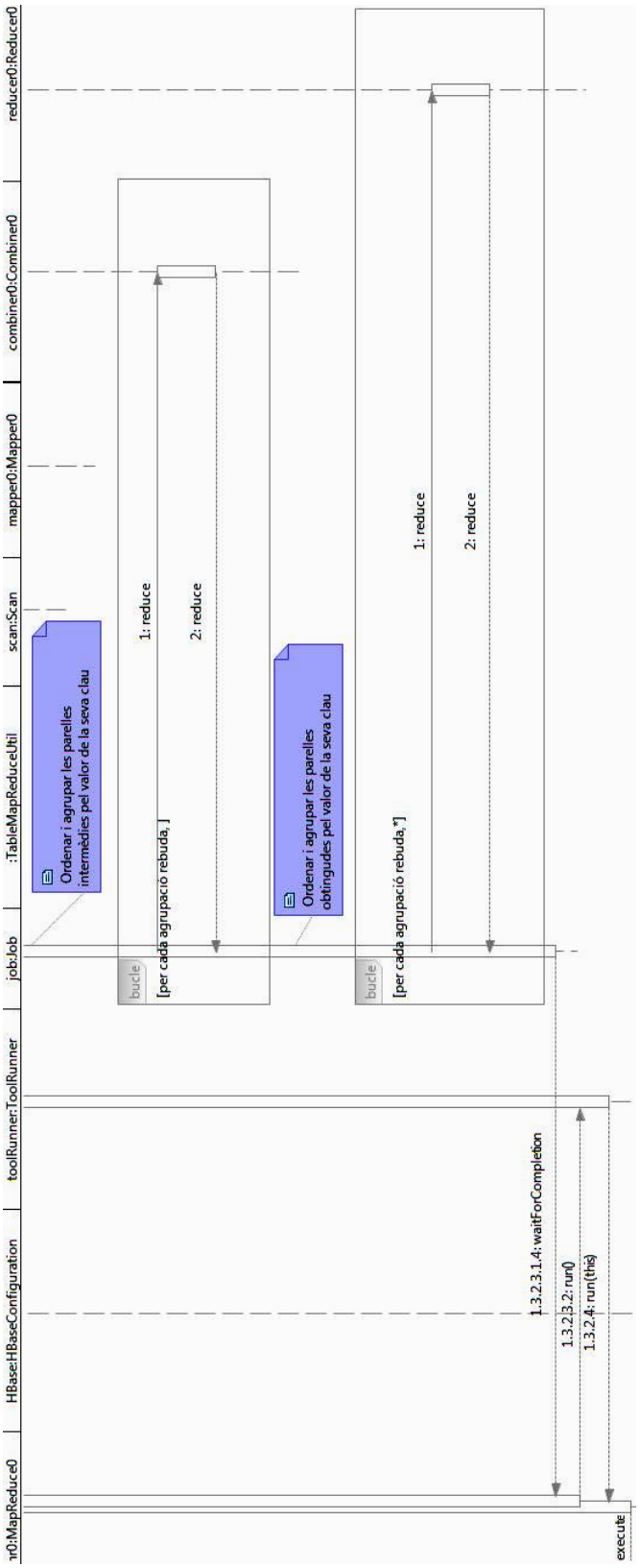
10.3.1 Diagrama de seqüència de la creació d'índexs



*Continua a la pàgina següent.



*Continua a la pàgina següent.


reducer0:Reducer0
combiner0:Combiner0
mapper0:Mapper0
scan:Scan
:TableMapReduceUtil
job:Job
toolRunner:ToolRunner
HBase:HBaseConfiguration
Ordenar i agrupar les parelles intermèdies pel valor de la seva clau
1: reduce
2: reduce
bucle
[per cada agrupació rebuda,]
Ordenar i agrupar les parelles obtingudes pel valor de la seva clau
1: reduce
2: reduce
bucle
[per cada agrupació rebuda, *]
1.3.2.3.1.4: waitForCompletion
1.3.2.3.2: run()
1.3.2.4: run(this)
execute

En aquest diagrama de seqüència hi podem veure un exemple d'execució per a la creació d'un índex per a una dimensió. En aquest cas, es tracta de la creació d'un índex d'un sol nivell (apartat 7.1.1). El cos de l'operació Map, Reduce i Combine apareixen buits perquè són els algorismes pertinents que ja he explicat en el capítol d'algorismes. El tractament d'errors referents al format de la comanda es fa dins de l'operació *parse* de la classe *Dimension*.

Ometo el diagrama de seqüència per a la creació d'índexs multinivell perquè és el mateix que aquest però utilitzant la classe Mapper0Multiple enlloc de la classe Mapper0.

10.3.2 Diagrama de seqüència d'una consulta (IRA)

Ara mostraré el diagrama de seqüència del procés d'execució d'una consulta fent servir l'algorisme IRA amb índexs d'un sol nivell. El diagrama resultat és llarg i complex. Degut a les seves dimensions l'he partit en 5 parts i he posat explicacions entre les diferents parts per tal de deixar clar què ha passat ens els punts de tall.

Aquesta primera part del diagrama conté la seqüència de passos que es segueixen per parsejar una consulta introduïda per l'usuari.

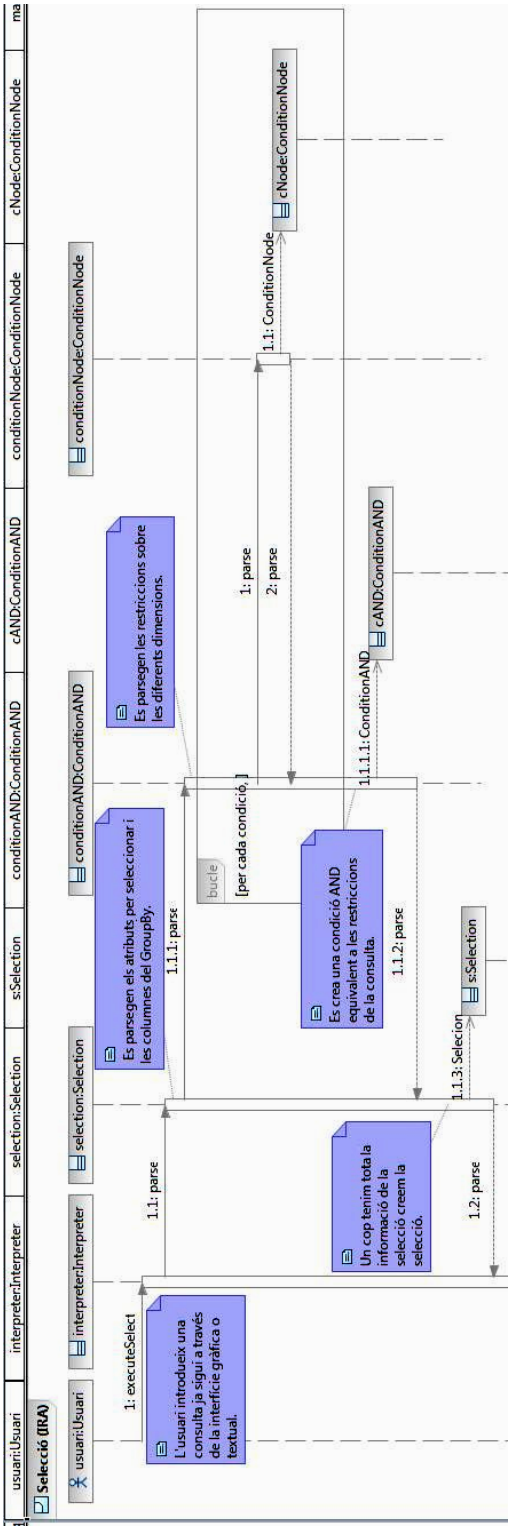


Figura 10: Diagrama de seqüència d'una consulta (part 1)

Un cop l'interpret ha obtingut l'objecte Selection a partir de la comanda de l'usuari, l'executa. Aquest és el punt en què comença el diagrama següent:

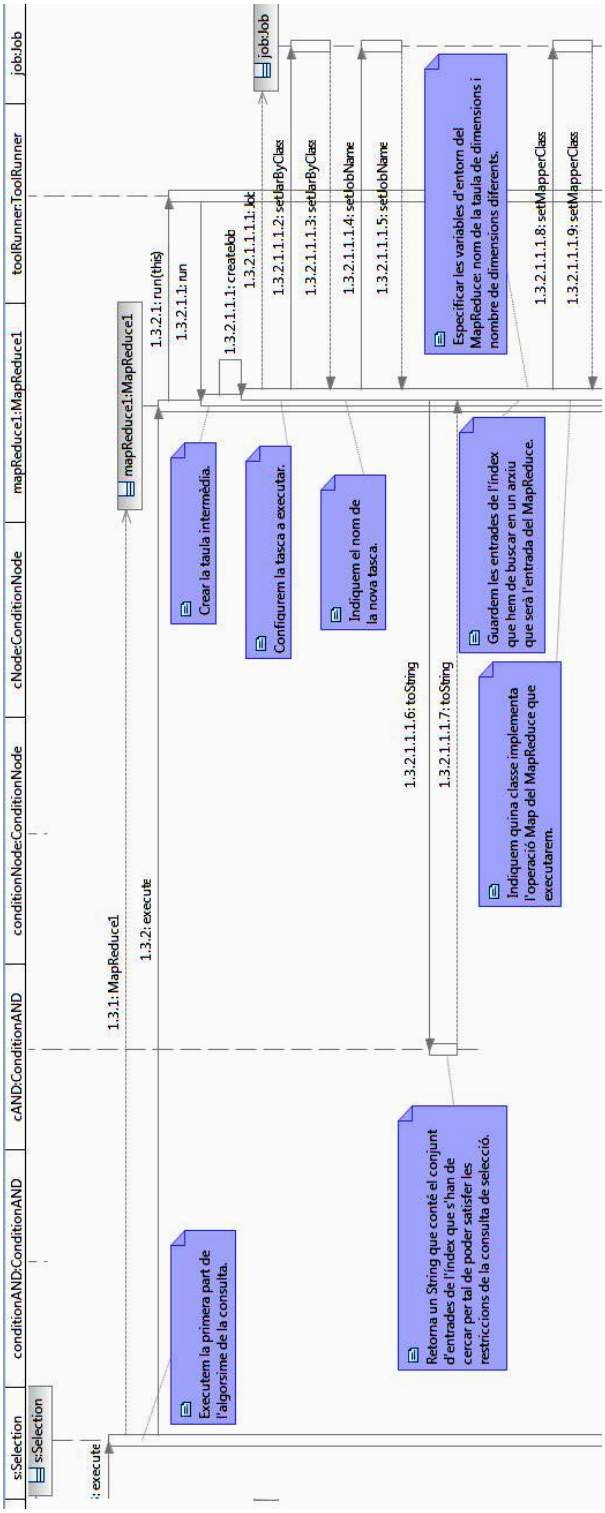


Figura 11: Diagrama de seqüència d'una consulta (part 2)

A la figura 11 es veu el procés que es realitza per configurar l'execució del primer MapReduce de l'algorisme de consulta. En aquest cas, és l'algorisme de les claus en un índex d'un sol nivell.

Un cop s'arriba al final del diagrama anterior veiem que ja s'ha configurat quina serà l'operació Map i que falta especificar quina serà l'operació Reduce. El diagrama següent parteix des d'aquest punt:

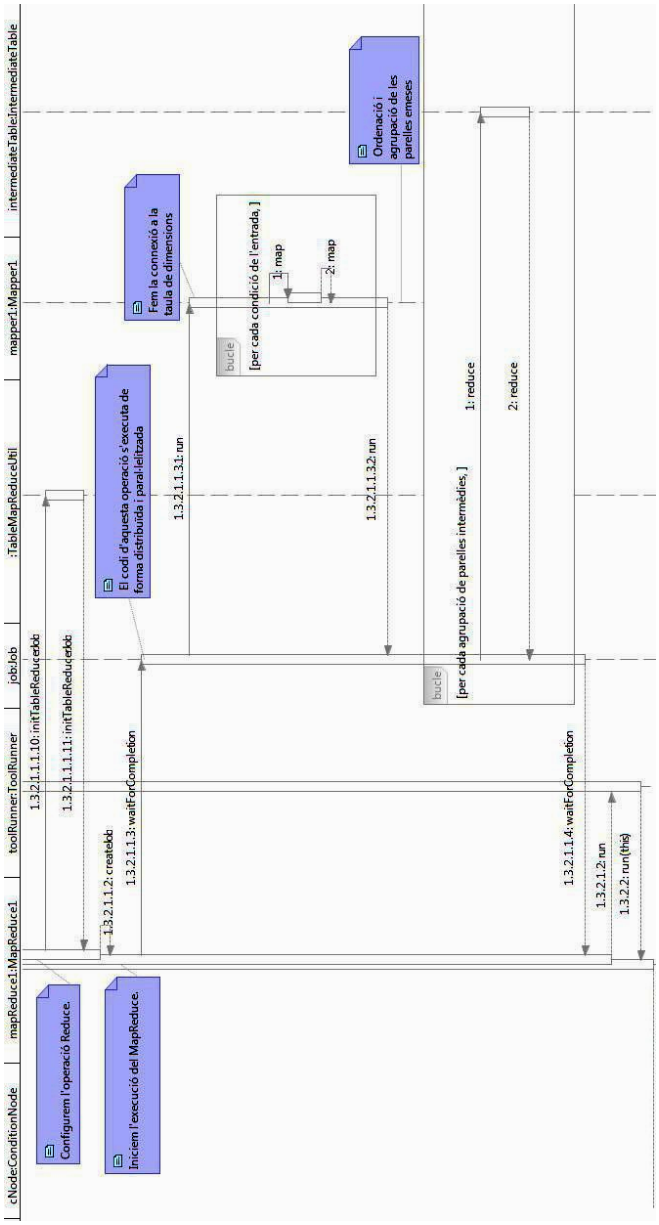


Figura 12: Diagrama de seqüència d'una consulta (part 3)

La figura anterior mostra el diagrama de seqüència fins el punt en que s'acaba l'execució del primer Map Reduce. El següent diagrama ens mostra la configuració del segon MapReduce:

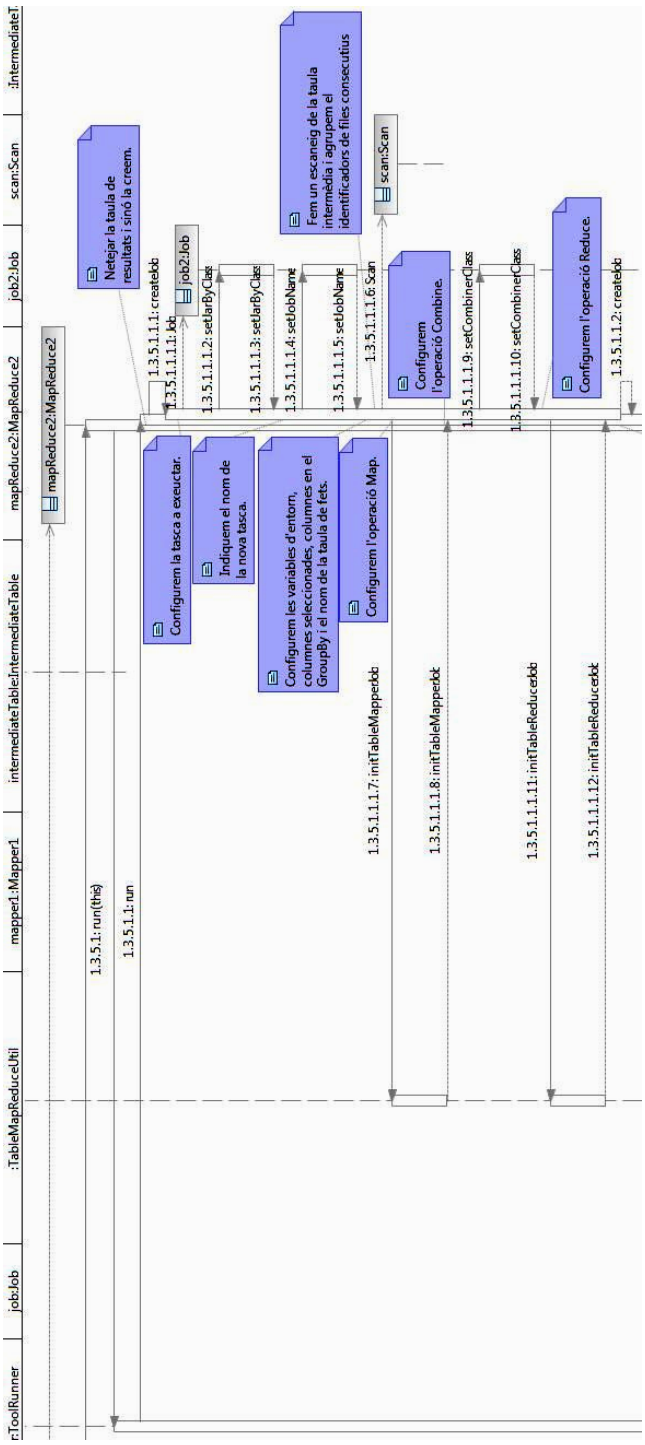


Figura 13: Diagrama de seqüència d'una consulta (part 4)

El diagrama següent mostra l'execució del segon MapReduce:

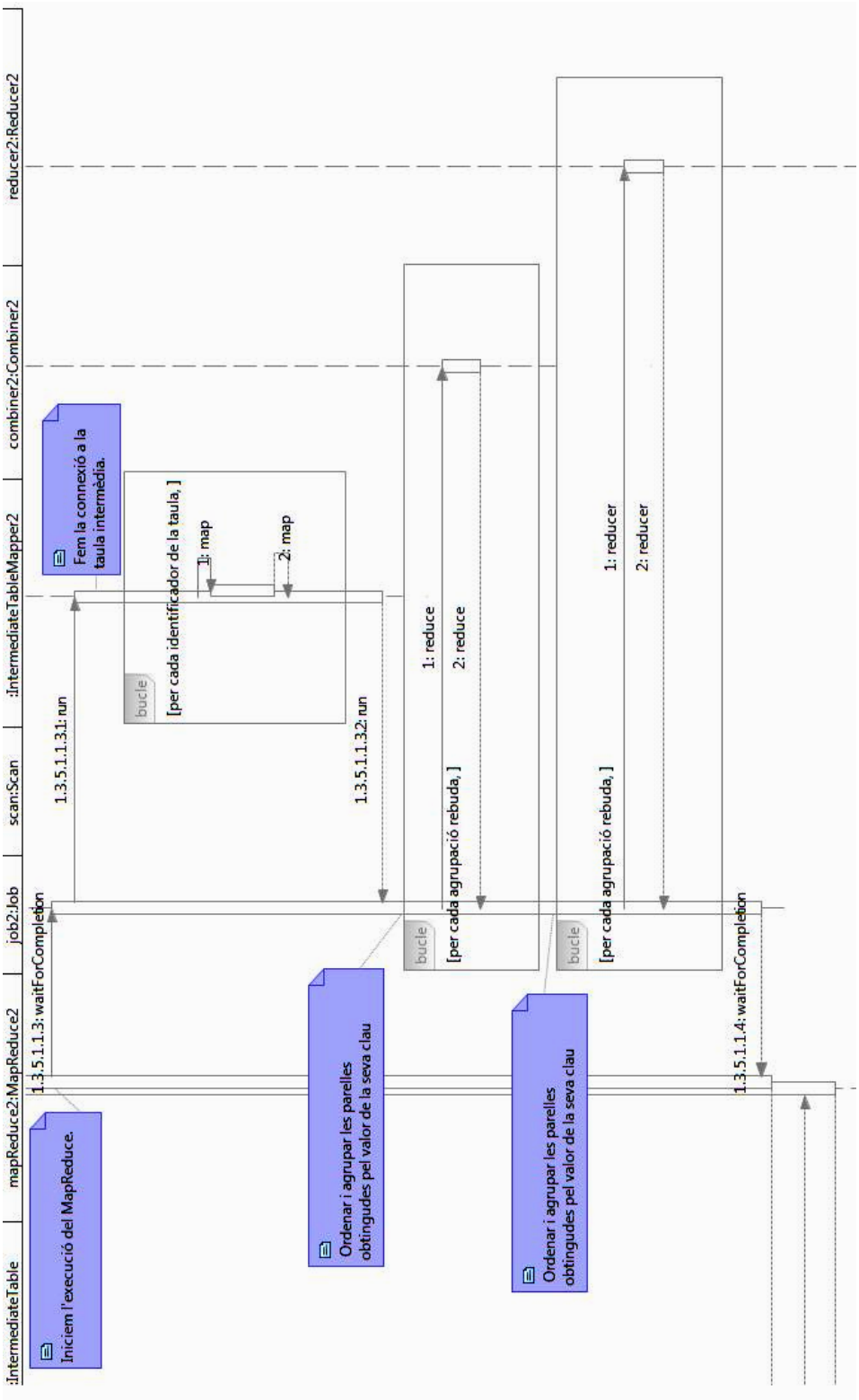


Figura 14: Diagrama de seqüència d'una consulta (part 5)

En el diagrama de seqüència en conjunt, hi podem apreciar les diferents etapes d'execució d'una consulta qualsevol. Aquest diagrama és fet per a una execució que utilitza els índexs d'un sol nivell, però en cas que es volgués fer servir l'índex de múltiples nivells únicament s'hauria de substituir la classe Mapper0 del primer MapReduce per la classe Mapper0Multiple.

Per a l'algorisme IFS la seqüència de passos és molt semblant. Canvien les classes que s'utilitzen en els MapReduce per les que implementen els algorismes que corresponen. L'altre canvi que hi ha és: el recorregut de la taula que es fa entre el primer i el segon MapReduce es canvia a l'algorisme IFS pel procés de llegir els identificadors de l'arxiu intermedi i guardar-los per poder crear el Bitmap dins de la operació Map del segon MapReduce executat.

El diagrama per a l'algorisme FSS canvia una mica més i només requereix d'un MapReduce. És a dir, que canviant les classes del primer MapReduce, canviant-ne la configuració de forma pertinent i eliminant el segon MapReduce també aconseguiríem el seu diagrama de seqüència.

Aquesta gran facilitat per canviar el funcionament de l'aplicació amb tants pocs canvis demostra que l'objectiu de què el projecte sigui modificable s'ha aconseguit amb escreix.

11 Implementació

La implementació d'aquest projecte ha estat llarga i complexa. En el capítol 5 ja he explicat el model de desenvolupament que he seguit. Ara, però, faré incidència en aspectes més tècnics de la implementació i diferents problemes que m'he anat trobant al llarg del desenvolupament del projecte.

11.1 Programa principal

El programa principal consta de la implementació dels diferents algorismes i de la interfície textual i la gràfica.

Al llarg del desenvolupament del projecte he fet diferents implementacions, a continuació comento les funcionalitats han tingut cadascuna d'elles.

- **Versió 1**: Aquesta primera versió permetia la creació d'índex d'un sol nivell, i les consultes amb condicions disjuntives i encara no es permetien la agrupacions. Només constava d'una interfície textual.
- **Versió 2**: En la segona versió vaig canviar el comportament de les condicions: les condicions van passar a ser disjuntives si eren sobre la mateixa dimensió o bé conjuntives si eren sobre dimensions diferents. En aquesta versió també vaig introduir les agrupacions respecte el valor de diverses columnes.
- **Versió 3**: Vaig modificar la implementació de l'IFS de tal manera que fes servir un arxiu intermedi a l'hora de crear el filtre per a la taula de fets, fins llavors es feia utilitzant una taula intermèdia. Vaig crear la interfície gràfica que permet crear els índexs i les consultes de forma més amigable.
- **Versió 4**: En aquesta versió es va afegir la possibilitat d'escollir quin algorisme s'havia de fer servir per executar una consulta, podies triar entre: el Index Filtered Scan, el Index Random Access i el Filtered Source Scan. Vaig afegir també les operacions Combine a l'algorisme de creació de dimensions i als algorismes IFS, IRA i FSS. Vaig incorporar, també, alguns cronòmetres en el projecte per tal de poder mesurar els temps clau per tal de poder fer una bona avaluació dels rendiments.

- **Versió 5**: Vaig afegir la possibilitat d'habilitar la compressió GNU Zip (gz) a les taules de l'HBase. Per poder treballar amb el TPC-H vaig canviar les operacions d'agregació per tal de treballar amb *floats* enlloc de *ints*. Vaig afegir el càlcul del factor de selecció i vaig canviar l'algorisme IFS perquè fes servir bitmaps enlloc d'un filtre d'HBase per a escanejar la taula.

  Entre la versió 4 i la 5, va aparèixer la versió 0.90.1 d'HBase amb algunes millores respecte l'anterior (0.20.6). Es va decidir actualitzar el sistema a la nova versió i vaig haver d'adaptar tot el codi ja que canviaven algunes operacions de la API.

Al llarg del desenvolupament del projecte m'he trobat amb diferents problemes que comento en el següent apartat:

11.1.1 Problemes trobats

Al llarg del desenvolupament del projecte m'he trobat diversos problemes que han implicat diversos canvis en la feina feta fins al moment i fins i tot modificacions en la planificació inicial (capítol 8.2):

- **ToolRunner**: El ToolRunner és una classe Java que està inclosa en la API de Hadoop MapReduce que serveix per a configurar l'execució del MapReduce mitjançant diferents paràmetres. En alguns dels manuals de Hadoop que he mirat fan servir aquesta classe, però en cap cas diuen que sigui obligatori el seu ús per tal de què el programa es pugui executar de forma distribuïda.

  Inicialment no utilitzava aquesta classe i les proves executades de forma local (en un sol ordinador) em funcionaven, però quan vaig voler-lo provar en el clúster de casa, em vaig trobar que no funcionava. Llavors vaig recórrer als exemples que havia agafat i em vaig adonar que en alguns d'ells es feia servir el ToolRunner i que els MapReduce implementaven una classe anomenada Tool.

  Vaig provar de fer aquests canvis en el desnormalitzador i, així, vaig aconseguir executar-lo de forma distribuïda. En conseqüència, vaig haver de modificar tots els algorismes del programa principal per tal de què s'executés de forma distribuïda.

- **Java Heap Space Exception**: Aquesta excepció va ser la primera excepció d'execució que vaig obtenir quan el projecte funcionava bé per a quantitats de dades petites. Em va aparèixer per generar les proves amb un Scale Factor de 0,05. Era conseqüència de què la mida de la memòria per defecte dels Threads de Hadoop és de 256MB i la vaig ampliar fins a 1024MB i ja no se'm va tornar a repetir el problema. Per fer-ho, vaig haver de modificar l'arxiu de configuració de Hadoop ($HADOOP_HOME[8]/conf/mapred-site.xml) i afegir el següent:

```
<property>
    <name>mapred.child.java.opts</name>
     <value>-Xmx1024M</value>
</property>
```

- **Scanner Timeout Exception**: Aquest problema també em va aparèixer al intentar generar els jocs de proves amb un Scale Factor de 0,05. És causat per l'escànner que obté la informació d'una taula per a ser processada. Si el sistema és lent, o té un disc dur lent[9], aquesta lectura pot tardar més que el temps estimat per defecte i llavors es produeix aquesta excepció. Per solucionar-ho vaig augmentar el temps de què disposa el escànner, que per defecte és de 60 segons, i es va solucionar el problema. A mesura que vaig anar fent proves més grans, vaig haver d'anar augmentant aquest temps. A l'arxiu de configuració de l'HBase ($HBASE_HOME[10]/conf/hbase-site.xml) hi vaig haver d'afegir:

```
<property>
    <name>hbase.regionserver.lease.period</name>
    <value>60000000</value>
</property>
```

  Aquest temps està representat en mil·lèsimes de segon i equival a 16 hores i 40 minuts.

- **Massa connexions**: en la implementació de l'algorisme IRA, es produïa l'error: *Error: unable to create new native thread*. Aquest es produïa perquè la màquina virtual es quedava sense memòria ja que per cada parella es feia una connexió a la base de dades per tal d'obtenir el valor que ens interessa i per molt que forcés la neteja de la connexió el *Garbage Collector* no era prou ràpid. Per solucionar el problema vaig estar

8 Directori d'instal·lació de Hadoop, veure l'Annex II: Manual d'usuari.
9 El disc dur del meu portàtil és de 5.400 rpm.
10 Directori d'instal·lació d'HBase, veure l'Annex II: Manual d'usuari.

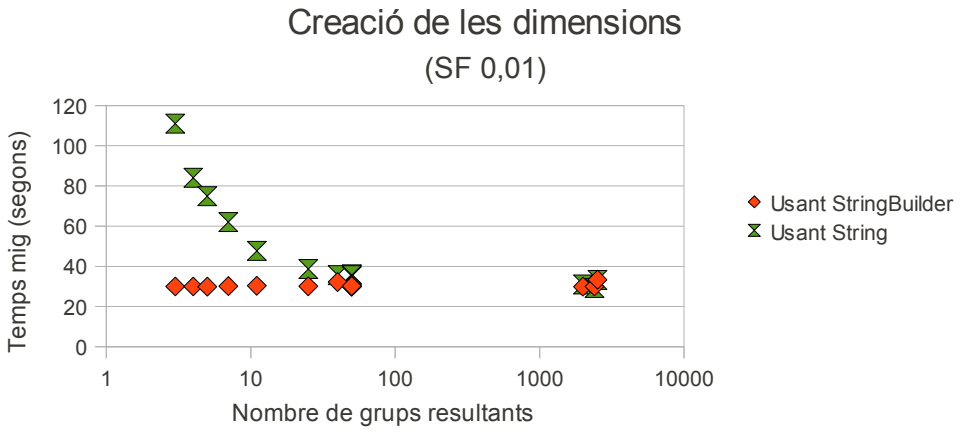
estudiant la API de Hadoop i vaig trobar que les classes Mapper i Reducer tenen el mètode

***public void** run(**Context** context)*

que s'executa un cop abans processar un bloc de dades amb les operacions Map o Reduce. Amb aquest s'invoca el cos de l'operació Map per a cadascuna de les parelles d'entrada. Així, dins de l'operació *run* hi vaig establir la connexió a la taula que ens interessava a la base de dades i llavors podia utilitzar la mateixa connexió per a totes les iteracions i netejar-la al final.

Aquest mateix estil de solució em va servir per a poder fer els bitmaps (veure apartat 7.2.2.2) de manera que es creessin només un cop per a cada bloc de dades que s'ha de processar, cosa que va suposar una gran millora de rendiment.

- **Rendiment de la creació de dimensions**: a la primera implementació que vaig fer de l'algorisme de creació d'índexs (la versió d'un sol nivell, apartat 7.1.1), vaig detectar-hi un problema de rendiment. Es pot veure a la gràfica qui hi ha a continuació:



Gràfica 1: Rendiments de String i StringBuilder en la creació dels índexs

Els resultats d'aquesta gràfica s'han obtingut amb un Scale Factor[11] de 0,01. Podem veure-hi clarament que usant la classe String de Java per fer la concatenació dels identificadors el cost creix exponencialment com menys grups té la dimensió. Això és degut a què la classe String, cada cop que es fa una concatenació es copien els dos Strings a una nova regió de memòria. Això fa que si has de concatenar *n* identificadors

11 Per més informació sobre el factor de selecció veure el capítol TPC-H (12).

el primer el copiaràs *n-1* vegades, el segon *n-2* vegades... i ja es veu que surt un cost exponencial.

Es veuen més afectats els índexs amb un nombre de grups menors perquè implica que els grups d'identificadors són més grans i, per tant, els identificadors s'han de copiar més vegades.

Per aquest motiu, vaig estar buscant en la API de Java i vaig trobar la classe StringBuilder que és expressa per fer concatenació de String de forma eficient i es pot veure que el cost de creació dels índexs no es veu afectat pel nombre d'agrupacions.

11.2 Generació dels jocs de proves

La generació dels jocs de proves es fa en diferents passos i utilitzant diferents programes que he creat o he modificat (el generador TPC-H). Són els següents programes:

11.2.1 Generador de les taules del TPC-H

He hagut de modificar el generador per tal de què posés totes les entrades de la taula en el mateix arxiu i a més, que guardés tota la informació necessària per poder fer les insercions (veure) en una taula de l'HBase.

Les modificacions en el generador del TPC-H, al final no han estat gaire complexes, però en un principi vaig intentar modificar-lo perquè generés les dades directament desnormalitzades, cosa que va resultar ser impracticable pels següents motius:

- El generador original està pensat per a generar jocs de proves de grans dimensions, pot generar proves de fins a 100TB (aproximadament) i per tant, és impossible poder tota la informació a la memòria RAM per tal de poder fer la desnormalització ràpidament. Val a dir que en les proves realitzades tampoc s'ha pogut arribar a volums de dades tant grans com perquè no cabessin a la memòria RAM (veure capítol de proves), però en el moment de prendre la decisió no ho sabíem.
- Tots els valors són aleatoris, no hi ha cap fórmula per calcular-los. A més, la llavor utilitzada va canviant de forma aleatòria i t'hauries d'anar guardant tots els

valors de les llavors i tindríem un problema d'espai com l'anterior o tornar-los a re-calcular cosa que alentiria terriblement el procés

Per aquests motius, es va decidir que la desnormalització s'havia de fer a posteriori i que, per fer-ho, hauria de programar el desnormalitzador. Per fer aquesta desnormalització hi havia dues opcions:

- Utilitzar un gestor de bases de dades relacionals: té l'avantatge no s'ha de programar res. Pots fer servir un conjunt d'instruccions SQL per fer la desnormalització.
- Utilitzar HBase i MapReduce: té l'avantatge que pot aprofitar la potència de càlcul del clúster on després farem les proves per generar, també, els jocs de proves. Té l'inconvenient que s'ha de programar.

Es va decidir realitzar la segona opció ja que ens permetia utilitzar més fàcilment el clúster, no calia un SGBD i ens estalviàvem la copia de les dades entre SGBDs. Així, vaig modificar la sortida del generador del jocs de proves perquè escrivís tota la informació en un sol arxiu en el format que accepta la shell de l'HBase.

Aquesta opció ens va permetre carregar les dades del test dins de l'HBase en una forma normalitzada, però era extremadament lenta. Per aquest motiu vaig programar un carregador de dades des de l'arxiu a l'HBase.

Del generador modificat del TPC-H n'he fet dues versions més a part de l'anterior. Aquestes dues versions, són expresses pel carregador de dades que he fet. N'he fet dues perquè les proves que es generen ocupen molt d'espai i en la segona versió vaig canviar el format de l'arxiu de sortida per tal de reduir-ne la mida. Inicialment el format de l'arxiu per a cada línia era el següent:

$$nom\_taula|clau|nom\_columna|valor$$

Es tenia una línia per cada valor de cada columna per a cadascuna de les taules.

Després de les modificacions, va passar a ser el següent:

$$nom\_taula|clau|nom\_columna_1|valor_1|nom\_columna_2|valor_2|...|nom\_columna_n|valor_n$$

Amb aquests canvis vaig aconseguir reduir la mida fins a una mica més de la meitat.

11.2.2 Carregador de dades

El Carregador de dades o també anomenat Loader és el programa encarregat de llegir l'arxiu generat i carregar-lo a l'HBase. Llegeix l'arxiu que se li indica d'entrada i en processa les files una per una. En cada fila espera trobar-hi el nom de la taula en què va la tupla, la clau de la tupla, i llavors un conjunt de parelles columna valor de la tupla. Per cada fila, crea una operació Put i la configura per tal d'inicialitzar tots els valors de la fila de cop.

11.2.2.1 Rendiment

El rendiment de la càrrega de les proves és importat, suposa aproximadament el 7%[12] del temps de càrrega dels jocs de proves. Al principi, amb un Put per cada valor de cada columna s'inserien de mitjana 3 valors per segon. Vaig provar les següents configuracions:

- **Desactivar l'autoflush**: de la connexió i fer el *flush* manualment per tal d'intentar optimitzar l'ús de la connexió va comportar una millora del rendiment que va augmentar la velocitat mitjana a 18 insercions per segon. Aquesta millora és deguda a què amb l'autoflush activat forces que s'enviïn les dades al servidor a cada inserció que es fa. En canvi, si es desactiva, s'enviaran quan tingui la *cache* d'operacions per fer plena. Les enviarà totes les operacions de cop al gestor de bases de dades cosa que fa estalviar un gran nombre de connexions. Ens podem permetre aquest retard perquè només estem fent insercions a la taula però no la llegim. Després de les insercions es força un *flush* per tal de garantir que s'insereixin totes les dades.

- **Desactivant WAL**: El WAL (Write Ahead Log) és un registre en què s'afegeix una entrada per cada operació que s'ha d'executar sobre la base de dades i que realitza una modificació en el contingut de la base de dades (inserir/esborrar/modificar dades).

  Per a cada petició que rep, l'enregistra en el log i retorna al flux del programa i després l'executa pel seu compte. En canvi, amb el WAL desactivat quan rep una petició, retorna el flux directament i estalvia l'escriptura del log. Així, correm el risc de perdre les dades inserires en cas que falli el sistema. Ens ho podem permetre perquè estem en un entorn de desenvolupament controlat, però per utilitzar aquesta tècnica en un entorn de producció és molt arriscat.

12 Carregant les dades del TPC-H amb un Scale Factor del 0,5, veure gràfica 3, pàgina 120.

La velocitat d'inserció de valors amb el WAL desactivat és de 8 insercions per segon.

- **Desactivant l'autoflush i el WAL simultàniament**: obtenim una velocitat mitjana de 21 insercions per segon.

Per tant, ha estat aquesta última configuració la que he utilitzat per a poder fer la càrrega i implica una velocitat 7 vegades superior a la velocitat sense configurar res.

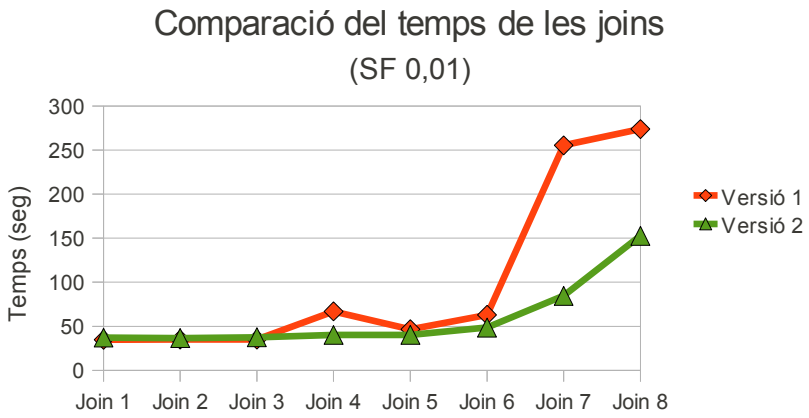
11.2.3 Desnormalitzador

El desnormalitzador s'encarrega de desnormalitzar les taules del TPC-H. He implementat el desnormalitzador de dues maneres diferents:

En la primera versió (explicada a l'apartat 7.3.1) feia la join entre A i B, sent A i B dues taules amb la taula A referenciant B. La A era l'entrada del MapReduce i s'agrupaven els valors per la clau de les columnes per fer la join a l'entrada del Reduce i llavors dins del Reduce s'accedia a la taula B per prendre la informació de la clau. Aquesta versió resultava, però, molt lenta.

En la segona versió (explicada a l'apartat 7.3) totes les taules per desnormalitzar, estan dins d'una sola taula d'HBase. Cada taula del TPC-H es troba en una família de columnes diferents. Llavors, s'agafen les dues famílies de columnes que es corresponen a les dues taules del TPC-H de què en volem fer la join.

A continuació, mostro una gràfica comparativa dels temps d'execució de les dues versions del desnormalitzador:



Gràfica 2: Comparació dels algorismes de desnormalització.

Podem veure per cadascuna de les 8 join entre taules (apartat 12.1) que el temps requerit per desnormalitzar els joc de proves del TPC-H amb primera versió és superior al de la versió 2. La reducció de temps que aconseguim en la versió 2 és de gairebé un 40% (38,6% en el cas de l'exemple) respecte la versió 1 amb un Scale Factor de 0,01.

12 TPC-H

El TPC-H consisteix en un conjunt de dades i consultes per a fer tests de rendiment de gestors de bases de dades. L'esquema de les dades utilitzat permet treballar amb grans volums de dades i pretén imitar el que es podria trobar en una empresa. Les consultes, per la seva banda, busquen avaluar situacions de test, sobre la base de dades, ideades per simular les consultes que s'executarien en una empresa. Aquestes consultes, tenen la finalitat d'obtenir informacions força complexes del negoci.

L'esquema de taules utilitzat pel TPC-H és el de la figura següent[13]:

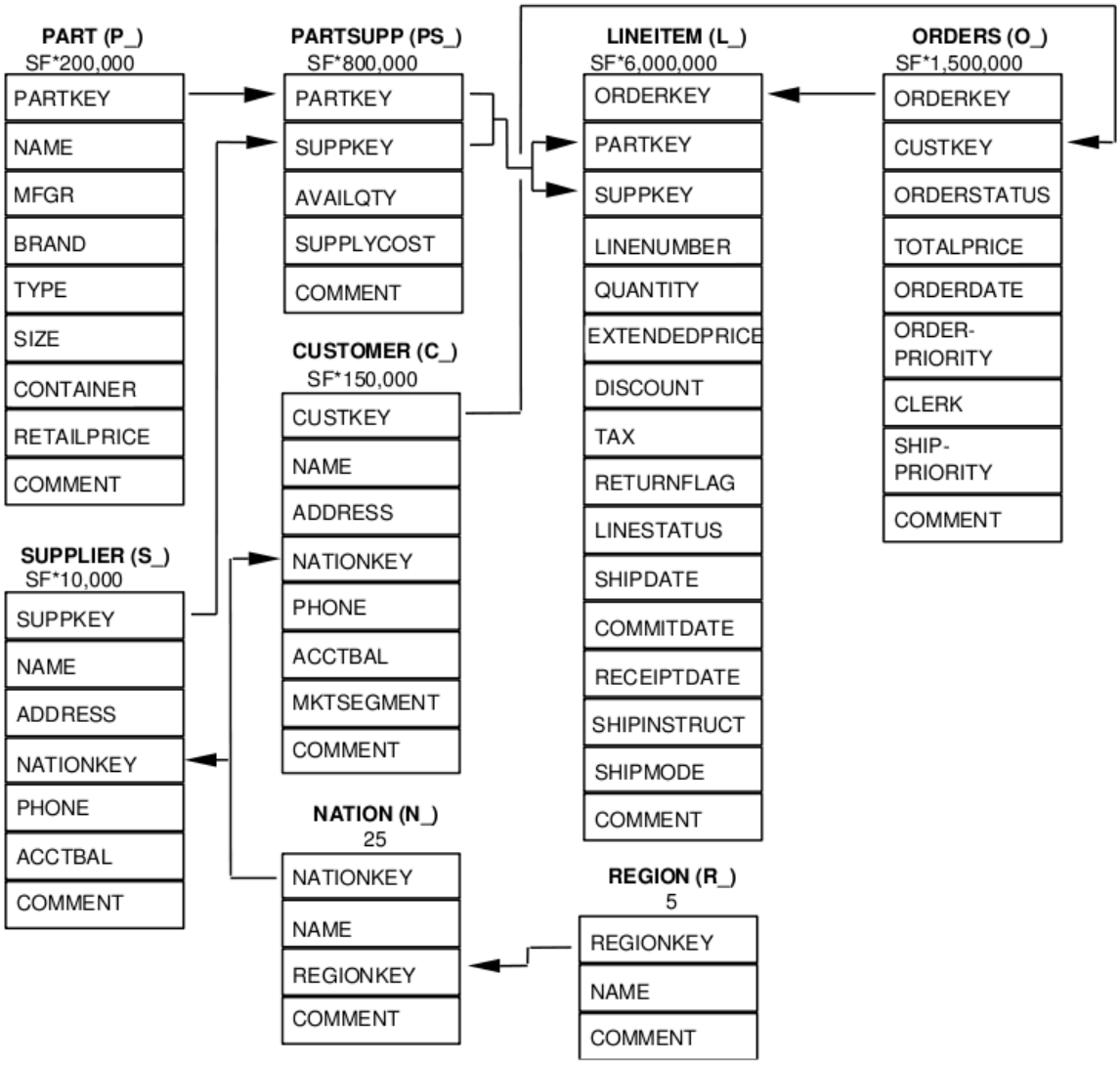


Figura 15: Esquema de taules normalitzat del TPC-H

13 Figura extreta de l'especificació del TPC-H[5].

TPC-H

La clau primària de cadascuna de les taules és l'atribut que es troba en la primer fila de cada taula en l'esquema. Amb les següents excepcions:

- La clau de la taula *partsupp* està formada pels valors de: *partkey* i *suppkey*.
- La clau de la taula *lineitem* està formada pels valors de: *orderkey* i *linenumber*.

Les dades creades amb generador que ofereix el consorci TPC, compleixen l'esquema anterior. L'usuari pot especificar la quantitat de dades que vol produir per a les proves. Ho fa especificant un valor anomenat Scale Factor (SF). A la figura anterior (figura 15) a sota del nom de cada taula (totes menys *nation* i r*egion*) hi posa *SF*número* i el resultat d'aquest producte serà el nombre de tuples que tindrà la taula per un SF determinat. Per exemple, la taula *partsupp* amb Scale Factor igual a 2 contindrà 1,600,000 entrades. Les taules *nation* i *region* tenen un nombre constant d'entrades.

Per aquest projecte, el conjunt de proves del TPC-H presenten el següent inconvenient: l'esquema de base de dades està normalitzat i els algorismes implementats (IFS, IRA, FSS) estan pensats per bases de dades desnormalitzades. Per tant, he hagut de desnormalitzar l'esquema de taules.

A continuació explico el procés de desnormalització que he dut a terme per a poder fer les proves dels algorismes i també quin procés he seguit per poder executar consultes similars a les del TPC-H amb el projecte desenvolupat:

12.1 Procés de desnormalització

El procés que he fet per obtenir l'esquema desnormalitzat del TPC-H a partir del normalitzat és el següent:

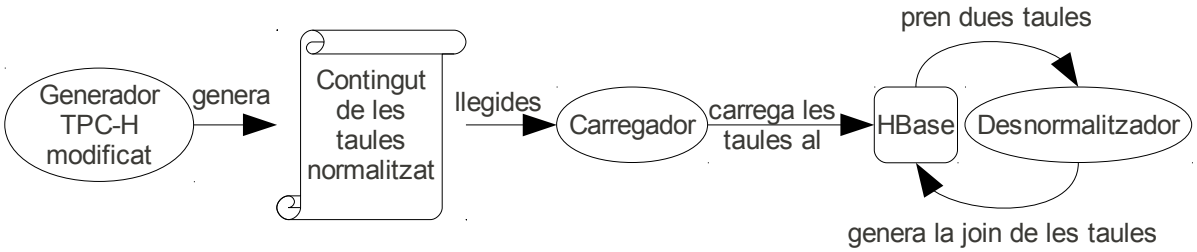


Figura 16: Procés de desnormalització realitzat

Tal com es pot veure a la figura 16 s'ha de fer servir la versió modificada del generador TPC-H per generar la informació de l'esquema de taules normalitzat. Aquest esquema de taules l'he entrat a l'HBase amb el carregador (o Loader). Un cop carregades les dades, es comença el procés de desnormalització de les taules inserides a l'HBase. Per fer la desnormalització del esquema, s'han de fer vuit joins següents:

	Taula Externa	**Taula Interna**	**Taula Resultant**
join 1	*nation*	*region*	*nat-reg*
join 2	*supplier*	*nat-reg*	*sup-nat*
join 3	*customer*	*nat-reg*	*cust-nat*
join 4	*partsupp*	*part*	*partsupp-part*
join 5	*partsupp-part*	*sup-nat*	*psupp-part-supp*
join 6	*orders*	*cust-nat*	*ord-cust*
join 7	*lineitem*	*ord-cust*	*line-ord*
join 8	*line-ord*	*psupp-part-supp*	*final*

Taula 6: Joins realitzades per a la desnormalització

El procés de desnormalització s'ha dut a terme fent servir l'algorisme de desnormalització explicat a l'apartat 7.3. Per a cadascuna de les joins de la taula 6 tenim que:

- El nom de la taula en la columna Taula Externa defineix el nom de la taula externa de l'algorisme de desnormalització.
- El nom de la taula en la columna Taula Interna defineix el nom de la taula interna de l'algorisme.
- El nom de la taula en la columna Taula Externa defineix el nom de la taula en què es desarà la join de les altres dues taules.

Un cop tenim l'esquema desnormalitzat s'han de canviar els identificadors de les tuples de tal manera que siguin consecutius. Es fa per les següents raons:

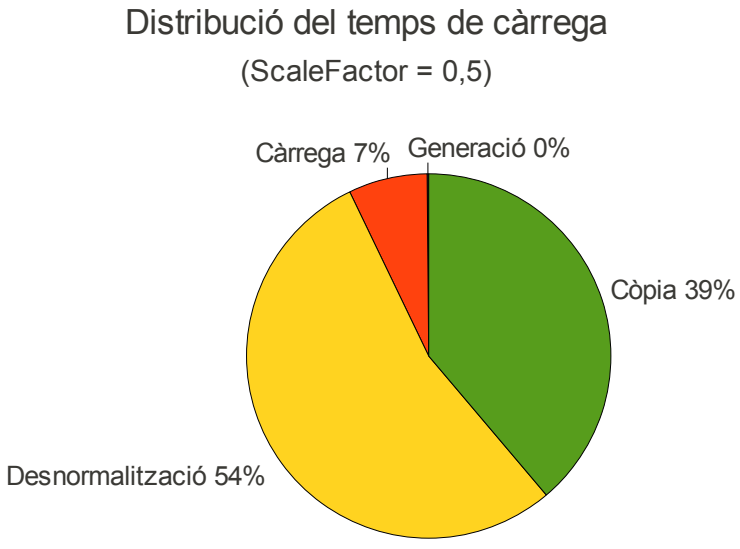
- Per tal de poder fer les agrupacions de les entrades a cercar en l'algorisme IRA i així, poder-lo executar més eficientment.
- En el cas de l'IFS ens permet limitar el recorregut de l'escaneig sobre la taula de fets ja que podem identificar quina és la primera i quina l'última tupla del rang a cercar.

El procés de canviar els identificadors, implica una còpia de la taula. El que he fet, ha estat copiar el contingut de la taula en una altra taula i numerar les entrades de forma consecutiva. Si es numerem les entrades sense més ni més hi ha el problema que no podem afirmar que dues tuples siguin consecutives lexicogràficament. Cal recordar, que en HBase les tuples s'ordenen lexicogràficament pel valor de la seva clau. Per exemple, suposem que tenim dues tuples amb identificadors *1* i *2*, no podem afirmar que siguin consecutives perquè no sabem si al mig hi ha una altra entrada, per exemple, una que tingui per identificador *11* que aniria després de l'*1* abans que el *2*. Per resoldre aquest problema els identificadors que s'assignen són de longitud fixa, 10 caràcters. És el número precedit de suficients zeros de manera que la seva longitud sigui 10. Per exemple l'identificador per la primera tupla seria:

000000001

12.1.1 Temps requerit

El temps necessitat per a la realització de tot el procés per deixar les proves apunt és elevat. Per exemple, per a preparar les proves amb un Scale Factor de 0,5 tarda més de 16 hores (16 hores 43 minuts, per més detalls veure l'apartat 8.1.5.2 del capítol de proves). El temps que s'utilitza per cada part del procés és el següent:



Gràfica 3: Distribució del temps de preparació de l'esquema desnormalitzat

Com podem veure a la gràfica, la major part del temps de preparació de les proves se'n va en el procés de desnormalització. Tampoc és menyspreable el temps que es tarda a realitzar la còpia de les dades per a canviar els identificadors de les tuples.

12.2 Consultes del TPC-H

El TPC-H consta de consultes per executar sobre les dades generades i mesurar-ne el rendiment. Aquestes consultes, però són massa complexes per executar-les amb el meu projecte. Contenen operacions d'agregació com la mitjana (només he implementat la suma), contenen comparacions entre els valors seleccionats en la consulta, cosa que tampoc he implementat... Per aquest motiu, com que les consultes del TPC-H no podien ser executades amb el meu projecte, vaig prendre diferents factors que podien influir en el rendiment de les consultes i els vaig mesurar en les consultes. Aquests factors van ser:

- Nombre de mesures seleccionades.
- Factor de selecció.
- Nombre d'atributs per a l'agrupació de valors.
- Nombre de clàusules sobre diferents dimensions en el predicat de la consulta.

Vaig calcular aquests factors per cadascuna de les 22 consultes que componen els tests del TPC-H i vaig obtenir els següents resultats:

	Mínim	**Màxim**	**Mediana**
Mesures seleccionades	1	9	3
Mida de l'agrupació	0	7	1
Nombre de clàusules	1	6	2
Factor de selecció	~ 0	~ 1	0,0089

Amb aquests resultats, a l'hora fer les proves es va optar per provar cadascun d'aquests factors individualment. Per fer-ho, vaig crear consultes que deixessin tres dels factors fixats al valor de la seva mediana i fer variar el quart factor entre el mínim i el màxim. Pels tres primers factors de la taula (el nombre de mesures seleccionades, la mida de l'agrupació i el nombre de clàusules) he provat tots els valors entre el mínim i el màxim. Per al factor de

selecció, he fet les proves en potències de 10, he provat des de 10^{-5} fins a 1, és a dir, he provat: 10^{-5}, 10^{-4}, ..., 1.

S'ha fet servir la mediana, enlloc de la mitjana, per trobar la tendència central de la mostra ja que és més adient per situacions en les quals les dades estan esbiaixades i hi contenen valors atípics com és el nostre cas.

Les consultes que he utilitzat per a mesurar la influència dels diferents factors sobre el rendiment, queden agrupades en les següents categories:

12.2.1 Factor de selecció

A continuació mostro les consultes que tenen per objectiu mesurar la influència del factor de selecció en el rendiment dels algorismes. Les he fet variant el factor de selecció entre el mínim i el màxim i deixant els altres valors fixats al valor de la mediana. Així, en les consultes es seleccionen dues mesures, tenen un atribut a l'agrupació i si és possible amb dos consten de dues clàusules en el predicat. Les consultes per aquesta categoria són les següents:

- `SELECT discount, extendedprice, quantity WHERE PartSupplierPartType = ECONOMY ANODIZED STEEL% :: Shipmode = MAIL% :: Discount = 0.01% :: SupplierNationRegionName = ASIA% GROUP BY shipdate_year`
    - Aquesta consulta té un factor de selecció de $1{,}66 \cdot 10^{-5}$. Amb un Scale Factor de 0,01 agafa una tupla. Aquesta consulta té quatre dimensions amb clàusules enlloc de dues perquè era impossible aconseguir un factor de selecció tant petit amb menys clàusules. Bé, l'única possibilitat era fixar els valors d'una de les claus del cub de dades, però llavors, al fer les proves amb factors de selecció més grans no hauria augmentat el nombre de tuples agafades, i per tant, no s'hauria mantingut el factor de selecció de la consulta.
- `SELECT discount, extendedprice, quantity WHERE PartSupplierPartType = ECONOMY ANODIZED STEEL% :: Shipmode = MAIL% :: Discount = 0.01% GROUP BY shipdate_year`
    - Aquesta consulta té un factor de selecció $9{,}97 \cdot 10^{-5}$ amb un Scale Factor de 0,01. Com en el cas anterior, s'ha hagut de fer amb més clàusules que les que indica la mediana. En aquest cas, però, han estat suficients tres clàusules sobre diferents dimensions:

- `SELECT discount, extendedprice, quantity WHERE PartSupplierPartType = ECONOMY ANODIZED STEEL% :: Shipmode = MAIL% GROUP BY shipdate_year`
    - Aquesta té un factor de selecció de $1{,}08 \cdot 10^{-3}$ amb un Scale Factor de 0,01. Aquest factor de selecció ja l'he pogut aconseguir amb dues clàusules tal com indica la mediana i d'aquí en endavant també.
- `SELECT discount, extendedprice, quantity WHERE Partbrand = Brand#11% :: Shipinstruct = NONE% GROUP BY shipdate_year`
    - Aquesta té un factor de selecció de $1{,}00 \cdot 10^{-2}$ per a un Scale Factor de 0,01.
- `SELECT discount, extendedprice, quantity WHERE Discount = 0.01% :: Partbrand = All% GROUP BY shipdate_year`
    - Aquesta té un factor de selecció de 0,0918 amb un Scale Factor de 0,01. En aquesta consulta, la clàusula sobre la dimensió *Partbrand* és All%, això vol dir que volem agafar tots els valors possibles i que, per tant, realment no està restringint la selecció. Aquesta, però, serveix per forçar l'accés a l'índex per aquest atribut en els algorismes IRA i IFS, encara que la clàusula no faci servei.
- `SELECT discount, extendedprice, quantity WHERE Discount = All :: Partbrand = All% GROUP BY shipdate_year`
    - Aquesta consulta té un factor de selecció de 1 amb un Scale Factor de 0,01. En aquest cas, com que havíem de seleccionar tota la taula i havíem de posar dues clàusules alhora vaig posar les dues clàusules amb el valor All.

12.2.2 Mesures seleccionades

Les consultes que he fet per estudiar la influència del nombre de mesures seleccionades són de l'estil següent:

```
SELECT @mesures@ WHERE Partbrand = Brand#11% :: Shipinstruct = NONE%
GROUP BY shipdate_year
```

El factor de selecció d'aquesta consulta és 0,01, molt proper al valor de la mediana del factor de selecció de les consultes del TPH-C. També veiem que té dues mesures seleccionades i una columna en a l'agrupació. Així, aquests tres factors queden fixats al valor calculat de la mediana.

Per preparar les consultes per a mesurar la influència del nombre de mesures, he fet 9 consultes diferents, substituint *@mesures@* pels *n* primers atributs de la següent llista:

```
discount, extendedprice, tax, quantity, orders-totalprice, partsup-
part-retailprice, partsup-supplycost, shipdate_year, shipdate_month
```

12.2.3 Mida de l'agrupació

El procés per a crear les consultes per mesurar la influència de la mida de l'agrupació sobre el rendiment de les consultes és similar al que he utilitzat en el cas anterior. He partit de la consulta base:

```
SELECT discount, extendedprice, quantity WHERE Partbrand =
Brand#11% :: Shipinstruct = NONE%
```

Aquesta consulta té un factor de selecció proper a 0,01. El nombre de mesures seleccionades és tres i tenim dues clàusules. Així, per aquests tres paràmetres els seus valors són els de les medianes calculades a partir de les consultes del TPC-H. Com podem veure, la consulta base no consta de cap agrupació.

Per a provar la influència de la mida de l'agrupació, provarem les seves possibles mides entre 0 i 7. Aquestes són les mides mínimes i màximes de les agrupacions de les consultes del TPC-H respectivament. Les consultes les he creat afegint els *n* primers elements a l'agrupació de la consulta:

```
linestatus, partsup-part-brand, partsup-part-size, partsup-part-type,
returningflag, shipinstruct, shipmode
```

12.2.4 Nombre de clàusules

Per a preparar les consultes que mesuren la influència del nombre dimensions que apareixen a les clàusules vaig tenir el problema que el mínim de clàusules és una, però amb una sola clàusula no vaig poder aconseguir el factor de selecció de 0,01. Així, que el que vaig fer va ser agafar una clàusula en la que m'apropés força a aquest valor i llavors vaig escollir una segona que acabés d'ajustar el resultat. Vaig fer la següent consulta:

```
SELECT discount, extendedprice, quantity WHERE PartSupplierPartType =
SMALL POLISHED NICKEL% :: PartSupplierPartType = LARGE ANODIZED BRASS%
GROUP BY shipdate_year
```

Com podem veure el nombre d'atributs seleccionat i la mida la mida de l'agrupació complixen amb els respectius valors de la mediana. Per a la resta de consultes per mesurar la influència del nombre de dimensions a les clàusules vaig de servir següent consulta:

```
SELECT discount, extendedprice, quantity WHERE Shipinstruct = NONE% ::
Partbrand = Brand#11% GROUP BY shipdate_year
```

A la qual vaig afegint les *n* primeres clàusules de la llista de clàusules següents per tal de crear totes les consultes.

```
SupplierNationRegionName = All%, Discount = All%, Returnflag = All%,
PartsupPartContainer = All%, Shipmode = All%, Linestatus = All%
```

Veiem que són clàusules que donen més feina a l'algorisme però que no influeixen en el resultat de la consulta i mantenen el factor de selecció de 0,01.

12.3 Índexs

Per poder resoldre totes les consultes preparades a l'apartat anterior, necessitem un índex per a cadascuna de les dimensions que apareixen sota alguna clàusula de les consultes. Les dimensions necessàries són les següents:

- **SupplierNationRegionName**: compost per la columna *partsup-supplier-nation-region-name*.
- **Discount**: compost per la columna *discount*.
- **Returnflag**: compost per la columna *returningflag*.
- **PartsupPartContainer**: compost per la columna *partsup-part-container*.
- **Shipmode**: compost per la columna *shipmode*.
- **Shipinstruct**: compost per la columna *shipinstruct*.
- **Partbrand**: compost per la columna *partsup-part-brand*.
- **Linestatus**: compost per la columna *linestatus*.
- **PartSupplierPartType**: compost per la columna *partsup-part-type*.

Veiem que les diferents dimensions només consten d'un atribut. S'ha fet així perquè, les columnes amb restriccions, en les consultes del TPC-H tenen restriccions en diferents atributs però entre elles no tenen relacions jeràrquiques en el seu significat o en la restricció.

13 Proves

En aquest capítol detallo algunes de les proves que he fet al llarg del projecte. Només incloc proves que es poden realitzar amb la versió definitiva del projecte. Al llarg de tot el desenvolupament he anat fent d'altres proves, algunes de les quals per a funcionalitats que han canviat o que s'han implementat d'altres maneres. Les proves que mostro primer tenen per objectiu provar a grans trets el correcte funcionament del projecte. L'objectiu de les que mostro més endavant és avaluar el rendiment dels algorismes:

13.1 Proves de funcionament

L'objectiu d'aquestes proves ha estat verificar el correcte funcionament dels diferents algorismes implementats per aquest projecte. Aquestes proves les he fet després del desenvolupament del projecte, però no cal oblidar que a mesura que anava programant els diferents programes anava fent proves per verificar el seu correcte funcionament.

Les proves que mostro al llarg d'aquest apartat les he fet utilitzant la taula de l'annex I.

13.1.1 Creació de dimensions

Recordem que per a la creació de dimensions només hem d'especificar el nom de la dimensió a crear i les columnes de la taula de fets que la componen. Les proves són les següents:

13.1.1.1 Dimensions simples

En aquesta prova, comprovarem que les dimensions que tenen un sol atribut es creen correctament. La dimensió que creem es diu *Gender* i conté un únic atribut *gender*. La comanda és la següent:

```
CREATE DIMENSION Gender ATTRIBUTES gender
```

Si mirem a la taula dels índexs, utilitzant índexs d'un sol nivell, veiem que han aparegut les entrades:

Clau de la fila	Keys
Gender%female%	3
Gender%male%	1,10,2,4,5,8,9
Gender%unknown%	6,7

Mirant a l'annex I podem veure fàcilment que aquests resultats són correctes. Que efectivament l'única dona que apareix a la taula és la de l'entrada 3. Les entrades 6 i 7 són les que tenen un sexe desconegut i les altres són les entrades que corresponen a homes.

Ara bé, si executem la mateixa instrucció per a la creació de la dimensió amb índexs multinivell s'afegiran els valors següents:

Clau de la fila	Keys
Gender%All	1,2,3,4,5,6,7,8,9,10
Gender%female%	3
Gender%male%	1,10,2,4,5,8,9
Gender%unknown%	6,7

Veiem que en aquest cas, si la dimensió només consta d'un atribut, només s'afegeix una entrada més a l'índex multinivell respecte els índexs d'un sol nivell. Aquesta nova entrada (All) conté tot el conjunt d'identificadors.

Així, hem comprovat que les dimensions amb un sol atribut es creen bé utilitzant índexs d'un sol nivell o multinivell.

13.1.1.2 Dimensions compostes

Ara provarem que les dimensions amb múltiples valors també es creen bé. Crearem la dimensió *DateOfBirth* amb els atributs de la taula *year*, *month* i *day*. La comanda utilitzada és la següent:

```
CREATE DIMENSION DateOfBirth ATTRIBUTES year month day
```

Les entrades que s'afegeixen a la taula d'índexs usant índexs d'un sol nivell són les següents:

Clau de la fila	Keys
DateOfBirth%1895%11%11%	5
DateOfBirth%1895%12%12%	4
DateOfBirth%1948%12%24%	2
DateOfBirth%1948%3%11%	1
DateOfBirth%1960%1%4%	8,9
DateOfBirth%1970%4%2%	10
DateOfBirth%1987%4%5%	3
DateOfBirth%2000%1%1%	6,7

Com en el cas anterior, podem comprovar que els resultats obtinguts són correctes mirant la taula de l'annex I. Ens hem de fixar, també que els valors de la dimensió s'ordenen en el mateix ordre en què els hem especificat a la llista d'atributs de la comanda.

D'altra banda, si executéssim la mateixa comanda amb índexs multinivell, a més a més de les entrades d'un sol nivell, s'afegirien les entrades següents:

Clau de la fila	Keys
DateOfBirth%1895%11%All	5
DateOfBirth%1895%12%All	4
DateOfBirth%1895%All	4,5
DateOfBirth%1948%12%All	2
DateOfBirth%1948%3%All	1
DateOfBirth%1948%All	1,2
DateOfBirth%1960%1%All	8,9
DateOfBirth%1960%All	8,9
DateOfBirth%1970%4%All	10
DateOfBirth%1970%All	10
DateOfBirth%1987%4%All	3
DateOfBirth%1987%All	3
DateOfBirth%2000%1%All	6,7
DateOfBirth%2000%All	6,7
DateOfBirth%All	1,2,3,4,5,6,7,8,9,10

Com podem veure, aquestes són les entrades que es corresponen als diferents nivells d'agregació de la dimensió *DateOfBirth*. Tenim el nivell que inclou totes les dades (p.e.

DateOfBirth%All), el nivell que classifica per any (p.e. DateOfBirth%2000%All), el que classifica per any i mes (DateOfBirth%1987%4%All) i, finalment, el que ja teníem en l'índex d'un sol nivell que classifica per any, mes i dia.

També podem comprovar que aquests resultats són correctes contrastant-los amb la taula de l'annex I.

Així, amb aquestes proves, hem comprovat el correcte funcionament dels índexs multinivell. Ens queda comprovar que es pot definir correctament l'ordre d'agregació dels diferents atributs de la dimensió. Ho veiem en el següent apartat:

13.1.1.3 Ordre en les dimensions compostes

Ara comprovarem que l'ordre dels atributs en la instrucció és el que determina que l'ordre dels atributs a l'hora de crear les dimensions. Ho farem amb una comanda semblant a la de la prova anterior. Crearem la dimensió RDateOfBirth amb els atributs de la taula *day*, *month* i *year*. La comanda utilitzada és la següent:

```
CREATE DIMENSION RDateOfBirth ATTRIBUTES day month year
```

Les entrades que s'afegeixen a la taules de dimensions usant índexs d'un sol nivell són les següents:

Clau de la fila	Keys
RDateOfBirth%1%1%2000%	6,7
RDateOfBirth%11%11%1895%	5
RDateOfBirth%11%3%1948%	1
RDateOfBirth%12%12%1895%	4
RDateOfBirth%2%4%1970%	10
RDateOfBirth%24%12%1948%	2
RDateOfBirth%4%1%1960%	8,9
RDateOfBirth%5%4%1987%	3

Mirant a la taula de l'annex I podem comprovar que aquests resultats també són correctes. També comprovem que l'ordre dels atributs de la dimensió és el mateix que en la comanda, en aquest cas: *day*, *month*, *year*.

D'aquesta manera també he provat el correcte funcionament amb índexs multinivell però n'ometo els resultats perquè no aporten nova informació.

Així, ja hem comprovat el correcte funcionament de la creació d'índexs ja siguin d'un sol nivell o multinivell.

13.1.2 Consultes

En aquesta part comento algunes de les proves que he fet per a comprovar que la implementació dels algorismes funciona correctament. Les proves les he realitzat són sobre la taula de l'annex I i utilitzen les dimensions creades en les proves de l'apartat anterior.

Totes les proves realitzades en aquest aquest apartat les he fet amb totes les configuracions possibles: amb i sense compressió, amb índexs multinivell i d'un sol nivell per a cadascun dels tres algorismes (IRA, IFS i FSS). Totes les possibles combinacions han donat els mateixos resultats que són correctes.

13.1.2.1 Consultes bàsiques

L'objectiu d'aquesta prova és comprovar que la selecció i l'agregació de valors funciona correctament. La instrucció que farem servir és la següent:

```
SELECT salary WHERE Gender = unknown%
```

Aquesta instrucció ens retornarà la suma del sou de tots els empleats de la taula que tenen un sexe desconegut. El contingut de la taula de resultats fent servir com a taula de fets la taula de l'annex I és el següent:

Clau de la fila	Salary
All	30

Mirant a la taula de l'annex podem veure que la suma dels sous dels empleats amb sexe indefinit és:

$$15 + 15 = 30$$

Per tant, aquest resultat està bé. Ara si hem fet servir l'algorisme IRA podem mirar el contingut de la taula de claus intermèdies. D'aquesta taula només ens interessen els identificadors de les diferents entrades. Veiem que el seu contingut és:

Clau de la fila	Key
6	true
7	true

Veiem que concorden amb els identificadors de les files amb els dels empleats que tenen un sexe indefinit.

En cas d'utilitzar l'algorisme IFS, si examinem el contingut de l'arxiu intermedi podrem comprovar que conté els identificadors 6 i 7.

13.1.2.2 Múltiples clàusules

Amb aquestes proves comprovarem el correcte funcionament dels algorismes quan tinguem consultes amb múltiples clàusules. Cal recordar que les condicions sobre la mateixa dimensió es tracten com la unió dels conjunts i que entre les diferents dimensions es tracten com la intersecció dels conjunts.

Una de les proves que he fet ha consistit a comprovar el correcte funcionament de la següent comanda sobre la taula de l'annex I:

```
SELECT expenses, salary WHERE Gender = female% :: Gender = unknown% ::
                             DateOfBirth = 1960%All%
```

Veiem que no obtenim resultats perquè no hi ha cap dona o ésser de sexe desconegut a la taula que hagi nascut a l'any 1960. D'altra banda, si executem la consulta:

```
SELECT expenses, salary WHERE DateOfBirth = 1960%All% :: Gender = male%
```

Obtenim els resultats:

Clau de la fila	Expenses	Salary
All	500,0	3000,0

Mirant a la taula de l'annex I podem comprovar que els dos homes que hi surten van néixer l'any 1960 i que les seves despeses i els seus sous sumen els valors indicats.

13.1.2.3 Agrupació de valors

Algunes de les proves que he fet per comprovar el correcte funcionament de l'agrupació de valors són les que explico en aquest apartat. Seguint treballant amb la taula de l'annex, si executem la consulta següent:

```
SELECT expenses, salary WHERE Gender = All% GROUP BY gender
```

Obtenim els resultats:

Clau de la fila	Expenses	Salary
female_	10,0	3000,0
male_	13500,0	18000,0
unknown_	0,0	30,0

Veiem, que en aquest cas el resultat consta de tres files. Cadascuna de les files hi ha una categoria diferent. Aquesta categoria ve determinada pel valor de les columnes respecte les que fem l'agrupació. Podem comprovar que aquests resultats també són correctes contrastant-los amb la taula de l'annex.

Ara, comentaré una consulta més complicada. Aquesta obtindrà la suma dels sous dels homes i les dones que treballen agrupades segons la secció a la que pertanyen. La consulta és la següent:

```
SELECT salary WHERE Gender = male% :: Gender = female% GROUP BY section
```

Els resultats obtinguts amb aquesta consulta són els següents:

Clau de la fila	Salary
Development_	6000,0
Direction_	7500,0
Human-Resources_	7500,0

Veiem que hi ha una fila per cada departament i que les sumes dels sous de cada departament es corresponen amb els de la taula de l'annex I. S'ha de tenir en compte que no he seleccionat les files corresponents als éssers amb sexe desconegut.

També podem fer agrupacions respecte múltiples columnes. Per exemple, si executem la consulta anterior agrupant per secció i tipus de feina tal com fem en la consulta següent:

```
SELECT salary WHERE Gender = male% :: Gender = female% GROUP BY job, secti-
                                                                    on
```

Obtenim:

Clau de la fila	Salary
Administrator_Direction_	3500,0
Designer_Development_	3000,0
Director_Direction_	4000,0
Master_Human-Resources_	6000,0
Programmer_Development_	3000,0
Public-Relations_Direction_	0,0
Senator_Human-Resources_	1500,0

Aquesta taula conté el resultat de la consulta. Veiem que hi ha una fila per cada combinació de les columnes de secció i tipus de feina que apareix a la taula de l'annex I. La primera fila és correspon a la suma dels sous dels homes i les dones que treballen en la secció de direcció i que realitzen la feina d'administració.

13.1.3 Desnormalitzador

En aquest apartat comento algunes de les proves que he fet amb el desnormalitzador. Les proves les he realitzat amb les dades del TPC-H un cop carregades en l'HBase amb l'esquema normalitzat.

Vaig realitzar la join entre la taula de nacions i la taula de regions i va funcionar correctament (veure capítol 12). Va generar 25 files cadascuna de les quals es corresponia amb una de les nacions originals i tenien, a més de les columnes originals, les columnes corresponents al nom de la regió i al comentari de la regió. El valor de totes les columnes era el que pertocava. A l'esquema de la taula del TPC-H (capítol 12) es pot comprovar que aquestes són les columnes que han d'aparèixer en el resultat de la join.

Després, vaig fer una prova similar amb la join entre la taula dels proveïdors i la taula obtinguda de la join anterior. La vaig comprovar tota, bé, vaig comprovar que en la taula resultant (100 files per a un Scale Factor de 0,01) per a cada nació de cada proveïdor s'havien afegit els valors que pertocaven de les regions i de les nacions.

Per assegurar-me que tot el procés de les joins estigués correctament definit, a més, vaig prendre una fila qualsevol de la taula resultant i vaig comprovar que els seus valors fossin correctes. Vaig agafar l'entrada que tenia per clau *final#10916|3* (entrada que es correspon a la fila que té *orderkey* 10916 i *linenumber* 3 de la taula *lineitem*) i vaig comprovar que tots els

valors que apareixien a la seva fila es corresponien a les referències que tenia. Tots els valors van resultar ser correctes.

13.2 Proves de rendiment

En aquest apartat comentaré les proves de rendiment que he dut a terme. Per fer les proves de rendiment dels algorismes de creació de d'índexs, de desnormalització i d'execució de consultes amb dades i consultes realistes he emprat el test TPC-H.

Les proves que he fet han estat amb dues configuracions d'entorn diferents:

- *Entorn no distribuït*: per a aquestes proves he fet servir el meu portàtil. Té un processador Intel Core 2 Duo 2,2GHz, 4GB de memòria SDRAM DDR2 a 667MHz i un disc dur SATA de 250GB a 5.400rpm.

  Aquest ordinador és el servidor de tots els serveis i a la vegada n'és el client.

- *Entorn distribuït*: per a aquestes proves he fet servir el portàtil anterior i un ordinador de sobre taula amb les següents especificacions: té per processador un Pentium 4 a 2,4GHz, 512MB de memòria RIMM a 533MHz i un disc dur IDE de 110GB a 7.200rpm.

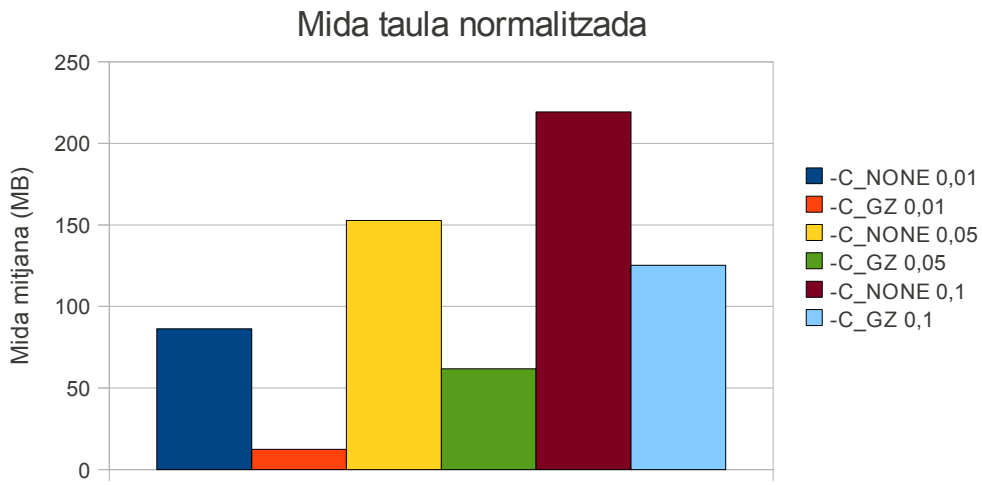
  El portàtil feia de servidor de Hadoop HDFS, del JobTracker i de l'HBase i al mateix temps feia d'esclau per a tots ells. L'ordinador de sobre taula feia d'esclau de l'HDFS i del JobTracker. Tots els processos s'iniciaven en el portàtil i, per tant, s'hi executaven tots els processos que no han estat paral·lelitzats.

A continuació, explico en diferents criteris els rendiments obtinguts dels projecte durant les proves i les conclusions que n'he extret:

13.2.1 Espai requerit de disc

En aquestes proves he mesurat l'espai necessitat per a la taula de fets i per a la taula dels índexs en funció de si s'utilitza o no compressió de les dades dins de les taules. L'espai requerit en un entorn distribuït és el mateix que en un entorn no distribuït perquè el volum de dades és el mateix, i per tant aquest criteri no ens serveix per a fer distincions.

Per començar estudiem l'espai requerit per la taula de fets abans de ser desnormalitzada:



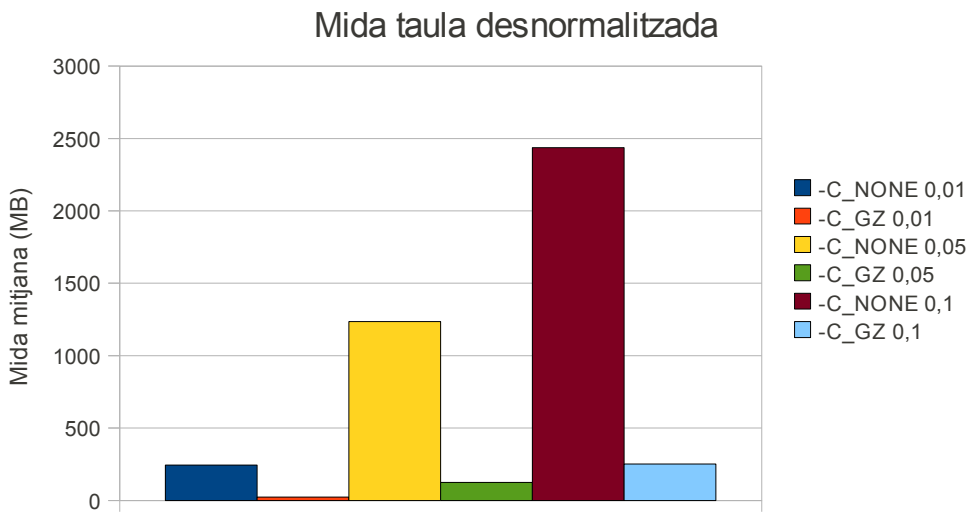
Gràfica 4 Mida de la taula normalitzada

Primer de tot, cal explicar la notació de les llegendes de les gràfiques. El nom de les categories de les gràfiques ve donada per la configuració de la prova. Si el nom d'una categoria conté el mot -C_NONE vol dir que no s'ha usat compressió, en canvi si conté -C_GZ vol dir que s'ha utilitzat compressió del tipus GNU Zip. El nombre que conté el nom de la categoria és el Scale Factor que s'ha usat. Per exemple -C_GZ 0,05 es correspon als resultats obtinguts amb un Scale Factor de 0,05 usant compressió.

A la gràfica no hi ha els resultats pel Scale Factor de 0,5. La mida de la taula normalitzada amb aquest Scale Factor amb compressió és de 287,35 MB. Aquest resultat, però, no el mostro a la gràfica perquè no tinc la mida de la taula sense compressió. És així, perquè, com veurem més endavant, el temps requerit per a la seva generació és extremadament gran i no hi ha gaire diferència de temps entre processar dades comprimides i descomprimides i ocupen molt menys comprimides.

A la gràfica 4 hi podem veure que amb l'esquema de taula normalitzat i sense comprimir les taules ocupen de mitjana 3,73 vegades més que les comprimides. Tot i així en aquesta gràfica es pot apreciar una tendència a anar-se reduint aquesta proporció.

Ara bé, un cop desnormalitzada podem veure com l'espai requerit augmenta significativament, sobretot en el cas en què no s'utilitza compressió:



Gràfica 5 Mida de la taula desnormalitzada

La mida de les taules sense compressió augmenta de mitjana 7,34 vegades amb una tendència tendència creixent a mesura que augmenta el Scale Factor. En canvi, si utilitzem taules amb compressió la mida de l'esquema augmenta en un factor mig de 2,6 amb tendència a mantenir-se.

També veiem que la compressió obtinguda a les taules desnormalitzades és més gran que en les normalitzades. Amb els resultats obtinguts, veiem que les taules desnormalitzades sense compressió ocupen 10,01 vegades més que les que utilitzen compressió amb tendència a mantenir-se. Aquest fet s'explica perquè al desnormalitzar la taula s'introdueixen repeticions de dades i, per tant, s'en facilita la compressió.

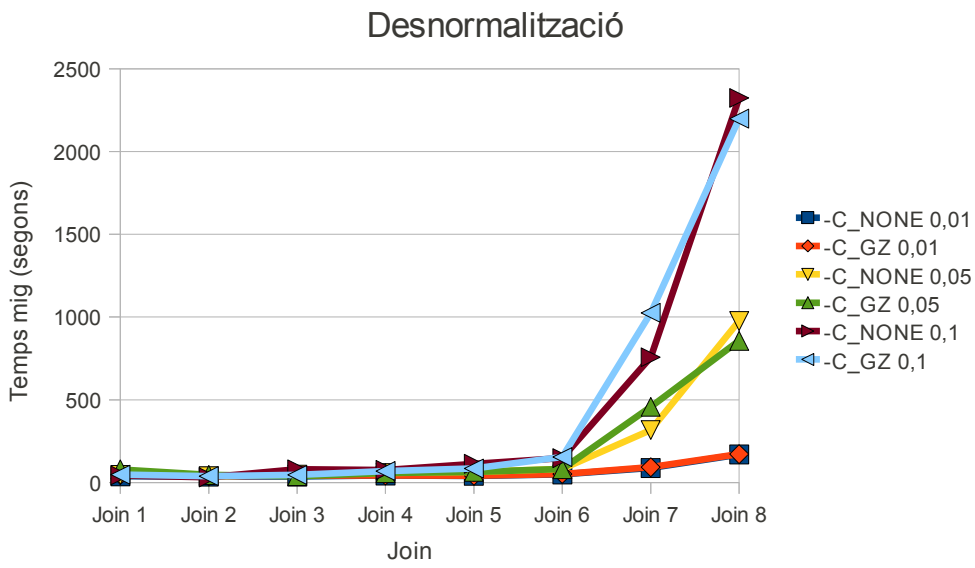
La mida de la taula desnormalitzada amb un Scale Factor de 0,5 i utilitzant compressió no apareix a la gràfica, però la seva mida és de 1,26 GB.

13.2.2 Desnormalitzador

Hem vist la necessitat d'utilitzar un desnormalitzador per preparar les proves de rendiment. També hem vist a la gràfica 3, de la pàgina 120, que el rendiment del desnormalitzador és crucial per a poder realitzar les proves ja que representa un 54% del temps total de la càrrega.

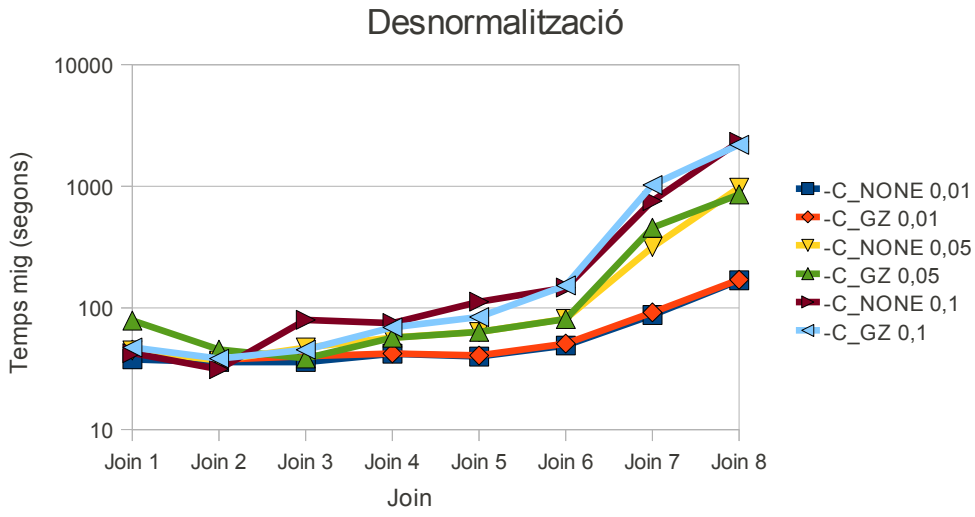
Els resultats de les proves que presento en aquest apartat han estat realitzats en la configuració d'un entorn no distribuït.

Primer de tot, mostro els temps de realització de cada join pels Scale Factor des del 0,01 al 0,1 utilitzant o no compressió:



Gràfica 6: Temps de desnormalització

Degut a les diferències que hi ha entre els diferents valors no podem apreciar gaire les variacions que hi ha entre els valors més petits. Si mostrem els mateixos resultats amb l'eix del temps en escala logarítmica veiem:



Gràfica 7 Temps de desnormalització (2ª gràfica)

En aquesta segona gràfica podem apreciar-hi millor les diferències entre els temps de les diferents joins, però és més difícil veure-hi la diferència en valor absolut entre els temps.

Comparant primer els resultats obtinguts amb els Scale Factor des del 0,01 al 0,1:

Entre les dues gràfiques, podem veure que el cost es comença a elevar a partir de la sisena join. També veiem que l'augment del temps és proporcional al Scale Factor que estem utilitzant: com més gran és el Scale Factor més gran és el pendent. També veiem que no influeix gaire negativament en el temps requerit l'ús de la compressió a les taules.

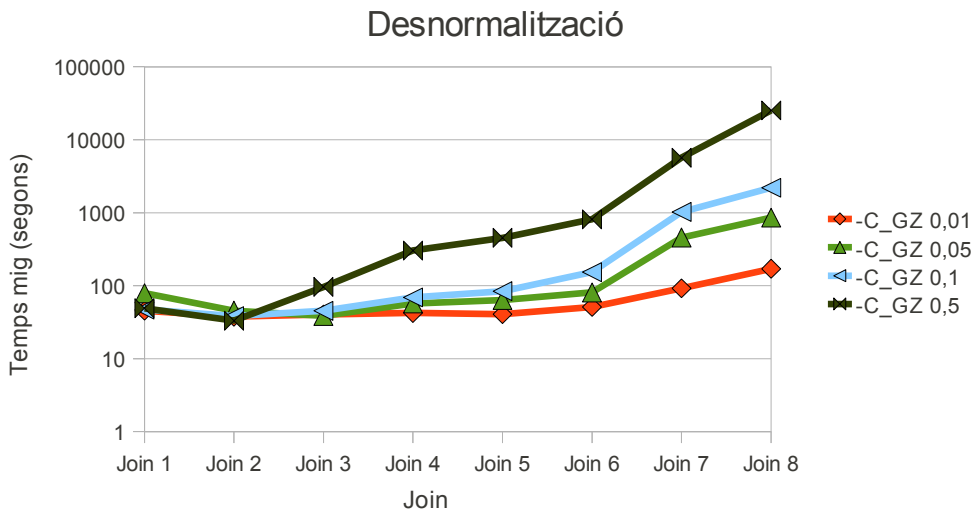
El temps de realització de les joins amb el Scale Factor de 0,1 és d'una hora i un minut aproximadament.

Com ja podem veure, els beneficis que ens aporta utilitzar compressió són els següents:

- Es requereix molt menys espai de disc, fins a 10 vegades menys per a una taula desnormalitzada.
- El temps de processar les dades comprimides és lleugerament superior al temps requerit per processar les dades sense compressió.

Per aquest motiu, les proves que he fet amb volums de dades més grans només les he fet amb compressió. Perquè l'espai requerit és molt inferior i el temps és molt semblant.

Vegem ara, els resultats que s'obtenen amb un Scale Factor de 0,5 (el més gran que he provat) amb compressió. Són els següents:



Gràfica 8 Temps de desnormalització (només compressió)

En aquesta gràfica només mostro el resultats obtinguts emprant compressió a les taules. Ara, si considerem els resultats obtinguts amb el Scale Factor de 0,5 veiem:

- Els seus temps es distancien de les altres configuracions a partir de la tercera join. Aquest fet s'explica perquè la primera join és entre les taules Nation i Region que són de mida fixa i, per tant, el temps ha de ser el mateix cosa que podem comprovar. Les dues següents joins són entre la taula Nat-reg amb Supplier i Nat-reg amb Costumer que creixen relativament poc segons el Scale Factor. Les joins que queden són amb taules més grans i que creixen més ràpidament i, per tant, disparen el cost abans.
- El temps d'execució de les joins amb un Scale Factor de 0,5 és de 9 hores i 4 minuts. Molt superior al temps obtingut d'una hora amb el Scale Factor de 0,1.
- També veiem a la gràfica 8 que el pendent a partir de la sisena join és encara més pronunciat que pels Scale Factor més petits.

Amb les proves que he fet, es pot comprovar, com ja sabíem, que la desnormalització és un procés amb un cost exponencial respecte el Scale Factor. També hem vist que el temps requerit és molt semblant, fins i tot una mica millor, en el cas de la compressió i que l'espai requerit és molt inferior.

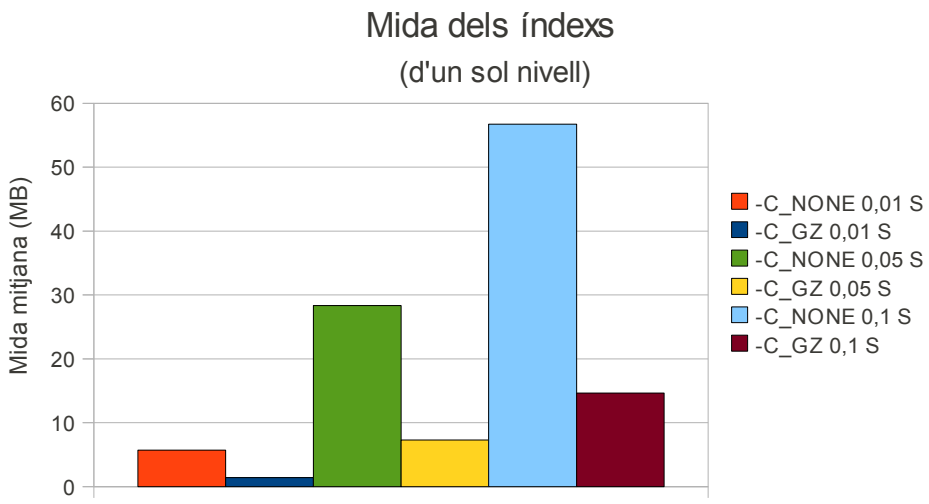
13.2.3 Comparació d'índexs d'un nivell amb índexs multinivell

Ara estudiarem l'espai requerit pels índexs. Com ja hem dit, tenim dues configuracions diferents d'índexs, els índexs d'un sol nivell (apartat 7.1.1) o bé multinivell (apartat 7.1.2). Primer veurem una comparació de les mides requerides pels índexs per a diferents configuracions, volums de dades i, després, els temps de creació d'aquests índexs.

13.2.3.1 Mida de la taula dels índexs

En aquest apartat comentaré la mida de la taula dels índexs segons les diferents configuracions utilitzades.

Vegem primer la gràfica que ens mostra la mida de la taula dels índexs d'un sol nivell:



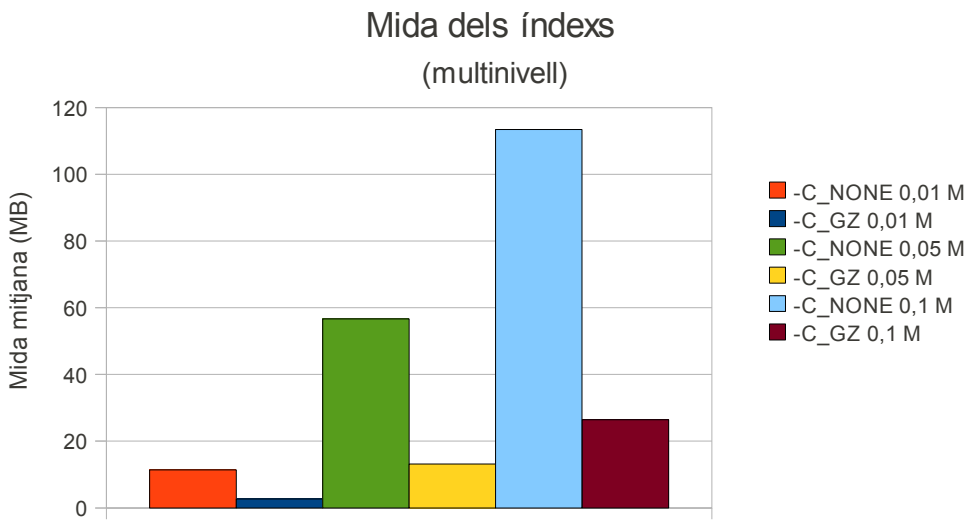
Gràfica 9: Mida dels índexs d'un sol nivell

A la gràfica hi podem veure la mida de la taula dels índexs un cop s'hi han carregat els índexs utilitzant diferents configuracions. Els resultats mostrats són dels índexs d'un sol nivell, des d'un Scale Factor de 0,01 a 0,1 amb compressió i sense compressió. Les categories marcades amb un S són aquelles que han fet servir índexs d'un sol nivell i les categories marcades amb una M són les que han fet servir múltiples nivells. Els resultats obtinguts amb un Scale Factor de 0,5 no els poso a la gràfica perquè només tinc els resultats emprant compressió, però la mida mitjana de l'índex d'un sol nivell amb compressió és de 76,13MB.

A la gràfica podem apreciar que quan no s'utilitza compressió els índexs d'un sol nivell ocupen de mitjana 3,92 vegades més que en els casos que utilitzem compressió. Aquesta raó de compressió és constant (té molt poques variacions) entre les proves que van del 0,01 al 0,1. Si estimem la mida dels índexs d'un sol nivell sense compressió amb un Scale Factor de 0,5 ens surt de la mida de: 76,13MB * 3,92 = 298,4MB.

Podem veure també que la mida de la taula dels índexs es multiplica aproximadament per cinc quan passes del Scale Factor de 0,01 al 0,05 tant si utilitzes compressió si no n'utilitzes. En el cas de passar del Scale Factor de 0,05 a 0,1 veiem que la mida de la taula es duplica. Pel cas de 0,1 a 0,5 utilitzant compressió es torna a multiplicar per 5 aproximadament perquè el nombre d'entrades és proporcional al Scale Factor. D'aquesta mantera hem comprovat que la mida dels índexs d'un sol nivell és lineal respecte el volum de dades amb què treballem.

A continuació, veurem i comentaré una gràfica similar a l'anterior però pels índexs multinivell:



Gràfica 10: Mida dels índexs multinivell

En aquesta gràfica veiem la mida de la taula dels índexs multinivell per a diferents volums de dades. Els volums de dades van des del Scale Factor de 0,01 al Scale Factor de 0,1 utilitzant o no compressió. Com en el cas anterior, els resultats obtinguts amb un Scale Factor de 0,5 utilitzant compressió no són a la gràfica però la seva mida és de 132,88MB.

A la gràfica 10 veiem que, de mitjana, les taules dels índexs multinivell sense compressió ocupen 4,30 vegades més que les que utilitzen compressió. Podem veure que aquesta raó de compressió és superior a l'obtinguda en els índexs d'un sol nivell de 3,92 (gràfica 9). Aquesta diferència es pot explicar per la redundància de dades. En el cas dels índexs multinivell afegim redundància en les dades i la redundància és un dels factors que més influeixen en la compressió de dades.

Si com en el cas anterior volguéssim estimar la mida de la taula dels índexs multinivell amb un Scale Factor de 0,5 sense usar compressió, obtindríem la mida de: 132,88MB * 4,30 = 571 MB. Aquesta estimació es pot considerar vàlida perquè la raó de compressió dels índexs multinivell per als Scale Factor de 0,01 a 0,1 és gairebé constant (amb molt poca desviació).

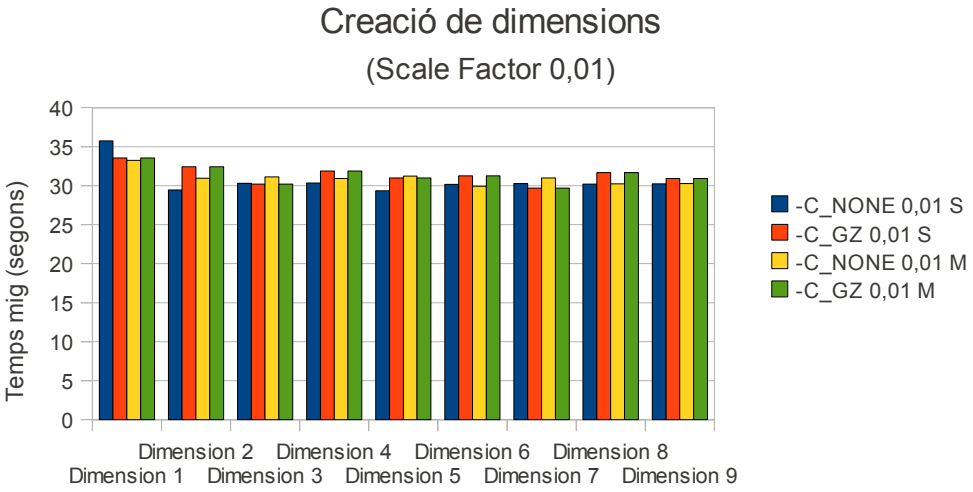
Com en el cas anterior, la mida de les taules és proporcional al volum de dades emprat. La mida de l'índex amb un Scale Factor de 0,1 ocupa el doble que el que obtenim amb un Scale Factor de 0,05 i deu vegades més que el d'un Scale Factor de 0,01, tant en els casos en què utilitzem compressió com si no.

D'altra banda, si comparem les gràfiques 9 i 10, podem apreciar que els índexs multinivell ocupen de mitjana el doble (2,00) que els d'un sol nivell en cas de no utilitzar compressió. Si utilitzem compressió, però, aquesta proporció mitjana baixa fins els 1,80. En ambdós casos la desviació obtinguda és molt baixa.

13.2.3.2 Temps de creació dels índexs

En aquest apartat veurem els temps requerits per a la creació dels índexs per a les diferents configuracions provades.

Primer vegem una gràfica que ens mostra el temps de creació de cadascuna de les dimensions definides a l'apartat 12.3 amb un Scale Factor de 0,01:

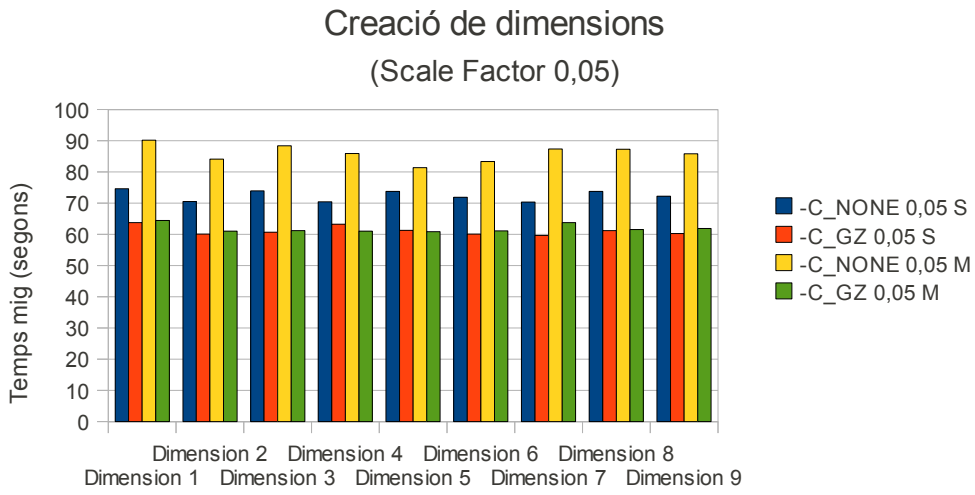


Gràfica 11: Temps de creació de les dimensions (Scale Factor de 0,01)

En aquesta gràfica veiem temps requerit per a crear cadascun dels índexs per les diferents configuracions.

Podem veure que per aquest volum de dades el temps de creació de tots els índexs està al voltant dels 30 segons. També veiem que en la majoria de casos el temps de la creació dels índexs sense compressió és lleugerament inferior als casos que utilitzen compressió.

Vegem ara la gràfica que ens mostra el temps de creació dels índexs multinivell amb un Scale Factor de 0,05:



Gràfica 12: Temps de creació de les dimensions (Scale Factor de 0,05)

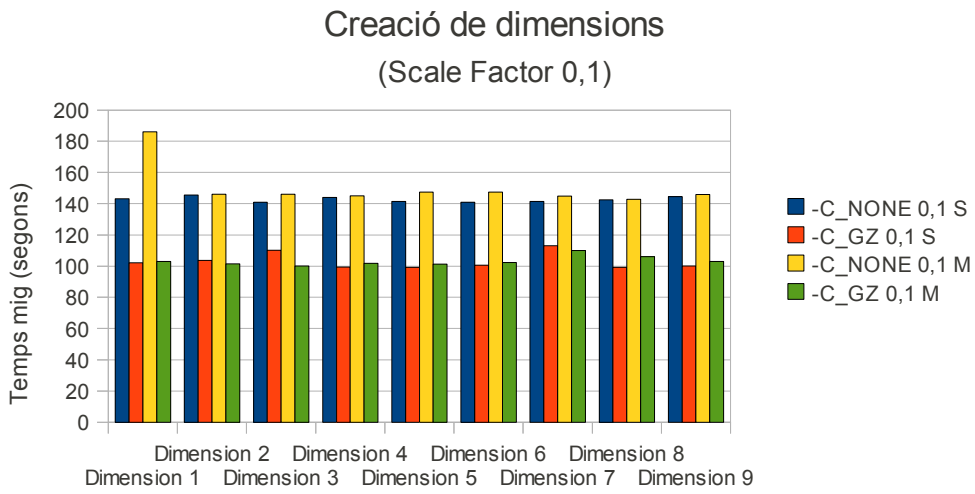
En aquesta gràfica veiem el temps de creació dels índexs multinivell és superior al temps requerit pels índexs multinivell s'utilitzi o no compressió. En cas de no utilitzar compressió el temps de creació dels índex multinivell és de mitjana 1,19 vegades superior (una diferència mitjana de 13 segons) al requerit pels índexs d'un sol nivell. No obstant, si s'utilitza compressió els índexs d'un sol nivell requereixen 1,01 vegades el temps (una diferència mitjana de 0,71 segons) dels índexs multinivell.

D'altra banda, es pot apreciar encara més clarament que el temps de creació dels índexs és més elevat si no s'utilitza compressió que en el cas d'utilitzar-ne. Pels casos dels índexs d'un sol nivell, quan no s'ha usat compressió requereixen de mitjana 1,18 vegades el temps requerit utilitzant compressió, representa una diferència mitjana de 11 segons. Aquesta diferència augmenta fins a una raó de 1,39 (una diferència mitjana de 24 segons) en cas de tractar-se d'índexs multinivell.

Així, ja podem veure que amb un Scale Factor de 0,01 la creació dels índexs resulta més ràpida usant compressió que sense utilitzar-ne. Aquest fet pot ser degut a que s'ha de llegir menys informació del disc perquè la compressió redueix el nombre de blocs que es necessiten del disc i al mateix temps l'algorisme de compressió utilitzat és ràpid.

Veiem també, que el temps mig de creació d'un índex amb un Scale Factor de 0,05 és de 70 segons, una mica més del doble del temps requerit per crear un índex amb un Scale Factor de 0,01.

Vegem ara, una gràfica similar a l'anterior però feta amb les dades obtingudes amb un Scale Factor de 0,1:



Gràfica 13: Temps de creació de les dimensions (Scale Factor de 0,1)

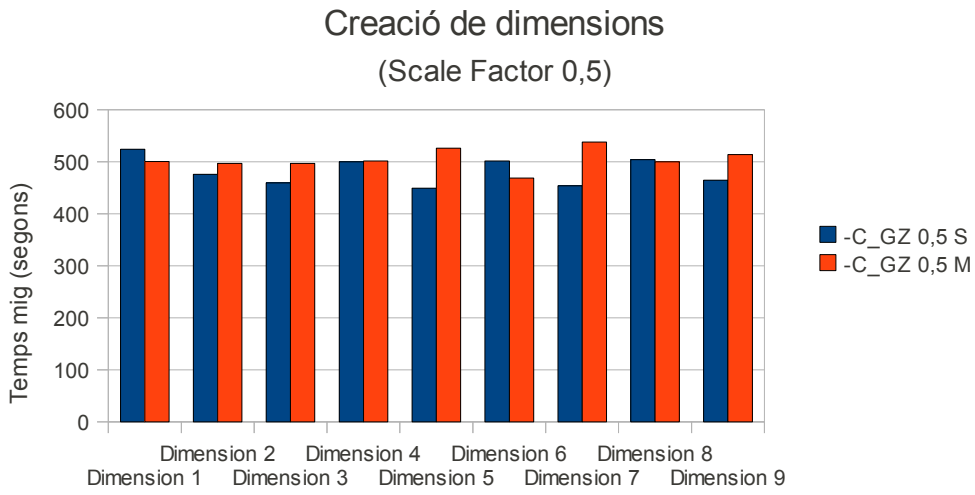
Podem veure a la gràfica 13 que els resultats obtinguts amb un Scale Factor de 0,1 tenen una forma similar als obtinguts amb un Scale Factor de 0,05.

Veiem que el guany de temps utilitzant compressió ha augmentat fins a un factor mig de 1,45 (una millora mitjana de 47 segons per consulta) pels índexs multinivell i pels índexs d'un sol nivell un 1,38 (una millora mitjana de 40 segons).

D'altra banda, el guany obtingut durant la creació d'índexs d'un sol nivell respecte els índexs multinivell ha disminuït lleugerament.

Veiem també que el temps mig de creació d'un índex amb un Scale Factor de 0,1 és de 124 segons (2 minuts i 4 segons). Aquest temps és gairebé el doble de l'obtingut amb un Scale Factor del 0,05 i aproximadament 10 vegades més que del Scale Factor de 0,01.

Vegem ara la gràfica obtinguda amb els temps de creació dels índexs utilitzant compressió i un Scale Factor de 0,5:



Gràfica 14: Temps de creació de les dimensions (Scale Factor de 0,5)

En aquesta gràfica veiem que els temps de creació dels índexs d'un sol nivell i multinivell són molt similars usant compressió fins i tot amb Scale Factor del 0,5. No obstant això, de mitjana la creació d'un índex d'un sol nivell s'executa 23 segons més ràpid que la d'un índex multinivell.

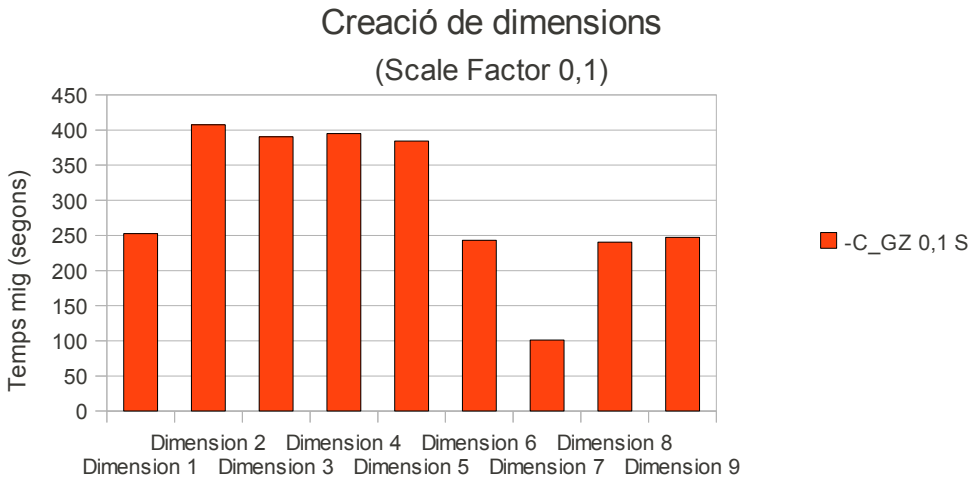
Podem veure també que els temps per a la creació dels índexs augmenta fins al voltant dels 500 segons (8 minuts i 20 segons). Concretament la mitjana pels índexs d'un sol nivell tenen una mitja de 504,21 segons i el multinivell, per la seva banda, 480,46 segons. Aquests temps són aproximadament 5 vegades els temps obtinguts amb un Scale Factor de 0,1; aquest resultats són bons perquè tenim 5 vegades més de dades. Això vol dir que el temps de creació dels índexs és directament proporcional a la quantitat de dades que hem d'indexar.

13.2.3.3 Creació dels índexs amb dos servidors

Les proves que he fet usant l'entorn distribuït han estat utilitzant els Scale Factors de 0,1 i 0,5 amb compressió i amb índexs d'un sol nivell. S'ha fet així, perquè com ja hem pogut comprovar el temps de creació d'un índexs usant compressió és inferior i el temps de creació dels índexs d'un sol nivell també és més baix que amb múltiples nivells.

Com és d'esperar, la mida dels índexs usant dos servidors és la mateixa que usant-ne només un. Així, en aquest apartat ens centrarem en el temps requerit per les execucions.

Vegem primer la gràfica amb els resultats obtinguts amb un Scale Factor de 0,1:

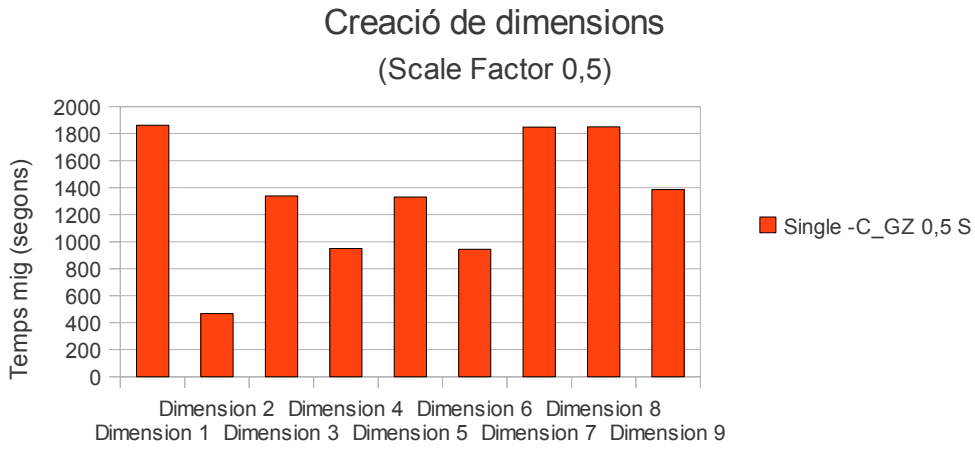


Gràfica 15: Temps de creació de les dimensions, entorn distribuït (Scale Factor de 0,1)

A la gràfica 15 podem veure que el temps de creació dels índexs és superior al obtingut en un entorn no distribuït amb el mateix Scale Factor (gràfica 13). L'únic cas en qual el temps és similar és per a la dimensió número 7. Els altres resulten més lents. Aquest fet, s'explica pel motiu següent:

Aquests resultats són la mitjana de dues execucions en l'entorn distribuït, però com que el volum de dades no és molt elevat, el procés de creació d'un índex l'executa tot un ordinador. Examinant els logs es pot veure que la dimensió 7 és l'única que s'ha creat en les dues execucions en el portàtil. La resta de dimensions s'han creat com a mínim un cop en l'ordinador de sobre taula. Aquest fet, implica que el rendiment sigui més baix perquè el procés s'executa a l'ordinador de sobre taula que és més lent i no perquè s'executi de forma distribuïda.

Passa exactament el mateix usant un Scale Factor de 0,5. Vegem-ho a la gràfica que hi ha a continuació:



Gràfica 16: Temps de creació de les dimensions, entorn distribuït (Scale Factor de 0,5)

Veiem que la situació és exactament la mateixa però, que en aquest cas, és la dimensió 2 la que té els temps similars als obtinguts en un entorn no distribuït (gràfica 14).

D'aquestes gràfiques és complicat extreure'n més conclusions fiables per culpa del soroll introduït per la diferència de potència dels equips.

13.2.3.4 Conclusions

Amb les dades obtingudes i comentades en aquest apartat se'n poden extreure algunes conclusions:

- Els índexs multinivell ocupen aproximadament el doble que els índexs d'un sol nivell. Molt possiblement perquè els índexs multinivell creats tenen dos nivells.
- La compressió de les dades pot reduir fins a gairebé una quarta part la mida de la taula dels índexs.
- El temps de creació dels índexs d'un sol nivell i de múltiples nivells són similars tot i que ha estat una mica més baix el dels índexs d'un sol nivell.
- L'ús de compressió a l'hora de crear els índexs afavoreix al rendiment del procés.

13.2.4 Comparació dels algorismes de consulta

En aquest apartat veurem la influència dels diferents factors de les consultes estudiants en el capítol del TPC-H (apartat 12.2) sobre el rendiment dels diferents algorismes. Per a cadascun dels quatre factor identificats, a més, he provat diferents Scale Factors.

He fet proves amb Scale Factors que van des de 0,01 fins a 0,5. Per a tots ells he mesurat el rendiment usant índexs d'un sol nivell (marcats a la llegenda de les gràfiques amb una S) i índexs multinivell (marcats amb una M). Pels Scale Factors que van de 0,01 a 0,1 he provat el rendiment amb compressió i sense compressió. Les proves amb el Scale Factor de 0,5 les he fet només amb compressió. S'ha fet així, perquè tal com podrem comprovar a cada tipus de prova els rendiments obtinguts de les diferents configuracions són molt semblants i és molt costós preparar les proves amb un Scale Factor de 0,5 (veure l'apartat 13.2.2), per tant, només he fet les proves del Scale Factor de 0,5 amb compressió.

Per a cada tipus de prova que he fet mostro una gràfica en què hi ha totes les configuracions provades pel Scale Factor de 0,01 per comprovar que els resultats obtinguts són similar. Després mostro només els resultats obtinguts amb índexs d'un sol nivell utilitzant compressió que és la combinació que de mitjana ha obtingut millor temps però que no disten gaire dels altres valors. D'aquesta manera, mostrant els resultats d'una sola configuració, es poden estudiar millor els valors obtinguts.

A més, he executat les diferents proves a l'entorn distribuït i n'he mesurat el rendiment. En mostro els resultats que n'he obtingut però, com podrem comprovar, no aporten més informació rellevant.

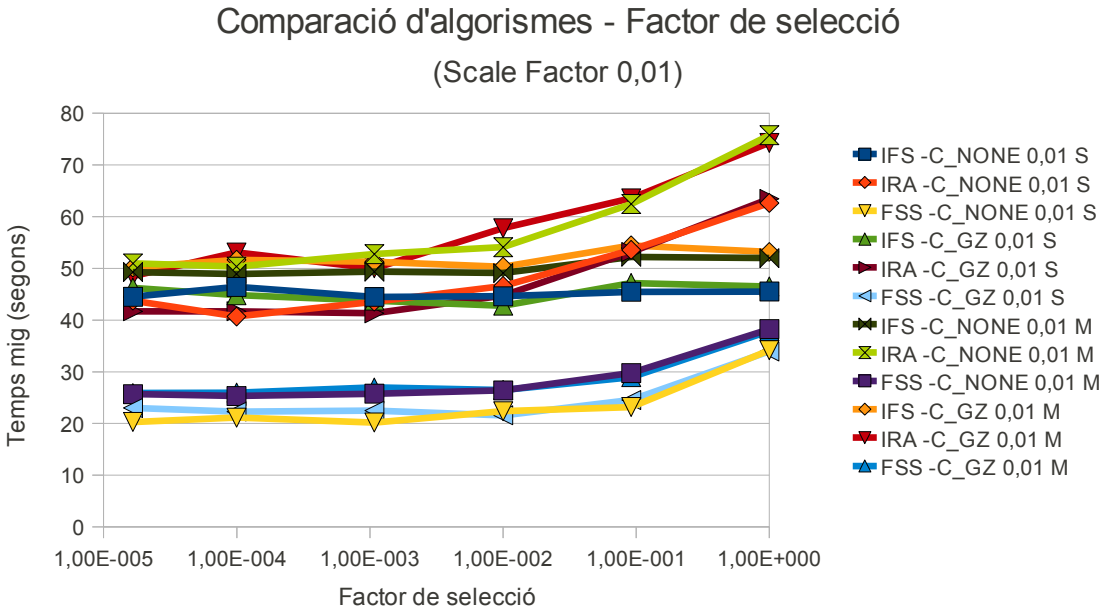
Vegem, a continuació, els resultats de cadascun dels algorismes segons les diferents característiques de les consultes i configuracions:

13.2.4.1 Factor de selecció

En aquest apartat estudiarem el resultat de les consultes explicades a l'apartat 12.2.1 que pretenen mesurar la influència del factor de selecció en el rendiment dels diferents algorismes.

Scale Factor 0.01

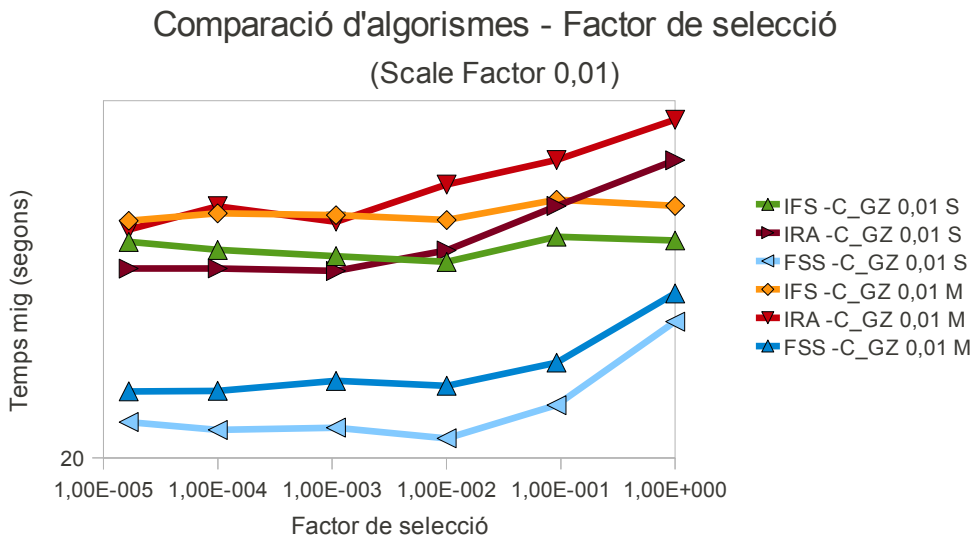
Començarem estudiant els resultats obtinguts amb un Scale Factor de 0,01. Examinant la següent gràfica:



Gràfica 17: Influència del factor de selecció (Scale Factor de 0,01)

Podem veure que l'algorisme FSS obté un rendiment molt superior als altres dos amb un temps el doble de ràpid. És així, perquè tal com veurem més endavant el temps mesurat en aquestes proves és el temps de posta en marxa dels MapReduce i la influència del temps de processament de les dades sobre la gràfica final és menyspreable. Així, com que els algorismes IFS i IRA requereixen de dos MapReduce resulta més lents que el FSS que només en necessita un i per això tarden el doble.

A la gràfica anterior podem veure-hi que els rendiments obtinguts utilitzant o no compressió són molt similars. De mitjana, resulten una mica més ràpids amb compressió que sense. Passa el mateix amb tots els Scale Factors provats, des de 0,01 fins a 0,5. Tal com he explicat, a partir d'aquest punt mostraré només els resultats obtinguts emprant compressió a les taules. Així, examinem ara la gràfica següent en la que mostro només els valors obtinguts amb compressió i un Scale Factor de 0,01:



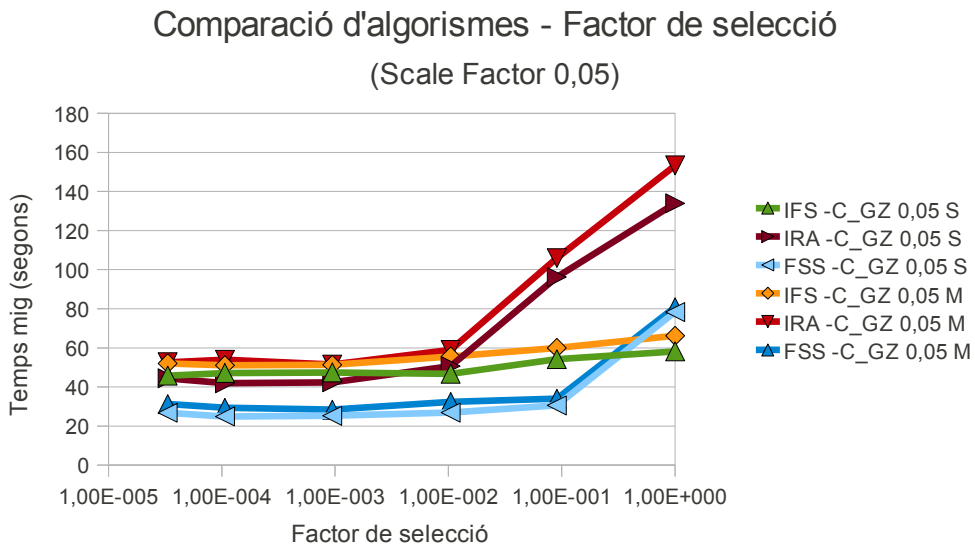
Gràfica 18: Influència del factor de selecció amb compressió (Scale Factor de 0,01)

La gràfica 18 mostra els resultats obtinguts usant compressió a les taules. He utilitzat escala logarítmica en l'eix vertical de la gràfica per tal d'augmentar la separació visual entre els valors més petits.

Podem veure-hi que, per tots els algorismes, els temps obtinguts han estat pitjors utilitzant índexs de múltiples nivells. També veiem més clarament la diferència de rendiment que hi ha entre el FSS i els algorismes IRA i IFS. Aquests dos últims algorismes presenten un comportament similar amb factors de selecció menors a 10^{-2}. El cost de l'algorisme IRA comença a créixer a partir del factor de selecció 10^{-2}. Aquest fet és previsible ja que l'IRA és l'algorisme que fa els accessos aleatoris. Cada accés implica una petició i, per tant, més temps a l'hora de resoldre el conjunt d'entrades a cercar.

Scale Factor 0.05

Ara examinarem els resultats obtinguts amb un Scale Factor de 0,05. Comencem amb la gràfica següent:



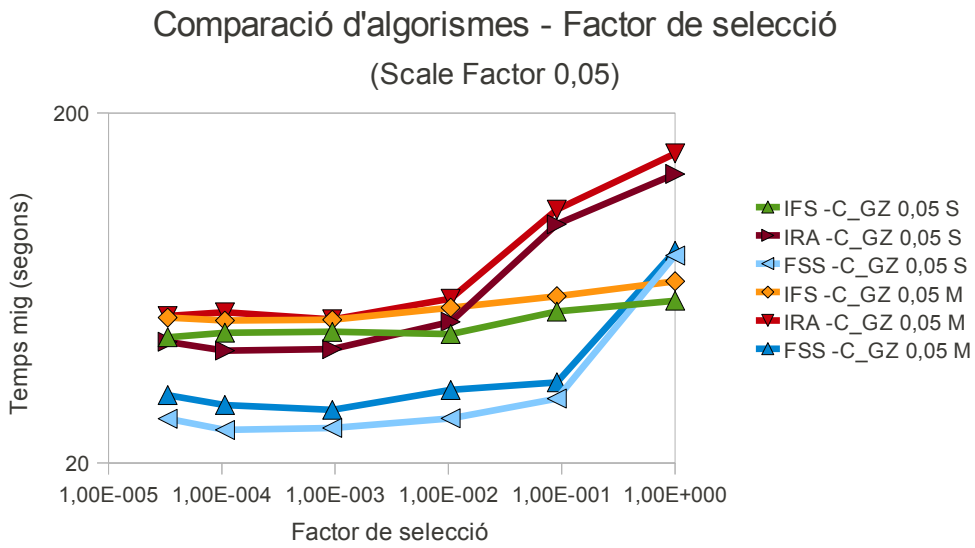
Gràfica 19: Influència del factor de selecció (Scale Factor de 0,05)

En aquesta gràfica veiem el rendiment de les consultes per mesurar la influència del factor de selecció utilitzant un Scale Factor de 0,05 i provant els índexs d'un sol nivell i multinivell usant compressió. Examinant la gràfica podem destacar dues influències clares del factor de selecció:

- Els factors de selecció grans influeixen molt negativament en el rendiment de l'IRA. Veiem que amb un factor de selecció de 1 el temps es multiplica per 2,5 respecte el temps requerit amb un factor de selecció de 10^{-2}.
- L'algorisme FSS també es veu influenciat negativament per factor de selecció 1. Veiem que respecte el factor de selecció de 0,1 el temps es multiplica per 2,7. Aquest increment situa el seu cost per sobre del cost de l'algorisme IFS.

Per la seva banda l'algorisme IFS no sembla veure's gaire perjudicat per factors de selecció alts.

Com que a la gràfica anterior és molt complicat analitzar la situació pels factors de selecció petit, he fet la gràfica següent utilitzant escala logarítmica a l'eix vertical:



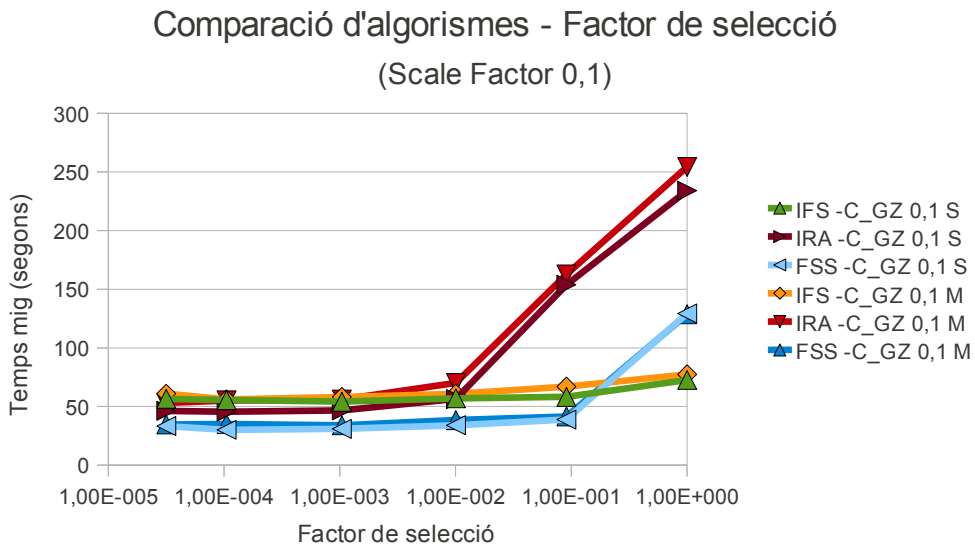
Gràfica 20: Influència del factor de selecció amb compressió(Scale Factor de 0,05)

A la gràfica 20 es pot apreciar clarament que el rendiment és inferior en els casos en què es treballa amb índexs multinivell. Aquesta diferència és més o menys gran en els diferents algorismes però sempre hi és present. Un fet que resulta sorprenent és que fins i tot l'algorisme FSS que no fa servir els índexs, podem veure que obté millors resultats quan els índexs creats són d'un sol nivell.

Tot i que no mostri els resultats a les gràfiques examinant els resultats numèrics es veu que els casos en què s'ha utilitzat compressió s'obtenen millors temps.

Scale Factor 0.1

Ara veurem l'efecte del factor de selecció en les consultes utilitzant un Scale Factor de 0,1. Comencem amb la gràfica que mostra les diferents configuracions provades:



Gràfica 21: Influència del factor de selecció (Scale Factor de 0,1)

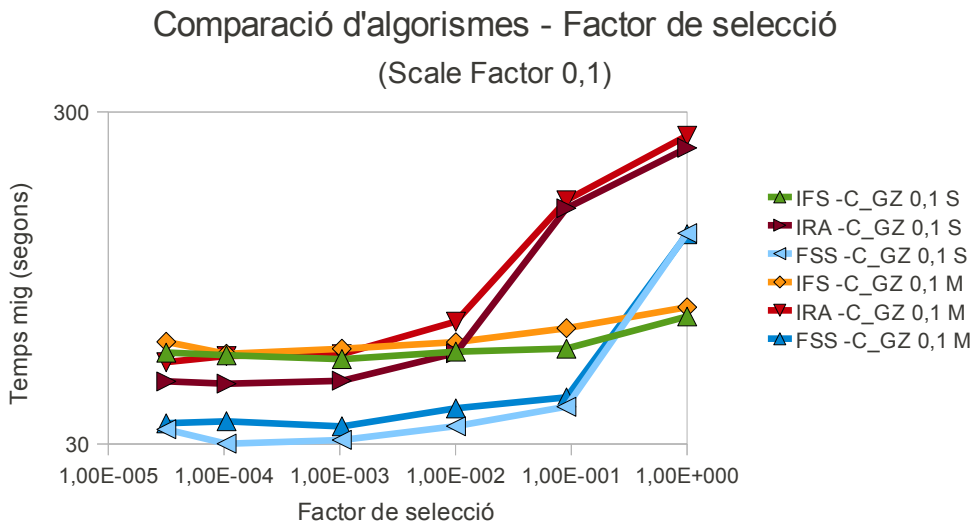
A la gràfica 21 podem apreciar-hi les tendències, observades amb un Scale Factor de 0,05, en els factors de selecció elevats. El cost de l'algorisme es comença a incrementar a partir del factor de selecció de 10^{-2}. El temps d'execució d'una consulta amb un factor de selecció de 1 és 5 vegades superior al temps requerit per les consultes amb un factor de selecció de 10^{-2}. Aquest és increment és el doble que el registrat amb el Scale Factor de 0,05.

Veiem també, com en el cas anterior, que el cost temporal del FSS incrementa a partir del factor de selecció de 0,1. La diferència que hi ha entre el factor de selecció de 1 i de 0,1 en el pitjor de casos és de 4,86 vegades superior.

L'algorisme IFS segueix amb la tònica de mantenir el seu temps estable pels factors de selecció més elevats i veiem que segueix essent el més ràpid quan el factor de selecció de la consulta és 1.

El temps mig de les diferents consultes amb un factor de selecció inferior a 10^{-2} segueix estant al voltant dels 50 segons. Tal com veurem més endavant el FSS segueix sent l'algorisme més ràpid per aquests volums dades, però la distància amb els altres algorismes es va reduint.

Com en els casos anteriors, la gràfica següent mostra els mateixos resultats en una gràfica on s'utilitza escala logarítmica a l'eix vertical per facilitar la comparació dels temps pels factors de selecció més petits:



Gràfica 22: Influència del factor de selecció amb compressió(Scale Factor de 0,1)

A la gràfica 22 veiem que els temps obtinguts amb els índexs multinivell són superiors als dels índexs d'un sol nivell.

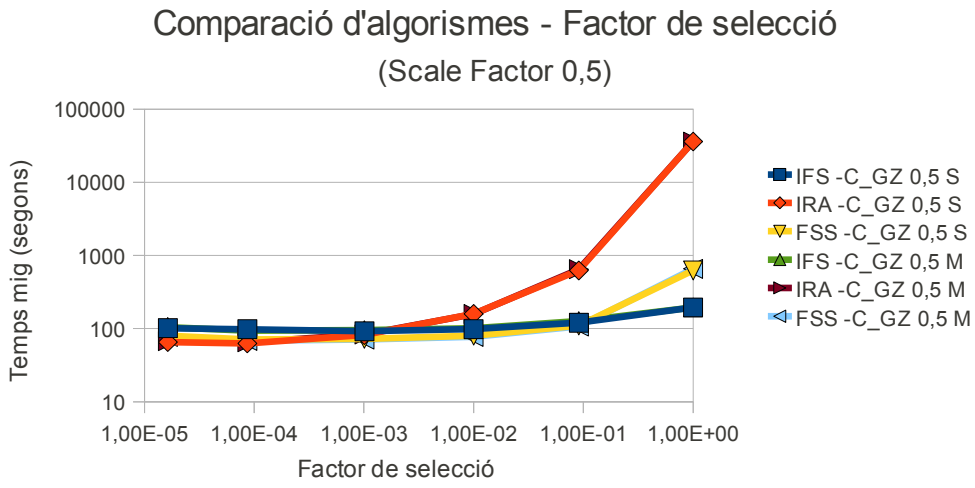
Podem apreciar com el temps d'execució de l'algorisme IRA resulta millor que el de l'algorisme IFS pel factor de selecció més baix. En cas d'usar compressió, tant amb índexs d'un sol nivell com multinivell resulten més ràpids.

Veiem també que els temps segueixen sent millors en els casos en què s'utilitzen els índexs d'un sol nivell, fins i tot quan l'algorisme és el FSS.

Scale Factor 0.5

Ara veurem la influència que té el factor de selecció en el rendiment de les consultes amb el volum més gran que he provat de dades, és a dir, amb un Scale Factor de 0,5. Les proves que he realitzat són utilitzant compressió.

Vegem els resultats obtinguts a la gràfica següent:



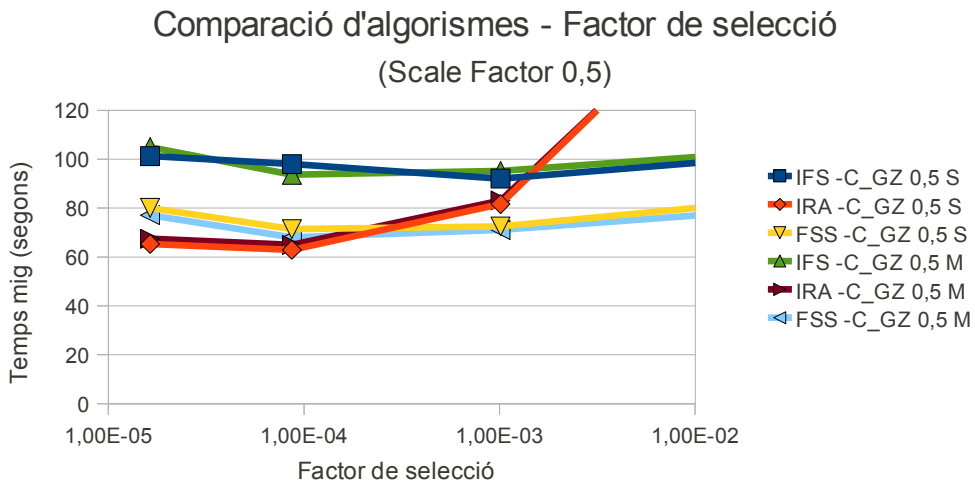
Gràfica 23: Influència del factor de selecció (Scale Factor de 0,5)

D'aquesta gràfica hem de fer notar que l'eix vertical està en escala logarítmica per tal que els valors quedin visualment més separats. A la gràfica veiem que els temps obtinguts usant o no índexs multinivell són molt similars i queden sobreposats.

Veiem, també, que amb aquest volum de dades el cost de l'algorisme IRA comença a distanciar-se dels altres algorismes a partir del factor de selecció 10^{-3}.

El temps requerit per l'algorisme FSS s'incrementa, com en els casos anteriors, a partir amb el factor de selecció de 1 i es situa per sobre de l'IFS.

En els casos amb factors de selecció molt petits els valors estan molt junts i costen de discernir. Per aquest motiu he fet la gràfica següent, per veure millor aquesta regió:



Gràfica 24: Influència del factor de selecció (Scale Factor de 0,5)

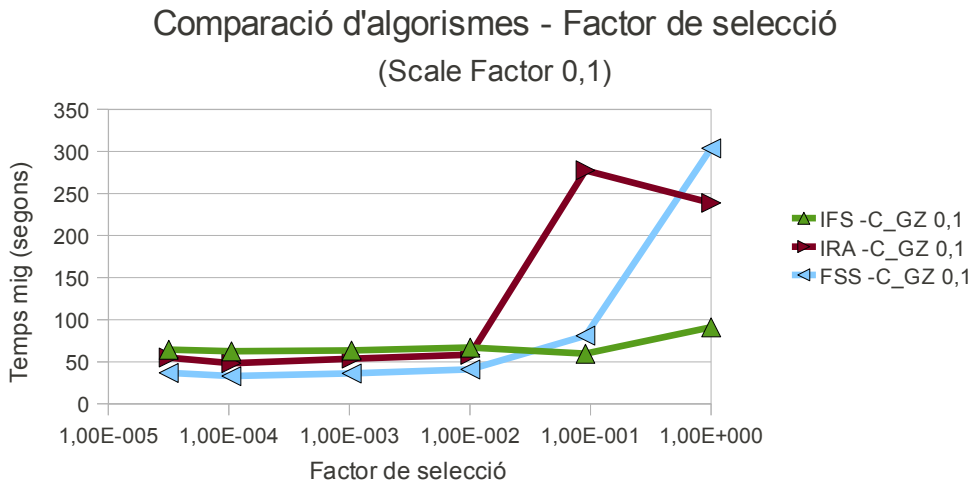
La gràfica 24 ens permet veure amb més detall l'interval que va del factor de 10^{-5} al factor de selecció de 10^{-2} (aproximadament). Hi veiem que l'ús o no d'índexs multinivell té una influència baixa en el rendiment dels diferents algorismes.

En els resultats obtinguts amb un Scale Factor de 0,5, ja podem apreciar-hi que depenent del factor de selecció de la consulta surt més a compte fer servir un algorisme o un altre. Pels factors de selecció de 10^{-5} i 10^{-4} és millor algorisme és l'IRA. A partir del 10^{-3} fins el 10^{-1} (mirant la gràfica de dalt) és el FSS. Finalment pel factor de selecció de 1 el millor algorisme és l'IFS. Podem veure també que el temps mínim per a resoldre les consultes està entre els 80 i els 300 segons.

Entorn distribuït

Les proves que presento en aquest apartat s'han dut a terme usant l'entorn distribuït, utilitzant els Scale Factors de 0,1 i 0,5 amb compressió i amb índexs d'un sol nivell, ja que aquesta és la configuració que ens ha donat més bon rendiment amb els índexs d'uns sol nivell.

Vegem primer la gràfica que hem obtingut utilitzant un Scale Factor de 0,1:

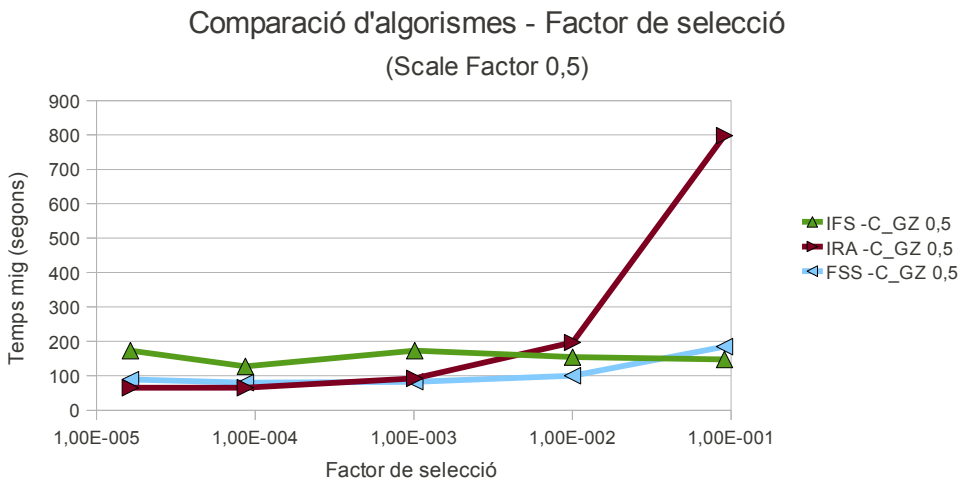


Gràfica 25: Influència del factor de selecció, entorn distribuït(Scale Factor 0,1)

A la gràfica 25 hi podem veure un comportament molt semblant al observat amb un Scale Factor de 0,1 amb un sol ordinador (gràfica 21). Hi ha una gran diferència i és que l'algorisme IRA és més lent obtenint una desena part de la taula que la taula sencera. Aquest fet és causat perquè les dues execucions pel factor de selecció 0,1 van coincidir que es feien a l'ordinador de sobre taula (que és més lent) i les del factor 1 van fer ambdues en el portàtil.

La resta de conclusions extretes per aquest Scale Factor amb un entorn no distribuït són aplicables al cas de l'entorn distribuït.

Vegem ara els resultats obtinguts amb un Scale Factor de 0,5:



Gràfica 26: Influència del factor de selecció, entorn distribuït(Scale Factor 0,5)

Per aquesta configuració, en l'entorn distribuït, obtenim uns resultats molt similars als de obtinguts en un entorn no distribuït. Tot i així, hem de tenir en compte que en aquesta gràfica s'hi ha introduït soroll perquè els dos ordinadors tenen unes prestacions molt diferents. No obstant això, podem veure que l'algorisme IRA és el que obté el millor rendiment pels factors de selecció 10^{-5} i 10^{-4} entre 10^{-3} i 10^{-2} és el FSS i l'IFS és el que va millor pels factors de selecció més grans.

Taula resum

A continuació mostro una taula comparativa dels diferents algorismes feta amb els resultats obtinguts de les proves:

Millor algorisme						
Factor de selecció	$\sim 10^{-5}$	$\sim 10^{-4}$	$\sim 10^{-3}$	$\sim 10^{-2}$	$\sim 10^{-1}$	~ 1
Scale Factor 0,01	FSS	FSS	FSS	FSS	FSS	FSS
Scale Factor 0,05	FSS	FSS	FSS	FSS	FSS	IFS
Scale Factor 0,1	FSS	FSS	FSS	FSS	FSS[14]	IFS
Scale Factor 0,5	IRA	IRA	FSS	FSS	FSS	IFS

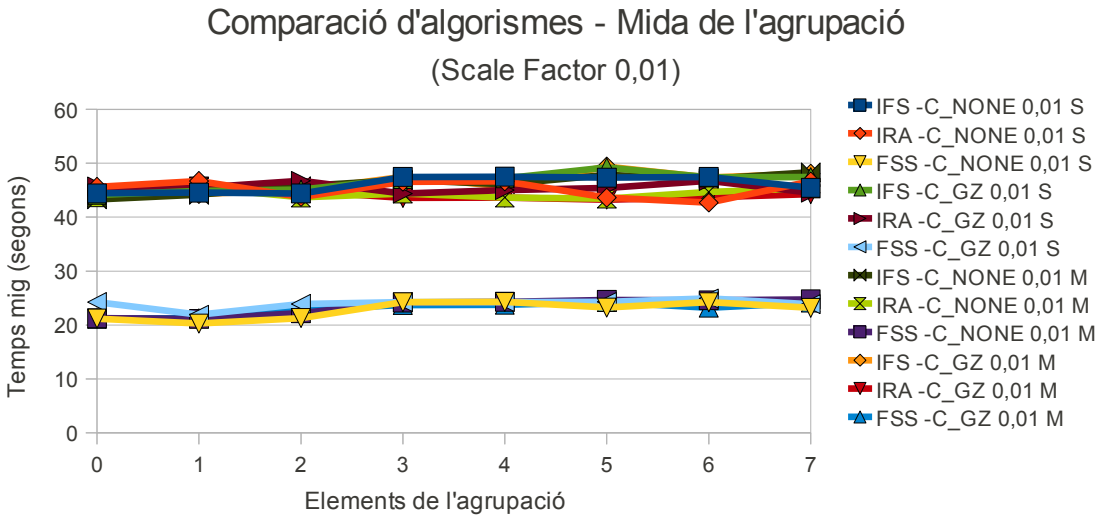
Aquesta taula mostra l'algorisme que obté els temps d'execució més baixos per a les diferents configuracions provades, amb compressió o sense compressió, amb índexs multinivell o d'un sol nivell i amb un o dos servidors.

13.2.4.2 Mida de l'agrupació

Ara veurem les gràfiques dels resultats obtinguts pels diferents algorismes a l'hora d'estudiar la influència de la mida de l'agrupació sobre el rendiment de les diferents consultes. Les consultes utilitzades per a fer les proves d'aquesta secció són les que he explicat a l'apartat 12.2.3. En les diferents consultes es varia la mida de l'agrupació de 0 a 7 atributs diferents. La resta de valors de les consultes s'han deixat fixats al valor de la mediana: un factor de selecció d'aproximadament 0,01, dues clàusules i seleccionant 3 atributs.

Scale Factor 0.01

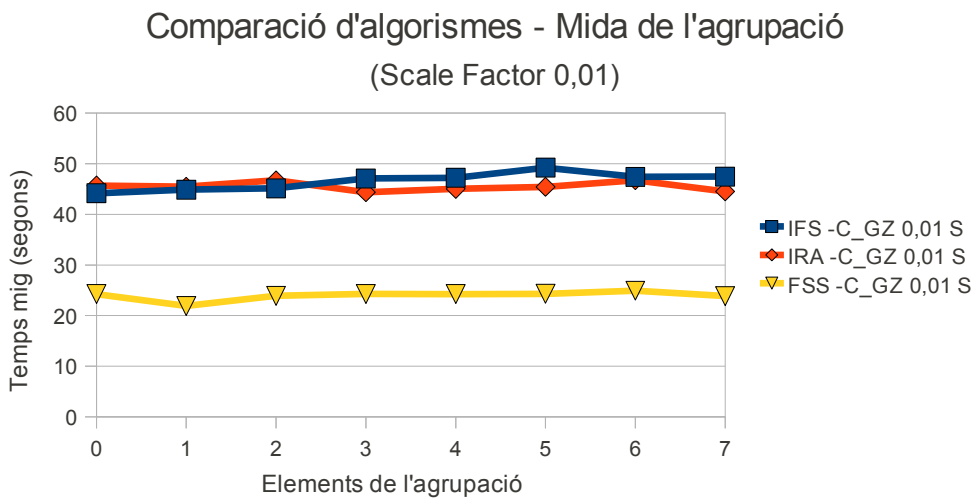
A la gràfica següent veiem els resultats que s'obtenen amb un Scale Factor de 0,01 amb les diferents configuracions que he provat:



Gràfica 27: Influència de la mida de l'agrupació (Scale Factor de 0,01)

En aquesta gràfica hi podem veure que la influència entre les diferents configuracions sobre el rendiment en les diferents consultes és molt baix. Per aquest motiu, al llarg d'aquest apart, mostraré només els resultats obtinguts amb compressió i utilitzant índexs d'un sol nivell. Vegem la gràfica pel Scale Factor de 0,01 amb aquestes restriccions:

14 En cas de fer servir dos servidors, la millor opció ha resultat ser IFS. Cal dir, que molt possiblement sigui causat per la diferència de potència dels servidors.



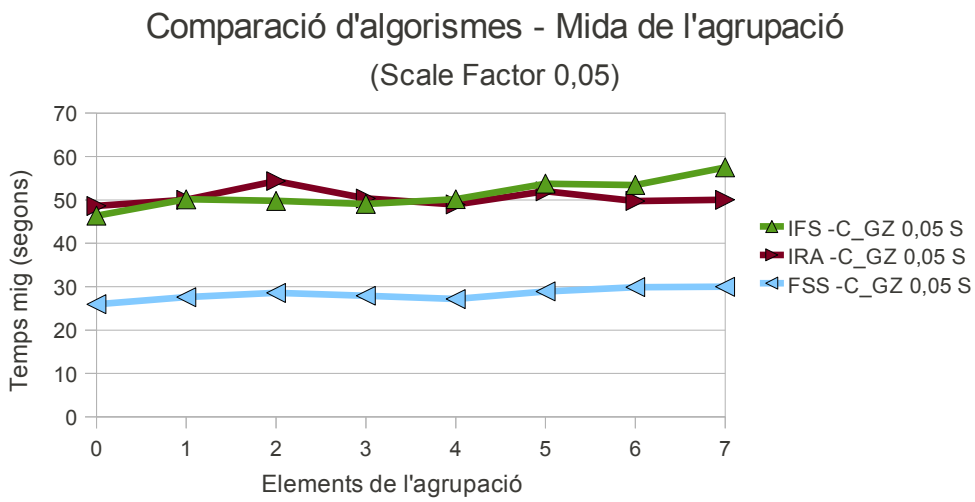
Gràfica 28: Influència de la mida de l'agrupació (Scale Factor de 0,01)

Podem apreciar clarament que el FSS té un rendiment molt superior als altres dos algorismes, va gairebé el doble de ràpid. Molt possiblement aquest fet sigui degut a què el volum de dades és baix i, per tant, el temps requerit per l'execució sigui el temps de gestió del MapReduce. Aquest fet s'explica perquè els algorismes IFS i IRA fan servir dos MapReduce i el FSS un.

Podem veure, també, que els algorismes IFS i IRA tenen uns rendiments molt similars. No es pot apreciar que hi hagi una relació entre el temps d'execució i el nombre d'elements de l'agrupació.

Scale Factor 0.05

En aquest apartat veurem els resultats obtinguts amb un Scale Factor de 0,05. A la gràfica que hi ha a continuació hi mostro els resultats obtinguts utilitzant compressió i índexs d'un sol nivell:



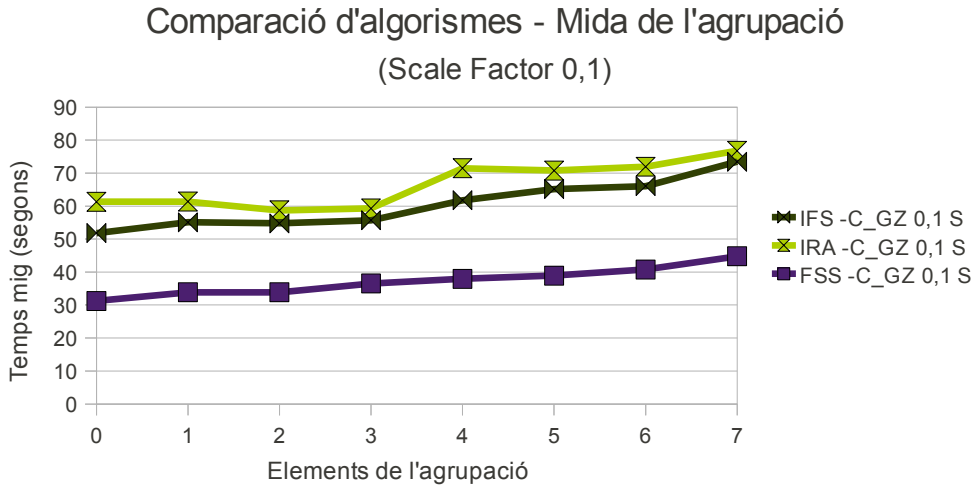
Gràfica 29: Influència de la mida de l'agrupació (Scale Factor de 0,05)

A la gràfica 29 hi veiem que els algorismes IRA i IFS es comencen a distanciar amb un Scale Factor de 0,05. A més, es pot apreciar un petita influència de la mida de l'agrupació en el rendiment dels algorismes IFS i FSS, com més gran és la mida de l'agrupació, més tarda la seva execució.

Podem veure a la gràfica, també, que el temps de tots els algorismes ha augmentat respecte el Scale Factor de 0,01.

Scale Factor 0.1

Ara veurem els resultats que s'obtenen amb un Scale Factor de 0,1:



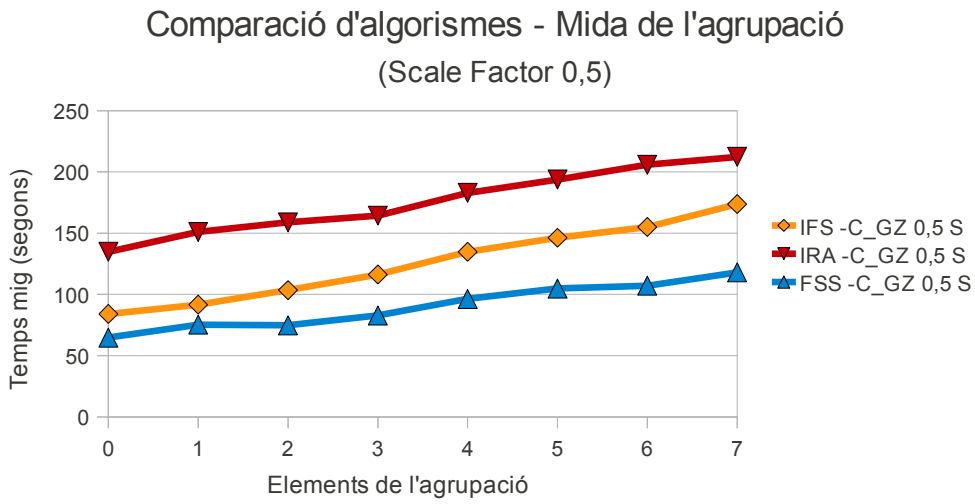
Gràfica 30: Influència de la mida de l'agrupació (Scale Factor de 0,1)

En aquesta gràfica hi podem veure que el temps de l'algorisme IRA és més gran que el de l'algorisme IFS. A més hi apreciem que el temps requerit per a l'execució del FSS ha augmentat més que el de l'IFS.

D'altra banda, ja es veu una influència de la mida de l'agrupació sobre el rendiment. Aquesta influència, però, és més gran els algorismes IFS i IRA que en FSS.

Scale Factor 0.5

Ara veurem els resultats obtinguts pel factor de selecció més gran que he provat. Són els següents:

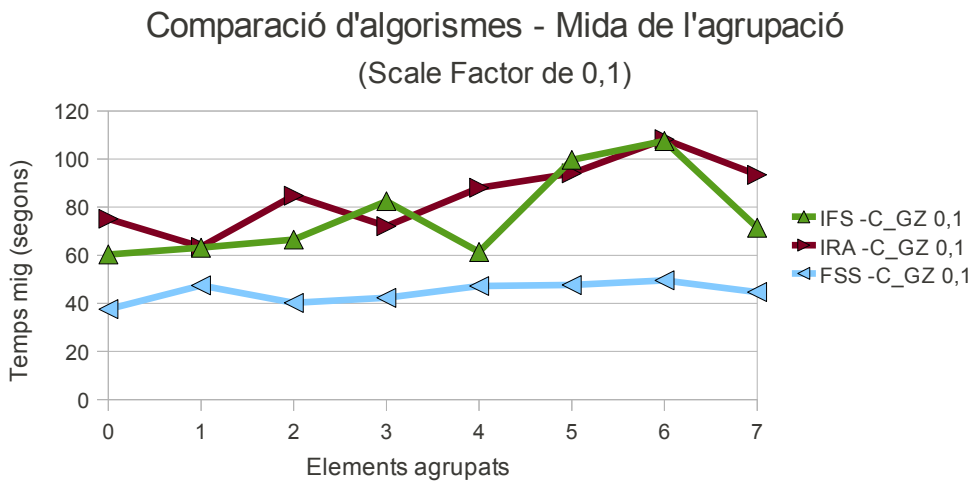


Gràfica 31: Influència de la mida de l'agrupació (Scale Factor de 0,5)

A la gràfica 31 veiem que la diferència de temps augmenta entre els algorismes IFS i IRA. Simultàniament, el temps de l'algorisme IFS i FSS són més propers pels Scale Factors més petits i es distancien per les agrupacions més grans.

Entorn distribuït

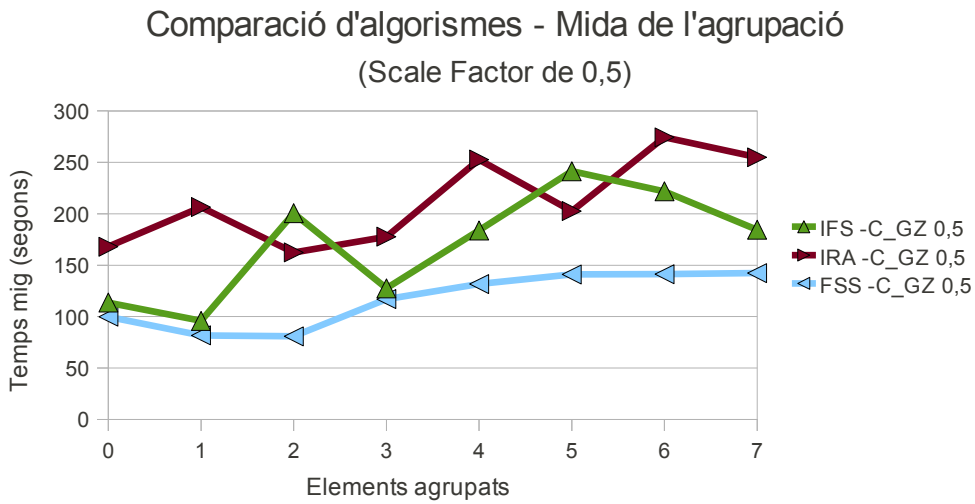
En aquesta secció comentaré la influència observada en els resultats del mida de l'agrupació sobre el rendiment dels algorismes en un entorn distribuït. Els resultats que mostro són utilitzant compressió i índexs d'un sol nivell. Vegem primer els resultats obtinguts amb un Scale Factor de 0,1:



Gràfica 32: Influència de la mida de l'agrupació, entorn distribuït (Scale Factor 0,1)

En aquesta gràfica s'hi pot entreveure la tendència general dels algorismes. El FSS resulta més ràpid que l'IFS i l'IRA, però es veu clarament quines són les execucions que s'han fet a l'ordinador de sobre taula perquè resulten més lentes que les fetes en el portàtil. Aquest causa l'he pogut comprovar examinant els logs de les tasques.

Vegem ara els resultats obtinguts amb un Scale Factor de 0,5 en un entorn distribuït:



Gràfica 33: Influència de la mida de l'agrupació, entorn distribuït (Scale Factor 0,5)

Aquesta gràfica és semblant a l'obtinguda amb un Scale Factor de 0,1. Els temps són més elevats, però també es pot apreciar una gran diferència de rendiment entre els ordinadors.

Comparant els resultats de la gràfica 33 amb els de la gràfica 31, veiem que els temps mesurats en el portàtil són molt semblant en els dos casos, una mica superior en l'entorn distribuït. D'aquí, en podem extreure que la configuració que he fet del clúster no influeix molt negativament respecte el rendiment d'un node.

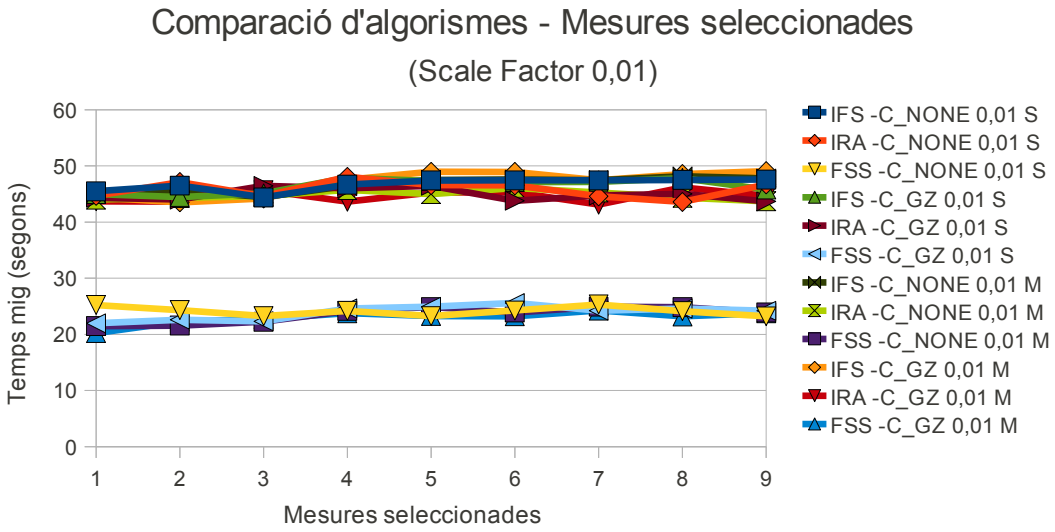
13.2.4.3 Mesures seleccionades

Ara estudiarem la influència que té el nombre de mesures seleccionades en el rendiment de la consulta. Les proves s'han fet utilitzant les consultes explicades a l'apartat 12.2.2. En aquestes consultes es proven diferents nombres de mesures seleccionades, entre 1 i 9. La resta de valors característics de les consultes (el factor de selecció, la mida de l'agrupació i el nombre de condicions) estan fixats al valor de la mediana.

A continuació veurem els resultats que hem obtingut per a cadascun dels Scale Factors:

Scale Factor 0.01

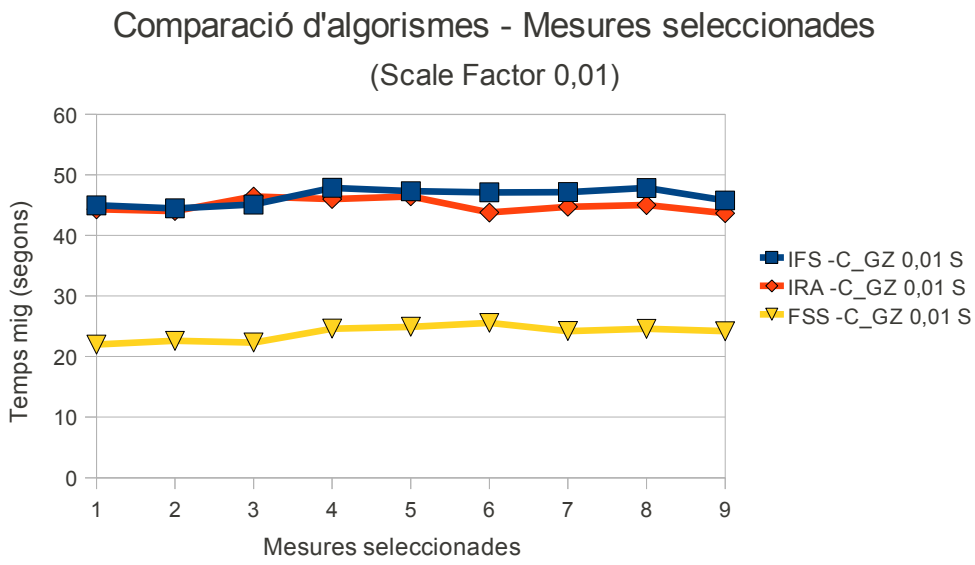
Vegem primer una gràfica en la que apareixen totes les configuracions que he provat (amb o sense compressió, amb índexs multinivell o d'un sol nivell):



Gràfica 34: Influència del nombre de mesures seleccionades (Scale Factor de 0,01)

En aquesta gràfica hi podem veure que la influència en el rendiment de l'ús o no de compressió a les taules és molt baixa. També és baixa la influència de l'ús d'índexs multinivell o d'un sol nivell. Per aquest motiu les gràfiques mostro a partir d'aquest punt, en aquest apartat, són utilitzant índexs d'un sol nivell i compressió a les taules.

Ara estudiem la taula següent que conté els resultats pel Scale Factor de 0,01, amb compressió i amb índexs d'un sol nivell:

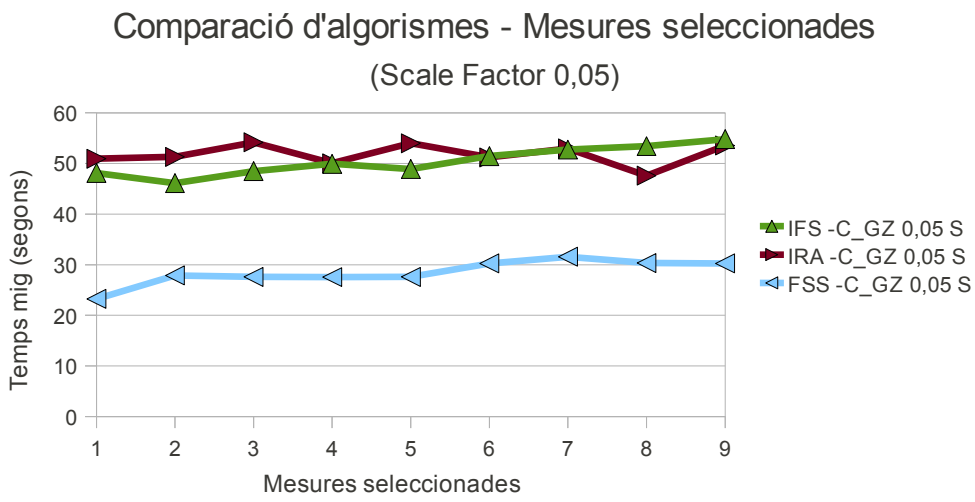


Gràfica 35: Influència del nombre de mesures seleccionades (Scale Factor de 0,01)

A la gràfica hi podem veure que els algorismes IRA i IFS tenen un rendiment similar. L'algorisme FSS té un temps d'execució inferior al dels algorismes IFS i IRA. Aquest temps és gairebé el doble de ràpid. Com ja he explicat anteriorment, aquesta diferència és causada perquè els algorismes IFS i IRA fan servir dos MapReduce i el FSS un.

Scale Factor 0.05

Ara veurem els resultats obtinguts amb un Scale Factor de 0,05:

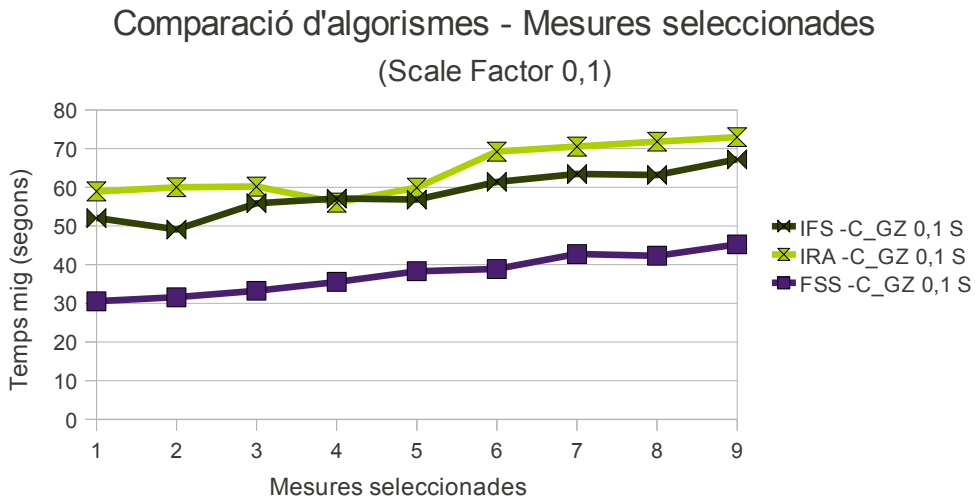


Gràfica 36: Influència del nombre de mesures seleccionades (Scale Factor de 0,05)

Veiem que es manté la distància, que havíem apreciat amb un Scale Factor de 0,01, de l'algorisme FSS respecte els algorismes IFS i l'IRA. Veiem, a més, que de mitjana l'algorisme IRA comença a resultat una mica més lent que l'algorisme IFS.

Scale Factor 0.1

Ara veurem els resultats obtinguts amb un Scale Factor de 0,1:



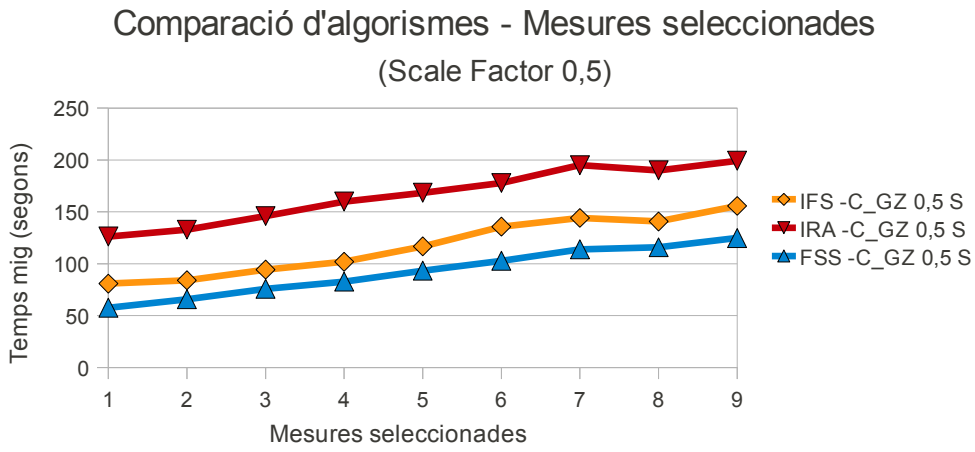
Gràfica 37: Influència del nombre de mesures seleccionades (Scale Factor de 0,1)

Respecte la gràfica 36, veiem que amb un Scale Factor de 0,1 s'incrementa la distància mitjana entre els temps dels algorismes IRA i IFS. L'IRA obté un millor rendiment seleccionant 4 mesures que seleccionant-ne 3, l'única explicació que hi he trobat és que: el nou valor que s'afegeix, *quantity*, és suficientment petit com perquè la informació extra que es necessita ja es trobi a memòria o bé que es mantingui a memòria (total o parcialment) la informació cercada de la consulta anterior i que, per tant, no facin falta tants accessos a disc.

D'altra banda, la distància en temps que hi ha entre l'algorisme IFS i el FSS es manté molt similar al de la gràfica 36, tot i que de mitjana s'ha reduït una mica.

Scale Factor 0.5

Vegem a continuació els resultats que hem obtingut amb un Scale Factor de 0,5:



Gràfica 38: Influència del nombre de mesures seleccionades (Scale Factor de 0,5)

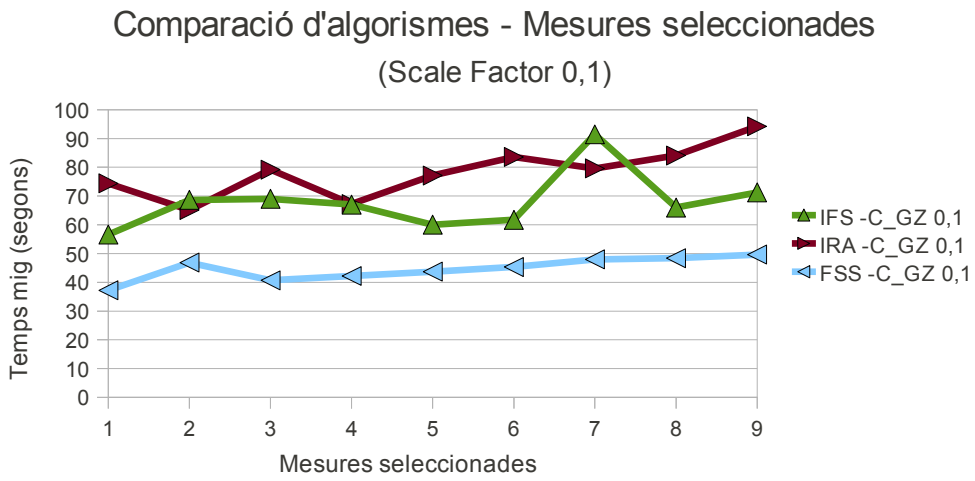
A la gràfica 38, hi veiem que els temps de l'algorisme IRA segueixen incrementant-se més ràpidament que els dels algorismes IFS i FSS. Aquest increment ha estat d'entre 70 i 130 segons respecte el Scale Factor de 0,1. Es produeix un increment més gran com més atributs es seleccionen. Si comparem els temps dels algorismes IFS i FSS de la gràfica 38 amb els de la gràfica 37 veurem que els temps han augmentat entre 20 i 90 segons.

En aquesta gràfica, la 38, hi podem apreciar una clara relació lineal respecte el temps necessitat per resoldre la consulta i el nombre d'atributs seleccionats.

Entorn distribuït

En aquest apartat veurem els resultats que he obtingut amb les consultes per estudiar la influència del nombre de mesures seleccionades quan s'executen en un entorn distribuït. Les proves s'han realitzat utilitzant compressió i índexs d'un sol nivell.

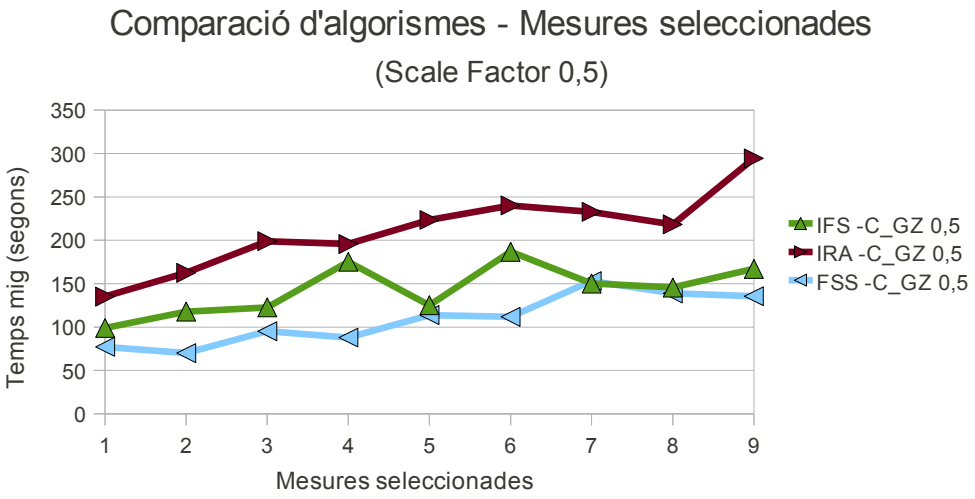
Vegem primer els resultats obtinguts amb un Scale Factor de 0,1:



Gràfica 39: Influència del nombre de mesures seleccionades, entorn distribuït(Scale Factor 0,1)

A la gràfica 39 veiem que els resultats que hem obtingut amb un Scale Factor de 0,1 en un entorn distribuït són semblants als obtinguts amb un Scale Factor 0,1 en un entorn no distribuït (gràfica 37). Veiem, però, que el cost d'execució en general ha augmentat al voltant d'uns 10 segons. Veiem, també, que la separació en els temps dels algorismes IFS i IRA no és tant clara. Aquest fet és degut al què l'execució a l'ordinador de sobre taula resulta més lenta en el portàtil. He pogut comprovar que efectivament és provocat per l'execució en l'altre ordinador examinant els logs generats en el procés.

Veiem ara els resultats que hem obtingut amb un Scale Factor de 0,5:



Gràfica 40: Influència del nombre de mesures seleccionades, entorn distribuït(Scale Factor 0,5)

En aquesta gràfica veiem que amb un Scale Factor de 0,5 en un entorn distribuït té un cost temporal superior al cost obtingut en un entorn no distribuït. A diferència dels resultats mostrats a la gràfica 39, els temps dels algorismes IFS i IRA queden clarament separats.

A més, a la gràfica 40 podem apreciar que hi ha una relació entre el nombre de mesures seleccionades i el temps d'execució de l'algorisme IRA. No és una relació lineal tant clara com la que podem apreciar a la gràfica 36 però és més clara que la de la gràfica 39.

També veiem a la gràfica 40 que els temps obtinguts amb els algorismes IFS i FSS tampoc són tant lineals com els obtinguts amb un Scale Factor de 0,5 en un entorn no distribuït. Aquest fet és causat com en els casos anteriors perquè les execucions fetes a l'ordinador de sobre taula resulten més lentes.

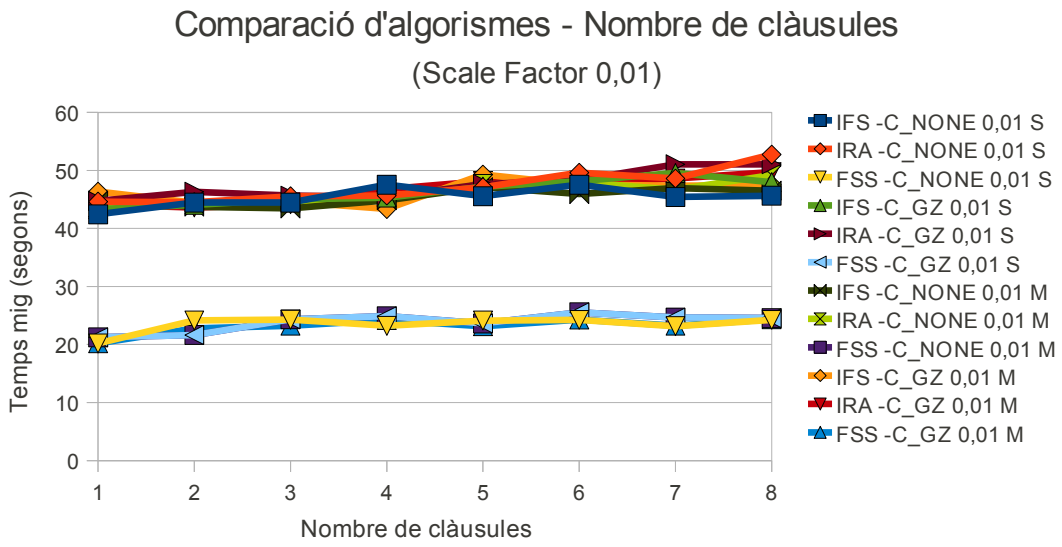
13.2.4.4 Nombre de clàusules

Ara veurem la influència de l'última de les característiques, el nombre de clàusules del predicat, sobre el rendiment de les consultes utilitzant els diferents algorismes. Per fer aquestes proves he fet servir les consultes que he explicat a l'apartat 12.2.4. Aquestes consultes proven consultes similars variant el nombre de clàusules del predicat entre 1 i 6. Els altres paràmetres característics de les consultes s'han deixat fixats als valors de les seves medianes (factor de selecció de 0,01, tres mesures seleccionades i un element a l'agrupació).

A continuació veurem els resultats obtinguts amb aquest conjunt de consultes per cadascun dels Scale Factor que hem provat:

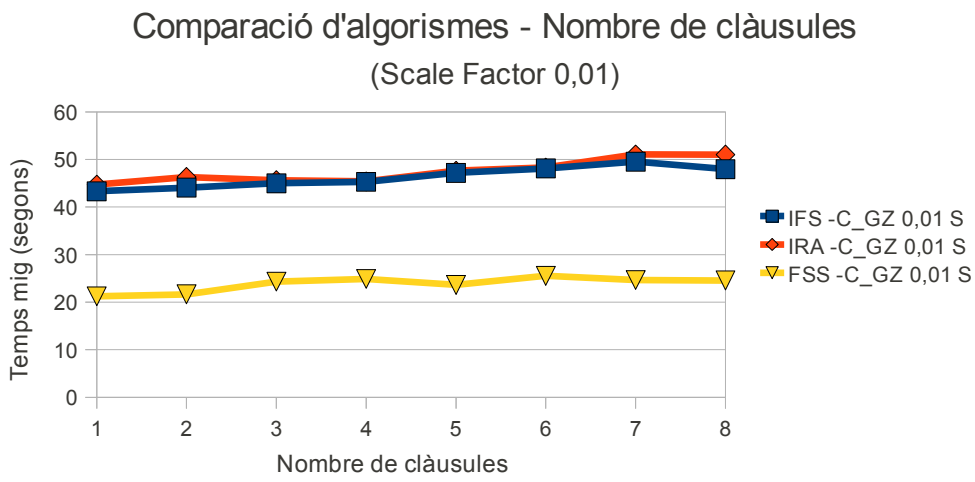
Scale Factor 0.01

A continuació mostro una gràfica on he representat els resultats que obtingut utilitzant un Scale Factor de 0,01:



Gràfica 41: Influència del nombre de clàusules (Scale Factor de 0,01)

En aquesta gràfica, com en els apartats anteriors, hi podem veure que la influència de les diferents configuracions, ús o no de compressió i l'ús d'índexs d'un sol nivell o multinivell, és molt baixa sobre el rendiment dels diferents algorismes. Per aquest motiu les gràfiques que mostraré d'aquí en endavant seran utilitzant compressió i índexs d'un sol nivell i, així, podrem estudiar més clarament els resultats. Vegem els resultats obtinguts amb aquesta configuració i un Scale Factor de 0,01:



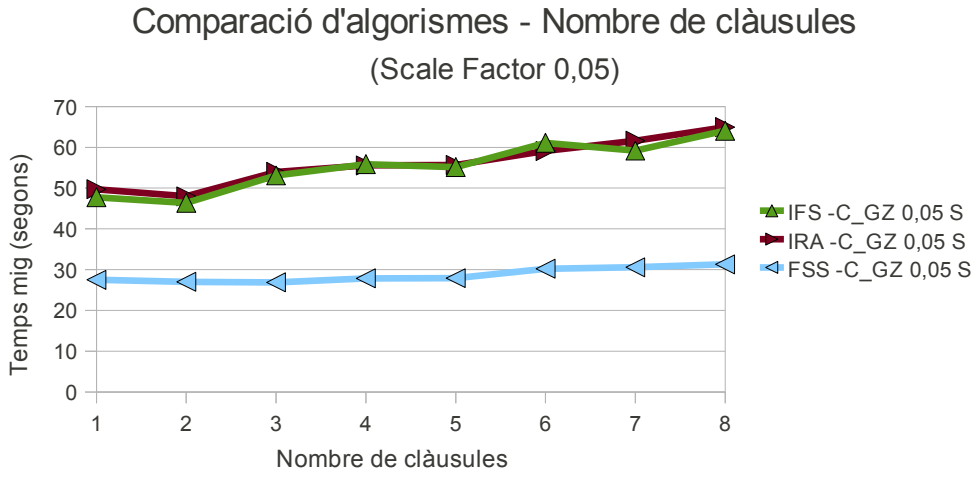
Gràfica 42: Influència del nombre de clàusules (Scale Factor de 0,01)

En aquesta gràfica podem veure que els algorismes IFS i IRA tenen un rendiment molt similar. Tot i així l'algorisme IRA requereix d'un temps una mica superior. Aquests dos algorismes, però, necessiten un temps gairebé el doble que el del FSS.

Veiem, a més, que amb un volum de dades petit el nombre de clàusules que té una consulta té una influència temporal molt baixa.

Scale Factor 0.05

Examinem ara els resultats obtinguts amb un Scale Factor de 0,05 mostrats a la gràfica que hi ha a continuació:

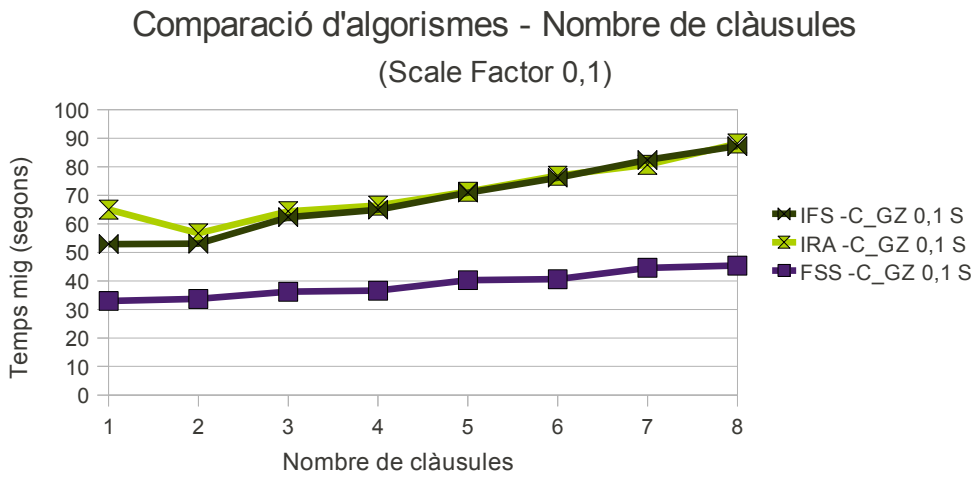


Gràfica 43: Influència del nombre de clàusules (Scale Factor de 0,05)

En aquesta gràfica veiem que amb un volum de dades cinc vegades més gran (respecte els resultats de la gràfica 42) ja es comencen a marcar diferències de rendiment. Com més clàusules té la consulta, més lenta resulta. Al mateix temps, sembla que la influència sobre el FSS és inferior, cosa que és completament normal perquè aquest últim algorisme consisteix en un escaneig de la taula. Podem veure com en aquesta diferència s'accentua en els resultats obtinguts amb Scale Factors superiors com els que mostro a més endavant.

Scale Factor 0.1

Vegem ara els resultats que s'han obtingut per aquestes proves amb un Scale Factor de 0,1:

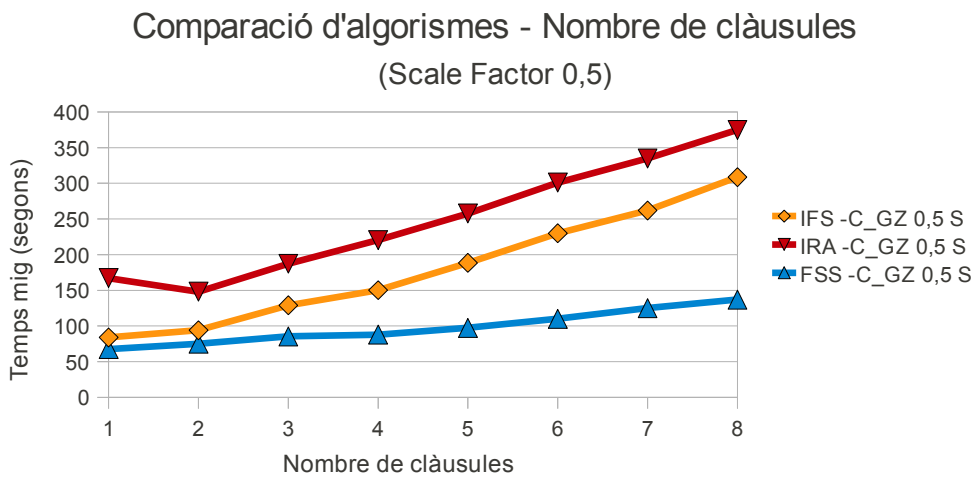


Gràfica 44: Influència del nombre de clàusules (Scale Factor de 0,1)

En aquesta gràfica veiem que encara s'accentua més la influència negativa del nombre de clàusules en els algorismes IRA i IFS. Aquesta mateixa influència també és pot apreciar però en menor grau per a l'algorisme FSS.

Scale Factor 0.5

Estudiem ara els resultats obtinguts amb un Scale Factor de 0,5:



Gràfica 45: Influència del nombre de clàusules (Scale Factor de 0,5)

Per aquest volum de dades, veiem que l'algorisme IRA ja obté un rendiment clarament inferior que l'IFS i el FSS.

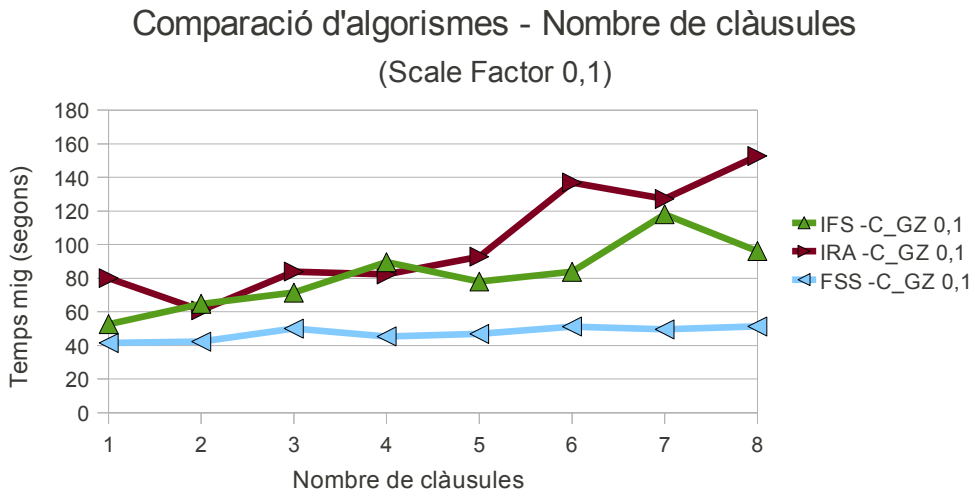
També veiem, com en els casos anteriors, que la influència del nombre de clàusules sobre l'algorisme FFS és inferior a la influència percebuda per l'algorisme IFS. No obstant això, l'algorisme IFS té un rendiment molt semblant al FSS quan el nombre de clàusules és baix.

La diferència de la influència del nombre clàusules sobre els algorismes és causada pel motiu següent: els algorismes IRA i IFS han de fer un o més accessos a l'índex per cadascuna de les clàusules i l'algorisme FSS no perquè només fa un escaneig de la taula de fets.

Entorn distribuït

Ara veurem la influència del nombre de clàusules en un entorn distribuït. Els resultats que mostro s'han obtingut utilitzant compressió i índexs d'un sol nivell.

Vegem primer la gràfica on represento els resultats obtinguts amb un Scale Factor de 0,1:

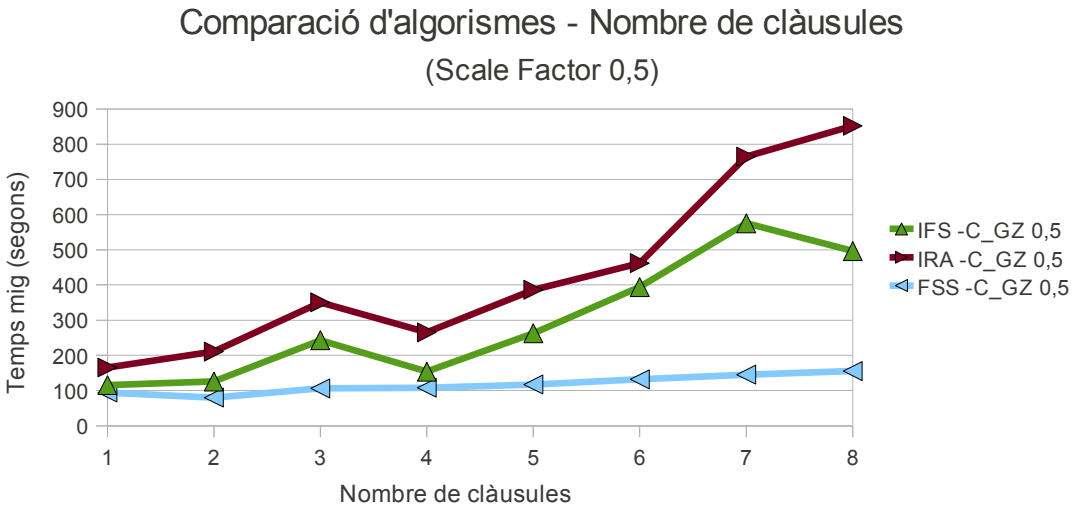


Gràfica 46: Influència del nombre de clàusules, entorn distribuït(Scale Factor de 0,1)

Respecte els resultats obtinguts en un entorn no distribuït (gràfica 44) veiem que s'han accentuat les diferències entre els algorismes IRA i IFS. D'altra banda, l'algorisme FSS té obté un rendiment més semblant a l'obtingut pels altres dos algorismes.

Podem apreciar que en l'entorn distribuït, s'ha perdut linealitat del nombre de clàusules i el temps requerit pels algorismes que accedeixen a l'índex. Com en les altres proves, aquest efecte és degut a la diferència de prestacions que presenten els ordinadors del clúster.

A la gràfica següent podem estudiar els resultats obtinguts amb un Scale Factor de 0,5 en un entorn distribuït:



Gràfica 47: Influència del nombre de clàusules, entorn distribuït(Scale Factor de 0,5)

Tal com hem vist a la gràfica 45, en aquesta gràfica hi podem veure la forta relació que hi ha del nombre de clàusules de les consultes amb els temps requerits pels algorismes IRA i IFS. També s'observa la baixa influència del paràmetre estudiat sobre el rendiment de l'algorisme FSS.

Veiem, com en el cas de l'entorn no distribuït, que utilitzant un entorn distribuït amb aquest volum de dades el rendiment de l'algorisme del FSS és molt semblant al de IFS quan el nombre de clàusules és baix.

La variància dels resultats en un entorn distribuït augmenta igual que en les altres proves.

13.2.4.5 Conclusions

En aquest apartat comento a grans trets els resultats obtinguts de les diferents proves que s'han dut a terme. Aquestes s'han fet utilitzant els jocs de proves del TPC-H que ens proporcionen una estructura de dades realista.

Amb les proves realitzades, s'ha pogut veure clarament que amb una quantitat de dades suficientment gran s'obtenen rendiments diferents per a la mateixa consulta amb els diferents algorismes:

Index Random Access (IRA)

L'algorisme IRA, que realitza un accés aleatori de la informació que s'ha d'obtenir després d'haver accedit a l'índex, obté el millor rendiment amb les consultes amb un factor de selecció suficientment petit i un volum de dades prou gran. Per contra, aquest algorisme obté els pitjors resultats quan el factor de selecció és massa gran.

L'efecte sobre el rendiment de la mida del conjunt d'atributs amb els que fem l'agrupació o el nombre de mesures escollides és equiparable al dels algorismes IFS i FSS. D'altra banda, el nombre de clàusules tenen un impacte més gran en l'IRA i l'IFS que no pas en el FSS. És així perquè per cada nova clàusula requereix un o més accessos addicionals a l'índex.

Index Filtered Scan (IFS)

L'algorisme IFS resulta la millor opció per a factors de selecció propers a 1 i amb volums de dades suficientment grans. La influència de les altres característiques de la consulta sobre el temps d'execució és semblant les observades en l'algorisme IRA. Cal dir, però, que tant en

les proves de la mida de l'agrupació com en el nombre de mesures seleccionades obté millors resultats que l'IRA tot i que la influència observada és la mateixa. Aquest fet, fa pensar que la diferència observada sigui deguda al factor de selecció de la consulta.

La influència del nombre de clàusules del predicat sobre el temps requerit segueix essent més gran que pel FSS, perquè l'algorisme IFS ha de fer un o més accessos a l'índex per cadascuna de les clàusules que tinguem.

Filtered Source Scan (FSS)

L'algorisme FSS resulta la millor opció per a aquelles consultes que tenen factor de selecció entre 10^{-3} i 10^{-1} amb els volums de dades que he provat en aquest projecte. El temps requerit per aquest algorisme es veu perjudicat, com els altres dos algorismes, a mesura s'augmenta el nombre d'elements que componen l'agrupació, el nombre de mesures seleccionades i el nombre de clàusules dels predicat que té la consulta. Val a dir, però, que el nombre de clàusules té una influència menor que en l'IFS i l'IRA perquè aquest algorisme sempre realitza un escaneig de la taula i, per tant, tot i que durant l'escaneig s'ha d'obtenir més informació per poder comprovar que es compleixen les clàusules, la pot obtenir per blocs i per tant pot aprofitar la localitat espacial. D'aquesta manera, obté un millor rendiment.

14 Conclusions

Un cop acabat el projecte i mirant en perspectiva el treball realitzat al llarg d'aquests mesos, en puc fer una valoració positiva. Personalment crec que el sistema desenvolupat compleix els objectius plantejats pel projecte.

He programat els diferents algorismes del projecte i he comprovat que funcionin correctament. El sistema desenvolupat, a més, aconsegueix satisfer en gran mesura els requeriments del projecte. Permet definir les dimensions del cub de dades i crear els índexs per a cadascuna de les dimensions. Al seu torn, el mecanisme per a la realització de consultes permet especificar diferents clàusules de predicat i criteris d'agrupació dels resultats de forma intuïtiva.

S'ha pogut constatar que l'arquitectura emprada pel projecte és modificable per la gran quantitat de canvis que he pogut fer en els algorismes al llarg del projecte. D'altres requeriments, com la tolerància a fallades i l'execució concurrent en ordinadors, els satisfem utilitzant les possibilitats que ens aporta Hadoop.

El requeriment de l'eficiència és més difícil de mesurar. He dedicat molt d'esforç a millorar el rendiment dels algorismes i he aconseguit algunes fites importants. D'altra banda, per a les proves no s'ha disposat d'un entorn hardware realista i, per tant, les dades obtingudes no són contrastables amb altres sistemes.

Amb les proves que he fet, s'ha pogut comprovar que el projecte es pot executar de forma concurrent amb dos ordinadors. La concurrència només l'he pogut comprovar en el cas del desnormalitzador, però l'arquitectura que he utilitzat en la implementació dels algorismes IRA, IFS i FSS és la mateixa i he pogut comprovar que es pot executar en diferents ordinadors. També he pogut constatar que no es fàcil configurar un clúster i aconseguir que funcioni encara que sigui en dos ordinadors.

Els resultats obtinguts del rendiment dels algorismes durant les proves són bons i esperançadors. Bons en el sentit que ja es pot començar a intuir en quines situacions és millor utilitzar cada algorisme. Però els resultats també són esperançadors perquè deixen entreveure que si es disposés de més recursos es podrien obtenir resultats més bons.

La gran varietat de configuracions a les proves demostra que el sistema és molt configurable i que es pot automatitzar el procés d'execució de les proves. Els scripts que he preparat per a realitzar les proves del projecte demostren que es pot automatitzar el procés d'execució del sistema.

Personalment crec que la interfície gràfica desenvolupada resulta clara i que costa poc acostumar-s'hi. És una interfície complexa, però ho requereix la complexitat del procés de les consultes. D'altra banda, he de reconèixer que m'ha resultat extremadament útil la generació automàtica de les comandes que he programat a la interfície gràfica per a la preparació de les proves. Així he pogut fer les comandes còmodament amb la interfície gràfica i llavors, per mesurar el rendiment en diferents configuracions, feia una execució en bateria de totes les consultes amb la interfície textual.

Aquest projecte m'ha aportat una oportunitat molt bona per a aprendre el funcionament de MapReduce i com implementar algorismes que es beneficiïn de les possibilitats que ofereix. És una tecnologia molt innovadora que es comença a implementar en moltes empreses i tinc l'esperança que aquests coneixements els podré aplicar en un futur.

14.1 Línies de futur

Aquest projecte és un projecte d'investigació i, com a tal, sempre s'hi pot seguir treballant. A continuació comento unes quantes línies que es poden seguir a partir d'on ha quedat el projecte:

Es podrien fer proves amb volums de dades més grans i en clústers de debò. Per a poder simular una situació realista farien falta, com a mínim, uns 10 ordinadors en paral·lel. D'altra banda, no es tracta només de configurar-los, sinó també d'optimitzar-ne l'execució per a les consultes i obtenir els millors resultats possibles.

S'hauria d'estudiar una modificació de l'IFS: en lloc de crear un bitmap per comprovar quines tuples ens interessen, fer dos recorreguts en paral·lel sobre la llista d'identificadors que s'han d'obtenir i els identificadors de les tuples que ha de processar el segon MapReduce de l'algorisme. Fent-ho així, estalviaríem memòria i l'algorisme tindria un cost lineal perquè les dues llistes arribarien ordenades.

De cara a la implantació d'aquest sistema en un entorn de producció real, s'hauria de fer que calculés automàticament, per a cada consulta, quin és el millor algorisme per a utilitzar. Per a poder-ho fer, s'haurien de fer proves amb volums de dades més grans i crear estadístiques de les dades de les taules.

Un altre requeriment que crec important per a la implantació d'aquest sistema seria l'actualització dels índexs de forma incremental. Així, podríem evitar haver de crear-los de nou cada vegada.

15 Glossari

A continuació defineixo diverses paraules que he fet servir al llarg de la memòria i que no són d'ús corrent:

- **Bases de dades operacionals**: Bases de dades que es fan servir quotidianament en una organització per tal de dur a terme les seves activitats.
- **Clau forana**: Conjunt de columnes d'una taula que referencien a una altra taula. Per tal que la referència sigui inequívoca se sol utilitzar la clau primària.
- **Clau primària**: Columna o columnes d'una taula els valors de les quals serveixen per identificar de forma inequívoca cadascuna de les seves possibles entrades.
- **Dimensió**: Conjunt d'atributs d'una taula de fets que tenen una relació semàntica jeràrquica.
- **Factor de selecció**: Proporció de tuples necessàries per a realitzar una consulta respecte al total d'entrades d'una taula. La fórmula és la següent:

$$\frac{tuples\ necessàries}{total\ tuples}$$

- **Garbage Collector**: Gestor automàtic de la memòria dels programes. El seu objectiu és alliberar de la memòria aquelles variables o regions ocupades pels programes que ja no siguin utilitzades.
- **Gestor de bases de dades relacional**: Gestor de bases de dades que pot suportar models de dades relacionals (vegeu *model de dades relacional* al Glossari).
- **Localitat espacial**: Proximitat de les dades que ens interessen en el disc. Diem que ens beneficiem de la localitat espacial quan la proporció $\frac{infomació\ requerida}{blocs\ de\ disc\ llegits}$ és elevada.
- **Localitat temporal**: Proximitat entre les peticions de dades per tal que el bloc de disc que conté la informació que ens interessa no s'hagi tret de la memòria cau o de la memòria principal.

- **Memòria cau (*cache*)**: Memòria ubicada dins de la CPU per tal de reduir la latència d'accés a les dades mantenint-les més properes al processador. La memòria cau és especialment útil quan ens podem beneficiar de la localitat espacial i temporal.
- **Model de dades relacional**: Esquema de taules típic en què una relació ve determinada per un identificador d'una entrada en una taula i per un conjunt d'atributs (les columnes). Va ser definit per Edgar F. Codd l'any 1969.
- **Nom de xarxa**: Nom que té un equip dins d'una xarxa. Es pot fer servir com a substitut de la direcció IP per tal de fer més amigable la connexió entre equips.
- **Scale Factor**: Mesura que té el test del TPC-H per indicar la mida de la prova. Un Scale Factor de 1 vol dir que un document pla que contingui la dades normalitzades ocuparà aproximadament 1 GB; si el Scale Factor és 2, la mida aproximada serà de 2 GB. Per a més informació, vegeu el capítol 12 que parla sobre el TPC-H.
- **Secure Shell**: Protocol de xarxa que s'estableix entre dos dispositius de xarxa i que permet la transmissió de dades per un canal segur.
- **SGBD**: Sigla de Sistema Gestor de Bases de Dades.
- **Taula de fets**: Taula que conté el conjunt de mesures que volem estudiar. Les mesures estan identificades per diferents valors anomenats dimensions.
- **TPC-H**: Conjunt de proves, dades i consultes, per a mesurar el rendiment de sistemes de suport de decisions.
- **Tupla**: Cadascuna de les entrades d'una taula, també anomenades files.

16 Bibliografia

16.1 Articles

[1] Cahng, Fay; *et al*. "Bigtable: A Distributed Storage System for Structured Data". *OSDI,* 2006. URL: <http://labs.google.com/papers/bigtable-osdi06.pdf>.

[2] Dean, Jeffrey; Ghemawat, Sanjay. "MapReduce: Simplified Data Processing on Large Clusters". *OSDI*, 2004. URL: <http://labs.google.com/papers/mapreduce-osdi04.pdf>.

[3] Jiang, Dawei; *et al*. "The Performance of MapReduce: An In-depth Study". *PVLDB*, 2010. URL: <http://www.comp.nus.edu.sg/%7Eepic/vldbexp844.pdf>.

16.2 Llibres

[4] Apache Software Foundation. *The Apache HBase Book*. Última revisió 5 d'octubre del 2010. URL: <http://hbase.apache.org/book.html>.

16.3 Manuals

[5] Transaction Processing Performance Council. *TPC BENCHMARK™ H: Standard Specification*. San Francisco: TPC, 2010. Revisió 2.13.0. URL: <http://www.tpc.org/tpch/spec/tpch2.13.0.pdf>.

16.4 Pàgines web

[6] Pàgina principal de Hadoop: http://hadoop.apache.org/

[7] Pàgina principal de Hadoop MapReduce: http://hadoop.apache.org/mapreduce/

[8] Pàgina principal d'HBase: http://hbase.apache.org/

[9] Pàgina principal d'HDFS: http://hadoop.apache.org/hdfs/

[10] Pàgina Wiki de Hadoop: http://wiki.apache.org/hadoop/

[11] Pàgina Wiki de Hadoop MapReduce: http://wiki.apache.org/hadoop/MapReduce

[12] Pàgina Wiki de Hadoop sobre AmazonEC2: http://wiki.apache.org/hadoop/AmazonEC2

[13] Pàgina d'instal·lació d'HBase: http://hbase.apache.org/quickstart.html

[14] Arquitectura de l'HDFS: http://hadoop.apache.org/hdfs/docs/current/hdfs_design.html

[15] Pàgina de TPCH: http://www.tpc.org/tpch/

[16] Agile Manifesto: http://agilemanifesto.org/

[17] Pàgina Wiki sobre Agile Software Development: http://en.wikipedia.org/wiki/Agile_software_development

[18] Pàgina d'instal·lació de l'HBase: http://hbase.apache.org/notsoquick.html

[19] Pàgina web on s'expliquen les diferències entre el BigTable de Google i l'HBase de Hadoop: http://www.larsgeorge.com/2009/11/hbase-vs-bigtable-comparison.html

[20] Shell d'HDFS: http://hadoop.apache.org/common/docs/r0.17.2/hdfs_shell.html

[21] Manual de Yahoo! sobre Hadoop: http://developer.yahoo.com/hadoop/tutorial/

16.5 Altres recursos

[22] Pàgina web de l'Enciclopèdia catalana: http://diccionari.cat/

[23] Cercador en línia de Google: http://www.google.com

[24] Viquipèdia: http://www.wikipedia.org/

17 Annex I: Taula de proves

A continuació mostro la taula que he fet servir per comprovar el correcte funcionament dels algorismes (apartat 13.1):

Id	*Name*	*Surname*	*Gender*	*Salary*	*Section*	*Level*	*Job*	*Year*	*Month*	*Day*	*Expenses*
1	Luke	Skywalker	male	4000	Direction	5	Director	1948	3	11	300
2	Obi-Wan	Kenobi	male	3500	Direction	4	Administrator	1948	12	24	400
3	Padmé	Amidala	female	3000	Development	5	Programmer	1987	4	5	10
4	Yoda		male	6000	Human-Resources	3	Master	1895	12	12	1900
5	Palpatine	Sidious	male	1500	Human-Resources	3	Senator	1895	11	11	400
6	R2	D2	unknown	15	Development	1	Programmer	2000	1	1	0
7	C	3PO	unknown	15	Development	1	Programmer	2000	1	1	0
8	Qui-Gon	Jinn	male	2000	Development	2	Designer	1960	1	4	200
9	Mace	Windu	male	1000	Development	2	Designer	1960	1	4	300
10	Jar-Jar	Binks	male	0	Direction	1	Public-Relations	1970	4	2	10000

18 Annex II: Manual d'usuari

En aquest annex explico la instal·lació dels diferents serveis necessaris per a l'execució del projecte i, seguidament, el funcionament dels diferents programes desenvolupats.

18.1 Instal·lació dels serveis

18.1.1 Hadoop MapReduce i HDFS

En aquest apartat explico els passos bàsics que s'han de seguir per instal·lar Hadoop MapReduce 0.20.2 en un clúster. S'aconsella fer-ne la instal·lació sobre Linux. Per tal de poder fer la instal·lació de Hadoop en un clúster requereix que:

- Tots els nodes tinguin instal·lat el JDK6[15].
- Tots els nodes tinguin un usuari en comú i que aquest usuari tingui suficients permisos com per executar tots els programes que instal·larem.
- Tots els nodes tinguin SSH[16] i RSync instal·lat.
- Tots els nodes es puguin connectar a través de SSH a tots els nodes del clúster (inclòs el mateix node) usant el compte de l'usuari comú sense que demani contrasenya.
- Tots els nodes del clúster han de tenir exactament la mateixa configuració, fins i tot, el directori d'instal·lació del Hadoop.

Ara explicaré la instal·lació de Hadoop 0.20.2 que inclou el MapReduce i l'HDFS[17] que va dins del mateix paquet. Jo l'he fet concretament en la versió d'Ubuntu 10.04 LTS. Per realitzar la instal·lació de Hadoop s'ha de fer el següent:

- Descarregar la versió estable de la pàgina web oficial de Hadoop [6].
- Descomprimir l'arxiu i moure'l al directori on es vulgui deixar instal·lat. D'aquesta carpeta en direm $HADOOP_HOME.

15 Java Development Kit 6
16 Secure Shell
17 Veure l'apartat Apache HDFS(6.1)

- Configurar l'entorn Java. Ho farem editant l'arxiu $HADOOP_HOME/conf/hadoop-env.sh. En aquest arxiu s'ha de configurar el Java amb la variable $JAVA_HOME. Per exemple, posarem:

  export JAVA_HOME=/usr/lib/jvm/java-6-sun

  que és la directori on s'ha fet la instal·lació de Java en el sistema.

- Configurar el funcionament de l'HDFS: S'ha d'afegir el següent contingut a l'arxiu $HADOOP_HOME/conf/hdfs-site.xml:

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
</configuration>
```

  On la propietat *dfs.replication* és el nombre de cops que es replicarà l'arxiu dins del clúster. Com més gran sigui el seu valor, més gran serà la seva disponibilitat però al seu torn més espai ocuparà l'arxiu.

- Indicar qui és el màster del clúster: S'ha de modificar l'arxiu $HADOOP_HOME/conf/mapred-site.xml i inserir-hi el següent contingut:

```
<configuration>
     <property>
         <name>mapred.job.tracker</name>
         <value>@servidor:9001</value>
     </property>
</configuration>
```

  On la propietat *mapred.job.tracker* és el nom del màster del clúster que dirigirà l'execució de els tasques. S'ha de substituir la direcció del servidor (*@servidor)* pel nom de l'ordinador dins de la xarxa o bé per la seva direcció IP.

- Indicar quin serà l'ordinador que controlarà i dirigirà l'HDFS: la següent configuració s'ha d'inserir a l'arxiu $HADOOP_HOME/conf/core-site.xml:

```
<configuration>
     <property>
         <name>fs.default.name</name>
         <value>hdfs://@servidor:9000</value>
```

```
    </property>
</configuration>
```

La propietat fs.default.name és l'adreça IP o nom de xarxa del màster de l'HDFS que el gestionarà i dirigirà les operacions sobre el sistema d'arxius distribuït. Cal fer notar, que en un entorn amb múltiples nodes, el màster de l'HDFS pot ser diferent del màster del JobTracker (definit en l'opció anterior).

Ara falta realitzar la configuració del clúster. A continuació explico les configuracions que falten per configurar el rol de cada node:

- Indicar els JobTrackers: s'ha d'introduir el nom de xarxa o la IP dels màsters del clúster a l'arxiu $HADOOP_HOME/conf/masters. S'ha d'escriure un nom o una IP per línia. Només un màster dirigirà les tasques, però en cas que el primer caigués, el segon supliria el primer. A més a més, tots els nodes especificats a l'arxiu es van sincronitzant per tal que no es perdi l'execució de les tasques que s'estan executant en cas que un d'ells caigui.
- Indicar els treballadors del clúster: s'ha d'introduir el nom de xarxa o la direcció IP dels nodes del clúster a l'arxiu $HADOOP_HOME/conf/slaves. Cada nom o IP s'ha d'escriure en una nova línia.

Aquesta, és la configuració bàsica, però a l'apartat 11.1.1, explico els canvis que hi he hagut de fer per solucionar els diferents problemes que m'han aparegut al llarg del desenvolupament del projecte.

18.1.1.1 Arrancada del servei

Un cop acabada la configuració, ja es pot iniciar el servei. Primer de tot s'ha de formatejar el sistema de fitxers de l'HDFS, es fa amb la comanda següent:

```
$HADOOP_HOME/bin/hadoop namenode -format
```

On *nodename* és el nom del sistema de fitxers que volem crear.

Finalment, l'únic que ens falta fer és iniciar els serveis. Els activarem tots amb la comanda:

```
$HADOOP_HOME/bin/start-all.sh
```

És possible que els nodes tardin un cert temps a configurar-se (al voltant d'un minut).

18.1.2 HBase

Ara explicaré el procés que s'ha de seguir per fer la instal·lació bàsica d'HBase. Per a poder fer la instal·lació d'HBase en un clúster es requereix haver fer prèviament la instal·lació de Hadoop explicada a l'apartat anterior. Un cop instal·lat Hadoop, els passos que s'han de seguir són els següents:

- Descarregar la versió estable de la pàgina web oficial de l'HBase [8].
- Descomprimir l'arxiu i moure'l al directori on es vulgui deixar instal·lat. D'aquesta carpeta en direm $HBASE_HOME.
- Especificar on volem guardar els arxius de l'HBase dins de l'HDFS: a l'arxiu $HBASE_HOME/conf/hbase-site.xml s'hi ha d'afegir:

  ```
  <property>
      <name>hbase.rootdir</name>
      <value>hdfs://@servidor:9000/hbase</value>
  </property>
  ```

  On *@servidor* és la direcció IP o el nom de xarxa del màster de l'HDFS definit en la configuració de Hadoop a l'apartat anterior.

- Indicar quin ordinador del clúster serà el màster d'HBase (HBaseMaster): s'ha d'afegir la configuració següent a l'arxiu $HBASE_HOME/conf/hbase-site.xml:

  ```
  <property>
      <name>hbase.master</name>
      <value>@servidor:60000</value>
  </property>
  ```

  S'ha de canviar el *@servidor* per la direcció IP o el nom de xarxa del node el qual volem que faci de màster d'HBase.

- Configurar el ZooKeeper: Per defecte HBase gestiona el ZooKeeper. El ZooKeeper és un servei que permet mantenir la configuració del clúster de forma centralitzada. Per a fer-ne la configuració a través de l'HBase hem d'afegir la configuració següent a l'arxiu $HBASE_HOME/conf/hbase-site.xml:

```
<property>
    <name>hbase.zookeeper.quorum</name>
    <value>@node1,@node2,...,@noden</value>
</property>
```

On *@node$_i$* és la direcció IP o bé el nom de xarxa del node i-èssim del clúster. En aquesta llista han d'aparèixer tots els nodes del clúster que han de poder fer servir l'HBase.

- Indicar si el servei d'HBase és distribuït: és possible que tot i disposar d'un clúster, només un dels nodes gestioni l'HBase. Per indicar que volem que sigui més d'un node, s'ha d'afegir la configuració següent a l'arxiu $HBASE_HOME/conf/hbase-site.xml:

```
<property>
    <name>hbase.cluster.distributed</name>
    <value>true</value>
</property>
```

En aquest exemple s'ha indicat com a certa la configuració distribuïda de l'HBase, en cas de no voler que sigui així, hauríem de substituir el *true* per un *false*.

- Indicar els HRegionServers: s'ha d'introduir el conjunt de direccions IP o noms de xarxa d'aquells nodes del clúster que volem que sigui HRegionServers. Se n'ha d'escriure un per línia a l'arxiu: $HBASE_HOME/conf/regionservers.
- Configuració de l'HDFS: HBase necessita accés a la configuració d'HDFS. Per tal d'evitar problemes d'actualització de la configuració, la forma més fàcil és crear un enllaç (*softlink*) de l'arxiu $HADOOP_HOME/conf/hdfs-site.xml a la ruta $HBASE_HOME/conf/hdfs-site.xml.

Tots els nodes del clúster que facin servir o gestionin l'HBase han de tenir la instal·lació feta a la mateixa carpeta i exactament la mateixa configuració.

18.1.2.1 Arrancada del servei

Un cop feta la configuració de l'HBase i tenint ja executant-se el Hadoop, s'ha d'executar la següent comanda per tal d'iniciar el servei:

$HBASE_HOME/bin/start-hbase.sh

Amb aquesta comanda s'iniciarà el ZooKeeper i després l'HBase.

18.2 Funcionament

En aquest apartat explicaré el funcionament dels diferents programes d'aquest projecte. Són els següents:

18.2.1 Programa principal

El programa principal és el programa que implementa els algorismes de creació d'índexs i consulta de cubs de dades que s'avaluen en aquest PFC. Té dues interfícies, una textual i una gràfica. A més a més, se li poden passar paràmetres a l'hora d'executar-lo per configurar-ne el funcionament. A continuació n'explico el seu funcionament:

18.2.1.1 Paràmetres

Tant en la versió del programa amb interfície gràfica com textual se li poden passar certs paràmetres per tal de tenir-lo ja configurat quan comenci l'execució. Els paràmetres amb què pot configurar són els següents:

- -IRA: Per a resoldre les consultes es farà servir l'algorisme de consulta IRA. No és compatible amb els paràmetres -IFS i -FSS . Aquesta opció està activada per defecte.
- -IFS: Per a resoldre les consultes es farà servir l'algorisme de consulta IFS. No és compatible amb els paràmetres -IRA i -FSS. Aquesta opció està desactivada per defecte.
- -FSS: Per a resoldre les consultes es farà servir l'algorisme de consulta FSS. No és compatible amb els paràmetres - IRA i -IFS. Aquesta opció està desactivada per defecte.
- -Tname *nom_taula*: Serveix per especificar el nom de la taula de l'HBase que es farà servir com a taula de fets (ja ha d'existir). Per defecte el seu valor és: *table*.
- -Dname *nom_taula*: Serveix per indicar el nom de la taula de l'HBase que volem fer servir per a desar-hi o obtenir-ne els index de dimensions per a aquesta execució. Si ja existeix, utilitza aquesta taula i sinó en crea una de nova amb el nom indicat quan sigui necessari. Per defecte el seu valor és: *dimension*.

- -TmpTable *nom_taula*: Aquesta taula serà creada i si existeix esborrada cada vegada que s'executi l'algorisme IRA. Aquesta és la taula intermèdia en la qual es guardaran els identificadors obtinguts dels índexs que s'han de cercar. Per defecte el seu valor és: *tmp*.
- -TmpFile *ruta_arxiu_hdfs*: Indica la ruta de l'arxiu, dins de l'HDFS, que es farà servir com a arxiu intermedi. Aquest arxiu serveix per desar els identificadors obtinguts dels índexs i que s'han de cercar a la taula de fets. En cas que l'arxiu existeixi, aquest serà esborrat. Per defecte el seu valor és: *tmpResult*.
- -LogPath *ruta_arxiu*: Serveix per indicar en quin arxiu es vol escriure el log de les operacions realitzades. Si l'arxiu no existeix, el crea i si ja existeix, continua escrivint al final de l'arxiu. Per defecte el seu valor és: *log*.
- -verboseLog: Amb aquesta opció els resultats de les estadístiques desats en els logs tindran una forma més entenedora indicant el significat de cada valor i les seves unitats. Aquesta opció no és compatible amb l'opció: *-NonVerboseLog*. Aquesta opció està activada per defecte.
- -NonVerboseLog: Utilitzant aquesta opció els resultats de les estadístiques que es desaran en els logs tindran una estructura més succinta. Aquests només contindran els valors numèrics separats per tabuladors i amb els resultats de cada execució en una línia diferent. Aquesta opció és incompatible amb l'opció: *-verboseLog.* Aquesta opció està desactivada per defecte.
- -PreserveIntermediateFile: Serveix per indicar que no volem que s'esborri l'arxiu de claus intermèdies després de cada execució de l'algorisme IFS. El seu propòsit és facilitar el depurat del projecte. En cas d'activar aquesta opció, l'arxiu intermedi s'haurà d'esborrar manualment després de cada consulta. Aquesta opció està desactivada per defecte.
- -C_NONE: Serveix per indicar que no volem utilitzar cap tipus de compressió en les taules creades durant l'execució del programa. Aquesta opció és incompatible amb l'opció -C_GZ. Per defecte aquesta opció està activada.

- -C_GZ: Serveix per indicar que volem utilitzar el mètode de compressió GZ en totes les taules creades durant l'execució del programa. Aquesta opció és incompatible amb l'opció -C_NONE. Per defecte aquesta opció està desactivada.
- -storeResultsToFile *path_file*: Serveix per indicar que volem que es desin els resultats de les consultes i al mateix temps indiquem en qui arxiu volem que es desin. Si l'arxiu no existeix el crearà i si ja existeix s'escriuran els resultats al final d'aquest. Per defecte aquesta opció està desactivada.
- -factsTableRowCount *int*: S'utilitza per indicar el nombre d'entrades que té la taula de fets. Així es pot evitar que es calculi aquest valor durant l'execució del programa. El valor ha de ser un nombre natural. S'utilitzarà aquest valor per calcular el factor de selecció de les consultes realitzades.
- -singleLevelIndex: Serveix per forçar l'ús dels índexs d'un sol nivell durant l'execució. Aquesta opció és incompatible amb -multipleLevelIndex. Per defecte està activada.
- -multipleLevelIndex: Serveix per forçar l'ús dels índexs multinivell durant l'execució. Aquesta opció és incompatible amb -singleLevelIndex. Per defecte està desactivada.
- -conditionFile *ruta_arxiu_hdfs*: S'utilitza per indicar la ruta de l'arxiu que es fa servir per passar-hi les entrades del índex que s'han de cercar per a realitzar la consulta. És la ruta del l'arxiu que es crearà i es farà servir d'entrada en els MapReduce dels algorismes d'obtenció de claus (apartat 7.2.1). Per defecte el seu valor és: *conditions*.

18.2.1.2 Interfície textual

La interfícies textual, ens permet realitzar operacions relacionades amb els índexs i també realitzar consultes. El seu funcionament és el següent:

Crear índexs

La creació dels índexs de dimensions es fa amb la comanda CREATE DIMENION que té la gramàtica següent:

CREATE DIMENSION *nom_dimensió* ***ATTRIBUTES*** *nom_columna*$^{+}$

El nom de la dimensió (*nom_dimensió*) que creem ha de ser únic, per tal d'evitar un conflicte de noms amb dimensions existents.

Els noms de les columnes (*nom_columna*$^+$) han d'estar ordenats pel nivell seu nivell d'agregació. Primer les que tenen un grau d'agregació més gran.

Executa els algorismes Índex 1-nivell (7.1.1) o bé Índex multinivell (7.1.2) segons la configuració que s'estigui fent servir.

Totes les columnes especificades a la consulta han de formar part de la taula de fets indicada en la configuració de l'execució.

Configurar índexs

Aquesta operació serveix per afegir la informació d'una dimensió ja existent a la taula dels índexs. La dimensió indicada ha estat creada en una altra execució i, per tant, el programa en la actual execució no en té informació (perquè el programa principal no es guarda la informació relativa als índexs). La gramàtica és la següent:

!CREATE DIMENSION *nom_dimensió* ***ATTRIBUTES*** *nom_columna*$^+$

Els paràmetres de d'aquesta instrucció tenen el mateix significat que en el cas del CREATE DIMENSION.

Aquesta operació només s'ha de fer servir en cas que el tipus d'índexs de l'execució actual sigui el mateix que amb el que hem creat l'índex. Tot i així, els algorismes d'un sol nivell poden treballar amb índexs multinivell.

Executar consultes

Les consultes sobre la taula de fets es realitzen amb la comanda SELECT que té la següent gramàtica:

SELECT *nom_columna*$^+$ ***WHERE*** *nom_dimensió = valor [:: nom_dimensió = valor]**

GROUP BY *nom_columna**

El nom de la dimensió, *nom_dimensió*, ha de ser d'una dimensió existent. El valor que fixem per a la dimensió ha de ser de la forma

$$v_1\%v_2\%...v_n\%$$

sent v_1, v_2, ..., v_n els valors que volem per cada nivell d'agregació v_i de la dimensió. Podem fer servir la forma següent si volem especificar els k primers valors de la dimensió i deixar la resta lliures (amb $k < n$):

$$v_1\%v_2\%...v_k\%All\%$$

Tots els valors han d'estar ordenats per ordre d'agregació. A més, totes les columnes especificades a la consulta han de formar part de la taula de fets especificada en la configuració de l'execució.

18.2.1.3 Interfície gràfica

La interfície gràfica d'aquest projecte ens permet crear índexs, realitzar consultes i inspeccionar-ne els resultats, configurar les opcions d'execució, generar el codi de les operacions executades fins al moment i executar comandes amb el mateix format que l'utilitzat en la interfície textual.

La finestra de la interfície gràfica té una estructura de pestanyes que permet el ràpid desplaçament entre les diferents funcionalitats i al mateix temps separar-les de forma molt visual tal com es pot apreciar a la figura 17.

Ara explicaré com funciona cadascuna de les pestanyes i finestres que conformen la interfície:

Creació de dimensions

La pestanya per a la creació de dimensions, és la pestanya que s'obre al iniciar l'aplicació i és la següent:

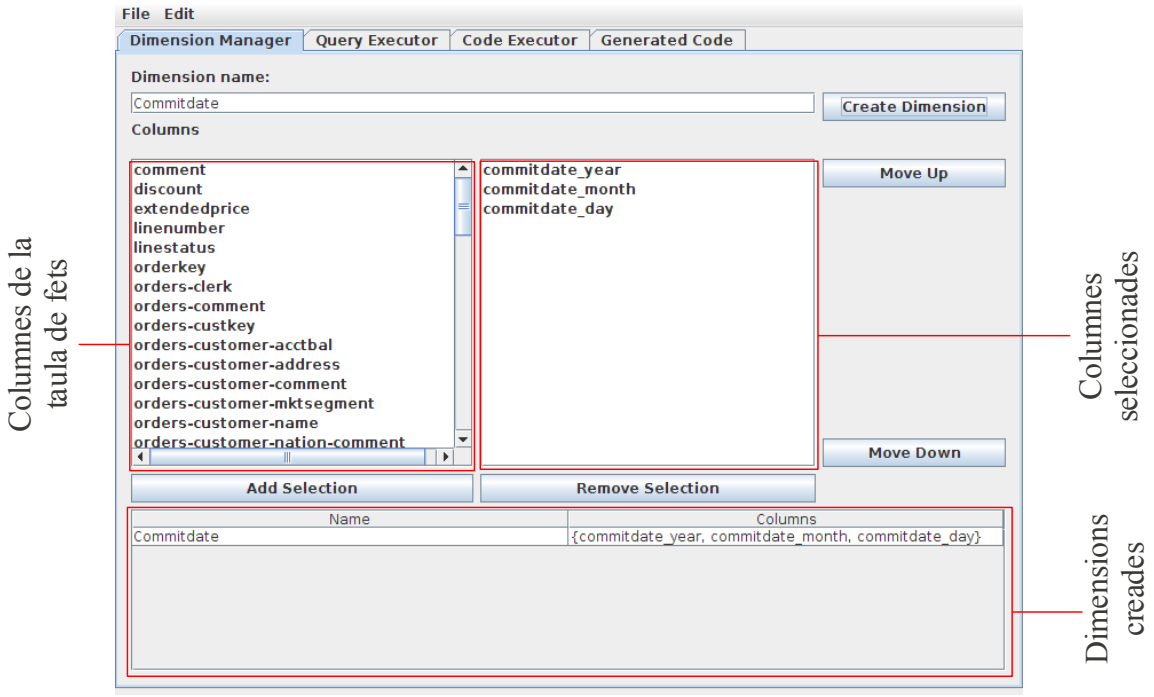


Figura 17: Pestanya de creació de dimensions

Podem veure que el contingut de la finestra és el següent: el camp *Dimension name* en el qual s'hi ha d'introduir el nom de la dimensió que volem crear. Hi ha, també, la llista de columnes de la taula de fets, de les quals en seleccionarem una per afegir-la a la dimensió i premerem el botó *Add Selection*. La columna seleccionada desapareixerà de la llista de columnes de la taula de fets i s'afegirà a la llista de columnes seleccionades. Si volem treure una columna de la llista de les columnes seleccionades, l'escollirem i polsarem el botó *Remove Selection*.

El nivell d'agregació de les columnes dins de la dimensió que crearem s'especifica ordenant els noms de les columnes. Com més amunt sigui el nom més gran serà el nivell d'agregació. Modificarem l'ordre d'agregació de les columnes escollint-les de la llista de columnes seleccionades i movent-les amunt prement el botó *Move Up* o bé avall amb el botó *Move Down*.

Un com hem configurat i definit el nou índex per crear, premerem el botó *Create Dimension*. El sistema crearà la nova dimensió i l'afegirà a la llista de dimensions creades especificant-ne el nom i les columnes que la componen.

Execució de consultes

Aquesta pestanya serveix per fer consultes sobre la taula de fets fent servir els índexs que s'han creat utilitzant la pestanya de crear dimensions:

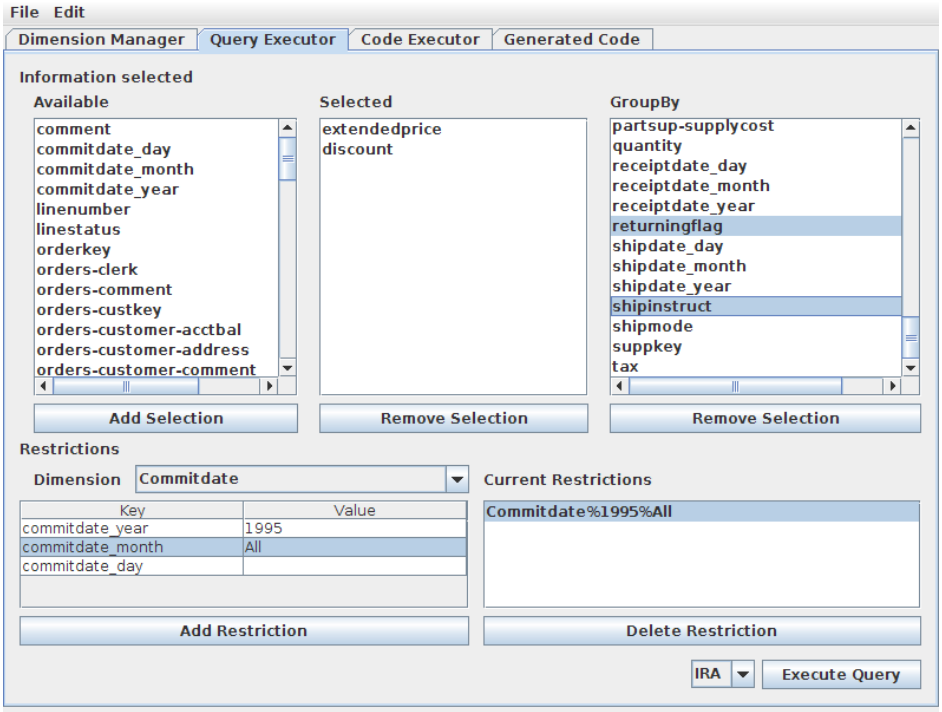


Figura 18: Pestanya d'execució de consultes

La pestanya d'execució de consultes, consta de diferents llistes. La llista de columnes disponibles (*Available*) que inicialment conté la llista de columnes de la taula de fets i serveix per afegir columnes a seleccionar per a la consulta. Per fer-ho, seleccionarem la columna que volem obtenir en la consulta i l'afegirem a la llista *Selected* prement el botó *Add Selection* i s'eliminarà de la llista *Available*. Per eliminar una columna de la llista de seleccionades, l'escollirem i polsarem el botó *Remove Selection*.

Per afegir les restriccions a la consulta, ho farem seleccionant del desplegable *Dimension* el nom d'una de les dimensions que hem creat, i emplenarem la taula que apareix a sota els valors que volem per a cadascun dels nivells d'agregació. En cas de no voler especificar un valor per a tots els nivells d'agregació, especificarem un valor pels primers i llavors el valor *All* pel següent tal com es mostra a la figura 18.

Especificarem les columnes pels valors dels quals volem fer l'agrupació dels resultats utilitzant la llista *GroupBy*. Aquesta llista conté el nom de totes les columnes de la taula de fets i

seleccionarem les columnes que ens interessin. Per seleccionar múltiples columnes farem servir la tecla *Control*. Si volem esborrar la selecció de columnes realitzada premerem el botó *Remove Selection.*

Un cop configurada la consulta a realitzar, seleccionarem l'algorisme que volem utilitzar del desplegable que hi ha al costat del botó *Execute Query*. Podem seleccionar entre els diferents algorismes: IRA, IFS i FSS.

Finalment, premerem el botó *Execute Query* per executar l'algorisme. Un cop executada la consulta, es mostrarà la pestanya de visualització de resultats amb el contingut de la informació de la taula resultant.

Visualització de resultats

Aquesta pestanya ens mostra els resultats obtinguts al realitzar l'execució de la consulta. Té la forma següent:

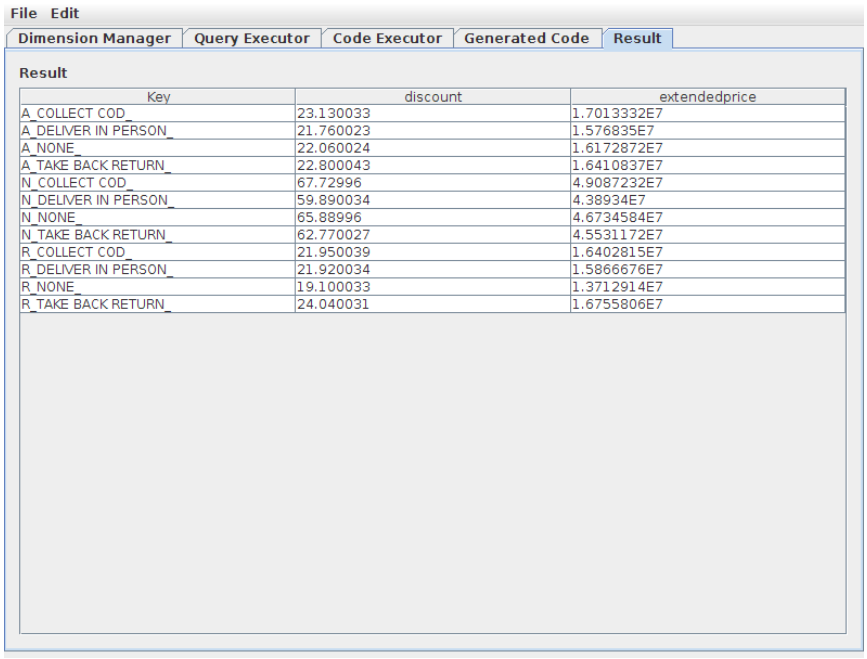


Figura 19: Pestanya de resultats

Aquesta pestanya conté una taula en la qual es mostra la informació resultant de l'última consulta executada. La primera columna contindrà la clau de cadascuna de les files, que al seu torn és cadascun dels valors de les columnes indicades en *GroupBy* separats per un guió baix

'_'. La resta de columnes contenen els valors seleccionats. Hi apareix una columna per cada atribut de la taula.

Codi generat

Aquesta pestanya ens mostra el codi per a la interfície textual equivalent a les operacions executades fins al moment amb la interfície gràfica. És la següent:

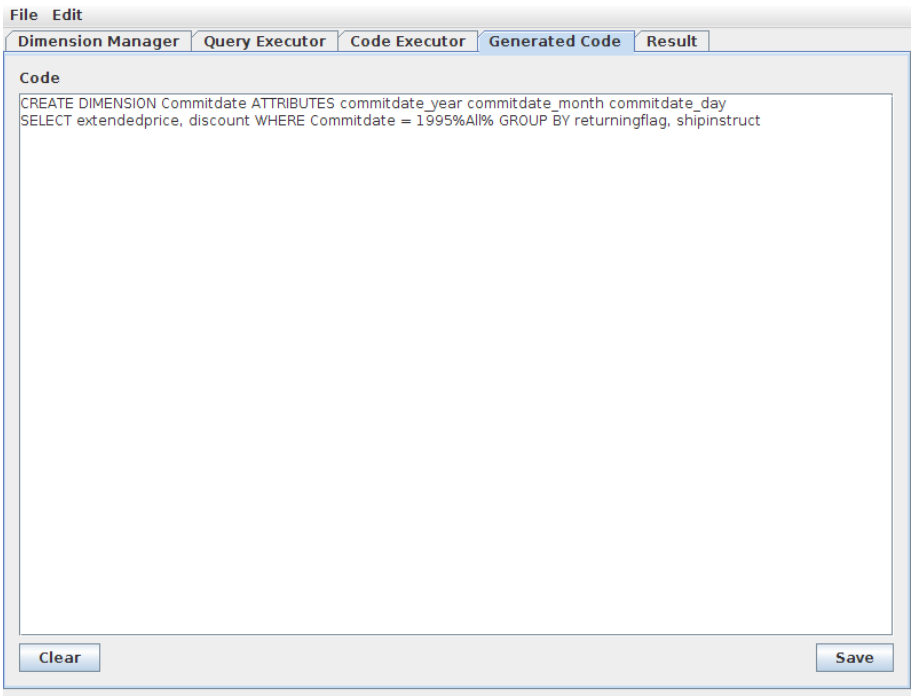


Figura 20: Pestanya del codi generat

Aquesta pestanya, a més de mostrar el codi de les operacions realitzades té dos botons. L'un és el botó *Clear* per a esborrar el tot el codi generat fins el moment i l'altre és el botó *Save* per desar el codi en el fitxer que se li indiqui.

Executor de codi

La pestanya de l'executor de codi ens permet executar les instruccions que hem vist a l'Interfície textual (apartat 18.2.1.2). L'estructura d'aquesta pestanya és la següent:

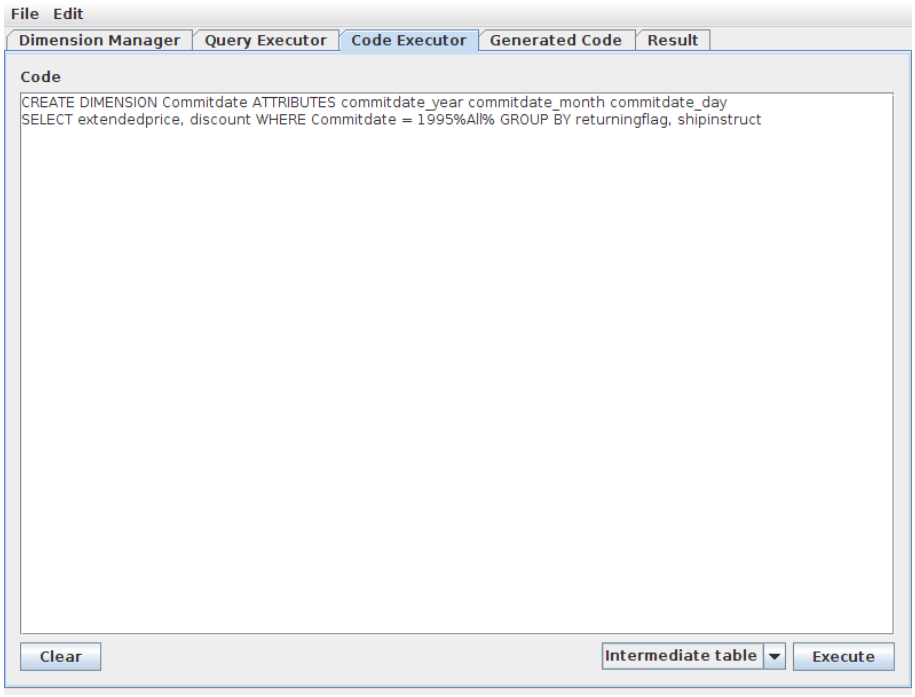


Figura 21: Pestanya de l'executor de comandes

Podem veure que la pestanya consta d'un quadre de text en el qual introduirem el codi que volem executar. El botó *Clear* serveix per netejat el contingut del quadre de text. També disposem d'un desplegable que ens permet escollir l'algorisme que volem fer servir per executar el codi i d'un botó, *Execute,* que iniciarà l'execució de totes les comandes introduïdes.

Finestra d'opcions

La finestra d'opcions ens permet canviar la configuració del programa en plena execució. Es pot modificar el valor de les opcions, definides o no al moment d'invocar el programa, d'una forma còmode i amigable.

Dins de la finestra de configuració les diferents opcions estan repartides en varies pestanyes. Les pestanyes que tenim són les següents:

Opcions generals

Aquesta pestanya ens permet canviar la taula de fets, la taula de dimensions (taula d'índexs), el tipus de compressió emprat i si utilitzarem índexs d'un nivell o multinivell. L'estructura d'aquesta pestanya és la següent:

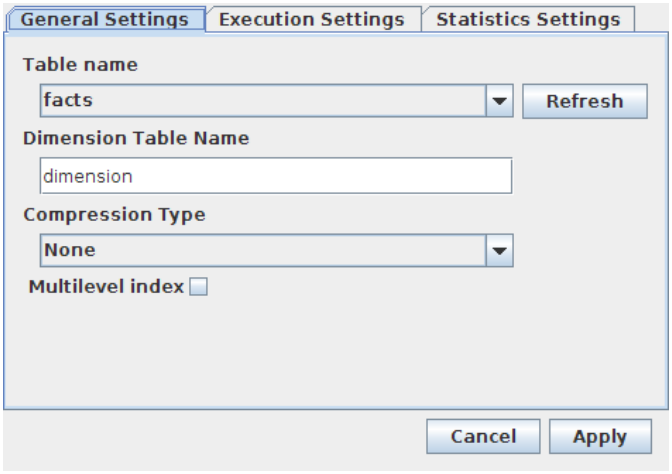


Figura 22: Configuració - Opcions generals

Podem veure que consta d'una llista desplegable en la qual hi apareix el nom de totes les taules de l'HBase. Amb aquest desplegable podem canviar la taula que fem servir com a taula de fets. També hi ha el botó *Refresh* que ens permet recarregar la llista de taules que hi ha. També trobem el camp *Dimension Table Name* en el qual introduirem el nom de la taula de dimensions que volem fer servir. Del desplegable anomenat *Compression Type* escollirem el tipus de compressió que volem fet servir GZ o bé None (sense compressió). Finalment podem activar l'ús dels índexs multinivell amb l'opció *Multilevel index*, en cas de no estar marcada es farà servir l'índex d'un sol nivell.

Opcions d'execució

Aquesta pestanya serveix per configurar diferents opcions de l'execució de les consultes. Té l'estructura següent:

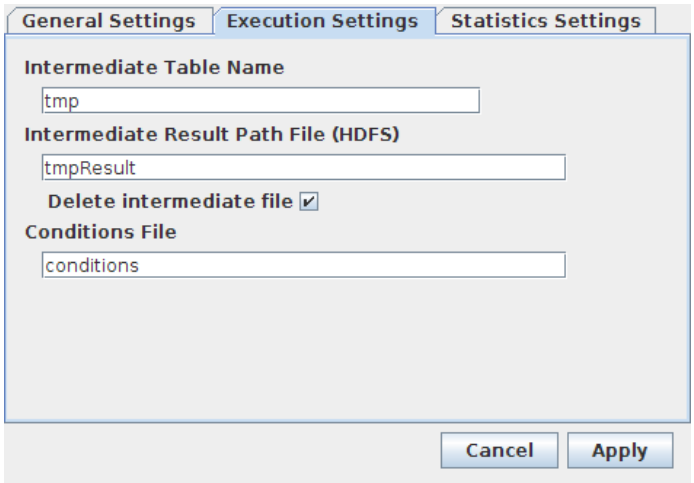


Figura 23: Configuració – Opcions d'execució

Podem veure que consta dels camps *Intermediate Table Name* en el qual podem modificar el nom de la taula intermèdia utilitzada en l'algorisme IRA. També podem modificar la ruta de l'arxiu intermedi utilitzat en l'algorisme IFS i activar-ne o no, al seu torn, el seu esborrat. Finalment també podem modificar la ruta de l'arxiu de condicions que es fa servir per passar les entrades de l'índex que ens interessen al primer MapReduce dels algorismes IRA i IFS.

Opcions d'estadístiques

Aquesta pestanya ens permet configurar diferents característiques referents a la generació de les estadístiques. Té el següent aspecte:

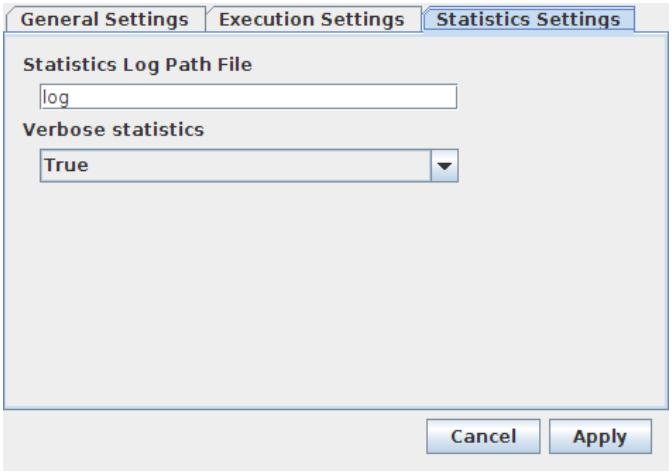


Figura 24: Configuració – Opcions d'estadístiques

Modificant el valor del camp *Statistics Log Path File* podem canviar la ruta de l'arxiu del log en què guardarem la informació de l'execució actual. També podem configurar les estadístiques per a què siguin molt explicatives posant *Verbose statistics* a cert. D'altra banda, si posem *Verbose statistics* a fals la informació serà més succinta.

18.2.2 Generador de taules del TPC-H

Aquest és el programa que genera les taules per carregar del TPC-H dins de l'HBase. El conjunt d'opcions que podem fer servir a l'hora d'invocar-lo és molt extens i el podem trobar al README proporcionat amb el generador de proves del TPC-H. El programa que s'ha d'invocar és el *dbgen* que requereix de l'arxiu de distribucions *dist.dss*. Les opcions que he utilitzat per invocar el generador són les següents:

- -v: Indica el progrés de generació de les dades.

- -f: Els fitxers existents seran sobreescrits.
- -s *scale_factor*: Serveix per especificar el Scale Factor de les dades generades. Un Scale Factor de 1 genera un arxiu per carregar d'una mida aproximada de 1GB. El factor d'escalat pot ser enter o decimal.

Per invocar el generador que he modificat s'ha de fer amb una comanda de l'estil següent:

./dbgen -vf -s scale_factor > tmp

En la qual s'ha de substituir el *scale_factor* per la mida que ens interessi, ja sigui un valor enter o decimal.

18.2.3 Carregador

Aquest programa serveix per carregar les dades creades pel generador del TPC-H a una taula de l'HBase. Totes les taules del TPC-H s'inseriran a la mateixa taula de l'HBase anomenada per defecte *supertable*. Cadascuna de les taules del TPC-H s'inserirà en una família de columnes diferent i cadascuna de les columnes de la taula del TPC-H anirà en una columna diferent dins de la família de columnes. Per executar aquest programa ho farem de la forma següent:

hadoop jar Loader.jar arxiu_taules

on l'*arxiu_taules* és la ruta de l'arxiu resultant del generador de taules del TPC-H. D'altra banda, si volem inserir les dades en una taula diferent, ho fer afegint un segon paràmetre a la invocació del programa de la forma següent:

hadoop jar Loader.jar arxiu_taules nom_taula

La taula indicada o bé la taula per defecte ja ha d'existir abans d'invocar el carregador.

18.2.4 Desnormalitzador

Aquest programa desnormalitza les taules del TPC-H contingudes en una única taula d'HBase en diferents famílies de columnes. On cada família de columnes té per nom el nom d'una de les taules del TPC-H i les columnes de cada família de columnes el nom de les columnes de l'HBase. La taula a la qual s'accedirà per fer-ne la desnormalització ha de tenir el nom de: *supertable*. Després de la desnormalització canviarà tots els identificadors de les files per tal que siguin consecutius.

A l'hora d'invocar-lo se li poden passar els següents paràmetres:

- -C_NONE: Serveix per indicar que no volem utilitzar cap tipus de compressió en les taules creades durant l'execució del programa. Aquesta opció és incompatible amb l'opció -C_GZ. Per defecte aquesta opció està activada.
- -C_GZ: Serveix per indicar que volem utilitzar el mètode de compressió GZ en totes les taules creades durant l'execució del programa. Aquesta opció no és compatible amb l'opció -C_NONE. Per defecte aquesta opció està desactivada.
- -NO_DELETE: Serveix per deshabilitar l'esborrat de les dades de la taula origen. Per defecte aquest opció està habilitada. També, cal dir que no és compatible amb l'opció -DELETE.
- -DELETE: Serveix per habilitar l'esborrat de les dades de la taula el més aviat possible de la taula origen. És a dir, esborrar una família de columnes quan no la necessitem més o, fins i tot, la taula que conté les dades quan ja no la necessitem. Aquesta opció està deshabilitada per defecte. Cal notar, també, que aquesta opció és incompatible amb l'opció -NO_DELETE.
- -VERBOSE: Amb aquesta opció indiquem que volem que la informació mostrada per la sortida estàndard del programa sigui explicativa. Aquesta opció està activada per defecte i és exclusiva amb l'opció -NON_VERBOSE.
- -NON_VERBOSE: Aquesta opció serveix per indicar que volem que ens mostri les estadístiques de l'execució del programa, per la sortida estàndard, d'una forma més succinta. Aquesta opció està desactivada per defecte i és exclusiva amb l'opció -VERBOSE.
- -it *num_iteració*: serveix per executar la join en la posició *num_iterció*[18] si *num_iteració* és menor o igual que 8. O bé, si és 9, executarem l'últim pas, que és la còpia de la taula desnormalitzada per tal de canviar els identificadors de les files.

Per invocar el desnormalitzador ho farem amb la comanda següent:

hadoop jar Desnormalitzador.jar [paràmetres]

18 Explico en què consisteix cadascuna de les joins en el capítol TPC-H.