Tesina Basi Di Dati

Pasquale del Moro, Dario Di Meo

**Indice**

1. Introduzione
2. Normalizzazione Master Table
3. Creazione Basi Di Dati e comandi DDL
4. Arrichimento base dati, alter table
5. Modello E/R
6. Analisi dei dati
   1. Query
   2. Grafici
7. Stored Procedure
8. Ottimizzazione Base di dati

**Appendice**

1. Creazione Master table
2. Fonti dei dati
3. **INTRODUZIONE**

La Tesina tratta la Creazione, la gestione e l’Analisi dei dati di un’ DB sui contagi **COVID-19** delle Provincie e Regioni italiane dal 24 Febbraio al 3 Maggio 2020.

Si tratta di una Base di Dati **Direzionale,** il cui scopo è essere una Base di dati attiva, con la quale poter poi interagire per l’analisi dei dati ed eventualmente costruire grafici e visualizzarli.

Il progetto parte da dei dati su regioni e province italiane, forniti dalla **Protezione Civile Italiana**, da convertire in tabelle(Master Table), da cui grazie il processo di Normalizzazione, si ricava lo schema principale della base dati. Era nostro compito aggiungere dati utili alle analisi per arricchire il nostro schema, **Stored Procedure** e i **Trigger** che rendono attiva la base dati, automatizzando alcuni processi che si sono ripresentati molto frequentemente durante la creazione di **Query** e **grafici**, a loro volta utili alla visualizzazione e all’analisi dei dati.

Una volta creato il supporto fisico di analisi dei dati, cioè il DataBase, abbiamo migliorato le prestazioni di alcune Query ricorrenti, attraverso la definizione di **Indici** e **Viste**.

Oltre alla realizzazione Fisica sul ambiente di sviluppo **Oracle SQL Developer,** parte della tesina è dedicata alla definizione, tramite un processo di **Reverse Engineering**, dello **schema logico** E/R, per rappresentare la realtà di nostro interesse.

1. **Normalizzazione Master Table**
2. **Creazione Basi Di Dati e comandi DDL**
3. **Arrichimento base dati, alter table**
4. **Modello E/R**

PARCHI

LAVORO

FARMACIE

VARIAZIONE\_SPOSTAMENTO

SVAGO

DATA

ETC…

(1,1)

**LUOGO\_**

**MISURAZIONE**

**3**

**MISURA MOVIMENTI**

CODICE\_REGIONE

DENOMINAZIONE\_PROVINCIA

CODICE\_PROVINCIA

DENOMINAZIONE\_REGIONE

CODICE\_REGIONE

(1,N)

ETC…

TAMPONI

DECEDUTI

TERAPIA\_INTENSIVA

TOTALE\_CASI

(0,N)

DATA

LONGITUDINE

LATITUDINE

POSIZIONE

(0,N)

**LUOGO\_**

**MISURAZIONE**

**2**

CODICE\_PROVINCIA

CODICE\_REGIONE

(1,1)

EN

ITA

NOTE

(1,1)

**SITO**

**MISURA PROVINCIA**

**MISURA REGIONE**

**MISURA COVID**

**REGIONE**

**PROVINCIA**

**LUOGO\_**

**MISURAZIONE**

**1**

(1,1)

(1,N)

ETC…

ETC…

DATA

1. **Analisi dei dati**

L’analisi dei dati, si avvalora di Query e ad alcuni tool Grafici, per mostrare i dati e le relazioni tra essi.

* **Query e Grafici sulle variazioni di Spostamenti Regionali:**

**--Query parametrizzata per il report in formato CSV sui movimenti di una regione presa in INPUT**

**CREATE OR REPLACE PROCEDURE** report\_movimenti\_regione (regione\_in **IN** regioni.denominazione\_regione%TYPE)

**AS**

found\_regione **INTEGER**;

**CURSOR** cursore\_regione **IS** (

**SELECT** mov.codice\_regione, mov.data, mov.variazione\_svago\_vendita\_dettaglio, mov.variazione\_farmacie\_alimentari, mov.variazione\_parchi, mov.variazione\_stazioni\_trasporto\_pubblico, mov.variazione\_sedi\_lavoro, mov.variazione\_zone\_residenziali

**FROM** movimenti\_regioni mov **JOIN** regioni r **ON** mov.codice\_regione = r.codice\_regione

**WHERE** **LOWER**(denominazione\_regione) = **LOWER**(regione\_in)

);

riga\_movimenti movimenti\_regioni%**ROWTYPE**;

regione\_not\_found **EXCEPTION**;

**BEGIN**

**--Controllo che la regione sia presente**

**SELECT** count(distinct(codice\_regione)) **INTO** found\_regione

**FROM** regioni

**WHERE LOWER**(denominazione\_regione) = **LOWER**(regione\_in) **AND** codice\_regione **IN** (

**SELECT** distinct(codice\_regione)

**FROM** movimenti\_regioni

);

**IF** found\_regione <= 0 **THEN**

**RAISE** regione\_not\_found;

**END IF;**

**--posso stampare le regioni**

**OPEN** cursore\_regione;

**dbms\_output.put\_line**('DATI MOVIMENTI REGIONE '|| regione\_in);

**dbms\_output.put\_line**('codice\_regione,data,variazione\_svago\_vendita\_dettaglio,variazione\_farmacie\_alimentari,variazione\_parchi, variazione\_stazioni\_trasporto\_pubblico,variazione\_sedi\_lavoro,variazione\_zone\_residenziali');

**LOOP**

**FETCH** cursore\_regione **INTO** riga\_movimenti;

**EXIT WHEN** cursore\_regione**%NOTFOUND**;

**dbms\_output.put\_line**(riga\_movimenti.codice\_regione || ',' || riga\_movimenti.data || ',' || riga\_movimenti.variazione\_svago\_vendita\_dettaglio || ',' || riga\_movimenti.variazione\_farmacie\_alimentari|| ',' || riga\_movimenti.variazione\_parchi|| ',' || riga\_movimenti.variazione\_stazioni\_trasporto\_pubblico|| ',' || riga\_movimenti.variazione\_sedi\_lavoro|| ',' || riga\_movimenti.variazione\_zone\_residenziali );

**END LOOP;**

**CLOSE** cursore\_regione;

**EXCEPTION**

**WHEN** regione\_not\_found **THEN dbms\_output.put\_line**('REGIONE NON PRESENTE');

**END** report\_movimenti\_regione;

* **MEDIA MOVIMENTI REGIONALI**

**------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

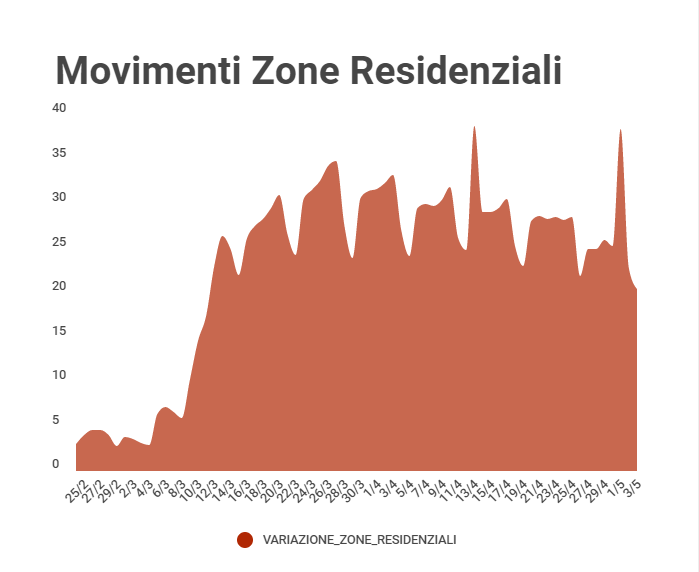
**--Media degli spostamenti degli italiani, verso una determinata categoria di luoghi**

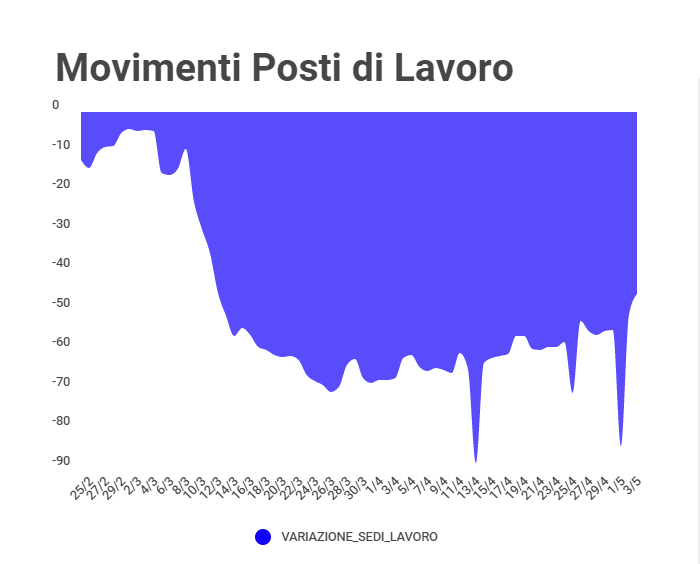
**SELECT** data**, AVG**(variazione\_\*\*\*\*)

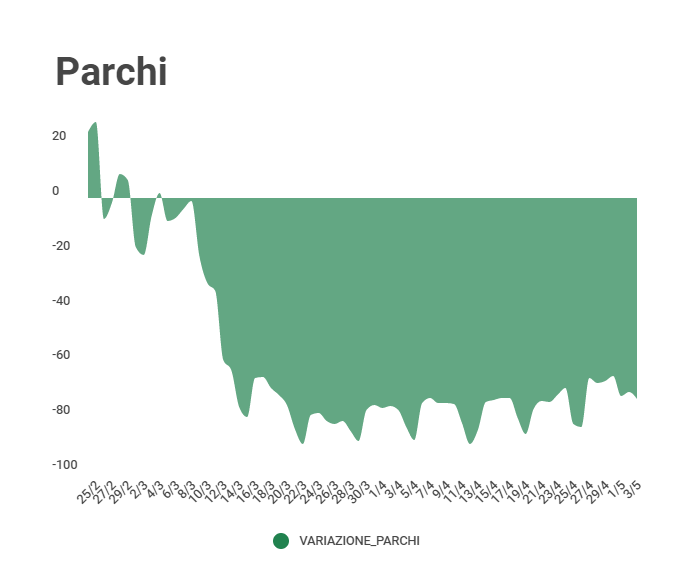
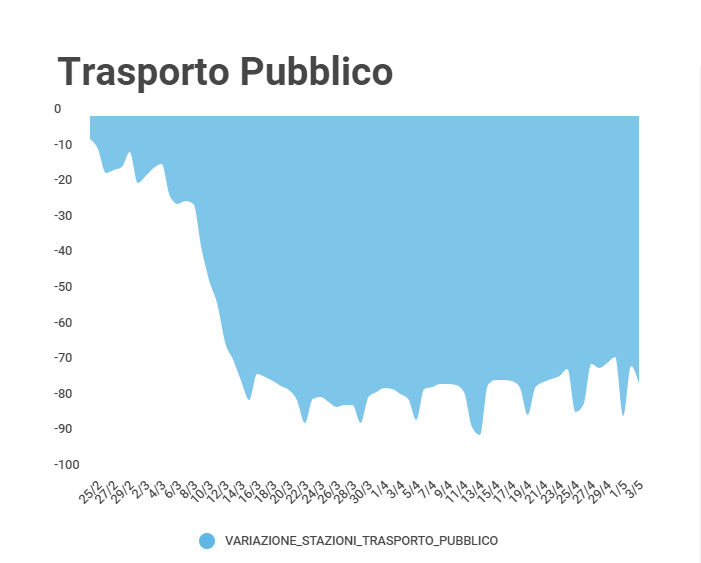
**FROM** movimenti\_regioni

**GROUP BY** data

**ORDER BY** data;

**------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**





* **Query che ritorna il numero di contagi e la posizione per ogni provincia, alla data più recente**

**SELECT** p.longitudine, p.latitudine, mis.totale\_casi

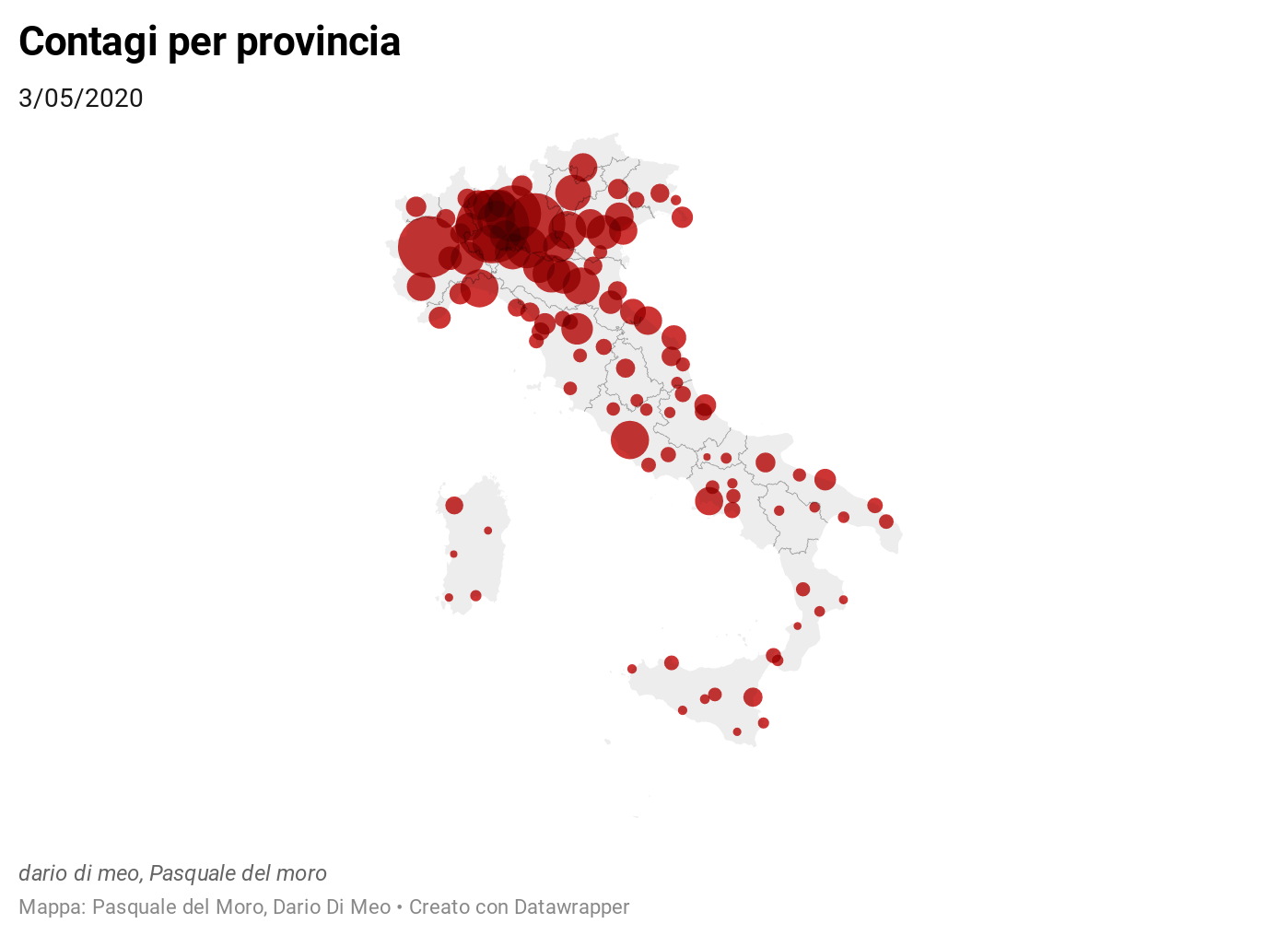
**FROM** misure\_province mis **JOIN** province p **ON** mis.codice\_provincia = p.codice\_provincia

**WHERE** data = (

**SELECT** max(data)

**FROM** misure\_province

);



1. Stored Procedure
2. **Ottimizzazione Base di dati**

L’ottimizzazione della base di dati, consiste nella creazione di **Indici** e **Viste**, per migliorare i tempi di esecuzione delle query.

Tramite le funzionalità di Cronologia SQL, su Oracle Live Sql **(**F8**)**, si può calcolare, in secondi, la durata di una query.

**Ad esempio:**

|  |
| --- |
| DURATA |
| 0.11s |

**SELECT** \*

**FROM** misure\_regioni 🡺

**WHERE** data = '3-mag-20';

**Indici**

Sono creati automaticamente sui campi chiave delle nostre tabelle. Generalmente le query sono basate su ricerche sui campi di tipo **DENOMINAZIONE**, cioè sui nomi delle regioni/province.

Per questo motivo si intorducono due **INDICI SECONDARI** sui campi non chiave, denominazione.

---------------------------------------------------------------------------------------------------------

**--Indice su denominazione regione**

**CREATE INDEX** idx\_denominazione\_regione

**ON** **regioni**(denominazione\_regione);

**--Indice su denominazione Provincia**

**CREATE INDEX** idx\_denominazione\_provincia

**ON province(**denominazione\_provincia**);**

---------------------------------------------------------------------------------------------------------

|  |  |  |
| --- | --- | --- |
| Tabella | Prima dell’indice | Dopo l’indice |
| Regioni | 0.005s | 0.002s |
| Province | 0.005s | 0.002s |

Grazie a questi due indici si nota un notevole decremento del tempo di esecuzione delle query, quantificato dallo strumento di analisi di Oracle Live Sql.

Valori ottenuti dalle seguenti Query di **Testing**:

**SELECT** \* **SELECT** \*

**FROM** regioni **FROM** province

**WHERE** denominazione\_regione = 'Campania'; **WHERE** denominazione\_provincia = 'Napoli';

**Note:**

Il calcolo del tempo di esecuzione è **Machine dipendent**, quindi è solo una stima del tempo necessario. Inoltre i tempi calcolati sono frutto di una **media aritmetica** su più prove ripetute.

**Viste**

Le viste (non materializzate), sono Query memorizzate che quando vengono chiamate vengono automaticamente eseguite per costruire una tabella temporanea su cui effettuare operazioni SQL.

Le viste vengono principalmente create per:

* Ridurre la complessità delle istruzioni sql (snellire il codice)
* Mostrare solo specifici attributi all’ utente utilizzatore della vista

Per questi motivi, si cercano **Query** frequentemente utilizzate sia per l’analisi dei dati, sia per gli eventuali utenti disposti ad utilizzare la base dati come **supporto** per una loro applicazione.

Si è pensato di creare una vista per la visualizzazione, alla **data più recente**, dei contagi per ogni **provincia**, alla cui è stata aggiunta la colonna *denominazione\_provincia* per facilitare eventuali ricerche sul campo.

**--Vista dati più recenti sui contagi per provincia**

**CREATE VIEW** last\_contagi\_province **AS**(

**SELECT** mis.data, mis.codice\_provincia, p.denominazione\_provincia, mis.totale\_casi

**FROM** misure\_province mis **JOIN** province p **ON** mis.codice\_provincia = p.codice\_provincia

**WHERE** mis.data **IN**(

**SELECT** max(data)

**FROM** misure\_province

)

);

**Viste Materializzate**

Le viste materializzate, a differenza delle viste classiche, sono uno **Snapshot** di una particolare Query. Quindi utilizzate per dati statici che mutano **molto raramente**, inoltre, sono molto più **efficienti** delle viste normali, in quanto non devono essere ricreate ad ogni utilizzo.

Una vista utile, è una sui dati riguardanti i posti letto presenti per ogni regione e il numero di positivi di ogni giorno.

**--Vista materializzata sui dati regionali mettendo in relazione i posti letto e i contagi**

**CREATE MATERIALIZED VIEW** posti\_regione\_positivi **AS**(

**SELECT** mis.data, r.codice\_regione, r.posti\_letto, r.posti\_terapia\_intensiva, mis.totale\_positivi

**FROM** misure\_regioni mis **JOIN** regioni r **ON** mis.codice\_regione = r.codice\_regione

);