INFORME DE SEGUIMENT 2

TREBALL DE FI DE GRAU

Autor: Ricard Tuneu Font

Data: 29/12/2024

Tutor: Eduardo Cèsar Galobardes

Universitat Autònoma de Barcelona

Índex

1. Objectius

En aquest últim informe de seguiment i per tant també, la última part de desenvolupament pràctic del projecte ja es pot fer una valoració contrastada dels objectius plantejats en l'inici del treball i els petits canvis que s'han fet durant la realització del mateix.

El primer dels objectius, ja complert en l'anterior informe era trobar el flux de càrrega de dades de la base de dades, és a dir, trobar les dependències que hi ha entre cadascuna de les taules que conformen el llac de dades per a poder fer la posterior optimització.

Seguidament, el segon objectiu, s'ha complert entre novembre i desembre tal com estava previst en la planificació. El que s'ha fet és realitzar una optimització en base al "lineage" obtingut anteriorment establint uns grups de càrrega nous per al nivell de "intermediate" en base als colls d'ampolla que generen les taules de nivell 1, tal com s'explica a continuació en el punt de desenvolupament.

Ja per últim, tenia l'objectiu de poder plasmar els resultats d'una manera visual i inicialment tenia la idea de fer-ho amb PowerBI, però donades les dades i l'eina que es volia generar per a l'empresa, vaig decidir fer-ho amb una llibreria de Python que s'anomena Streamlit [1]. Aquesta llibreria permet construir aplicacions web interactives ideals per a visualització de dades.

Així doncs, aquests 3 objectius que han evolucionat durant el treball, però he pogut complir-los tots ells, aconseguint realitzar una part més de "backend" amb tota l'extracció de dades i de "lineage" per acabar plasmant-la en un entorn "frontend" com és l'aplicació web que he creat.

2. Metodologia i Planificació

La metodologia seguida en aquesta última part del treball és la mateixa que he seguit des del principi el que m'ha permès tenir tota la feina ben organitzada i la realització de les taques en més o menys el temps previst.

En l'empresa hem seguit amb la idea Agile de "sprints" de dues setmanes i amb reunions diàries cada dia per a comentar el progrés de la feina o si ha sorgit qualsevol dubte o problema.

Actualment, el "backlog" final que plasma les diferents tasques que s'han anat fent durant el projecte es pot veure en la següent imatge:

User Story	Lineage dependencies	Active	Business	Laboratory
Task	Github + Prefect API's acces	Closed		Laboratory
Task	Entrega informe inicial (universidad)	Closed		Laboratory
Task	Dataframe and algorithm design	Closed		Laboratory
Task	Find out dependency tables (PY)	Closed		Laboratory
Task	Create Lineage (PY)	Closed		Laboratory
Task	Take schedules from tables (anomalies detection) wi...	Closed		Laboratory
Task	Calculate num of dependencies for each mart	Closed		Laboratory
Task	Create prototype with fiction data to do the testing ...	Closed		Laboratory
Task	Optimization algorithm (delta dataframe) with prot...	Closed		Laboratory
Task	Get redshift tables info	Closed		Laboratory
Task	Apply anomalies algorithm	Closed		Laboratory
Task	Apply optimitzation algorithm	Closed		Laboratory
Task	Create Tests to check optimization algorithm	Active		Laboratory
Task	Extract screenshot from actual times, before optimiz...	Closed		Laboratory
Task	Create Streamlit app	Closed		Laboratory
Task	Create What if?	Closed		Laboratory
Task	Extract Cron schedules from Github	Closed		Laboratory
Task	Generate Optimization Dataframe	Closed		Laboratory

Imatge 1. Històries d'usuari del projecte

Com es pot veure totes ja estan en estat tancat, excepte una que és en relació a alguns testos que també he volgut fer per a verificar la qualitat i la validesa de les dades durant el procés d'optimització a més del testejar el codi primerament amb un menor nombre de dades com és el prototip realitzat i usant dades sintètiques que permetien provar tots els casos peculiars.

Per acabar, la planificació inicial va ser força correcta ja que he pogut acabar el treball en les dates que em vaig marcar, que era abans de finalitzar l'any i tot i que no totes les històries d'usuari s'han fet en el temps pensat, el conjunt d'elles si que les he pogut finalitzar en el temps marcat.

3. Desenvolupament

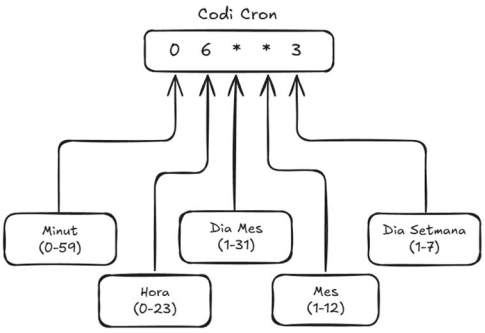
3.1. Obtenció taula d'optimització

En l'anterior informe l'última cosa que teníem era la taula d'optimització, però tenia totes les columnes relacionades amb temps, que recordem eren el "cron schedule", "reload time", "median load time" i "bottleneck time" en dades sintètiques. Va ser usat per a fer les primeres proves i testejar l'algorisme d'optimització abans de traslladar-ho a les dades reals que és el que s'ha afegit ara.

Per obtenir aquestes dades, s'han extret des de dos llocs diferents que és on l'empresa té aquesta informació. Primerament necessitem l'extracció del "cron schedule" que és un codi que dictamina a quina hora es recarrega una taula. Per a aquesta dada s'ha accedit al repositori de Github [2] on estan guardats tots els models de les taules del "datalake" i en aquests models hi ha un camp anomenat "schedule". D'aquest camp hem realitzat un regex (cerca dins l'script del model) de forma similar al que vam fer per a obtenir el llinatge de les taules i així doncs, de cada taula de nivell 1 ("staging") s'ha extret aquesta informació.

Cal remarcar que les taules d'"intermediate i els "marts" venen precedides per l'hora de càrrega de les seves "staging" i per tant necessitem el "schedule" d'aquestes ("staging") per a poder fer l'anàlisi i optimització.

Amb això ja tenim emplenada la informació de les columnes "cron schedule" i "reload time", ja que aquest últim és simplement la conversió del codi a segons des de la mitja nit.



Imatge 2. Conversió codi cron

La següent informació que ens falta és el temps mitjà que tarda una taula en recarregar les seves dades. Ha estat escollida la mediana, en comptes de la mitjana per a evitar "outliers" de recàrregues que hagin pogut patir algun problema.

Per aconseguir-ho l'empresa emmagatzema aquesta informació en una taula del "datalake" provinent de Prefect que és l'orquestrador que gestiona la càrrega de dades de l'empresa i d'on s'origina tota la informació en relació a temps de càrrega. Per obtenir les dades simplement hem hagut de fer una consulta de SQL i guardar les dades en la taula que usarem per a la optimització. La connexió realitzada és a través de AWS Redshift [3], i usant variables d'entorn per a que qualsevol usuari amb accés pugui carregar les dades.

La consulta utilitzada és la següent:

SELECT table_name, PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY duration) AS median_duration

FROM it.it_prefect_data

GROUP BY table_name;

Ara la darrera informació que ens queda és el "bottleneck time", però aquest simplement s'obté sumant l'hora que es recarrega amb el temps que tarda en carregar-se que són les dades que acabem d'obtenir.

Un cop això, ja tenim la taula per a aplicar el procés d'optimització. Com a detall final, però s'ha afegit en el "dataframe" dues columnes que són les "destination" i "destination domain" que pertanyen a la taula de "lineage" que ja teníem al inici, però que pensàvem que no serien útils, però finalment ens són d'ajuda per a fer part de l'aplicació de Streamlit que a continuació s'explica.

El que es veu a continuació és una part de la taula final que hem construït per a poder optimitzar a continuació:

Mart	Mart_Domain	Destination	Destination_Domain	Source	Source_Domain	Cron Schedule	Reload_Time	Median_Load_Time	Bottleneck_Time
cn_action	cn	cn_action	cn	cn_zsd0_call_subtyt	cn			4.57	
cn_action	cn	cn_zsd0_call_subtyt	cn	int_zsd0_call_subtyt	intermediate			4.66	
cn_action	cn	int_zsd0_call_subtyt	intermediate	sap_zsd0_call_subtyt	staging	['0 1 * * *']	3600.0	9.585	3610.0
cn_action	cn	cn_action	cn	cn_actiongroup	cn			2.83	
cn_action	cn	cn_actiongroup	cn	int_actiongroup	intermediate			4.65	

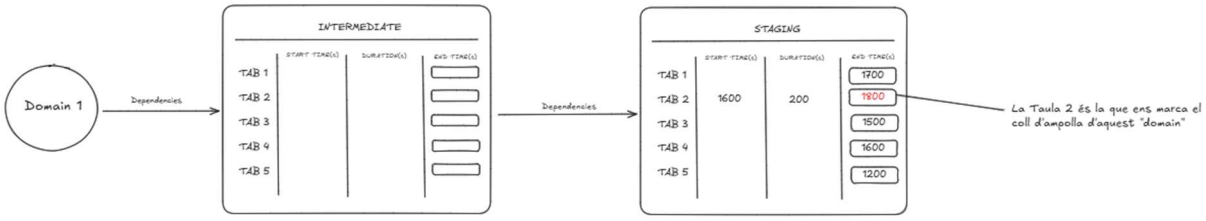
Imatge 3. Capçalera de la taula d'optimització

Cal remarcar que abans de procedir a l'optimització s'han tingut en compte uns detalls i és que en el cas que tinguem alguna taula de "staging" sense hora de recarrega, queda fora de l'optimització, és a dir, no es té en compte. I per altra banda, si qualsevol taula de nivell 1 o 2 no tenim dades sobre el temps que tarda en carregar-se també queda fora del procés d'optimització.

Aquestes casuístiques s'han tingut en compte, perquè per exemple hi ha algunes taules de "staging" que no tenen "schedule" diari ja que són carregues on-demand i per tant aquestes no les hem tingut en compte per fer l'optimització del flux de recàrrega diari.

3.2. Procés d'optimització

Un cop ja tenim el "dataframe" amb tota la informació necessària ja podem iniciar a codificar l'algorisme que ens optimitzarà el flux de càrrega de les taules dins el llac de dades.



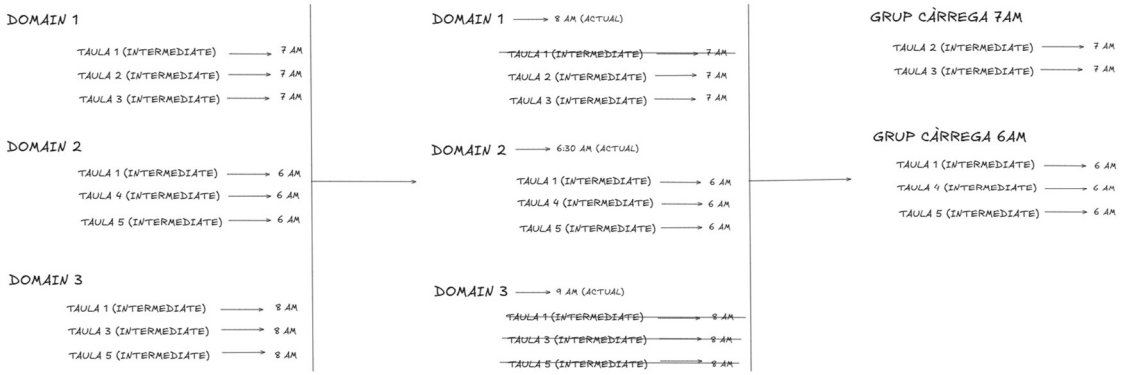
Imatge 4. Detecció coll d'ampolla

Recordant que un dels objectius és intentar carregar els diferents “domains” el més abans possible a nivell individual, un dels punts claus és detectar quina taula de nivell 1, que són les que ens venen amb un “schedule” establert i que hem recollit en la taula és la que es carrega en última instància.

Aquesta taula és la que limita que les “intermediate” (nivell 2) no puguin ser carregades abans. I amb això d’entrada el que farem és crear un grup de càrrega de taules just en el moment que aquesta última taula s’ha carregat, com si fos un “trigger”.

Aquesta detecció la realitzem per a tots els dominis de forma paral·lela, de manera que de cadascun d’ells obtenim la taula limitant i com a conseqüència una hora en la que es poden carregar les “intermediate” que usa aquell domini. Això ho podem saber gràcies al llinatge trobat en l’inici del projecte.

En la següent imatge, en la part de més a l’esquerra podem veure la situació que tenim:



Imatge 5. Creació dels grups de càrrega

Així doncs, quan tenim detectades les taules limitants de cada domini que ens han donat una hora de disponibilitat per a les taules de segon nivell, cal mirar aquestes taules de nivell 2 si apareixen en més d’un domini, si és així aquesta taula es carregarà juntament amb el domini que es pugui carregar abans.

Per entendre-ho, és una optimització on minimitzem l’hora de càrrega, és a dir, ens quedem amb la hora més baixa possible d’entre totes les opcions que tinguem, que ho marcarà el nombre de dominis els quals aquella taula hi tingui influència.

Si una taula només s’usa en un domini, aleshores no queda altra opció que assignar-la al grup de càrrega originat per la taula coll d’ampolla d’aquell domini.

Un cop analitzades totes les taules ja tenim els nous grups de càrrega, que actualitzen les dades el més aviat possible d’acord amb les limitacions que presenten les taules de “staging”.

Un cop tenim els grups de càrrega del nivell 2 (“intermediate”), ara només falta aquí veure quina taula de cada grup tarda més en carregar-se i serà aquella la que ens marca la disponibilitat per carregar els dominis (taules de nivell 3).

3.3. Aplicació Web

Un cop feta la optimització, cal poder mostrar les dades i resultats en un entorn visual en el qual poder analitzar resultats i extreure'n les respectives conclusions.

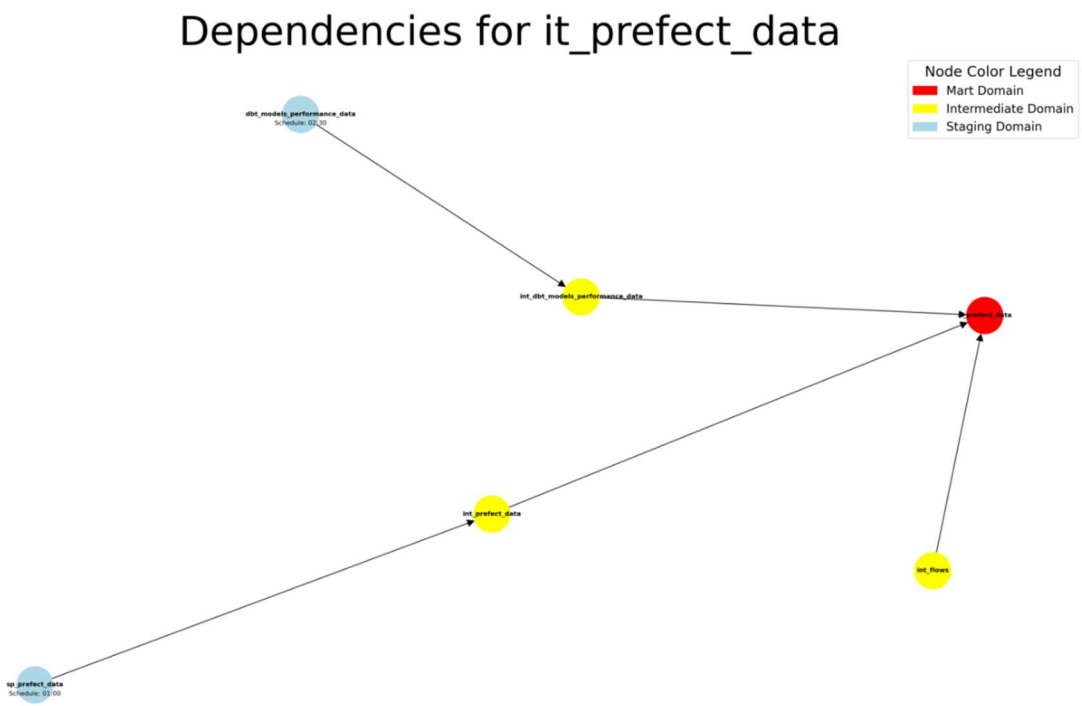
Tal com s'ha explicat en l'inici, la idea de fer un dashboard a PowerBI ha derivat i finalment s'ha creat una webapp amb algunes eines on hi ha la possibilitat de aplicar filtres per tenir visualitzacions més detallades i específiques del que es vulgui analitzar.

L'eina per realitzar la webapp és Streamlit que és una llibreria de Python i està especialment dissenyada per a la visualització de dades de forma dinàmica.

En l'aplicació tenim 3 funcionalitats diferents on es pot seleccionar qualsevol de les 3 a través d'un menú.

La primera d'elles és el "Lineage Display", aquesta funcionalitat és per poder visualitzar el flux de càrrega de dades d'una taula de nivell 3, és a dir, es mostra totes les taules que acaben alimentant aquesta taula "mart" ja sigui de forma directa o indirecta. Així doncs, aquí tenim 2 filtres, un primer per a escollir el domini on es troba la taula que volem buscar i un segon per escollir la pròpia taula.

Un exemple del que es visualitza és el següent:



Imatge 6. "Lineage Display"

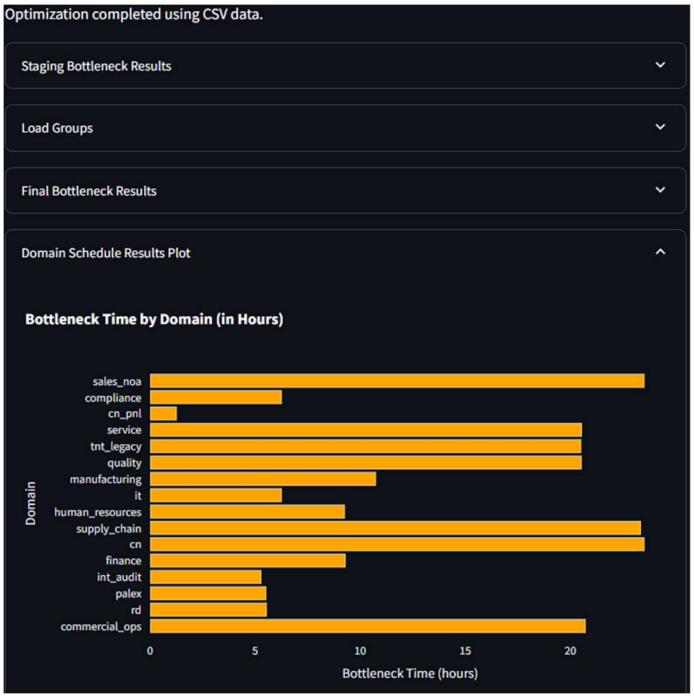
Com es pot veure, les fletxes apunten a la taula que alimenten i al final totes s'acaben unint en el node vermell, que és el mart que hem filtrat en l'aplicació. A més a més, en els nodes blaus, que simbolitzen les taules de nivell 1, hi ha la informació sobre el "schedule" que tenen assignat.

Amb la informació que es presta en aquest visual es pot fer detecció d'errors i poder veure on s'ha pogut trencar el camí en cas d'haver algun error o simplement pot servir d'informació per saber quines taules influeixen en el contingut d'altres.

La següent funcionalitat, anomenada "Schedule Optimization" és un simulador, per veure l'efecte que pot tenir un canvi d'hora de recàrrega en una o més taules a la vegada. Aquesta funcionalitat és clau per a l'empresa per a poder veure quins canvis a nivell d'hores de recàrrega es produeixen en els dominis, sense haver de aplicar els canvis realment, per això l'empresa podrà testejar certes idees que tinguin i en cas que la simulació sigui positiva llavors ho poden fer realitat.

Abans de provar de simular canvis horaris, el que cal saber és quines taules pot ser intel·ligent canviar l'hora, probablement perquè siguin les taules limitants. En l'app hi ha un botó per fer l'optimització sense canvis, és a dir la que proposa purament l'algorisme.

Aquí es mostra els resultats obtinguts:



Imatge 7. Resultats d'optimització

Com es pot veure en els resultats apareixen 4 desplegables diferents. En el primer d'ells hi ha la informació de les taules "bottleneck", és a dir mostra la última taula que es carrega de cada domini juntament amb la hora limitant en la qual es pot iniciar la recarrega de les "intermediate".

El segon desplegable i en funció de la informació obtinguda en el primer es troben els grups de recarrega generats. És a dir, per cada domini quines taules d'"intermediate" es poder carregar a la vegada i a quina hora es recarreguen.

Seguidament, el tercer, és resultat del segon i mostra l'hora definitiva a la que es poden carregar els dominis (nivell 3) segons la taula de nivell 2 que tardi més en carregar.

Per últim, aquests resultats finals es poden visualitzar en el gràfic de barres horitzontals de color taronja que apareix en l'últim desplegable.

Un cop tenim això, el més interessant ve ara. Si analitzem els resultats podem usar ara el simulador. Per exemple, una de les taules limitants era la sap_zsv_zequz, si apliquem un canvi d'hora com es pot veure en la següent captura:

Imatge 8. Simulació canvi d'hora

Si aplico aquest canvi, els resultats que es visualitzen en el gràfic són els següents:

Imatge 9. Resultats post simulació

Com es pot veure, ara el gràfic apareix amb una doble barra horitzontal per domini, la blava indica els resultats que teníem fins ara, mentre que la taronja mostra els canvis que s'han produït després de fer el canvi d'hora de recarrega.

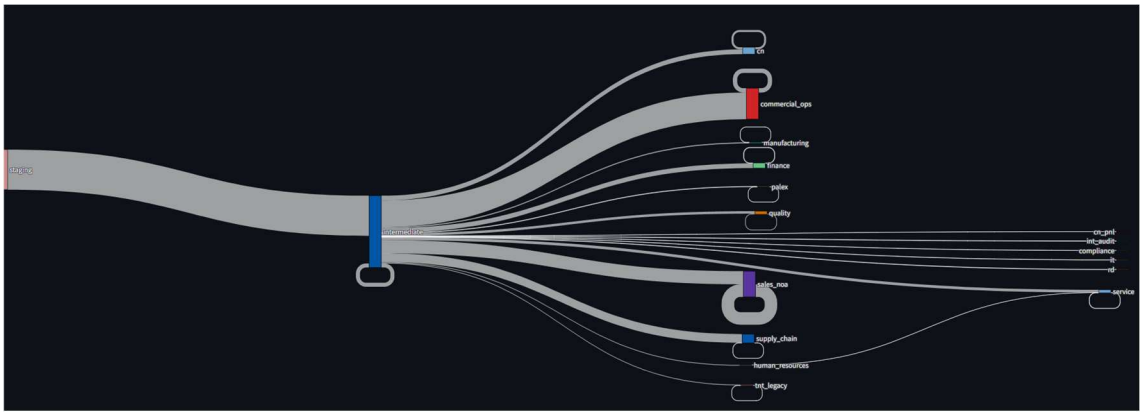
Si ens hi fixem, en els dominis de "service", "quality" i "commercial_ops" l'hora en el que ens diu que es poden carregar ara ha avançat molt, per tant la taula que hem avançat que és la sap_zsv_zequz, gràcies al simulador hem vist que és clara candidata a

analitzar i poder realment actualitzar la seva hora de recarrega en la mesura del possible.

Cal dir també que en el simulador a banda de canviar l'hora d'una o més taules podem eliminar-ne com a tal i que no estiguin dins la optimització, és a dir, eliminar completament que aquella o aquelles taules que indiquem puguin ser una taula limitant.

Per últim, tenim la tercera funcionalitat. Aquesta consta de 2 diagrames de Sankey diferents.

El primer d'ells i que es pot veure en la següent imatge, mostra de forma global les dependències que hi ha entre cadascun dels nivells de taules.



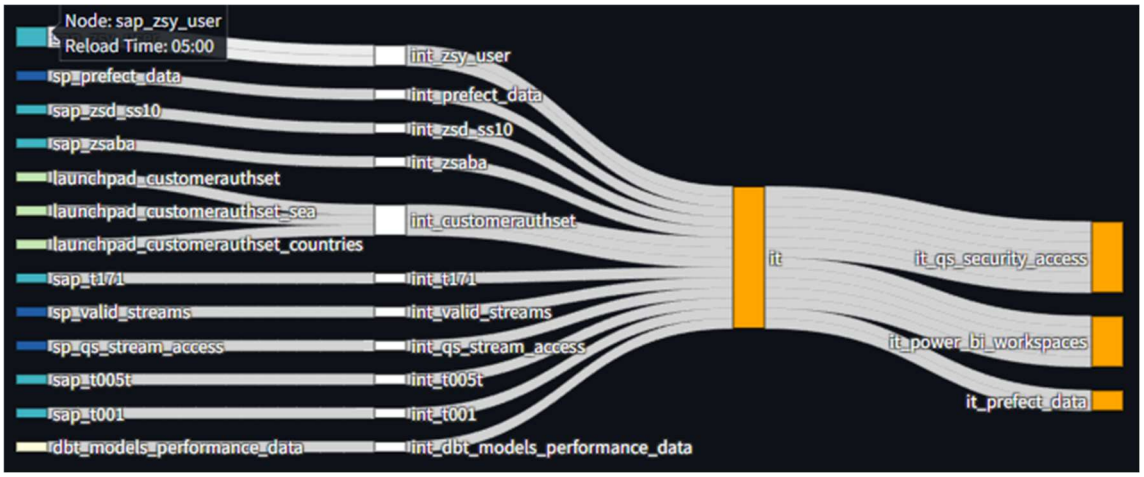
Imatge 10. Diagrama de Sankey global

Aquest diagrama és molt útil per detectar algunes anomalies que poden sorgir i en aquesta última imatge se'n pot veure una.

La empresa té completament la idea de que un domini no pot alimentar un altre domini, és a dir hi ha d'haver independència entre tots ells en el nivell 3. Poden compartir taules de nivell 1 i 2, però mai de nivell 3.

En aquesta imatge es detecta clarament com el domini de "human resources" té un enllaç apuntant a "service", així doncs això és una informació valuosa per a l'empresa per a estudiar aquest cas, el perquè de que estigui fet així i poder-ho corregir com abans millor.

L'altre diagrama, té uns filtres on podem indicar quines taules de nivell 1 volem visualitzar i de quin domini.



Imatge 11. Diagrama de Sankey fluxe de recàrrega

Aquest diagrama és similar al de llinatge inicial, però aporta la visió contraria gràcies als filtres, és a dir, en comptes de veure quines taules estan alimentant a una taula de nivell 3, el que es pot veure millor aquí és, les taules de nivell 1, a quines taules i domini alimenten.

A més a més s'ha aplicat una paleta de colors en les "staging" on cada color determina un origen de dades ja que n'hi ha de diferents com pot ser SAP, SharePoint(SP) o launchpad.

També en la visualització es pot veure posant-se a sobre de cada node l'hora de recàrrega i en cadascuna dels enllaços, el temps que tarda en carregar-se.

D'aquesta manera amb totes aquestes eines que te l'aplicació web l'empresa pot jugar amb elles per extreure'n informació de valor a banda de la pròpia optimització que realitza l'algorisme codificat.

4. Bibliografia i Webgrafia

[1] Documentació Streamlit. Enllaç: https://docs.streamlit.io/ (última consulta 20/12/2024)

[2] Github. Enllaç: https://github.com/ (última consulta 08/12/2024)

[3] Connexió Amazon Redshift. Enllaç: https://pypi.org/project/redshift-connector/ (última consulta 16/12/2024)