INFORME DE SEGUIMENT 1
TREBALL DE FI DE GRAU

Autor: Ricard Tuneu Font

Data: 11/11/2024

Tutor: Eduardo Cèsar Galobardes

Universitat Autònoma de Barcelona

ÍNDEX

1. Objectius

Després de vàries setmanes desenvolupant el projecte ja puc fer una valoració inicial sobre els objectius plantejats en el primer informe.

El primer dels objectius, indispensable per a la resolució del treball, era ser capaç de trobar el camí que uneix les taules “mart” amb les que les alimenta inicialment com són les “staging” passant entremig per les taules de “intermediate”.

Aquest primer objectiu ja l’he pogut completar, la informació està emmagatzemada en una “dataframe” a punt per a poder ser visualitzada en un “dashboard”.

L’objectiu número 2, que era optimitzar els fluxos de càrrega de taules i garantir que es tenen les dades actualitzades a l’hora que les aplicacions les necessiten, cal matisar-lo. Amb la mateixa realització de l'exercici s’ha definit de forma més detallada la funció que es vol aconseguir per a optimitzar els fluxos de càrrega del "datalake".

Aquest objectiu és el següent: Consisteix a crear un algoritme per definir grups de taules a “intermediate” que es puguin executar el més d'hora possible (i en paral·lel) per aconseguir les dades disponibles als marts finals dels "domains" a una hora òptima. D’aquesta manera també indicar quines són les ingestes (taules “staging”) que penalitzen el flux recarregant-se més tard que la resta.

Per acabar, l’últim objectiu es manté igual, que és el de plasmar els resultats obtinguts tant de “lineage” com d’optimització de fluxos a través d’un quadre de control amb l’eina de PowerBI.

2. Metodologia i Planificació

Pel que fa a la metodologia s'està aplicant la que ja vaig comentar en el primer informe on estic dividint la feina en diferents "sprints" i on fem una reunió diària els diferents membres del departament per a tractar els temes del dia a dia i poder resoldre qualsevol dubte o problemàtica que sorgeixi.

També amb l'eina Azure Devops vaig plasmant totes les tasques que estan en procés, les que ja estan fetes i les que estan pendents de fer per tal de poder-me organitzar bé i que els meus companys d'equip puguin anar veient com avança el desenvolupament del treball.

Aquí en la següent imatge es pot veure un estat actual del "backlog" amb les diferents tasques relacionades amb el treball:

User Story	Lineage dependencies	Active
Task	Github + Prefect API's acces	Closed
Task	Entrega informe inicial (universidad)	Closed
Task	Diseño algorismo + dataframe final	Closed
Task	Encontrar dependencias tablas (PY)	Closed
Task	Crear Linage (PY)	Closed
Task	Take schedules from tables (anomalies detection) wi...	Closed
Task	Calculate num of dependencies for each mart	Closed
Task	Create prototype with fiction data to do the testing ...	Closed
Task	Optimization algorithm (delta dataframe) with prot...	Active
Task	Obtain total gain from delta and for each mart path...	New
Task	Get redshift tables info	Active
Task	Apply anomalies algorithm	New
Task	Apply optimitzation algorithm	Active
Task	Store delta dataframe results in a historic	New
Task	PowerBi Dashboard development	New

Imatge 1. Estat actual del "backlog" del projecte

Per al moment la planificació dels temps s'està seguint força ja que al mes d'octubre vaig poder acabar amb la part de trobar el camí entre taules i actualment estic treballant amb l'algorisme d'optimització que planifico que durarà tot el mes de novembre tal com ja vaig esmentar en el primer informe.

3. Desenvolupament

3.1. Obtenció del flux de taules dins el “datalake”

Per poder obtenir el flux de taules del llac de dades el que hem de fer és analitzar les dependències que té cadascuna de les taules “mart” i les taules “intermediate” de la base de dades.

En un repositori de Github, l’empresa hi guarda tots els models que generen les taules del “datalake” i amb un “token” de la web[1] i la llibreria requests[2] he pogut accedir als codis de cadascuna de les taules per obtenir quines dependències tenen.

Com que les sentències de les taules estan en codi SQL el que he fet amb la llibreria re de Python[3] és buscar patrons dins la part de la clàusula “from” per saber de quina/es dades s’alimenta cada taula. D’aquesta manera obtenim totes les dependències de les “mart” i les “intermediate” que concatenant la informació podem obtenir el camí complet que segueix cada taula des de la seva part inicial on s’ingereix les dades fins a l’estructura final utilitzada per a una finalitat de negoci.

Un cop obtinguda la informació l’he emmagatzemat en un “dataframe” de pandas que té l’estructura que es pot veure en la següent imatge:

Mart	Mart_Domain	Destination	Destination_Domain	Source	Source_Domain
cn_action	cn	cn_action	cn	cn_actiongroup	cn
cn_action	cn	cn_actiongroup	cn	int_actiongroup	intermediate
cn_action	cn	int_actiongroup	intermediate	sp_actiongroup	staging
cn_action	cn	cn_action	cn	cn_zsd0_call_subtyt	cn
cn_action	cn	cn_zsd0_call_subtyt	cn	int_zsd0_call_subtyt	intermediate
cn_action	cn	int_zsd0_call_subtyt	intermediate	sap_zsd0_call_subtyt	staging
cn_action	cn	cn_action	cn	cn_cnhr	cn
cn_action	cn	cn_cnhr	cn	int_vacantuser	intermediate
cn_action	cn	int_vacantuser	intermediate	sp_vacantuser	staging
cn_action	cn	cn_cnhr	cn	_salesrep_area_combine	intermediate
cn_action	cn	_salesrep_area_combine	intermediate	_salesrep_area_combine	staging
cn_action	cn	cn_cnhr	cn	int_cnhr	intermediate
cn_action	cn	int_cnhr	intermediate	sp_cnhr	staging
cn_action	cn	cn_action	cn	cn_zsd0_calls	cn
cn_action	cn	cn_zsd0_calls	cn	int_zsd0_calls	intermediate
cn_action	cn	int_zsd0_calls	intermediate	sap_zsd0_calls	staging
cn_action	cn	cn_action	cn	int_zfv_201t	intermediate
cn_action	cn	int_zfv_201t	intermediate	sap_zfv_201t	staging

Imatge 2. Estructura emmagatzematge “lineage”

En aquesta imatge en concret he agafat el “mart” cn_action que pertany al domini de cn. En aquesta taula cadascuna de les files representa una dependència, per exemple si ens fixem en la primera ens està dient que la cn_actiongroup ("source"), està alimentant a la cn_action ("destination") i si després mirem la segona, podem veure com la int_actiongroup ("source") alimenta a la cn_actiongroup.

A més a més de cadascuna de les taules que conformen el camí obtingut aquesta taula ens indica a quin nivell pertanyen, ja sigui a “staging”, “intermediate” o al domini en cas que la taula sigui un “mart”. Així doncs, amb aquest "dataframe" podem extreure el flux de càrrega que hauria de seguir la taula cn_action.

No descarto en un futur canviar o crear una versió de "dataframe" amb un format diferent en cas que sigui necessari o més còmode per a les tasques del projecte.

3.2. Anàlisi de l'estructura del "datalake"

Un cop tenim ja el camí que uneix cadascuna de les taules de la base de dades he volgut fer una mica d'anàlisi de com poden ser de complexos aquests camins, ja que una taula es pot alimentar per tantes taules com faci falta i, per tant, els camins entre el "mart" i la ingesta no tots tenen perquè seguir el patró senzill que seria que un "mart" sigui alimentat per una "intermediate" i aquesta per una "staging".

Gràcies a fer aquesta anàlisi he pogut veure com hi ha taules amb moltíssimes dependències. Les que ocupen el top 5 amb alimentació de diferents taules són les que podem veure en la següent imatge:

	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 3. Top 5 taules amb més dependències

Amb això no és tot, ja que cal veure bé una distribució de com és de normal que una taula tingui aquesta magnitud tan gran de dependències. D'aquesta manera el que he fet és mirar la freqüència de taules que tenen x dependències i així poder veure la tendència o estructura general de la base de dades.

	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 4. Top 5 percentatge de taules amb x nombre de dependències

Els resultats obtinguts són aproximadament els que ja esperava perquè com podem veure la majoria de les taules "mart", un 68.13%, tenen tan sols 2 dependències que serien la "staging" i la "intermediate", però evidentment com hem vist hi ha camins més complexos, però que suposen una part més petita dins l'estructura del "datalake".

3.3. Creació prototip

Un cop tenim la informació sobre les dependències de les taules que ens componen els camins que han de seguir les recarregues diàries d'informació el següent pas és dissenyar l'algorisme d'optimització d'aquest flux de càrrega on poder crear nous grups de càrrega de les taules d'"intermediate" definits pels "bottlenecks" que obtindrem de les taules "staging" de cada domini i que d'aquesta manera puguem avançar la càrrega del propi "domain" per tenir les dades actualitzades diàriament tan aviat com es pugui.

Abans, però de ficar-se a generar tot el codi i infraestructura que ens permetrà fer això el que he volgut fer és crear un prototip a petita escala que pugui servir-me per fer proves i testejar l'algorisme abans d'aplicar-ho en les dades reals.

El que necessitem per optimitzar el flux de càrrega són els "marts" que tenim, amb la indicació del domini al qual pertanyi juntament amb les "sources" de la relació de "lineage" que hem vist anteriorment i també al nivell al qual pertany, ja sigui "mart", "intermediate" o "staging". Això últim ens permetrà agrupar les taules i poder analitzar el "bottleneck" de les "staging" per generar els nous grups de càrrega de les "intermediate" i per acabar poder optimitzar l'hora de càrrega de cadascun dels dominis.

A banda, el que necessitem és l'hora en la qual es recarrega cada taula i els temps que tarda a actualitzar-se.

Davant de tota aquesta informació que necessitem i donat que la informació dels "marts" i fonts ja les hem obtingut amb el "lineage", el que he fet és quedar-me amb un domini en concret, que en aquest cas es diu "human_resources", ja que és un dels més petits per a fer proves. En aquest domini li he afegit dues columnes amb una funció de generació de valors aleatoris feta juntament amb la IA de ChatGPT[4] per assignar a cada taula una hora de "schedule" i un temps de recàrrega.

D'aquesta manera amb això ja podríem començar a implementar l'algorisme d'optimització a escala local d'un únic domini. La idea és que quan funcioni pugui anar ampliant el nombre de dominis fins a aplicar l'algorisme d'optimització amb tots ells.

El prototip que començaré a usar presenta aquest format:

	Mart	Mart_Domain	Source	Source_Domain	Reload_Time	Average_Load_Time	Bottleneck_Time
1	hr_ccemails_sheet1	human_resources	int_ccemails_sheet1	intermediate	15795	2913	18708
2	hr_ccemails_sheet1	human_resources	sp_ccemails_sheet1	staging	860	1429	2289
3	hr_ccemails_weekly_report	human_resources	int_ccemails_weekly_report	intermediate	38158	1800	39958
4	hr_ccemails_weekly_report	human_resources	sp_ccemails_weekly_report	staging	11284	1002	12286
5	hr_cskt	human_resources	int_cskt	intermediate	6265	2749	9014
6	hr_cskt	human_resources	sap_cskt	staging	16850	3077	19927
7	hr_departments	human_resources	int_departments	intermediate	37194	1879	39073
8	hr_departments	human_resources	sp_departments	staging	21962	461	22423
9	hr_lfa1	human_resources	int_lfa1	intermediate	16023	501	16524
10	hr_lfa1	human_resources	sap_lfa1	staging	41090	2281	43371
11	hr_red_green_light_groups_sheet1	human_resources	int_red_green_light_groups_sheet1	intermediate	1685	1295	2980
12	hr_red_green_light_groups_sheet1	human_resources	sp_red_green_light_groups_sheet1	staging	769	2617	3386
13	hr_supervisors_csv	human_resources	int_supervisors_csv	intermediate	2433	1115	3548
14	hr_supervisors_csv	human_resources	sp_supervisors_csv	staging	5311	755	6066
15	hr_workforce_report_accesses	human_resources	int_workforce_report_accesses	intermediate	37819	1575	39394
16	hr_workforce_report_accesses	human_resources	launchpad_workforce_report_accesses	staging	39188	1316	40504
17	hr_zhro_emp	human_resources	int_zhro_emp	intermediate	17568	2643	20211
18	hr_zhro_emp	human_resources	sap_zhro_emp	staging	19769	3067	22836
19	hr_zhro_sup	human_resources	int_zhro_sup	intermediate	28693	637	29330
20	hr_zhro_sup	human_resources	sap_zhro_sup	staging	6396	1178	7574
21	hr_zhrt_group_emp	human_resources	int_zhrt_group_emp	intermediate	27480	1376	28856
22	hr_zhrt_group_emp	human_resources	sap_zhrt_group_emp	staging	41434	1091	42525
23	hr_zhrt_groups	human_resources	int_zhrt_groups	intermediate	25658	2564	28222
24	hr_zhrt_groups	human_resources	sap_zhrt_groups	staging	18942	1063	20005

Imatge 4. Dataframe prototip

Cal destacar que els temps de recàrrega s'han passat a segons des de les 00:00 h, és a dir, si tenim un "reload_time" de 3600 s voldrà dir que aquesta taula es carrega a la 01:00 h. A més a més, amb això he afegit una columna on se suma l'hora de càrrega amb el temps que tarda a carregar per obtenir el "bottleneck_time", és a dir, l'hora que haurem d'analitzar per optimitzar el nostre flux.

Per acabar, dir que el següent pas és aquest algorisme d'optimització que es treballarà primer amb aquestes dades sintètiques de temps, ja que les dades reals les obtindré pròximament i així també em servirà per testejar l'algorisme a més petita escala.

4. Bibliografia i Webgrafia

[1] Documentació Github. Enllaç: https://docs.github.com/es/apps/creating-github-apps/authenticating-with-a-github-app/generating-a-user-access-token-for-a-github-app (última consulta 17/10/2024)

[2] Llibreria requests (Python). Enllaç: https://pypi.org/project/requests/ (última consulta 17/10/2024)

[3] Llibreria d'expressions regulars (Python). Enllaç: https://docs.python.org/es/3/library/re.html (última consulta 21/10/2024)

[4] Intel·ligència artificial ChatGPT. Enllaç: https://chatgpt.com/ (última consulta 4/11/2024)