Optimització del flux de recàrrega diària d'una base de dades

Ricard Tuneu Font

Resum — Aquest treball aborda l'optimització del flux de recàrrega diària d'una base de dades complexa, en el context d'una empresa amb un volum creixent d'informació emmagatzemada. Es plantegen diversos objectius, com identificar les dependències entre taules dins d'un llac de dades, optimitzar l'ordre de recàrrega per assegurar la coherència i la disponibilitat de dades, i desenvolupar eines que permetin visualitzar els resultats de l'optimització.

A més, s'ha desenvolupat una aplicació web amb Streamlit que ofereix visualitzacions clares del llinatge de dades, diagrames de flux i un simulador per provar els impactes dels canvis en els horaris de recàrrega. Els resultats han demostrat millores significatives en l'eficiència del procés i en la disponibilitat de dades diàries, reduint colls d'ampolla i augmentant la flexibilitat per adaptar-se a les necessitats de negoci. Aquest projecte no només optimitza el flux existent sinó que també proporciona a l'empresa una eina interactiva per a analitzar i implementar millores contínues.

Paraules clau — llac de dades, optimització, flux de recàrrega, dades, aplicació, temps de càrrega, taula, arquitectura, dependències, origen de dades, visualització, simulació, models, orquestrador, empresa, anàlisi.

1 Introducció - Context del treball

Aquest projecte sorgeix a través d'un problema real detectat en l'empresa Werfen S.A. Una dificultat provinent de l'augment exponencial d'informació que s'obté i s'emmagatzema diàriament en la base de dades.

L'empresa passa actualment per a la construcció d'una estructura d'emmagatzematge conformada de tres nivells de profunditat, on de forma directa o indirecta les taules es relacionen entre elles i, per tant, es generen dependències. A més la informació es divideix en diferents dominis representant diferents àrees de l'empresa com pot ser finances o recursos humans.

El projecte sorgeix en el moment en el qual aquesta estructura va creixent i on per a poder prendre decisions encertades és necessari tenir una actualització diària en la major part de les taules que componen el llac de dades. Antigament, el volum d'informació era molt petit i es va designar unes hores de recàrrega de les taules a ull, però avui dia amb la quantitat de dependències entre taules que hi ha es requereix un ordre precís en l'hora d'actualitzar la informació per a poder tenir les dades en el moment que es necessiti i que a la vegada no hi hagi inconsistència en les dades.

Així, doncs, a partir d'aquí es plantegen una sèrie d'objectius sobre els quals desenvolupar el treball.

- Trobar el llinatge entre taules, és a dir, les dependències que hi ha entre les diferents relacions de la base de dades.
- Generació de nous grups de taules per a la recàrrega optimitzada de dades en els diferents dominis.
- Crear una eina de visualització i simulació per a mostrar els resultats obtinguts.

A continuació en el document es mostren una sèrie de seccions en les quals es desenvolupa la solució proporcionada davant del problema esmentat i com a conseqüència la materialització dels objectius plantejats.

Cal remarcar també que la metodologia aplicada que s'ha seguit durant la implementació de la solució és Agile, és a dir, la realització del treball a partir de cicles curts i iteratius amb l'objectiu d'obtenir una progressió constant amb la possibilitat d'adaptar-se a canvis fàcilment.

2 Arquitectura del Sistema i Components Clau

2.1 Llac de dades

L'empresa com ja he explicat, està en el procés de la construcció d'un llac de dades on emmagatzema tota la informació que obté sigui quin sigui l'origen. Sovint provinent de SAP[1], però també de SharePoint entre d'altres.

Aquest llac de dades presenta una estructura definida en 3 nivells. El primer d'ells és "staging", és la fase inicial on s'insereixen les dades de forma pura a com es reben des de la font. Les taules d'aquest nivell deriven en taules de nivell 2, anomenat "intermediate"; en aquest nivell, es declara el tipus de dada que hi ha en cada columna i la longitud en cas de ser de tipus text a més de gestionar els "null" que hi puguin haver. També, a petició més en l'àmbit de "business" es pot aplicar certa lògica per a una finalitat concreta. Finalment, tenim els "marts"; taules generades a través de les "intermediate" i l'objectiu d'aquestes és poder ser llegides per eines de visualització i mostrar informació de valor. Aquí trobarem "marts" creats a partir de taules de nivell 2 i 3, és a dir de propis "marts" també.

A banda de tot aquest flux que es genera amb l'estructura de 3 nivells tenim que els "marts" es divideixen en diferents dominis tal com s'ha explicat anteriorment. Per a l'optimització serà un punt clau, ja que es tractarà cada domini per separat perquè les dades d'una mateixa classe interessa tenir-les actualitzades en el mateix moment.

Totes les taules que formen part de l'estructura la companyia les té emmagatzemades a Amazon Redshift[2]. Es tracta d'un núvol que et permet guardar dades i t'ofereix sobretot una alta escalabilitat, ideal per al creixement en volum de dades que té l'empresa, i també, ofereix una alta velocitat de consulta sobre la base de dades.

2.2 Models

Les dades com bé hem vist es troben emmagatzemades al núvol, però és interessant tenir també guardat el model que origina cadascuna de les taules del llac de dades.

Les taules es creen a través de sentències de SQL i aquestes es guarden en un repositori de Github que té l'empresa. És clau comptar amb aquesta informació per a poder trobar el llinatge entre taules, ja que a partir d'aquí podrem saber les dependències que hi hagi entre taules. A banda que els models a l'empresa li serveixen per tenir control de l'estructura i també és de gran valor comptar amb ells per a la detecció d'errors.

En el repositori trobem diferents carpetes segons el nivell d'estructura en el que es troba una taula i en el cas del nivell 3, els "marts", els tenim també dividits per dominis.

2.3 Orquestrador

Per últim, però peça clau per a la realització d'aquest treball és l'orquestrador. Aquest s'encarrega de gestionar, supervisar i automatitzar els fluxos de recàrrega de les diferents taules que conformen el llac de dades. En el cas de l'empresa l'eina que s'utilitza és Prefect[3]. Bàsicament el que es fa és assignar una hora de recàrrega diària a les taules i aquest instrument el que fa és actualitzar les dades de forma automàtica.

3 Obtenció i preparació de dades

3.1 Extracció llinatge

El primer dels objectius se centra principalment en el que s'exposa en aquest punt. Per a poder optimitzar el flux de càrrega dels diferents dominis de l'empresa necessitem saber quines taules de nivell 1 i 2 alimenten als dominis, ja que aquestes taules poden alimentar a més d'una àrea a la vegada i aquí, és on més endavant veurem que és un punt clau per a l'optimització.

Com bé s'ha explicat, tenim els models guardats en un repositori de Github amb sentències de SQL, per tant, el primer que hem de fer és connectar-nos a aquest repositori. Cal esmentar que l'algorisme implementat al llarg del projecte és fet a través de Python. Així doncs, el que farà primerament el codi és accedir al repositori a través de la llibreria "requests"[4]. Amb l'enllaç del repositori i les credencials atorgades per l'empresa obtenim la visió de tots els arxius que hi trobem dins, en el nostre cas ens interessen els models amb els quals s'han creat totes les taules del llac de dades.

A partir d'aquí la idea de l'algorisme és iterar primerament dins els dominis dels "marts", ja que en els models d'aquests tenim les referències de les taules que les alimenten, la referència la podem detectar usant una tècnica de cerca dins de la clàusula "FROM" del SQL, en el meu cas usant la llibreria "re" que el que fa és buscar patrons en el text que tu li passes, en el nostre cas, les sentencies SQL que representen els models de les taules.

Un cop tenim el mètode de búsqueda en funcionament, el que fem és iterar, és a dir, quan trobem la taula o taules que alimenten un model, aleshores busquem quines taules alimenten aquelles i iterem fins al punt que trobem una taula de nivell 1 o "staging", ja que aquestes fan de taula origen i, per tant, no tenen cap taula que les alimenta.

Tota aquesta informació l'emmagatzem en un "dataframe" on guardem el "mart" inicial en una columna, seguit de dues columnes més que són la "source" i la "destination". Com bé diu el nom, la taula que fa de "source" alimenta a la taula destí. Amb això el que aconseguim és en cada fila del "dataframe" mostrar una dependència i en el global, totes les dependències d'un únic mart.

Cal dir també que juntament amb les dependències, de cada taula guardem el nivell dins l'estructura, és a dir, si és "staging" o "intermediate" o en el cas dels "marts" guardem el domini al qual pertany, per exemple, finances.

Amb això ja tenim el llinatge del llac de dades guardat en un dataframe, el qual més endavant es realitza unes visualitzacions per a facilitar i agilitzar la lectura d'aquest.

3.2 Anàlisi inicial de l'estructura

Un cop tenim el llinatge, hem volgut fer un petit estudi inicial per veure la magnitud de l'estructura en la qual estem treballant i així poder entendre millor els resultats obtinguts al final del document.

Per saber com són els fluxos que té la base de dades, és a dir, les dependències que hi ha entre taules hem extret un petit rànquing de les cinc taules de nivell 3 que depenen de més taules.

	Mart	Dependency Tables
1	commercial_ops_main_customers	203
2	_deals	179
3	_service_contracts	178
4	snoa_top_generallog	161
5	_instrument_master	157

Imatge 1. Top 5 taules més dependents.

Com podem veure el la imatge 1, els cinc "marts" que tenim depenen de més de 150 altres taules, és a dir, aquestes taules abans de poder ser carregades se n'han de carregar moltes d'altres per a poder mantenir la consistència en les dades de les mateixes taules.

Tanmateix, cal veure si aquesta magnitud de dependències és habitual dins el llac de dades i per això hem extret un segon estudi que indica quantes taules tenen x dependències i el percentatge que això simbolitza dins el total de relacions que tenim.

	Dependency Tables	Frequency	Percentage
1	2	650	68.13
2	4	33	3.46
3	5	27	2.83
4	3	21	2.2
5	10	19	1.99

Imatge 2. Top 5 percentatges de taules amb x dependències.

Amb aquest estudi el que podem veure en la imatge 2 és que la major part de les taules, concretament el 68,13% depenen de només dues taules, cosa que fa sentit tenint en compte que tenim tres nivells en l'estructura i, per tant, això significa que el "mart" en qüestió s'alimenta únicament per una "staging" (nivell 1) i una "intermediate" (nivell 2).

Després de fer aquesta petita anàlisi ja podem continuar amb l'obtenció de la resta d'informació que necessitarem per a optimitzar el flux de recàrrega del llac de dades.

3.3 Temps de recàrrega

Un cop tenim el llinatge guardat, el que ens falta ara és informació respecte als temps de recàrrega de les taules, és a dir, quant tarda cada taula diàriament a actualitzar-se. Això ens és interessant per a poder marcar la disponibilitat que té una taula en carregar-se en funció de si les taules de les quals depèn ja s'han carregat o no.

Aquesta informació s'obté a través de Prefect, que com bé he dit és l'orquestrador que utilitza l'empresa per a gestionar el flux de càrrega del llac de dades. De totes maneres, aquesta informació queda guardada en una taula del "datalake" i, per tant, per extreure la informació no ens cal accedir al Prefect, sinó que podem fer-ho a través d'una consulta a l'Amazon Redshift.

La consulta que s'ha fet per a poder extreure la informació és la següent:

```
query = """SELECT table_name, PERCENTILE_CONT(0.5)
WITHIN GROUP (ORDER BY duration) AS median_dura-
tion
        FROM it.it_prefect_data
        GROUP BY table_name;"""
```

El que es pot apreciar en aquesta consulta és que s'obté la informació d'una taula del domini d'IT, en concret, de la taula "it_prefect_data" i el més interessant és que s'extreu la mediana del temps de cada taula. Ha estat decidit així, ja que de vegades falla la càrrega de les taules i no mostra temps reals i com a conseqüència la generació d'"outliers". D'aquesta manera, agafant la mediana evitem que aquests valors fora del rang habitual tinguin impacte sobre les nostres decisions.

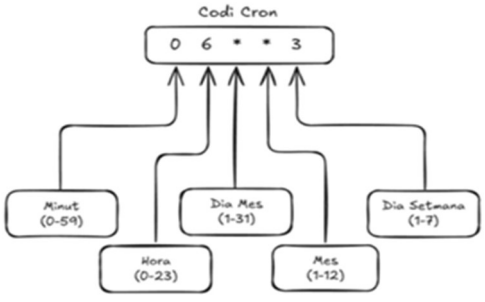
Així doncs, amb el que ens quedem és amb una taula de dues columnes, una corresponent al nom de la taula i l'altre el temps que tarda de mediana en actualitzar-se diàriament.

3.4 "Staging Schedules"

En aquest punt ve una de les parts clau per a poder desenvolupar l'algorisme d'optimització, necessitem saber a quina hora es carreguen les taules de nivell 1, ja que són les que ens marquen l'inici de l'algorisme i a la vegada la nostra limitació per al moment. En el punt següent a través d'un simulador es posa a prova l'efecte que pot tenir canviar el "schedule" d'aquestes taules inicials.

Per ara, però, el que volem és extreure l'hora de recàrrega de les taules i això ho podem trobar al Github, de la mateixa manera com ho hem fet per trobar el llinatge, ara l'objectiu és poder entrar als models de les taules de "staging" i llegir-ne el codi cron[5] que és el que ens indica a quina hora es recarrega aquella taula diàriament.

El "cron code" és la forma amb la qual Prefect interpreta quan ha de recarregar una taula, ja que hi ha moltes possibilitats en funció del dia, hora... i amb aquest codi que funciona de la manera que es pot veure en la imatge 3 el que farem és transformar-lo i quedar-nos amb l'hora diària a la qual es carrega, ja que és la informació que ens interessa per a l'optimització.



Imatge 3. Funcionament codi cron.

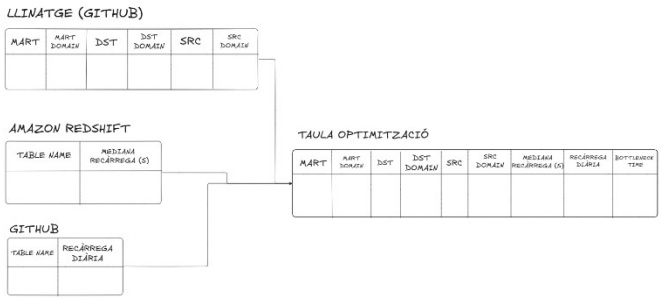
Així doncs, iterem per tots els models dins del repositori de Github i ens guardem també amb una taula el nom de la taula de nivell 1 i l'hora de recàrrega d'aquella taula posteriorment d'haver aconseguit el codi cron amb una tècnica de cerca a través de la llibreria "re" i d'haver transformat el codi en format HH:MM.

3.5 Creació de la taula d'optimització

A continuació, tota la informació recopilada l'hem d'ajuntar en un mateix arxiu per a facilitar l'algorisme i tenir una única font d'informació. Per tant, el que farem és concatenar en una taula el llinatge, el temps de recàrrega de les taules i per últim, l'hora de recàrrega de les taules de nivell 1.

Per ajuntar-ho tot farem "joins" a partir de la taula de llinatge i el que farem amb les dues altres taules és fer un join simple a partir de la columna "source" de la taula de llinatge amb la columna "table_name" de les altres dues taules.

En el següent esquema es pot veure com s'uneixen les 3 parts per a formar la taula que ens servirà per a l'optimització:



Imatge 4. Unió de les dades.

En l'esquema de la imatge 4 es pot observar també que la taula d'optimització té una última columna anomenada "bottleneck time", aquesta és una columna és la suma de l'hora de recàrrega de les taules de nivell 1 amb el temps que tarden a actualitzar-se, és a dir, aquesta columna ens indica a quina hora estan disponibles les taules del primer nivell.

Cal tenir en compte també que és possible que sobre alguna de les taules ens falti certa informació. Per exemple, en el cas que ens trobem que una taula de nivell 1 o 2 no tinguem dades registrades sobre el temps que tarda a actualitzar-se, aquesta relació serà eliminada de la nostra taula d'optimització, ja que no pot marcar els límits o coll d'ampolla. Per altra banda, també tenim una sèrie de taules que no es carreguen diàriament, per exemple, un cop al mes per motius de negoci de l'empresa. Aleshores, aquestes taules tampoc les tindrem en compte perquè el principal objectiu és centrar-nos en els problemes que deriven de la recàrrega diària de les dades.

En últim lloc, i més important és evitar taules de nivell 1 en les que no hi hagi una hora de recàrrega programada, ja que a partir d'elles s'inicia l'optimització i aquesta dada és vital per a poder prosseguir establint les hores de recàrrega de les taules que depenen d'ella.

De forma habitual, tindrem disponibilitat de totes les dades que necessitem, aplicant aquestes restriccions és una manera d'assegurar-nos que les dades que el nostre algorisme utilitzarà serà les que realment ens són d'interès per a abordar els objectius de la tesi.

4 APLICACIÓ STREAMLIT

A través del "dataframe" que hem construït hem decidit crear una aplicació web amb la llibreria Streamlit[6] de Python. L'objectiu d'aquesta aplicació és poder veure els resultats que s'obtinguin de l'optimització en forma de visualitzacions fàcils d'interpretar a més de poder realitzar diferents simulacions respecte al possible canvi d'hora de taules de nivell 1 i per últim com a eina simplement informativa per a facilitar la visió de les dades en els empleats de l'empresa.

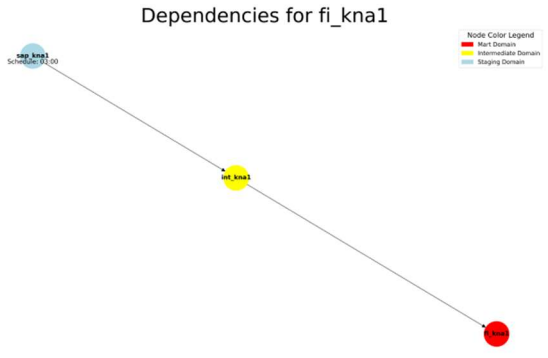
D'aquesta manera, l'aplicació s'ha dividit en tres apartats diferents que a continuació s'explica el seu desenvolupament.

4.1 Visualització llinatge

La primera part de l'aplicació el que tracta de fer és mostrar a partir de qualsevol "mart", és a dir, taula de nivell 3 un graf on es veuen totes les dependències que té. Per a fer això, primerament se'ns deixa filtrar per domini i seguidament pel "mart" que volem visualitzar.

Per a fer el graf s'ha utilitzat la llibreria "networkx" de Python. Aleshores com que l'aplicació llegeix el dataframe que s'ha extret amb tota la informació i que tenim dues columnes on es mostren les dependències que són la "source" i la "destination" el que fem és crear enllaços entre els nodes d'aquestes taules i de forma dirigida, és a dir, la font apuntarà al node destí amb l'objectiu que totes les fletxes s'acabin dirigint al "mart" que hem escollit a visualitzar.

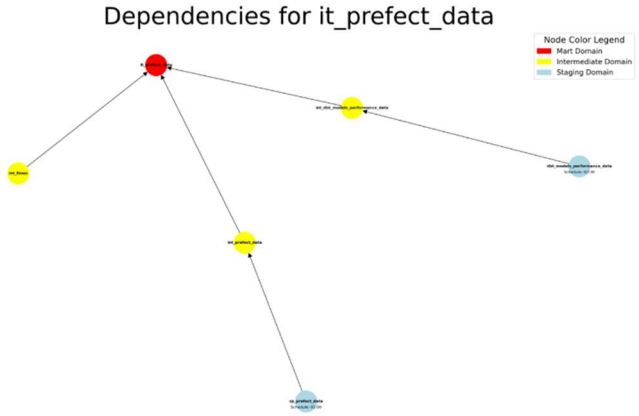
Aquí podem veure, en la figura 5, un exemple senzill de com es mostraria, en aquest cas per a la taula de nivell 3 "fi_kna1":



Imatge 5. Graf llinatge fi_kna1.

Com es pot veure en el dibuix, en aquest cas tenim que aquesta taula és de les que té tan sols dues dependències. Amb color blau clar representant la taula de nivell 1, on podem veure inclús l'hora de recàrrega; seguidament, el node groc que representa la taula de nivell 2 i ja per últim de color vermell la taula de nivell 3 per la qual hem filtrat.

També, però, podem tenir grafs més complexos tal com hem detectat en l'estudi inicial com és per exemple el graf de la taula "it_prefect_data" que és la taula d'on traiem la informació relacionada amb Prefect.



Imatge 6. Graf llinatge it_prefect_data.

La imatge 6 mostra que el "mart" depèn de tres taules de "intermediate" que són les de color groc i veiem també que n'hi ha dues de nivell 1 amb la seva respectiva hora de recàrrega. D'entrada tota taula s'origina d'una de nivell 1 per tant en la branca que no tenim node de color blau, no significa que no s'alimenti d'una "staging" sinó que el que significa és que o no tenim informació del "schedule" d'aquella taula o més probable que aquella taula es recarregui "on-demand".

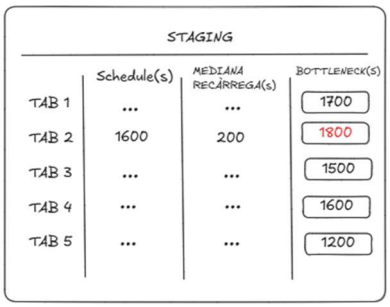
Així doncs, aquesta primera part de l'aplicació, a banda de mostrar-nos de forma informativa quines dependències té una taula i, per tant, ajudar-nos a comprendre de forma visual d'on provenen les dades, també ens ajuda a detectar anomalies com aquesta en la que s'hauria de mirar la taula de nivell 1 que falta per quin motiu no apareix i si convé poder-ho arreglar.

4.2 Optimització i simulador

La segona part de l'aplicació és on podrem optimitzar a través de l'algorisme creat, el flux de recàrrega diari de les dades. A banda d'això i que a continuació s'explicarà aquesta part també inclou un simulador per a poder canviar les hores de recàrrega de les taules de "staging" i veure com es comportaria l'optimització aplicant aquests canvis.

Com bé sabem, l'hora a la qual està marcada la recàrrega de les dades és el que ens marcarà l'inici de la nostra optimització per dominis. Així doncs, el primer pas del nostre algorisme serà separar la taula d'informació que hem obtingut per dominis i, per tant, cadascun d'ells s'optimitzarà per separat.

Un cop tenim els dominis per separat, de cadascun d'ells hem d'iterar per totes les taules de nivell 1 que tinguin influència en cadascun dels dominis i veure quina d'elles és la que tenim disponible més tard. Si recordem, en la taula d'informació recopilada hem calculat una columna que és la "bottleneck time" que està pensada expressament per això, ja que ens indica a quina hora tenim disponible cada taula. Aleshores simplement el que hem de fer és quedar-nos amb la taula que tenim disponibles les dades més tard i aquella serà la nostra taula limitant. Com que ho hem separat per dominis, en tindrem una per cada àrea de l'empresa.



Imatge 7. Exemple de taula limitant.

En aquest cas, podem veure com en un domini en concret la taula que ens limitaria és la segona, ja que l'hora a la qual està disponible és la més allunyada des de la mitjanit que és on comencem a contar el temps en segons.

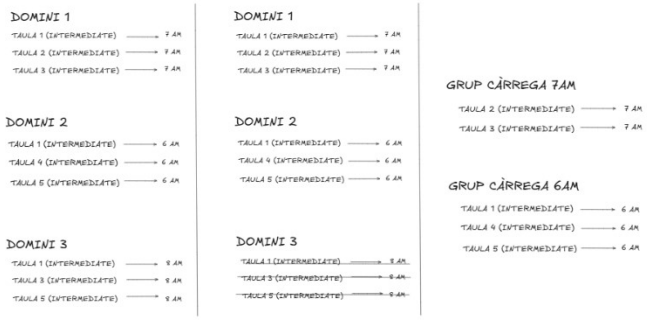
Així doncs, un cop tenim detectades totes les taules limitants, una per domini, ara és torn de col·locar la recàrrega de les "intermediates" immediatament després, cosa que és algo que no s'estava aplicant en l'empresa fins al moment, ja que com s'ha explicat en l'inici, les hores de recàrrega estaven programades a ull.

Ara ve una part interessant i és que des de l'empresa s'ha decidit realitzar una sèrie de grups de càrrega per a les taules de segon nivell. De la mateixa manera que hem fet amb les de primer nivell, de cada domini el que farem és detectar de quines taules d'"intermediate" en depèn. Aquí, però s'hi afegeix una casuística i és que en el llac de dades hi ha taules de segon nivell que alimenten a diferents dominis a la vegada, és a dir, per cada domini no hi ha un conjunt de taules de segon nivell que pertanyin únicament a aquell domini.

Per posar un exemple, una taula de segon nivell pot alimentar a la vegada un "mart" del domini de finances i una del domini de recursos humans.

Sabent això, l'algorisme d'optimització, a través de les taules que ens fan de coll d'ampolla que hem detectat

anteriorment, seguirà la idea que es pot veure en l'esquema de la imatge 8:



Imatge 8. Grups de càrrega.

Com podem apreciar, per elaborar els grups de càrrega tenim 2 passos previs a obtenir-los. En la part de més a l'esquerra de la imatge veiem 3 dominis amb tres taules d'"intermediate cadascun on l'hora de proposta de càrrega és a les 7 del matí en el primer domini, a les 6 en el segon i a les 8 en el tercer. Aquestes hores provenen de la taula limitant de cadascun dels dominis. És a dir, en el primer la taula limitant s'actualitzarà abans de les 7 i és per això que les taules de segon nivell posa que es poden carregar a aquesta hora.

Cal esmentar que d'ença que les dades d'una taula estan disponibles se suma 10 minuts i s'arrodoneix a quarts per assegurar que l'hora en què es carregarà la taula de nivell 2, prèviament ja s'hagi carregat l'última taula de nivell 1. Per posar un exemple, si la taula que fa de coll d'ampolla està disponible a les 6:10, aleshores l'hora de càrrega de les "intermediates" podrà iniciar a les 6:30 ja que és 6:10 + 10 minuts i arrodonir al quart següent, que és a les 6:30.

Un cop tenim aquest primer pas clar, cal passar al segon. Com bé ja hem dit anteriorment una taula de segon nivell pot alimentar a més d'un domini, però aquests dominis és possible que tinguin una hora de disponibilitat de càrrega diferent segons la seva "bottleneck table". En aquests casos i de forma totalment lògica la taula només farem que s'actualitzi un cop i ho farà en l'hora més avançada possible. En la imatge 8 si veiem totes les taules del domini 3 estan també en altres dominis i aquests la seva hora de recàrrega és prèvia, per tant, les taules del domini 3 es carregaran dins dels grups de càrrega d'altres dominis.

Com a resultat final si veiem hem obtingut dos grups de càrrega diferents, un a les 6 i un a les 7 del matí i si analitzem les taules que hi havia en l'últim domini, podem veure com les taules 1 i 5 es carregaran a les 6 en comptes de les 8 que era l'hora que es proposava en el domini i per últim la taula 3 es carregarà a les 7 en comptes de les 8.

Això si ho fem tenint en compte tots els dominis que tenim en el llac de dades acabarem aconseguint diferents grups de càrrega segons les taules limitants de nivell 1 i segons la redundància de les taules en diferents dominis.

Una vegada estem en aquest punt, ara ja només ens fa falta actualitzar les taules de nivell 3. Com és de forma lògica aquestes taules es podran carregar quan totes les taules del grup de càrrega del domini s'hagin actualitzat. Aquesta hora l'obtenim gràcies a l'hora proposada per la taula limitant de nivell 1 i ara el que hem de fer és detectar dins de cada grup de càrrega quina taula tarda més en actualitzar-se ja que aquesta serà la que ens limiti l'hora de càrrega dels "marts".

De la mateixa manera que hem fet anteriorment, ara el que farem és iterar dins els grups de càrrega i trobar la taula que més temps tarda a carregar, en la informació que hem recopilat en l'inici tenim les dades de la mediana de temps que tarden, així que simplement hem d'agafar l'hora de recàrrega de cadascun dels grups d'"intermediate" i sumar el temps que tarda en actualitzar-se la taula que tardi més.

A banda d'això, aquí també aplicarem la suma de 10 minuts i l'arrodoniment al següent quart per assegurar-nos que realment les dades en el segon nivell han estat actualitzades abans de carregar les definitives taules de tercer nivell.

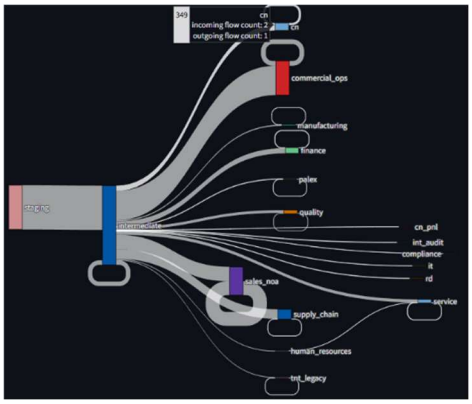
L'hora la qual obtinguem per domini després de tot aquest procés és l'hora optimitzada per a la recàrrega dels dominis.

Per acabar aquest punt, cal dir que en l'aplicació a banda d'optimitzar de forma natural amb les dades que tenim hi ha la possibilitat d'editar l'hora de recàrrega de taules de nivell 1 per veure l'impacte que això pot tenir en el flux de recàrrega general. En l'apartat de resultats es podrà analitzar i veure el funcionament de tot aquest algorisme.

4.3 Diagrames de Sankey

Abans de passar a veure resultats sobre la optimització, en l'aplicació hi ha un tercer apartat dedicat a un parell de diagrames de Sankey que de forma similar al llinatge ens donen una altra visió sobre l'estructura de la base de dades i ens pot permetre detectar errors com el que a continuació mostraré.

El primer dels diagrames, el de la imatge 9 mostra el flux que hi ha en l'estructura en relació els tres nivells que tenim, és a dir, mostra les dependències que hi ha entre cadascun d'aquests nivells.



Imatge 9. Estructura Sankey.

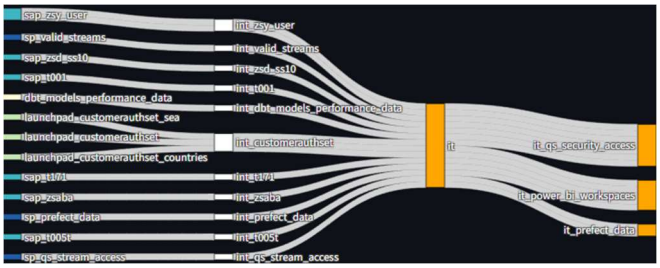
Tal com es pot apreciar en el diagrama tenim a l'esquerra el grup de nivell 1, després "intermediate" i seguidament tenim diferents ramificacions que són cadascun dels dominis de l'empresa. El gràfic a banda és interactiu i posant-se a sobre d'un node es pot veure les dependències que té, per exemple en la imatge veiem que el domini "cn" té 349 dependències amb taules d'"intermediate o amb taules del mateix domini, que ens indica que en té els "loops" que apareixen en cadascun dels nodes.

Aquest gràfic dona una visió de la importància o magnitud de dependències que tenen cadascun dels dominis respecte al total del llac de dades, però també ens serveix per a analitzar i detectar errors com el que es pot veure en la foto.

Si bé hem dit abans que una taula d'"intermediate" pot alimentar taules de diferents dominis, en l'empresa està totalment fora de les bones pràctiques que un "mart" d'un domini alimenti a un domini que no sigui el seu, és a dir, d'una altra àrea de l'empresa. Amb aquest diagrama s'ha detectat que "human resources" està alimentant al domini de "service" i, per tant, això és una cosa que s'ha de corregir com més aviat millor.

Per altra banda, tenim un segon diagrama de Sankey, similar a la visualització de llinatge, però aquest dona una visió des de les taules de primer nivell en comptes de tercer.

En aquest nou diagrama, el de la imatge 10, el que pots fer és filtrar per domini i escollir les taules de "staging" que vulguis mostrar.



Imatge 10. Primer nivell Sankey.

Com es pot apreciar, les taules de primer nivell tenen una paleta de colors per poder diferenciar l'origen de les dades ja sigui SAP, SharePoint, Launchpad entre d'altres.

A més a més, també podem veure les hores de recàrrega de les taules posant el ratolí a sobre i en els enllaços entre taules tenim el temps que tarden a carregar-se.

Així doncs, aquesta és l'última visió sobre l'estructura del llac de dades i que també, doncs ens permet poder fer anàlisis des d'una altra perspectiva.

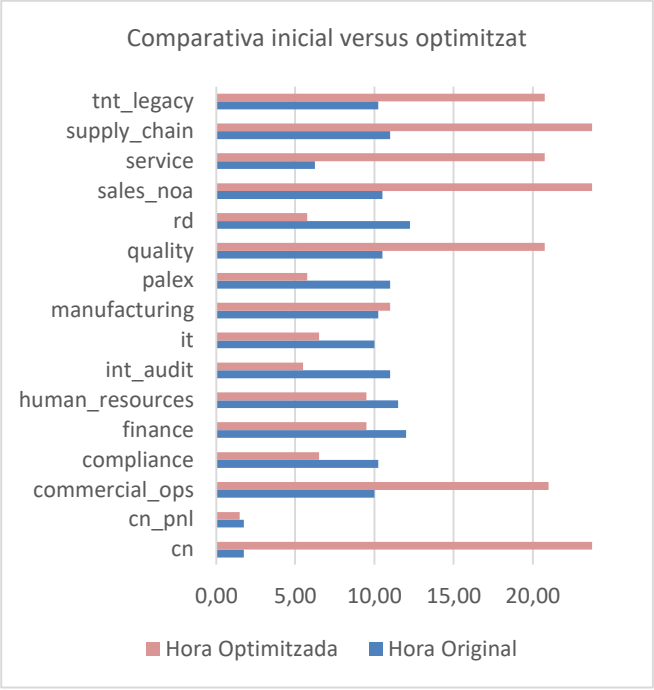
5 RESULTATS I AVALUACIÓ

Després d'haver exposat tota l'obtenció i preparació de dades a més de l'arquitectura amb la qual treballa l'empresa i haver mostrat una gran part de l'aplicació web creada, ara és moment realment de poder mostrar resultats i veure la utilitat de l'algorisme codificat.

5.1 Comparació foto inicial versus optimització

Com bé he dit, en l'aplicació web hi ha un apartat on és per optimitzar el flux de recàrrega diari de les dades, per tant, d'entrada el que volem veure és una comparació de les hores de recàrrega que tenien els dominis antigament en l'empresa contrastant amb les hores que l'algorisme ens indica.

Per veure-ho de forma senzilla, la següent gràfica, de la figura 11, mostra la comparativa entre la foto inicial del flux, és a dir, abans de l'optimització i les hores de recàrrega de dominis proposades després d'optimitzar.



Imatge 11. Comparativa inicial versus optimització.

Podem extreure vàries conclusions davant d'aquesta gràfica. D'entrada podem veure que hi ha diversos dominis que la línia blava que representa l'hora original de càrrega és més gran que la taronja, com en els casos de rd, palex, it... això és els resultats esperats en fer aquest gràfic, però s'han de tenir en compte dues possibilitats més que pugui fer que això no es doni.

El primer cas té relació amb els resultats del domini de "cn" on podem veure que l'hora original de l'optimitzada varia molt. Aquest en concret es deu al fet que cn és una àreea de la Xina dins l'empresa i en comptes de carregarse a la 1:45 indica que es pot carregar a les 23:45 hora espanyola, la qual és una bona notícia, ja que indica que les dades poden estar disponibles abans i en horari xinès vol dir que al matí a l'iniciar la jornada laboral ja tindran les dades carregades.

La resta d'optimitzacions que no milloren els temps inicials és degut al fet que en l'hora inicial en la qual es carreguen els dominis no s'estan tenint en compte totes les taules de les quals depèn aquell domini i per tant les dades diàries no estan ben actualitzades diàriament. Per posar un exemple el domini de "commercial_ops", l'hora optimitzada ens indica que es pot actualitzar a les 21h, això és degut al fet que hi haurà una taula limitant que es carrega una estona abans d'aquesta hora, i per això indica que podem carregar el domini a aquesta hora.

Si nosaltres carreguem el domini a les 10h tal com posa en l'hora inicial, ens estarem perdent informació diària al recarregar ja que hi hauran taules que encara no s'hauran recarregat aquell propi dia, és a dir, la informació d'avui podríem assegurar que la tenim demà. Això és una mala pràctica i aleshores cal detectar quines taules ens estan limitant i com podem millorar-ho per poder tenir les dades actualitzades molt més abans que a les 21h.

És per a aquests casos que ens és ideal l'eina del simulador que es presenta a continuació en el següent punt.

5.2 Simulació

Després d'haver vist la millora significativa amb l'optimització en alguns dominis on veiem que els podem carregar abans del que ho estaven fent fins ara, hem volgut anar un pas més enllà per intentar encara millorar més la disponibilitat de les dades o almenys poder oferir a l'empresa una eina per a testejar amb relació a les hores de recàrrega de les taules de primer nivell i veure quin possible impacte a futur podrien tenir certs canvis, sobretot per als dominis que hem vist que a causa d'alguna taula limitant no podem tenir les dades actualitzades en l'hora que ens agradaria, que és com més aviat millor.

Així doncs, la web presenta un apartat on podem aplicar aquests canvis d'hora en taules de primer nivell. Primer, però veurem que a l'optimitzar sense canvis apareixen 4 desplegables amb els resultats de l'optimització.

1. Resultats taules limitants primer nivell
2. Grups de càrrega generats
3. Hora optimitzada de càrrega pels diferents dominis
4. Gràfic de resultats

El primer d'ells és el que ens guiarà per a saber quines taules són candidates a simular un canvi d'hora, ja que aquestes són les que ens limiten que no puguem actualitzar els dominis abans.

El segon punt serveix a l'empresa per a saber quines taules s'han de carregar a cada hora i així indicar-li a Prefect que és l'orquestrador per a poder tenir les dades disponibles de les taules de segon nivell a temps per poder també tenir els "marts" en l'hora proposada per l'optimització.

El tercer punt simplement mostra a quina hora indica l'algorisme que podem carregar els dominis perquè la resta de taules de les qual depèn ja s'han actualitzat. I per últim, surt un deplegable amb un gràfic per veure de forma més visual a quina hora es carregaran els dominis.

D'aquesta manera, tenint tota aquesta informació sobre l'optimització directa de la informació que hem extret en l'inici, ara és torn per a poder jugar amb aquest simulador, per tant, anem a veure un exemple de com aplicar-ho.

En el primer desplegable s'ha detectat que hi ha una taula que és la "sap_zsv_zequz" que ens està limitant diversos dominis, per tant, és un bon exemple per a usar el simulador.

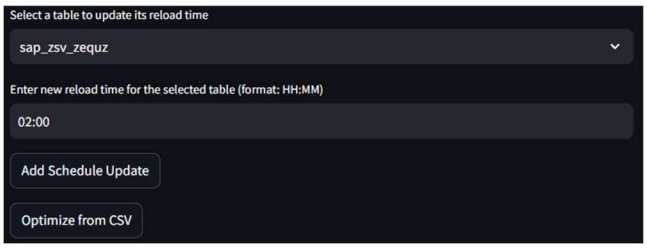
Domain	Bottleneck_Table	Bottleneck_Time(HH:MM)
cn	sap_vbrk	23:30
cn_pnl	sap_covp	01:15
commercial_ops	sap_zsv_zequz	20:30
compliance	sp_fmv08_ivd	06:15
finance	sap_zgplmo	09:15
human_resources	sap_cskt	09:15
int_audit	sap_makt	05:15
it	sap_zsaba	06:15
manufacturing	tc_bedford_materi	10:30
palex	sap_mvke	05:30
quality	sap_zsv_zequz	20:30
rd	sap_zsy_user	05:15
sales_noa	sap_vbrk	23:30
service	sap_zsv_zequz	20:30
supply_chain	sap_zsvw_wapp_c	23:15
tnt_legacy	sap_zsv_zequz	20:30

Imatge 12. Taules limitats post optimització.

Veiem com en els dominis "commercial_ops", "quality", "service" i "tnt_legacy la mateixa taula està impedint que no puguem carregar aquests dominis abans, per tant,

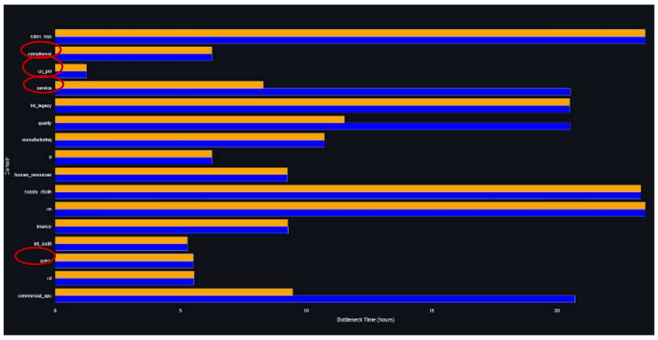
simularem que aquesta taula es carrega per exemple a les 02:00 en comptes de l'hora que s'està carregant ara que serà proper a les 20:30.

En la següent captura de l'aplicació web (imatge 13) podem veure com fer això:



Imatge 13. Actualització "schedule" taula de nivell 1.

Si apliquem els canvis i tornem a optimitzar, podem observar en la imatge 14 canvis en els resultats de l'optimització inicial.



Imatge 14. Simulació.

Com es pot veure, en l'últim desplegable que és la gràfica de resultats, tenim la comparativa entre abans i després de la simulació, i si ens fixem en els dominis els quals aquesta taula hi tenia influència veiem que a excepció de tnt_legacy que és la segona marca vermella, els altres 3 la línia taronja que és la que és la que utilitza l'hora simulada passa a ser molt menor a l'hora blava que és la proposada originalment per a l'algorisme.

Això ens és de molt alt valor per saber que si aquesta taula es carregués a les 02:00 tal com hem posat aquests 3 dominis es podrien actualitzar molt abans i per tant, tenir les dades disponibles diàriament a una hora molt més atractiva.

A partir d'aquí l'empresa pot lluitar per a canviar l'hora de recàrrega d'aquesta taula, de la mateixa manera que ho pot fer amb moltes més, ja que gràcies a aquest simulador es pot veure l'impacte que té l'hora de recàrrega d'una o més taules en concret sense haver d'aplicar els canvis directament. Per tant, és una eina per a testejar sobre el flux de recàrrega de dades més enllà de la pròpia optimització que proposa l'algorisme codificat.

5.3 Testos i avaluació

Després d'haver mostrat els resultats i el funcionament de l'algorisme d'optimització juntament amb la creació de l'aplicació web cal dir que tot això s'ha realitzat seguint una metodologia Agile amb la realització d'una sèrie de tasques dins de diferents "sprints" d'una duració de dues setmanes. La qual cosa comporta també reunions diàries dins l'empresa per a tractar qualsevol imprevís o problema que sorgís.

A banda d'això, que el que fa és ja evitar molts errors, ja que el progrés del treball és iteratiu també s'ha realitzat un prototip de dades.

El prototip és una taula de dades de només un domini que era dels més petits com era el de recursos humans i amb aquest i l'ajuda d'una intel·ligència artificial com ChatGPT[7] hem generat una taula de dades sinètiques per a fer proves al llarg de l'algorisme d'optimització, ja que s'executa per cada domini per separat, així doncs amb aquesta taula petita s'ha anat testejant que l'output de cada funció del codi fos l'esperat.

També s'han fet testos pel que fa a codificació, és a dir, a l'obtenir el codi cron i transformar-lo a hora diària s'ha comprovat que aquella taula s'executi diàriament, ja que el codi cron també permet indicar que es carregui un cop al mes per exemple, cosa que a nosaltres no ens interessa per a abordar els objectius.

6 Conclusió

Aquest projecte representa un pas endavant en la gestió i optimització de grans volums de dades, proporcionant solucions pràctiques que milloren la presa de decisions. A més, la creació d'una aplicació interactiva no només ha servit per fer més accessible el coneixement tècnic, sinó que també ha permès als usuaris de l'empresa visualitzar i entendre millor el funcionament del llac de dades. Aquesta eina, amb capacitat per simular diferents escenaris, posa a disposició de l'empresa una forma flexible i poderosa de prendre decisions informades i de planificar ajustos futurs.

Mitjançant l'ús de tècniques avançades d'anàlisi i desenvolupament, com la identificació del llinatge de dades i la implementació d'un algoritme d'optimització, s'han solucionat reptes complexos, demostrant la importància de comprendre i gestionar la interdependència entre els diferents nivells d'un llac de dades.

Des d'un punt de vista personal, el projecte ha estat un procés d'aprenentatge constant, posant en pràctica habilitats tècniques, com el desenvolupament en Python i la utilització d'eines com Amazon Redshift, Streamlit i Prefect, però també habilitats interpersonals com el treball en equip, l'adaptabilitat i la resolució de problemes en entorns dinàmics. L'adopció de la metodologia Agile ha demostrat

ser fonamental per garantir un avanç progressiu i efectiu en el desenvolupament del treball, permetent una gestió òptima del temps i dels recursos.

En definitiva, aquest projecte no només ha estat un exercici tècnic, sinó també una oportunitat per créixer professionalment i personalment. La combinació de reptes tecnològics amb la necessitat de desenvolupar solucions pràctiques i intuïtives ha contribuït a una experiència única, amb resultats que tenen un impacte real i positiu en el context empresarial.

Per acabar, m'agradaria dedicar una petita part als agraïments per a les persones que m'han donat suport durant la realització d'aquest projecte. I també vull deixar constància de les fonts d'informació utilitzades per a la realització de la memòria.

Agraïments

Primer voldria agrair al Eduardo Cèsar Galobardes per a acceptar ser el meu tutor acadèmic del treball i per l'ajuda i consells que m'ha donat al llarg de les sessions de seguiment portades a terme.

I com no pot ser d'altra manera, agrair al Raul Chaves, Víctor Pizarro i Simón Flores per donar-me la oportunitat de desenvolupar aquesta tesi dins d'una empresa, a més del seu suport constant durant el progrés del treball.

Bibliografia

[1] SAP, "SAP Software Solutions." Disponible a: https://www.sap.com [Últim accés: setembre 2024].

[2] Amazon Web Services, documentació Amazon Redshift. Disponible a: https://docs.aws.amazon.com/redshift/ [Últim accés: setembre 2024].

[3] Prefect, documentació Prefect. Disponible a: https://www.prefect.io/ [Últim accés: novembre 2024].

[4] Python Requests, llibreria requests. Disponible a: https://docs.python-requests.org/ [Últim accés: novembre 2024].

[5] "Cron Expressions," transformador expressions cron. https://crontab.guru/ [Últim accés: desembre 2024].

[6] Streamlit, documentació llibreria Streamlit Disponible a: https://docs.streamlit.io [Últim accés: gener 2025].

[7] OpenAI, ChatGPT per OpenAI. Disponible a: https://openai.com/chatgpt [Últim accés: gener 2025].