Il tuo bagaglio è al sicuro (forse)

Antonio Pennacchia
Matricola:808205

Lorenzo Mauri
Matricola:807306

Introduzione

"L'assicurazione sul bagaglio non la pago, tanto figuriamoci se viene smarrito". È la classica frase di chi è in procinto di effettuare un viaggio in aereo. La perdita del bagaglio aereo da parte degli aeroporti è un fenomeno che, per fortuna, accade piuttosto raramente, ma quando accade potrebbe rivelarsi piuttosto spiacevole, soprattutto se il possessore di tale bagaglio si appresta ad un lungo soggiorno nel luogo di destinazione.
La nostra indagine si basa proprio sull'evento della perdita di bagagli aerei, tramite l'analisi di grandi quantità di dati fornitici gentilmente dall'azienda di consulenza data-driven "YouthQuake". Youthquake, la quale collabora con un'azienda che lavora con dati relativi ai voli, ci ha fornito un ricco database contenente informazioni (opportunamente anonimizzate) su voli aerei che collegano aeroporti europei [1], ma l'informazione di maggiore interesse è rappresentata da una variabile che indica per ogni volo e per ogni singolo passeggero[2] se il bagaglio registrato sia stato perso oppure no nel tragitto effettuato; sfortunatamente non disponiamo della variabile che indica quale tra l'aereoporto di partenza e quello di arrivo sia "colpevole" dello smarrimento.
Sfruttando tali dati, proveremo a capire se vi è un legame tra la perdita del bagaglio e il suo colore. Nello specifico, dei bagagli presenti nel nostro campione di dati, quanti di un determinato colore ne sono stati persi in percentuale? Nei dati forniti da Youthquake, questa variabile è registrata tramite una codifica numerica. Per associare tali numeri a un colore abbiamo dovuto sfruttare un altro dataset di diversa fonte fornitoci dall'azienda cliente. Questa tabella comprende due variabili, ovvero la codifica e l'effettivo nome del colore (Es. "red").
In seconda battuta effettueremo un'analisi più specifica riguardante alcune rotte che coinvolgono (o in partenza o in arrivo) i tre aeroporti nei pressi di Milano-Bergamo, ovvero: Orio al Serio, Linate, Malpensa. Per la precisione, calcoleremo quanti dei bagagli del nostro campione sono stati persi, in percentuale, nei voli delle rotte aeree che collegano uno dei tre aereoporti milanesi ad uno dei top 30 aereoporti europei per traffico annuale di passeggeri, secondo la classifica stilata dalla "Airport Council International Europe" nel 2018 (anno in cui sono avvenuti tutti i voli su cui disponiamo di informazioni sul bagaglio). Queste due semplici analisi verranno rappresentate tramite infografiche sulla piattaforma di visualizzazione dati "Tableau", previa gestione ed elaborazione dei dati tramite il DBMS "MongoDB", sfruttato direttamente tramite la scrittura di codice su shell di comandi.
MongoDB si è rivelato molto utile grazie alla sua capacità di gestire in maniera

[1] Per motivi computazionali, nonché di privacy, ci è stato fornito solo un campione dei dati effettivamente a disposizione di YouthQuake; le informazioni contenutevi, quindi, riguarderanno alcuni dei bagagli registrati e dei voli effettuati presso tratte europee.

[2] Non vi è alcun dato che possa ricondurre all'identità del passeggero, semplicemente l'ID è legato al singolo bagaglio registrato

estremamente flessibile i dataset utilizzando un paradigma “NoSql”, ma soprattutto grazie alla scalabilità orizzontale, sfruttabile tramite il metodo definito “sharding” e che nel nostro caso è stato utilizzato per “filtrare” il dataset relativo ai voli (che da solo supera i 2GB di volume) sulla base del problema di nostro interesse. Forse vista la mole di dati, si poteva giungere ai risultati ottenuti anche utilizzando la piattaforma MySQL, però abbiamo ritenuto utile creare un’architettura potenzialmente in grado di gestire molti più dati, nell’ipotetico caso in cui servisse svolgere un problema simile ma considerando più voli (per esempio la totalità dei voli mondiali di qualche decennio).

Dati

Il database fornitoci da Youthquake è un database MYSQL che comprende tre dataset: flight, flight_bag, airport.

Database	Table	Size_in_GB
dati_datamanagement	flight	2.21
dati_datamanagement	flight_bag	0.20
dati_datamanagement	airport	0.01

Fig.1

Nella figura sopra (Fig. 1) sono rappresentate le dimensioni dei tre dataset. Come anticipato nel paragrafo predecente, il solo Flight supera i 2GB
Esso comprende 44 attributi, la cui chiave è id_flight, ovvero l’identificativo numerico del volo. Buona parte delle variabili sono legate all’utenza dell’azienda cliente; quelle rilevanti per il nostro problema, oltre alla chiave citata poc’anzi, sono le variabili “departure” e “arrival”, che contengono gli iataCode rispettivamente dell’aeroporto di partenza e di quello di arrivo del volo. Il dataset flight, di cui noi disponiamo solo un campione, è aggiornato da svariati anni e comprende quindi voli europei anche risalenti a una decina di anni fa, a differenza di flight_bag. Quest’ultimo, infatti, contiene l’informazione che più ci interessa, ovvero quella relativa alla perdita o meno del bagaglio, rappresentata dalla variabile binaria “lost” (0=” bagaglio non smarrito”, 1=”bagaglio smarrito”). L’azienda cliente ha iniziato ad essere interessata a questo tipo di informazione per il proprio business solamente dal 2018. Per i nostri scopi accademici, ci sono stati messi a disposizione solamente dati del 2018, ma in ogni caso non sarebbe stato possibile recuperare informazioni sulla perdita del bagaglio che fossero precedenti, poiché non sono mai stati raccolti dall’azienda.
Il dataset flight_bag include 37 attributi, tra cui quelle di nostro interesse sono la già citata lost, “id_flight_bag” che identifica il singolo bagaglio, “id_flight” che indica il volo per cui è stato imbarcato il bagaglio (variabile in comune con il dataset flight) e “bagColor”. L’ultima rappresenta il colore del bagaglio, espresso tramite un numero. Per integrare questa informazione utilizziamo, di conseguenza, anche la tabella (Fig. 2) fornitaci dall’azienda cliente per la codifica.

ColorCode	Color
100	black
110	brown
120	yellow
130	red
140	beige
150	grey
160	green
170	purple
180	blue
190	white

Fig.2

Il dataset airport contiene 55 attributi, di cui ci interessano prevalentemente l’identificativo (“id_airport”), “name”, “city” , “latitude”, “longitude” (utili principalmente per la visualizzazione finale), e soprattutto “iataCode”, che si legherà alle variabili di altre tabelle (o meglio, documenti, dato che per le query utilizzeremo MongoDB, che è un DBMS document-based) in cui è espresso lo iataCode dell’aereoporto di riferimento, al

fine di arricchire le informazioni. Oltre a questi dati, per l'analisi sulla perdita percentuale di bagagli per le rotte che collegano gli aeroporti "milanesi" ai principali aeroporti europei ci serve un riferimento di quali siano effettivamente questi aeroporti. Prendiamo in considerazione a tale scopo la classifica europea [1] stilata dall'ACI (Airports Council International) sui primi 30 aeroporti continentali per numero di passeggeri annuo del 2018. Tale tabella, ricavata tramite web-scraping utilizzando Python, contiene le variabili "Rank", "Code" (iataCode), "City", "Passengers" ed ha lo scopo principale di individuare le principali mete aereoportuali europee, nonché di aggiungere il dettaglio dei passeggeri annuali dell'aeroporto per arricchire l'infografica finale. Milano Malpensa è tra questi 30 aereoporti, verrà dunque esclusa dalla tabella, poiché, banalmente, non siamo interessati a considerare i voli che vanno da Linate a Malpensa o da Bergamo a Malpensa!

Architettura

Come già accennato in precedenza, utilizziamo MongoDB, con il supporto di una macchina virtuale Azure, per costruire un'architettura di dati in grado di garantire la scalabilità orizzontale, doverosa nel caso ipotetico in cui servisse gestire una mole di dati ingente (per esempio dell'ordine dei Terabytes). Mongo DB attua la scalabilità orizzontale tramite il meccanismo di sharding [2], che consiste nel suddividere i dati inseriti nell'architettura in più "shard". Uno shard è un'istanza "mongod" o un "Replica Set" che contiene una certa porzione dei dati totali; la spartizione dei dati tra i vari shard avviene sulla base della chiave del dataset di interesse. Nel nostro caso utilizziamo come chiave di sharding "id_flight" del dataset flight, poiché esso è il dataset che contiene più dati e per cui serve maggiormente un processo "parallelo" nello svolgimento della prima query. La variabile "id_flight" suddividerà i dati seguendo il meccanismo "range-based", basato su intervalli di valori (fig. 3).

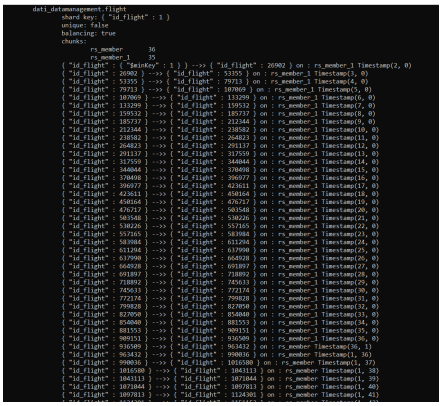

Fig.3

La nostra architettura si compone di nove istanze mongod e un'istanza mongos. Il "router" mongos fa da tramite tra i dati e le query, indirizzando le richieste del client verso lo shard che contiene i dati di interesse. Le nove istanze mongod, invece, compongono i due Replica Set e il CSRS (Config Server Replica Set). I Replica Set [3],ciascuno formato da tre istanze mongod, hanno lo scopo di immagazzinare e replicare i dati. Il nodo (mongod) primario è quello dove vengono caricati i dati [3], mentre i nodi secondari dello stesso Replica Set hanno lo scopo di replicare i dati inseriti nel nodo primario, aggiornandoli ad ogni operazione. La replicazione consente di evitare il rischio di perdere dati qualora si interrompa il nodo primario. In tal caso, esso viene sostituito da un nodo secondario nel ruolo di primario. Creiamo due Replica Set, poiché essi saranno i due shard che si spartiranno i dati sulla base della chiave di sharding scelta (nel nostro caso un RS

[3]I vari dataset, una volta caricati nel mongod tramite la funzione mongoimport (o ,nel caso dei dataset flight,flight_bag,airport, tramite un codice Python, per motivi computazionali), automaticamente diventeranno collezioni di documenti, pur essendo tabellare il formato iniziale

conterrà circa una metà dei voli, l'altro RS l'altra metà). Il Config Server, composto da ulteriori tre istanze mongod, contiene i metadati riguardanti la collocazione dei dati nei vari Replica Set e nei vari "chunks" (sottounità di 64MB degli shard).
L'ideale in una situazione più realistica (ad esempio quella in cui si hanno molti più dati da gestire) sarebbe quella di costruire un'architettura distribuita su più macchine, in modo tale da garantire la tolleranza al guasto. Nel nostro caso, per semplicità, svolgiamo le operazioni su una sola macchina, utilizzando due shard, un config server e un'istanza mongos (fig. 4 [2]).

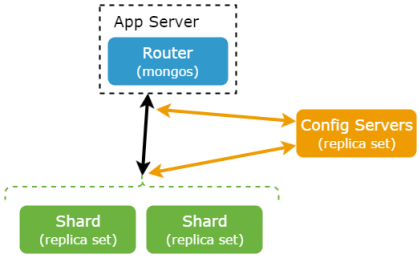

Fig.4

Analisi

Svolgiamo le query accedendo, dalla nostra Virtual Machine (4vCPU, 12GB RAM,128GB Disk,Debian 10 OS), alla porta del mongos e usufruendo del meccanismo di pipeline per unire più operazioni e generare di volta in volta nuovi documenti, fino ad ottenere quelli finali, utili per le infografiche.
Per prima cosa ci occupiamo dell'analisi della perdita percentuale di bagagli per colore. Escludendo dal calcolo le poco più di 1000 osservazioni per cui la variabile "lost" è un missing value, effettuiamo per il dataset flight_bag(documento, dopo averlo importato in MongoDB) sia la somma di tutti i bagagli (che definiamo tramite la variabile "sum_bagages"), sia la somma dei bagagli persi ("sum_losts") raggruppati per colore, tramite una query. Successivamente, integriamo entrambe la tabella ottenute con l'informazione contenuta nella tabella di codifica. Effettuiamo tale operazione tramite un lookup (equivalente del left join in SQL), mantenendo solamente le informazioni relative ai colori presenti nei documenti di codifica ed escludendo anche i missing values.
Ne risulta il documento "lost_color_complete", dove è presente la somma di bagagli e la somma di bagagli persi (da cui si potrà ricavare il rapporto direttamente attraverso la dashboard di visualizzazione) per ognuno dei dieci colori presenti nella tabella di codifica. Il documento verrà estratto tramite codice Python che esporterà il corrispondente JSON tralasciando l'oggetto ID, inutile per le analisi, poiché è già univoco il colore stesso del bagaglio.
In seconda battuta ci occupiamo di ricavare sul nostro campione di voli la percentuale (anche stavolta direttamente dalla visualizzazione) di bagagli persi nelle rotte che collegano i tre aeroporti lombardi alle mete europee presenti nella top 30 di ACI per numero di passeggeri annuo (2018), esclusa Malpensa poiché, di fatto, è uno dei tre aeroporti lombardi. Per questa seconda analisi vi sono più query da svolgere.
Innanzitutto, da "ranking_aci" e "airport" manteniamo, tramite query, solo i dati che ci servono, quindi togliamo Malpensa dalla classifica ACI e di airport teniamo solamente iataCode, nome, città e coordinate geografiche. In seguito, analogamente a quanto fatto per colore e tipologia di bagagli, dal documento flight_bag aggreghiamo i dati in modo da avere per ogni singolo volo la somma dei bagagli e la somma dei bagagli persi tra quelli presenti nel campione, sfruttando ovviamente la variabile lost. Otteniamo così il documento che chiameremo "sum_lost_for_flight".
Successivamente eseguiamo la query più

pesante dal punto di vista computazionale, ovvero quella con cui alleggeriamo la mole di dati presente nel documento flight, sfruttando la chiave di sharding per ottimizzare il processo. Nello specifico, dato che le rotte di nostro interesse coinvolgono aereoporti "milanesi", filtriamo il documento flight estraendo solo quei voli per i quali la partenza o l'arrivo sia presso uno tra Malpensa, Orio al Serio e Linate. Dal documento risultante, tramite due lookup (iataCode comune) con ranking_aci otteniamo due documenti che comprendono le caratteristiche dei voli delle rotte di nostro interesse. In un JSON vi sono le caratteristiche dei voli che partono dagli aeroporti lombardi e arrivano verso una delle mete della classifica ACI, nell'altro il percorso inverso. Di tutte le variabili di flight manteniamo solo quelle che ci servono, ovvero la partenza e l'arrivo (scritte sotto forma di iataCode), nonché il numero annuo dei passeggeri presenti nell'aeroporto di riferimento della classifica ACI e l'identificativo del singolo volo. Sfruttiamo quest'ultimo per effettuare un lookup rispetto a sum_lost_for_flight (ricordiamo che questi JSON hanno in comune appunto l'ID del volo). Dall'operazione di lookup si generano due ulteriori documenti: come prima, una per le partenze da aeroporti lombardi e l'altra per il percorso inverso; per ognuno risulta l'identificativo del volo, la partenza, l'arrivo, il numero ACI dei passeggeri annui per l'aeroporto "non lombardo" e la coppia numero di bagagli/numero di bagagli persi per quel volo. Queste ultime due variabili si ricavano dall'informazione presente in sum_lost_for_flight, la quale nel passaggio successivo viene aggregata per ciascuna rotta, portando ad avere due documenti costituiti dalla coppia partenza/arrivo, dai passeggeri dell'aereoporto non milanese nel 2018 e dalle somme dei bagagli e dei bagagli persi per ciascuna rotta (coppia partenza/arrivo, non più singolo volo).

Gli ultimi due documenti ricavati sono quelli che contengono i dati che ci interessa esporre nella visualizzazione, gli ultimi passaggi consistono nell'arricchirli (per rendere possibile la visualizzazione) e infine nell'unirli. L'arricchimento consiste nell'annidare (il cosiddetto "embedding") alle variabili "departure" e "arrival" un ulteriore documento con all'interno le caratteristiche dell'aereoporto a cui si riferisce lo iataCode espresso. L'embedding avviene tramite un'altra operazione di lookup, che lega lo iataCode dell'aereoporto a quello della variabile della partenza o dell'arrivo. Il risultato di tutte queste operazioni è una coppia di documenti: "milan_deps_complete" (per le rotte che partono dalla Lombardia) e "milan_arrs_complete" (per le rotte di percorso opposto). Ciascuno dei due comprende la coppia partenza/arrivo con annidate le caratteristiche principali dell'aereoporto (quelle selezionate nella query spiegata a inizio paragrafo), nonchè le due variabili numeriche sul numero di bagagli e sul numero di bagagli persi nella rotta ("sum_bagages" e "sum_losts") e il numero di passeggeri annuo del 2018 dell'aereoporto non milanese ("annual_psg_2018").

Abbiamo portato avanti l'aggregazione di due diversi documenti per le partenze da Milano e gli arrivi da Milano solo per comodità dal punto di vista sintattico e computazionale, perché in realtà non ci interessa sapere se il bagaglio sia stato perso per un volo che sia partito da Milano o che sia arrivato a Milano, non sapendo in quale dei due aeroporti il bagaglio viene smarrito (informazione di cui non disponiamo). I rulli con cui vengono gestiti i bagagli in partenza in un aeroporto sono esattamente analoghi a quelli che gestiscono i bagagli in arrivo; di conseguenza non ci interessa distinguere le due situazioni. Dopo aver esportato i due documenti, quindi, tramite un codice Python creiamo il documento definitivo da utilizzare per l'infografica, ovvero un JSON in cui per ogni rotta

(indipendentemente da andata e ritorno) vi sia l'informazione sugli aereoporti e sui bagagli persi, "milan_complete".

Risultati e conclusioni

"lost_color_complete" e "milan_complete" sono i due file JSON da cui faremo le visualizzazioni finali. Sfruttiamo il primo documento per produrre l'infografica in Fig. 5.

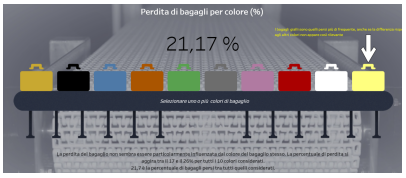

Fig.5

Tale infografica interattiva è finalizzata a segnalare all'utente la percentuale di bagagli di un certo colore che vengono persi. Cliccando uno o più bagagli [4] si ottiene la percentuale di bagagli, aventi il colore selezionato, che sono stati persi. Non vi sono particolari differenze nelle percentuali di perdita tra i diversi colori. Tutti i valori oscillano, infatti, tra il 17,9% (bianco) e il 25,7% (giallo). La domanda di ricerca che ci eravamo posti, ovvero se il colore del bagaglio può influire sulla sua perdita, almeno apparentemente sembra dare un riscontro negativo. Gli utenti più prudenti, però, potrebbero ritenere utili queste informazioni. Poniamo, ad esempio, che un utilizzatore possieda un bagaglio nero (18,20% di bagagli persi, Fig. 6) e uno giallo. Qualora dovesse avere a che fare con questa infografica, potrebbe optare per la prudente soluzione di scegliere il bagaglio nero per il suo prossimo viaggio aereo.

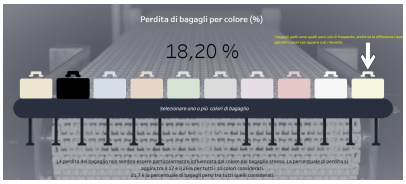

Fig.6

Passando al secondo problema, dal documento milan_complete generiamo un'infografica interattiva (Fig. 7) da cui è possibile selezionare l'aereoporto milanese di riferimento (Malpensa, Orio al Serio o Linate) e verificare quanti bagagli, di quelli di cui disponiamo informazione, sono stati persi in percentuale su ogni singola rotta.

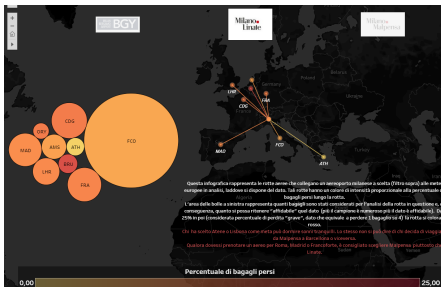

Fig.7

Ogni rotta è colorata di un colore di intensità proporzionale alla percentuale di perdita di bagagli associata alla rotta. La visualizzazione è arricchita con bolle di stesso colore della rotta corrispondente e di area proporzionale al numero di bagagli considerato. Chiaramente, più è ampio il campione considerato dei bagagli di quella rotta, più l'informazione è "affidabile". Escludiamo dalla visualizzazione le rotte per cui si ha un campione di meno di 30 bagagli, poichè l'informazione da esse ricavate è irrilevante e rischia di essere fuorviante. In generale possiamo concludere che i cittadini lombardi che hanno intenzione di viaggiare per mete come Atene e Lisbona possono sentirsi relativamente tranquilli per il proprio bagaglio. Lo stesso non si può dire di coloro che hanno prenotato un volo A/R per Barcellona da Malpensa; in questa rotta viene perso più del 30% dei bagagli (stando ai dati del nostro database). Per le mete di Roma, Francoforte e Amsterdam conviene scegliere Linate piuttosto

[4] Di default sono selezionati tutti i bagagli e la percentuale espressa è quella riferita alla totalità dei bagagli considerati

che Malpensa come aeroporto "milanese". Uno dei limiti di questa visualizzazione è quello di non disporre di informazioni su moltissime delle mete della classifica ACI prendendo come riferimento lombardo Orio al Serio. Dall'infografica, infatti, selezionando l'aereoporto di Bergamo appaiono solo tre rotte, tutte e tre di colore giallo poichè vi è associata bassa percentuale di perdita di bagaglio (sotto l'1%). Tuttavia, per quanto riguarda Linate e, soprattutto, Malpensa, l'utente che usufruisce dell'infografica prodotta può ricavare informazioni interessanti, oltre quelle già citate. Gli basta selezionare uno degli aereoporti di Milano e controllare il colore della rotta di interesse, sperando che questo, dal punto di vista cromatico, sia quanto più vicino possibile al giallo.
Nel complesso le due infografiche sono molto intuitive, anche se le informazioni da esse espresse sono talvolta poco approfondite (vedi per Orio al Serio). Il lavoro risulterebbe molto più interessante avendo a disposizione molti più dati, magari riferiti a voli di diversi anni. Una mole più consistente di dati genererebbe informazioni più affidabili e visualizzazioni più ricche e utili agli utenti finali:passeggeri da un lato, compagnie aeree e aziende produttrici di bagagli dall'altro. Proprio per questo motivo abbiamo costruito un'architettura dotata di scalabilità orizzontale. Così da poter, seppur ipoteticamente, gestire molti più dati e rendere le informazioni espresse dalle infografiche molto più incisive.
Le visualizzazioni citate sono disponibili al seguente link: `https://public.tableau.com/profile/antonio.pennacchia6765#!/vizhome/Iltuobagaglioalsicuroforseoriginal/Viz`

Sitografia

[1] https://bluesyemre.files.wordpress.com/2019/02/aci-europe_top30_2018.pdf

[2] https://docs.mongodb.com/manual/sharding/

[3] https://docs.mongodb.com/manual/replication/