

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE



CORSO DI LAUREA MAGISTRALE IN INFORMATICA
A.A. 2017-18

Progetto Basi di Dati II Modulo B

Autori

Bizzarri Flavio
N97000281

Cuomo Daniele
N97000270

Sommario

Si vuole realizzare un data warehouse destinato all'analisi di dati riguardanti misurazioni effettuate su veicoli da parte dell'Istituto Motori di Napoli. Le misurazioni effettuate si concentrano sull'emissione di particelle di inquinanti in tratti stradali misti e presentano anche la componente spaziale rappresentata dalle coordinate GPS dei rilevamenti. Il DW realizzato ed esposto in questo documento è di tipo ROLAP, implementato con PostgreSQL, il quale fornisce sia tutte caratteristiche necessarie al data warehousing sia un'estensione spaziale.

Indice

| | | |
|----------|--|----------|
| 1 | Progettazione | 1 |
| 1.1 | Interrogazioni | 1 |
| 1.2 | Diagrammi | 2 |
| 1.3 | ETL | 3 |
| 1.4 | Viste Materializzate | 4 |
| 2 | Analisi | 5 |
| 2.1 | ETL | 5 |
| 2.1.1 | Trasformazione in CSV | 5 |
| 2.1.2 | Import in tabella temporanea | 6 |
| 2.1.3 | Import nello schema | 6 |
| 2.1.4 | Aggiornamento degli indici | 7 |
| 2.1.5 | Aggiornamento delle viste materializzate | 7 |
| 2.2 | Analisi performance query | 7 |
| 2.2.1 | Query 1 | 7 |
| 2.2.2 | Query 2 | 7 |
| 2.2.3 | Query 3 | 8 |
| 2.2.4 | Query 4 | 8 |
| 2.2.5 | Query 5 | 8 |
| 2.2.6 | Query 6 | 8 |
| 3 | Conclusioni | 9 |

Capitolo 1

Progettazione

1.1 Interrogazioni

| # | Query | Obiettivo |
|---|--|--|
| 1 | Impatto ambientale medio in corse da 5 km (CO2 e NOx, massa) | Pensata per analizzare varianti della stessa interrogazione su diversi livelli di granularità |
| 2 | Consumo medio per intervalli di velocità prefissati, su tutto il dataset | Utile all'implementazione e l'analisi di viste materializzate che raggruppano i dati secondo delle fasce di velocità. Le fasce scelte, espresse in km/h, sono le seguenti: 0-50, 50-90, 90-130 |
| 3 | Efficienza dell'auto per intervalli di RPM (rotazioni per minuto) | Altra interessante interrogazione creata allo scopo di sfruttare le viste materializzate. Il dataset fornisce tutti i parametri necessari al calcolo dell'efficienza, o rendimento istantaneo |
| 4 | Per ogni test, media di NOx, CO2, Potenza e Velocità | Quest'interrogazione serve a valutare le prestazioni ottenute dall'esecuzione su di un partizionamento verticale con le colonne sparse tra più tabelle |
| 5 | Media e deviazione standard delle temperature | Quest'interrogazione serve a valutare le prestazioni ottenute dall'esecuzione con le colonne concentrate su di un'unica tabella restituita da un partizionamento verticale |

Le query sopra riportate hanno guidato lo sviluppo del sistema in ogni sua fase e hanno permesso di sperimentare differenti tecniche di implementazione di un sistema ROLAP. Una definizione formale delle stesse in linguaggio SQL sarà presentata più avanti nel corso di questo documento.

1.2 Diagrammi

Di seguito è riportato il diagramma UML che rappresenta lo schema dei fatti implementato secondo il modello relazionale ROLAP.

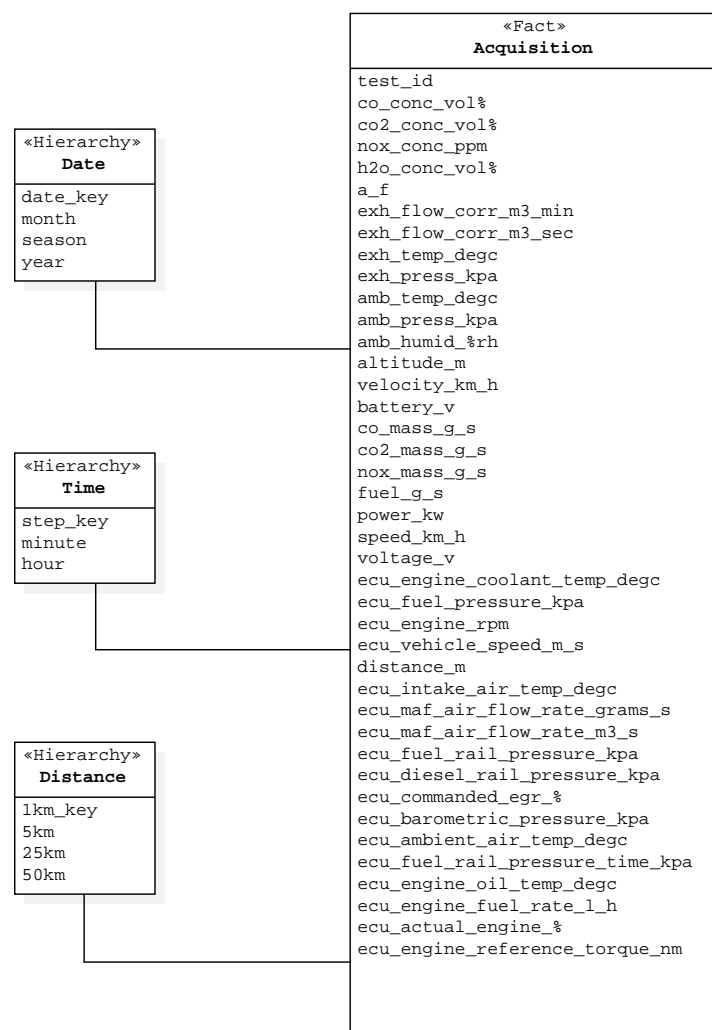


Figura 1.1: Diagramma UML schema dei fatti

È stata inoltre progettata e implementata una variante dello schema proposto che sfrutta la tecnica del partizionamento verticale ovvero la possibilità di dividere la tabella dei fatti in più tabelle, ognuna delle quali rappresenta una particolare sfaccettatura del *fatto*. Il partizionamento viene solitamente adoperato per agevolare quelle interrogazioni che riguardano solo una particolare area di interesse. Questa tecnica presenta però l'inconveniente di dover aggiungere una chiave tra le tabelle al fine di poterle ricongiungere per analizzare il dato nella sua completezza e la duplicazione delle chiavi surrogate associate alle dimensioni di analisi.

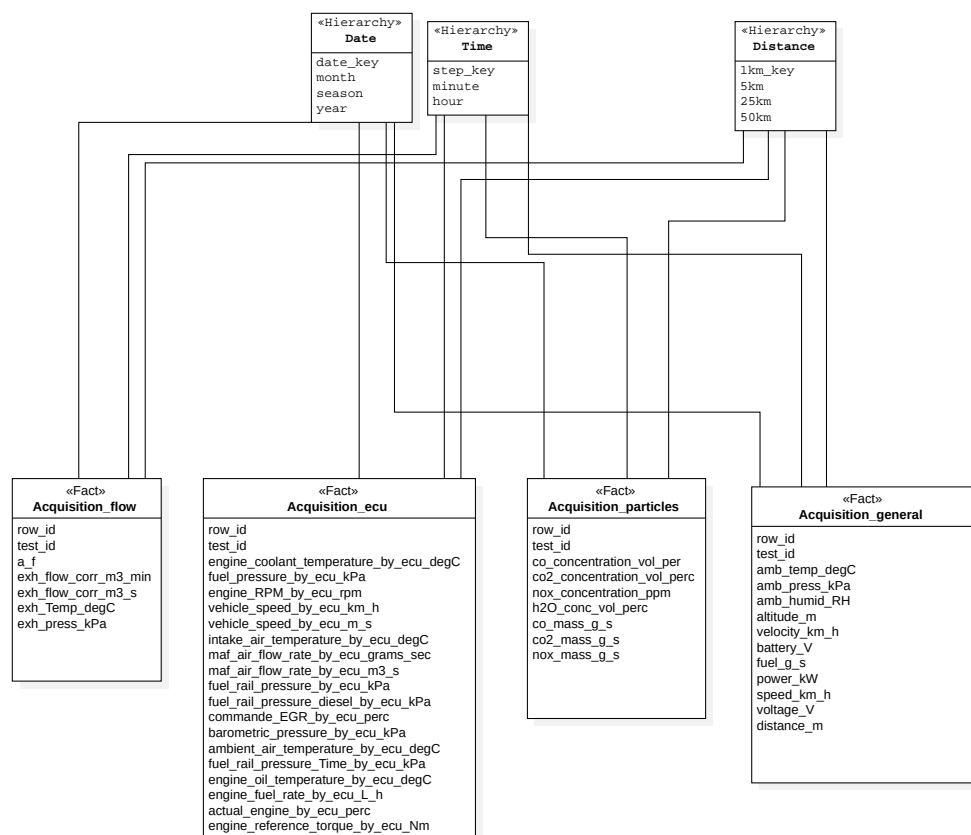


Figura 1.2: Diagramma UML schema dei fatti partizionato verticalmente

1.3 ETL

Al fine di avere un numero rilevante di dati per l'analisi dei tempi è stato implementato un meccanismo di duplicazione dei file. La procedura ETL sviluppata si compone di differenti passaggi descritti nella seguente tabella.

| # | Passaggio | Descrizione | Implementazione |
|---|--|---|-----------------|
| 1 | Trasformazione dei file XLSX in formato CSV | Eliminazione delle righe contenenti dati inconsistenti | Java |
| | | Calcolo dei campi formula | |
| | | Creazione di una data fittizia | |
| | | Creazione di un test_id | |
| | | Salvataggio in formato CSV | |
| 2 | Importazione dei dati in tabella provvisoria | Viene sfruttata il comando copy del DBMS per importare i dati in maniera efficiente | PostgreSQL |
| 3 | Shutdown degli indici | | PostgreSQL |
| 4 | Aggiornamento dello schema | Aggiornamento delle dimensioni Aggiornamento tabella dei fatti | PostgreSQL |
| 5 | Riattivazione degli indici | | PostgreSQL |
| 6 | Aggiornamento viste | | PostgreSQL |

Per righe inconsistenti si intendono tutte quelle righe ove per valori indicanti volumi e/o concentrazioni vi sono valori negativi; inoltre viene dedotta la distanza percorsa ove mancante sfruttando la distanza percorsa e le velocità istantanee rilevate negli istanti precedenti.

1.4 Viste Materializzate

Sono state create le seguenti viste materializzate al fine di implementare efficacemente le query 3 e 4. TODO codice viste qui

Capitolo 2

Analisi

In questo capitolo sono riportati i risultati e i tempi ottenuti per ogni fase: dalle procedure ETL all'esecuzione delle query.

2.1 ETL

2.1.1 Trasformazione in CSV

In questa sezione copriremo un'analisi dei tempi ottenuti nella fase che permette, a partire dai dati originali, di ottenere un file in formato CSV contenente i dati ripuliti da inconsistenze e in un formato adatto ad essere importato nel Datawarehouse. In particolare nella fase di pulizia, a partire dal file XSLX, si estraggono unicamente i record dove non appaiono valori negativi per quantità intrinsecamente positive (concentrazione, volume, ecc.); per i record ove manca il valore "relative" e/o la distanza percorsa si procede ad un calcolo a partire dall'ultima rilevazione valida estratta. Inoltre tutte le righe che presentano una chiara assenza di dati (70% delle colonne) vengono scartate. Infine, durante la fase di trasformazione, per ogni file viene generata una data e un test_id e per separare la parte intera da quella decimale si sostituisce la virgola con il punto.

I dati ottenuti si riferiscono ad una media aritmetica ottenuta testando 200 file (~1 milione di righe) generati a partire dal file originario fornito dall'Istituto Motori di Napoli.

| Righe XSLX | Righe CSV | Righe perse | Peso file XSLX | Peso file CSV |
|------------|-----------|-------------|----------------|---------------|
| 5764 | 5706 | 1% | 2.138KB | 2.283KB |

| Fase di pulizia | Fase di trasformazione | Tempo totale |
|-----------------|------------------------|--------------|
| 3.04s | 0,05s | 3,05s |

Questa fase mette in luce la buona qualità dei dati forniti dall'istituto: ci si aspetta in media di perdere pochissimi dati a causa di inconsistenze. Per quanto riguarda invece

le dimensioni si nota come il formato Comma-separated values sia leggermente meno efficiente nella compressione rispetto al formato proprietario di Microsoft® ma al tempo stesso permetta una più veloce elaborazione.

2.1.2 Import in tabella temporanea

Al fine di importare i file CSV nella tabella dei fatti ci si appoggia ad una tabella temporanea al fine di agevolare le operazioni successive. Questa tabella viene troncata alla fine della procedura. L'import sfrutta la funzione *COPY* messa a disposizione dal DBMS PostgreSQL.

| Numero file | Numero righe | Tempo |
|-------------|--------------|---------|
| 1 | 5706 | 214ms |
| 50 | 285.300 | 8,428s |
| 100 | 570.600 | 16.522s |

Tabella 2.1: I dati si riferiscono ad una media di 5 esecuzioni

I risultati mostrano la bontà della funzione *COPY* che sfruttando un inserimento batch di 1000 righe per volta riesce ad abbattere in modo consistente i tempi.

2.1.3 Import nello schema

In questa fase i dati, precedentemente inseriti in una tabella temporanea, vengono travasati nello schema proposto dopo aver disabilitato tutti gli indici. Le prove effettuate prendono in considerazione diverse dimensioni della tabella dei fatti oltre al caso in cui lo schema risulti partizionato.

| Dimensione tab. fatti | Numero righe importate | Partizionamento | Tempo |
|-----------------------|------------------------|-----------------|--------|
| 0 | 285.300 | No | 3,925s |
| 0 | 285.300 | Si | 7,753s |
| 570.600 | 285.300 | No | 3,730s |
| 570.600 | 285.300 | Si | 7,404s |
| 1.141.200 | 285.300 | No | 4,535s |
| 1.141.200 | 285.300 | Si | 8,572s |

Tabella 2.2: I dati si riferiscono ad una media di 5 esecuzioni

Dalla tabella emerge come il partizionamento comporti quasi un raddoppio del tempo necessario all'inserimento: ciò è dovuto al dover spalmare un singolo record della tabella temporanea su 4 differenti tabelle. Questo slow-down è risolvibile pensando a 4 inserimenti in parallelo: infatti le 4 tabelle, anche se logicamente collegate, durante l'inserimento non necessitano di condividere alcuna informazione.

2.1.4 Aggiornamento degli indici

In questa fase vengono riattivati gli indici delle chiavi tra tabella dei fatti e dimensioni. Questi indici sono assolutamente necessari per velocizzare le query ma possono essere disabilitati durante la fase di update dello schema.

| Dimensione tab. fatti | Partizionamento | Tempo |
|-----------------------|-----------------|---------|
| 285.300 | No | 2,530s |
| 285.300 | Si | 11,534s |
| 570.600 | No | 6,194s |
| 570.600 | Si | 24,715s |
| 1.141.200 | No | 16,903s |
| 1.141.200 | Si | 67,469s |
| 1.426.500 | No | 24,141s |
| 1.426.500 | Si | 98,218s |

Tabella 2.3: I dati si riferiscono ad una media di 5 esecuzioni

Dai dati sopra mostrati emerge ancora una volta come l'introduzione di un partizionamento verticale comporti un notevole rallentamento. In particolare la forbice tra i tempi registrati aumenta all'aumentare della dimensione dei fatti. Ciò induce a pensare attentamente all'introduzione di un partizionamento in fase di progettazione valutando il rapporto costo/benefici.

2.1.5 Aggiornamento delle viste materializzate

TODO.

2.2 Analisi performance query

In questa sezione verranno analizzate le query implementate nelle differenti implementazioni.

2.2.1 Query 1

TODO.

2.2.2 Query 2

TODO.

2.2.3 Query 3

TODO.

2.2.4 Query 4

TODO.

2.2.5 Query 5

TODO.

2.2.6 Query 6

TODO.

Capitolo 3

Conclusioni