UNIVERSITÀ DI PISA

Laboratory of Data Science Report

Federica Di Pasquale (493195)
email: f.dipasquale1@studenti.unipi.it

Ferri Lorenzo (607828)
email: l.ferri11@studenti.unipi.it

Anno accademico 2020/2021

Indice

1 Part 1

1.1 Assignment 0

Il Constellation Schema assegnato è stato riprodotto mediante la query presente nel file "01_DB_creation.sql". Valori nulli sono ammessi solo per gli attributi diversi dalle Chiavi Primarie presenti nelle Dimension Tables. Nelle tre Fact Tables non sono ammessi valori nulli per le Chiavi Esterne, al fine di non perdere ennuple a seguito di uno Star-Join.
I valori della Chiave Primaria della Dimension Table Time, "time_code", sono stati trasformati nel formato yyyy-mm-dd per renderli compatibili con il Data Type "Date" di SQL Server. Sono stati imposti vincoli di Chiave Primaria nelle Dimension Tables e di Chiave Esterna nelle Fact Tables, dunque i file vengono forniti nell'ordine in cui devono essere eseguiti per riempire prima le Dimensions e poi i Facts.

1.2 Assignment 1

Il file "fact.csv" contenente l'intera Fact Table, è stato suddiviso in tre Fact Tables separate, ciascuna per ogni linea di prodotto (cpu, gpu, ram), tramite il file "04_fact.py".
Date le dimensioni complessive della Fact Table, si è scelto di non creare tre file ".csv" ma di suddividere e caricare immediatamente il loro contenuto nel Constellation Schema.
A tale scopo è stato fatto uno scan dell'intero file durante il quale si individua per ogni record la Fact Table corrispondente; si è scelto di non caricare nel DW un record per volta, ma di salvare il loro contenuto in tre liste separate e solo al termine effettuare tre chiamate "executemany(sql, *params)".
Tale scelta si è rivelata molto più efficiente rispetto all'esecuzione di "execute(sql, *params)" effettuata su ogni singolo record; è necessario tuttavia notare che è stato possibile procedere in questo modo solo perché le dimensioni della Fact Table in esame non sono così elevate, mentre in generale sarebbe necessario bilanciare il contenuto caricato in memoria e le chiamate al DB.

1.3 Assignment 2

Le Dimension Tables sono state riempite con i due file: "02_dimensions.py", per tutte le dimensioni tranne Time, e "03_time_dimension.py", per la sola dimensione Time.

- **Dimensions:**
  Una volta effettuata la connessione al DB e definita una query parametrica, sono stati letti i file .csv contenenti le Dimension Tables tramite la funzione "reader" della libreria standard "csv", caricando uno per volta i record così ottenuti.

- **Time Dimension:**
  Sono state preparate le funzioni getQuarter(month) e getDayOfWeek(day, month, year), che ottengono rispettivamente il quarter dato il mese e il giorno della settimana data la

data del giorno. La seconda funzione ha l'obiettivo di calcolare il resto della divisione di x/7, dove x è dato dalla formula:

$$x = year + (year - 1)//4 - (year - 1)//100 + (year - 1)//400 + t \tag{1}$$

questo resto sarà l'indice del giorno della settimana partendo da 'Sabato'. Per ottenere x c'è bisogno di calcolare t, cioè il numero di giorni trascorsi dall'inizio dell'anno. Per farlo si è diviso il calcolo in:

$$t = daysToMonth(month, year) + day \tag{2}$$

dove la funzione daysToMonth(month, year) è una recursione che, dato il mese e l'anno (che serve per verificare se l'anno è bisestile), calcola il numero di giorni trascorsi fino all'inizio del mese corrente sommando i giorni dei mesi a ritroso lungo l'anno.

Il riempimento della tabella Time è avvenuto leggendo il contenuto del file "time.csv" e assegnandolo ad una Dictionary con la funzione DictReader(). Dopo essersi connessi al DB ed aver definito la query parametrica di inserzione, si è iterato sulle righe presenti nella Dictionary per ottenere i valori da passare ai parametri della query che ha riemepito il DB, applicandovi prima le funzioni sopra citate quando necessario.

2 Part 2

Si vuole rispondere ad alcune *business questions* utilizzando *Sequel Server Integration Services* (SSIS) e sviluppando delle soluzioni che calcolino i risultati dal lato client. In particolare è stata sviluppata un' unica soluzione contenente tre progetti relativi alle tre *business questions* assegnate.

2.1 Assignment 0

Business Question: For every year, the brand of gpu ordered by sales.

Tramite il nodo "Origine OLE DB" sono state lette solo le colonne necessarie della tabella Gpu_sales (gpu_code, time_code, sales_usd). Con il nodo "Ricerca" è stata operata la giunzione (tramite indice) con la colonna "brand" della tabella Gpu_product. Anziché operare una nuova ricerca per l'attributo "year" nella tabella Time, si è sfruttato il nodo "Colonna derivata" con cui si è estratto l'anno dal "time_code": quest'operazione permette un risparmio computazionale ed è stata resa possibile dalla trasformazione dell'attributo time_code effettuata al momento del popolamento della DW. E' stato poi eseguito il raggruppamento per gli attributi "year" e "brand", calcolando la funzione di aggregazione SUM(sales_usd) con nome "tot_sales_usd". Infine si è ordinato il risultato per (year, tot_sales_usd), in modo da avere per ogni anno la somma dei sales ordinata, come da richiesta. Il risultato finale viene scritto su un file di testo.

2.2 Assignment 1

> *Business Question:* For any given country, a product is said to have full regional spread if it was sold in all the regions of that country. List all the AMD brand cpus that do not have full regional spread in Germany.

Per rispondere a questa domanda, è stato necessario accedere alle tre tabelle Cpu_product, Cpu_Sales e Geography. Inizialmente è stato fatto un accesso alla tabella Cpu_Sales tramite un nodo "Origine OLE DB" selezionando solo le colonne necessarie. Successivamente, tramite un nodo "Ricerca" è stata recuperata la colonna relativa al "brand" delle Cpu dalla tabella Cpu_Product. In questo modo è stato possibile selezionare solo le cpu relative al brand "AMD" attraverso un nodo "Conditional Split".
Dal momento che si cercano le cpu che non hanno "full regional spread" è stato necessario recuperare anche le informazioni relative al luogo di vendita. A questo scopo, tramite un ulteriore nodo "Ricerca", sono state recuperate le colonne "Country" e "Region" dalla tabella Geography (utilizzando geo_code) e sono state selezionate solo le cpu vendute nella nazione Germania. Successivamente il flusso di dati è stato suddiviso tramite un nodo "Multicast" in modo da poter calcolare nei due branch rispettivamente il numero di regioni distinte per ogni nazione e, per ogni cpu e nazione, il numero di regioni distinte in cui è stata venduta tale cpu. Una volta che sono state calcolate tutte le informazioni necessarie, è stato possibile selezionare le cpu che non hanno "full regional spread" facendo una Join fra i risultati dei due flussi di dati precedenti (preventivamente ordinati rispetto a "Country") e infine selezionando solo le cpu per cui il numero di regioni distinte in cui sono state vendute è minore del numero totale di regioni distinte per nazione. Infine il risultato è stato scritto in un file tramite un nodo "Destinazione file flat".
Nel caso specifico di questa Business question, l'ultimo ordinamento rispetto a "Country" risulta superfluo, in quanto è stata precedentemente selezionata solo la regione Germania; tuttavia si è preferito costruire una soluzione più generale che funzioni anche nel caso in cui non ci si voglia restringere ad una sola nazione.

2.3 Assignment 2

> *Business Question:* Calculate which processor manufacturer yields the most sales, for each country and year.

Analogamente ai casi visti fin quì, dalla Fact Table "Gpu_sales" si ottengono gli attributi geo_code, time_code, gpu_code e sales_usd. country e processor_manufacturer si ottengono mediante una ricerca con Indice, mentre year si ottiene di nuovo con una funzione applicata su time_code. A questo punto sono necessarie due aggregazioni successive: una per (country, year, processor_manufacturer), per calcolare la somma dei sales; e un'altra per (country, year), in modo da ottenere il valore massimo della somma dei sales già calcolata, per ogni country e year. La prima aggregazione viene calcolata subito e dopo di essa la tabella viene duplicata con un nodo di "Multicast". In questo modo la seconda aggregazione può essere prodotta solo su una delle due tabelle identiche, lasciando l'altra invariata. Le due tabelle vengono poi ricongiunte con un nodo "Merge join" su uguaglianza degli attributi country e year (previo ordinamento per tali attributi). La tabella risultante conterrà sia la

somma dei sales per l'aggregazione (country, year, processor_manufacturer) che il valore della somma dei sales massima ripetuto per ogni country e year distinti. Basterà quindi imporre un'uguaglianza tra queste due metriche per ottenere solo le ennuple dove la somma dei sales è effettivamente pari a quella massima. Il risultato viene scritto su file di testo.

3 Part 3

3.1 Assignment 0

E' stato creato un *datacube* contenente i dati relativi alle tabelle "Gpu_Sales", "Gpu_product", "Vendor", "Geography" e "Time" tramite *SSAS* (SQL Server Analysis Services).
Dopo aver stabilito una connessione al Database tramite protocollo http, è stato settato come origine dati il DW precedentemente creato. Successivamente, allo scopo di sfruttare le gerarchie presenti in "Time", sono stati creati tre nuovi attributi ("day_forTheHierarchy", "week_forTheHierarchy" e "month_forTheHierarchy") concatenando rispettivamente i valori di year-month-week-day, year-month-week e year-month, in modo da introdurre, per queste colonne derivate, le dipendenze funzionali necessarie a definire la gerarchia.
Essendo i giorni, le settimane e i mesi rappresentati come numeri interi, prima di effettuare la concatenazione di questi valori, è stato anteposto uno 0 a tutte le quantità minori di 10, che ha permesso una migliore visualizzazione ed un corretto ordinamento dei dati. Tuttavia notiamo che, dal momento che negli attributi originali le settimane sono espresse come interi nell'intervallo [1, 52], la dipendenza funzionale week $\rightarrow$ month era già presente nel DW, ma si è scelto comunque di concatenare i valori di week e month per una rappresentazione più coerente dei dati.
Sono state successivamente definite le dimensioni del cubo e le gerarchie relative a "Time" e "Geography". Per quanto riguarda "Geography", le gerarchie presenti sono:

$$\text{Region} \rightarrow \text{Country} \rightarrow \text{Continent} \qquad \text{e} \qquad \text{Country} \rightarrow \text{Currency} \tag{3}$$

Grazie agli attributi precedentemente creati è stato possibile definire la seguente gerarchia per "Time":

$$\text{day} \rightarrow \text{week} \rightarrow \text{month} \rightarrow \text{year} \tag{4}$$

Sono inoltre presenti anche le seguenti dipendenze funzionali:

$$\text{day} \rightarrow \text{day\_of\_week} \qquad \text{e} \qquad \text{month} \rightarrow \text{quarter} \tag{5}$$

che, per una maggior pulizia dei dati, sono state inserite ma non incluse nella gerarchia principale. Tutte le dipendenze funzionali sono state definite aggiungendo delle relazioni tra attributi di tipo "rigido", cioè che non cambiano nel tempo.
Infine è stato creato il *datacube* mantenendo come misure quelle fornite di default dal sistema.

3.2 Assignment 1

> Show the total sales for each country and vendor and the grand total with respect to the continent.

E' stata scritta una query MDX che riporta sulle colonne la misura "Sales_usd" e sulle righe le dimensioni relative al paese e ai venditori. In particolare si vogliono visualizzare i ricavi totali di ogni venditore per ogni paese, ed il grand total rispetto al continente. E' stato dunque utilizzato un set che contiene due tuple: la prima permette la visualizzazione di tutte le coppie (paese, venditore) ed è scritta come: ([Geography].[Geography_Hierarchy].[Country], [Vendor].[Name].[Name]); la seconda permette invece di visualizzare il grand total per ogni continente: ([Geography].[Geography_Hierarchy].[Continent], [Vendor].[Name].[All]).

3.3 Assignment 2

> Let diff be the difference between the sales usd and sales currency. Show the total sales usd, total sales currency, total diff for each month and the running diff starting from the same year in Germany.

Per visualizzare il risultato richiesto è stato necessario scrivere una query MDX che calcola due membri derivati: "diff" e "running_diff". Il primo è stato calcolato semplicemente come differenza tra "Sales_usd" e "Sales_Currency", mentre per il secondo è stato necessario utilizzare la funzione PERIODSTODATE per specificare l'intervallo su cui aggregare la misura "diff" precedentemente definita. In particolare si vuole il "running_diff" a partire dall'inizio dell'anno corrente, dunque l'intervallo è stato specificato come: PERIODSTODATE([Time].[Time_Hierarchy].[year], [Time].[Time_Hierarchy].currentmember), diff).
Infine ci si vuole restringere a considerare solo i dati relativi alla Germania, dunque è stata aggiunta una clausola WHERE che permette di effettuare un'operazione di SLICE che seleziona solo la Germania.

3.4 Assignment 3

> Show the top 5 gpu brands w.r.t the monthly average sales for each region in Europe.

E' stata scritta una query MDX che calcola come membro derivato "monthly_avg_sales" tramite la funzione AVG a cui vengono passati due argomenti: il primo è l'espressione che definisce il set dei mesi, e il secondo è la misura "Sales_usd" di cui si vuole calcolare la media. Per mostrare il risultato richiesto, è necessario utilizzare la funzione GENERATE che permette il calcolo dei top 5 gpu brands rispetto al "monthly_avg_sales" per ogni regione e restituisce l'unione dei risultati ottenuti.
Infine è stata aggiunta una clausola WHERE che permette di selezionare il continente Europa.

3.5 Assignment 4

E' stato usato il "Grafico ad aree" dove sulle ordinate abbiamo la somma dei "Sales Usd" e sulle ascisse gli attributi della gerarchia "Time Hierarchy", in ordine crescente. E' possibile navigare la gerarchia con le frecce in alto nel grafico per ottenere l'andamento dei Sales per ogni anno, mese, settimana e giorno. Anche se non richiesto, l'attributo "Processor Manufacturer" è stato aggiunto alla legenda in modo da ottenere i Sales per ognuna di quelle

categorie nello stesso grafico. Quest'ultima caratteristica può comunque essere facilmente esclusa se non interessa.

3.6 Assignment 5

E' stata usata la "Mappa" di Bing in stile "Aereo". La somma di "Sales Usd" ed il "Conteggio di Gpu Sales" (misura aggiunta appositamente solo nel Cubo OLAP) possono essere visualizzati spostandosi sopra alle Bolle, il cui raggio è proporzionale a "Sales Usd". La gerarchia "Geography Hierarchy" può essere navigata tramite le frecce in cima al grafico, in modo da visualizzare le Bolle per ogni continente, stato e regione. Anche in questo caso, per rendere la visualizzazione più interessante, è stato aggiunto il "Name" del Vendor alla legenda, pur non essendo richiesto. Le Bolle dunque assumono la forma di grafico a torta.

3.7 Assignment 6

Il primo grafico è di tipo "Grafico a linee ed istogramma a colonne in pila". Viene rappresentato l'andamento di "Sales_usd" sovrapposto ad un istogramma che rappresenta il numero di gpu vendute, al variare di tutti i brand. In questo modo è possibile fare delle analisi riguardo i prezzi delle gpu dei vari brand, infatti, confrontando il guadagno totale a parità di numero di prodotti venduti, si può avere un'indicazione su quali siano i brand più o meno costosi. Ad esempio si nota come il brand AMD sia quello con il rapporto più favorevole tra guadagno e numero di prodotti venduti, quindi probabilmente si tratta del brand con i prodotti più costosi. Analogamente i brand ASUS e Gygabyte sembrano essere quelli relativi ai prodotti meno costosi.

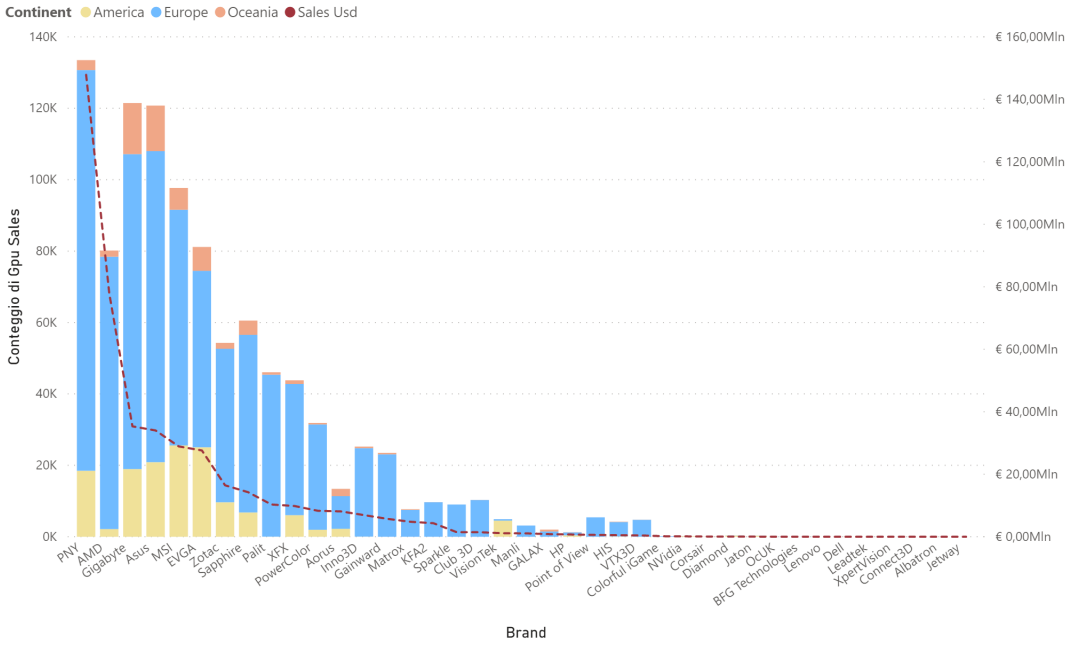

Figura 1: Sales_usd e conteggio per brand

Inoltre è stata aggiunta all'istogramma l'informazione relativa al continente, rappresentando su ogni colonna la distribuzione del numero totale di gpu vendute tra i diversi continenti.

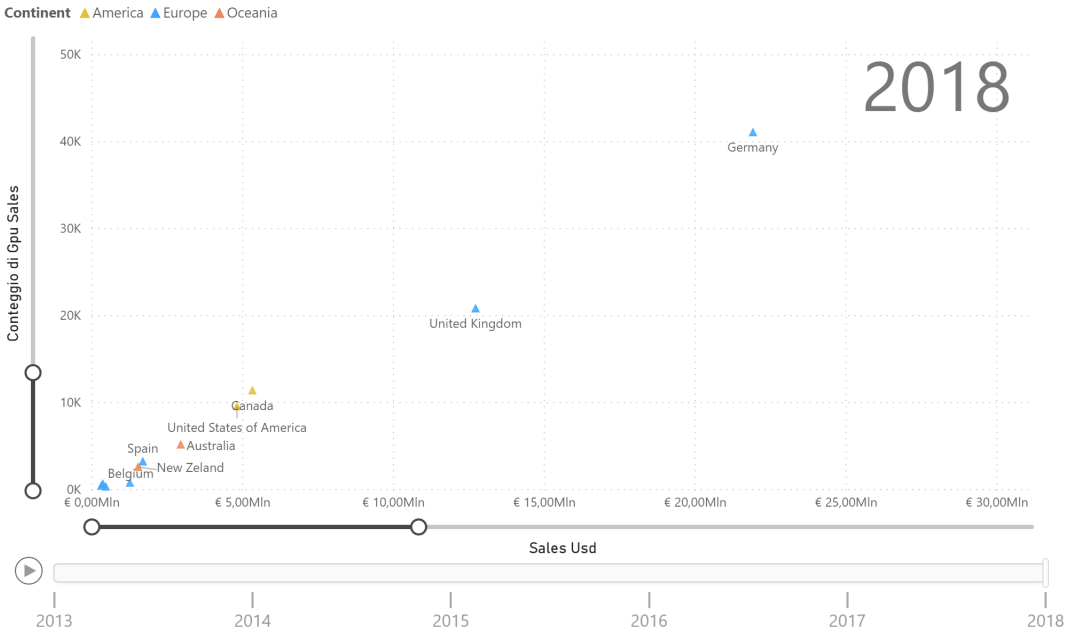

Figura 2: Sales_usd e conteggio per Country

Il secondo grafico è invece di tipo "Grafico a dispersione". Anche in questo caso si vogliono fornire delle informazioni sui prezzi dei vari prodotti, ma questa volta in relazione al paese in cui sono venduti. Ogni paese è dunque rappresentato da un punto in un grafico 2D le cui coordinate sono il guadagno "Sales_usd" sulle ascisse ed il conteggio dei prodotti venduti sulle ordinate. Come era possibile prevedere i vari paesi si distribuiscono approssimativamente su un retta a pendenza positiva, segno che mediamente un maggiore guadagno corrisponde ad un numero più elevato di gpu vendute. Tuttavia, si è scelta proprio questa rappresentazione perché, confrontando la posizione di un paese rispetto all'andamento medio, è possibile visualizzare meglio l'eventuale presenza di paesi il cui prezzo medio di vendita si discosta notevolmente da quello degli altri paesi. E' stata poi aggiunta una legenda che permette di individuare più facilmente il continente di ogni paese. Infine il grafico è di tipo dinamico, ovvero è possibile visualizzare dinamicamente il cambiamento al trascorrere degli anni, oppure selezionare l'anno di interesse.