|  |  |
| --- | --- |
|  |  |
| **CORSO DI BASI DI DATI**  **Professore Masciari Elio**  **COVID-19 ANALISYS** | |

**Letizia Arena N46004314 Veronica D’Ambrosio N46004507**

**Anno Accademico 2019-2020**

|  |  |
| --- | --- |
| *Sommario*  [Introduzione e Specifiche 3](#_Toc43419458)  [1. Creazione tabella master 4](#_Toc43419459)  [1.1. Accorgimenti preventivi all’importazione su Datagrip: data cleansing del file 4](#_Toc43419460)  [1.2. Creazione di uno schema “ad hoc” su Datagrip: 5](#_Toc43419461)  [1.3. Creazione della tabella master e relativo popolamento 5](#_Toc43419462)  [2. Verifica delle forme normali e decomposizione 7](#_Toc43419463)  [2.1. Verifica della 3NF 7](#_Toc43419464)  [2.2. Creazione delle tabelle in Oracle e definizione dei vincoli mediante DDL 8](#_Toc43419465)  [3. Arricchimento dello schema 9](#_Toc43419466)  [3.1. Aggiunta dati regionali sui casi e violazione vincolo di integrità referenziale 9](#_Toc43419467)  [3.2. Ampliamento dati per regione sugli spostamenti 10](#_Toc43419468)  [3.3. Inserimento dati sulla popolazione residente 11](#_Toc43419469)  [3.4. Inserimento dati sull’inquinamento dell’aria 11](#_Toc43419470)  [4. Schema Concettuale E/R 12](#_Toc43419471)  [4.1. Identificazione del modello relazionale dello schema 12](#_Toc43419472)  [4.2. Identificazione del modello entità/relazione 12](#_Toc43419473)  [5. Query per l’analisi del contagio 14](#_Toc43419474)  [6. PL/SQL 26](#_Toc43419475)  [6.1. Specifica in PL/SQL di Stored Procedure 26](#_Toc43419476)  [6.2 Specifica in PL/SQL di Trigger 29](#_Toc43419477) |  |

# Introduzione e Specifiche

Si vuole realizzare una base di dati, a partire dal dataset fornito dalla specifica e reperibile sul profilo [GitHub della Croce Rossa](https://github.com/pcm-dpc/COVID-19/tree/master/dati-province), utile all’analisi dei dati relativi all’epidemia COVID-19 in Italia nella finestra temporale che va dal 25/02/2020 al 03/05/2020.

Per lo sviluppo della base di dati si è scelto di utilizzare il DBMS ORACLE Database (v.18c XE) per mezzo del tool Datagrip (v.2020.1).

Una volta importati i dati su Datagrip, si è proceduto alla verifica dei requisiti della terza forma normale e alla successiva normalizzazione in caso questi non fossero soddisfatti.

A questo punto si è passati all’arricchimento dello schema con ulteriori informazioni riguardanti il fenomeno.

Una volta individuato lo schema definitivo, è stato estrapolato, attraverso un processo di reverse engineering, il relativo schema concettuale E/R della base di dati.

Infine, si è proceduto allo sviluppo di interrogazioni, procedure e trigger con lo scopo di analizzare e manipolare i dati presenti nel database

# Creazione tabella master

## 1.1. Accorgimenti preventivi all’importazione su Datagrip: data cleansing del file

Alcuni accorgimenti sono stati necessari al fine di importare correttamente i dati nella tabella master. In particolare, perché datagrip riconoscesse correttamente le date, si è resa necessaria la modifica del tipo e formato data nel CSV su Excel (es: da 20/05/2020 18:00:00 a 2020-05-20). La stessa modifica è stata apportata a tutti gli altri dataset utilizzati successivamente. Poi si è proceduto ad eliminare gli spazi vuoti prima dei nomi delle Regioni, poiché questi avrebbero dato problemi nell’ordinare alfabeticamente le tuple.  
Una osservazione dei dati presenti nella tabella ha inoltre fatto notare che i dati relativi ai numeri dei contagiati fossero cumulativi: ogni valore era quindi somma di tutti quelli cronologicamente precedenti.   
Si è ritenuto necessario, dunque, per poter permettere il futuro utilizzo del dato in varie query, di convertire il valore da cumulativo a giornaliero. Questa conversione, per la cui riuscita si è fatto uso di Microsoft Excel, è consistita nel calcolare la differenza, giorno per giorno, tra il valore di quel giorno e quello del precedente.

## 1.2. Creazione di uno schema “ad hoc” su Datagrip:

Di default Oracle permette di lavorare sull’user SYSTEM configurato durante l’installazione. Tuttavia, all’interno di questo schema sono già presenti varie tabelle di default che rendono difficile l’organizzazione delle tabelle di interesse.   
In Oracle non è possibile creare esplicitamente un *Database schema*. Ciò nonostante, quando viene creato un utente viene creato -ed identificato con esso- anche il suo schema , che è l’insieme degli oggetti posseduti da un utente e inizialmente è vuoto. Quindi si è pensato di creare l’utente (e il relativo schema) PROGETTO da popolare con gli oggetti ad esso inerenti. Per fare ciò si sono sfruttati i poteri amministrativi e di *account management* di SYSTEM. Il codice con il quale è stato possibile ottenere questo risultato è il seguente:

connect system/manager as sysdba  
ALTER SESSION SET "\_ORACLE\_SCRIPT"=true  
   
create *USER* PROGETTO identified by password;  
  
GRANT CONNECT to PROGETTO

GRANT ALL PRIVILEGES TO PROGETTO

alter session set current\_schema = PROGETTO

Le prime due linee di codice servono per abilitare i permessi di SYSTEM, le altre per creare l’utente, per consentire la connessione dell’utente al server e garantire gli stessi privilegi di SYSTEM (particolarmente utile per rimuovere il limite di 5000 tuple inseribili con un solo comando) e infine l’ultima per settare lo schema sul quale lavorare in console come predefinito.

## 1.3. Creazione della tabella master e relativo popolamento

È finalmente possibile creare la tabella master con il seguente comando SQL:

CREATE TABLE TABELLA\_MASTER  
(

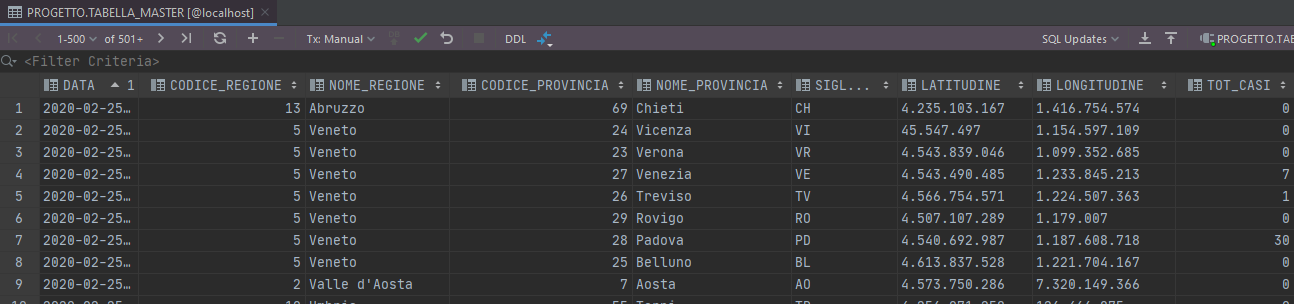
DATA DATE,

ID\_REGIONE INTEGER,  
NOME\_REGIONE VARCHAR2(50),  
ID\_PROVINCIA INTEGER,  
NOME\_PROVINCIA VARCHAR2(50),  
SIGLA\_PROV VARCHAR2(10),  
LATITUDINE VARCHAR2(50),  
LONGITUDINE VARCHAR2(50),  
TOT\_CASI INTEGER

PRIMARY KEY (DATA, ID\_PROVINCIA)  
);

Per popolarla, si è usata l’interfaccia grafica di Datagrip che supporta di default l’inserimento da file .CSV automatizzando il processo di INSERT.



Risultato:  
A questo punto è stata necessaria una nuova fase di data cleansing. Con il comando:

DELETE  
FROM TABELLA\_MASTER  
WHERE DATA NOT BETWEEN DATE '2020-02-25' AND DATE '2020-05-03'

sono state eliminate le date non interessate dall’analisi come da specifica. È stata inoltre riscontrata la presenza di tuple con numerosi attributi NULL e codici provincia tutti maggiori di 900 che non riferivano a nessuna provincia o regione realmente esistente. Essendo tuple inconsistenti, sono state eliminate con il comando:

DELETE

FROM PROGETTO\_PROVINCIA  
WHERE ID\_PROVINCIA>900

# Verifica delle forme normali e decomposizione

## 2.1. Verifica della 3NF

Si procede all’analisi della tabella master per verificare che sia in 3NF come richiesto dalla specifica.

Poiché lo schema non possiede attributi multivalore né strutturati ma solo attributi semplici, lo schema della tabella master è sicuramente 1NF. Tuttavia, così com’è proposto, NON è in forma 2NF. Per esserlo, oltre ad essere in forma 1NF, bisogna che ogni attributo non primo di R(X) è in dipendenza funzionale completa da ogni chiave di R(X), ossia tutti gli attributi non primi devono dipendere dall’interezza della chiave, non solo da parte di questa.

Per lo schema di relazione analizzato, sarebbe possibile scegliere come chiave primaria gli attributi DATA e CODICE\_PROVINCIA, in modo da rispettare tutti i vincoli ad essa legati. È possibile notare, però, che sussistono le seguenti FD:

FD1: {DATA,ID\_PROVINCIA}🡪PROVINCIA

FD2:{ID\_PROVINCIA}🡪PROVINCIA

E lo stesso discorso è applicabile per REGIONE e ID\_REGIONE.

La definizione di 2NF non è quindi rispettata dalla dipendenza DP2, per cui provincia dipende solo da parte della chiave. È possibile però normalizzare lo schema di 1NF attraverso l’operazione di decomposizione, per cui gli attributi vanno assegnati a tabelle separate. Si passa dunque ad una risoluzione specifica dello schema. Data l’originale TABELLA\_MASTER:

**TABELLA\_MASTER** (Data, Id\_Provincia, Provincia, Sigla\_Provincia, Id\_Regione, Latitudine, Longitudine, Tot\_Casi)

Si considerino, per isolarli, gli attributi che dipendono unicamente da parte della chiave primaria, più in particolare dall’attributo primo Id\_Provincia:

Id\_Regione, Nome\_Provincia, Sigla\_Provincia, Nome\_Regione, Latitudine, Longitudine

Con le quali è possibile creare due relazioni:

**CASI\_PROVINCE** ( Data, Nome\_Provincia, Tot\_casi)

**PROVINCIA** (Id\_Provincia, Nome\_Provincia, Sigla\_Provincia, Nome\_Regione, Codice\_Regione, Latitudine, Longitudine)

I due schemi relazionali proposti fanno sì che la relazione sia in 2NF.

Ciononostante, il nuovo schema della base di dati così creato non è comunque in forma 3NF. Infatti la seconda caratteristica che deve soddisfare è che ogni attributo non primo di R(X) non dipenda transitivamente da ogni chiave di R(X), ossia non possono esserci dipendenze tra attributi se non a quelli appartenenti alla chiave. Nello schema proposto in 2NF il problema nasce nello schema PROVINCIA, più in particolare nell’attributo Regione, che dipende dall’attributo Id\_Regione, pur non essendo questo un attributo primo della tabella. È necessario dunque attuare una seconda operazione di decomposizione:

**CASI\_PROVINCE** (Data, Id\_Provincia, Tot\_Casi)

**PROVINCIA** (Id\_Provincia, Nome\_Provincia, Sigla\_Provincia, Regione, Latitudine, Longitudine)

**REGIONE** (Id\_Regione, Nome\_Regione)

L’attributo Regione di PROVINCIA, ovviamente, avrà Id\_Regione(Regione) come chiave esterna. Lo schema così proposto è quindi in 3NF.

## 2.2. Creazione delle tabelle in Oracle e definizione dei vincoli mediante DDL

Di seguito viene allegato il codice per la creazione delle tabelle in Oracle popolate in seguito con il consueto metodo. Per le chiavi esterne si è scelta l’opzione: *on delete Cascade* per mantenere l’integrità dei dati.

CREATE TABLE CASI  
(

DATA DATE,  
ID\_PROV INTEGER REFERENCES PROVINCIA (ID\_PROVINCIA) ON DELETE CASCADE,  
TOT\_CASI INTEGER,

PRIMARY KEY (DATA, ID\_PROV)  
);

CREATE TABLE PROVINCIA  
(

ID\_PROVINCIA INTEGER,  
NOME\_PROVINCIA VARCHAR2(50),  
COD\_REGIONE INTEGER REFERENCES REGIONE (ID\_REGIONE) ON DELETE CASCADE,  
SIGLA\_PROV VARCHAR2(10),  
LATITUDINE VARCHAR2(50),  
LONGITUDINE VARCHAR2(50),

PRIMARY KEY (ID\_PROVINCIA)  
);

CREATE TABLE REGIONI  
(

ID\_REGIONE INTEGER,  
NOME\_REGIONE VARCHAR2(50),

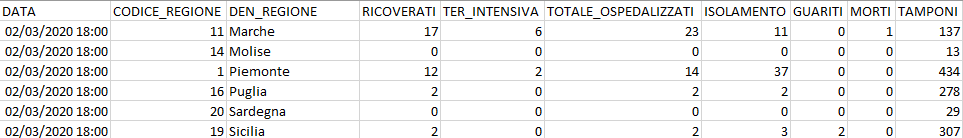
PRIMARY KEY (ID\_REGIONE)  
);

# Arricchimento dello schema

In questa fase si è proceduto all’arricchimento dello schema con ulteriori informazioni utili all’analisi del fenomeno. In particolare, sono stati aggiunti dati riguardo il numero di residenti sia per provincia che per regione, mentre per regione e data il numero di morti, ricoveri in strutture ospedaliere, ricoveri in terapia intensiva, dimessi guariti, isolamenti domiciliari e tamponi effettuati e spostamenti in varie categorie di luoghi.

## 3.1. Aggiunta dati regionali sui casi e violazione vincolo di integrità referenziale

Dalla stessa fonte della tabella master ([dpc-covid19-ita-regioni.csv](https://github.com/pcm-dpc/COVID-19/blob/master/dati-regioni/dpc-covid19-ita-regioni.csv)) sono state importate ulteriori informazioni categorizzate per data e regione, selezionando i domini di interesse operando le consuete modifiche al tipo data e eliminando le tuple non oggetto di interesse. Un esempio del file ottenuto:



A questo punto si è deciso di creare una nuova tabella che contenesse questi dati per migliorare la gestione e la visualizzazione degli stessi, con l’accortezza di lasciare momentaneamente l’attributo DEN\_REGIONE per facilitare l’inserimento di successivi dati che, presi da altre fonti, difficilmente avrebbero avuto anche l’attributo codice regione. Di seguito lo script:

CREATE TABLE CASI \_REGIONI  
(  
DATA DATE,  
CODICE\_REGIONE INTEGER REFERENCES REGIONE(ID\_REGIONE) ON DELETE CASCADE,

DEN\_REGIONE VARCHAR2(50),  
RICOVERATI\_CON\_SINTOMI INTEGER,  
TERAPIA\_INTENSIVA INTEGER,  
TOTALE\_OSPITALIZZATI INTEGER,  
ISOLAMENTO\_DOMICILIARE INTEGER,  
DIMESSI\_GUARITI INTEGER,  
DECEDUTI INTEGER,  
TAMPONI INTEGER,

PRIMARY KEY (DATA, CODICE\_REGIONE)   
)

All’atto dell’insert del file .csv tramite interfaccia grafica dei nuovi dati per regione, Datagrip ha notificato un’inconsistenza sui dati: la chiave CODICE\_REGIONE di Trento della nuova tabella DATI\_COVID\_REGIONI non trovava la chiave madre (della tabella REGIONE) alla quale riferirsi, generando una violazione del vincolo di integrità referenziale. In questo modo è stato scoperto un errore nello schema tabella\_master.

Nel file originale di Github, a causa di un refuso (corretto poi nei successivi file sulle regioni) le REGIONI “P.A. Trento” e “P.A. Bolzano” avevano lo stesso codice regionale, cioè 4. Sicché, all’atto dell’insert dalla tabella master alla tabella REGIONI, solo la tupla riferita a Bolzano veniva importata, ignorando quella di Trento. Ciò è stato dovuto alla violazione del vincolo di unicità di chiave primaria. Come soluzione il DBMS impedisce l’inserimento di altre tuple con stessa chiave primaria tralasciando così la tupla riferita alla P.A. di Trento e producendo quindi una tabella REGIONI mancante di una tupla. La strategia adottata per correggere l’errore è stata la seguente:  
nella tabella REGIONE è stata aggiunta la regione autonoma mancante:

INSERT

INTO REGIONI (ID\_REGIONE, NOME\_REGIONE)

VALUES (22,'P.A. Trento')

nella tabella PROVINCIA è stato impostato il codice regione giusto di Bolzano, modificandolo dal 4 della precedente tabella master sbagliata a 21.

UPDATE PROVINCIA

SET COD\_REGIONE = 21

WHERE COD\_REGIONE = 4

Lo stesso è stato fatto per quello di Trento. Qui è possibile notare che le regioni autonome e le omonime province hanno lo stesso codice sia per provincia che per regione, quindi è bastato, per Trento, cambiare il codice regione a 22 dove anche quello di provincia era 22.

UPDATE PROVINCIA

SET COD\_REGIONE = 22

WHERE ID\_PROVINCIA = 22

## 3.2. Ampliamento dati per regione sugli spostamenti

Grazie ai dati messi a disposizione da Google (reperibili [qui](https://www.google.com/covid19/mobility/index.html?hl=it)), è stato possibile ampliare la precedente tabella, aggiungendo informazioni riguardanti gli spostamenti relativi a luoghi riguardanti il tempo libero, alimentari e farmacie, luoghi residenziali, di lavoro, trasporto pubblico, vendita e attività ricreative. Questi dati vengono calcolati sulla base della variazione rispetto a un valore standard di riferimento precedente all’emergenza. Questo è il comando di alter table necessario alla modifica della precedente tabella:

alter table CASI\_REGIONI ADD   
 (  
VENDITA\_E\_ATTIVITA\_RICREATIVE INTEGER,   
ALIMENTARI\_FARMACIE INTEGER,   
TRASPORTO\_PUBBLICO INTEGER,   
LUOGHI\_LAVORO INTEGER,   
LUOGHI\_RESIDENZIALI INTEGER  
);

Per aggiungere e correlare correttamente i dati dei due dataset, sono state prima importate le nuove informazioni in un’appena creata tabella SPOSTAMENTI e poi è stata effettuata una join:

SELECT DATI\_COVID\_REGIONI.\*,   
SPOSTAMENTI.LUOGHI\_LAVORO,   
SPOSTAMENTI.LUOGHI\_RESIDENZIALI,

SPOSTAMENTI.TRASPORTO\_PUBBLICO,

SPOSTAMENTI.ALIMENTARI\_FARMACIE,

SPOSTAMENTI.VENDITA\_E\_ATTIVITA\_RICREATIVE  
FROM DATI\_COVID\_REGIONI INNER JOIN SPOSTAMENTI  
 ON DATI\_COVID\_REGIONI.DEN\_REGIONE = SPOSTAMENTI. NOME\_REGIONE AND DATI\_COVID\_REGIONI.DATA = SPOSTAMENTI.DATA;

A questo punto è possibile droppare l’attributo den\_regione che aggiungerebbe una ridondanza inutile alla base di dati:

alter table DATI\_COVID\_REGIONI drop column NOME\_REGIONE

Infine, è bastato aggiornare con i nuovi dati la tabella CASI\_REGIONI ampliata.

## 3.3. Inserimento dati sulla popolazione residente e sugli over 65

I dati relativi ai residenti sono stati reperiti sul [sito dell’Istat](http://dati.istat.it/Index.aspx?DataSetCode=DCIS_POPRES1) scegliendo una visualizzazione personalizzata e più recente possibile. Sono stati quindi selezionati “territori” comprendenti sia regioni che province, e come “periodo” il 2019. A questo punto i dati sono stati divisi in due file .csv, uno per le province e uno per le regioni e sono state corrette alcune incongruenze quali spazi e trattini extra che avrebbero potuto generare inconsistenze nella successiva importazione sul database.

Con i comandi di alter table sono state quindi aggiornate le tabelle REGIONI e PROVINCE:

alter table REGIONI ADD   
 (

RESIDENTI INTEGER  
);

alter table PROVINCE ADD   
 (

RESIDENTI INTEGER,

PERCOVER65 INTEGER  
);

Popolate poi con le consuete modalità.

3.4. Inserimento dati sull’inquinamento dell’aria

Dal seguente [link](https://github.com/andreapas79/COVID-19/blob/master/Data/air_pollution_data/Tabella%204.xlsx) sono stati recuperati i dati circa la percentuale di inquinamento da polveri sottili PM 2.5 per regione. È stato necessario operare, dopo il consueto data cleansing, una somma dei dati delle varie stazioni di analisi per regione con una semplice query. È stato poi aggiunto il nuovo campo alla tabella regioni con il comando di alter table, popolandolo poi di conseguenza.

SELECT nome\_regione *SUM*(perc\_inquinamento\_aria)  
from INQUINAMENTOARIA  
order by nome\_regione;

alter table REGIONI ADD   
 (

INQUINAMENTO\_ARIA INTEGER,  
);

# Schema Concettuale E/R

## 4.1. Identificazione del modello relazionale dello schema

Il primo step nel processo di reverse-engineering del progetto è quello di identificare il modello relazionale a cui si riferisce lo schema. Partendo dalle tabelle, anche mediante lo studio delle chiavi primarie e delle chiavi esterne, è quindi possibile ricavare le cinque seguenti relazioni:

**REGIONI** (id\_regione, nome\_regione, residenti, Air\_Pollution)

**PROVINCE** (id\_provincia, nome\_provincia, cod\_regione:REGIONI, sigla\_prov, longitudine, latitudine, residenti, PercOver65)

**CASI\_PROVINCE** (data, id\_prov:PROVINCE, tot\_casi)

**CASI\_REGIONI** (data, codice\_regione:REGIONI, ricoverati\_con\_sintomi, terapia\_intensiva, totale\_ospedalizzati, isolamento\_domiciliare, dimessi\_guariti, deceduti, tamponi)

**SPOSTAMENTI** (data:CASI\_REGIONI, codice\_regione:CASI\_REGIONI, alimentari\_e\_farmacie, trasporto\_pubblico, vendita\_e\_attività\_ricreative, luoghi\_residenziali, loughi\_di\_lavoro) 

## 4.2. Identificazione del modello entità/relazione

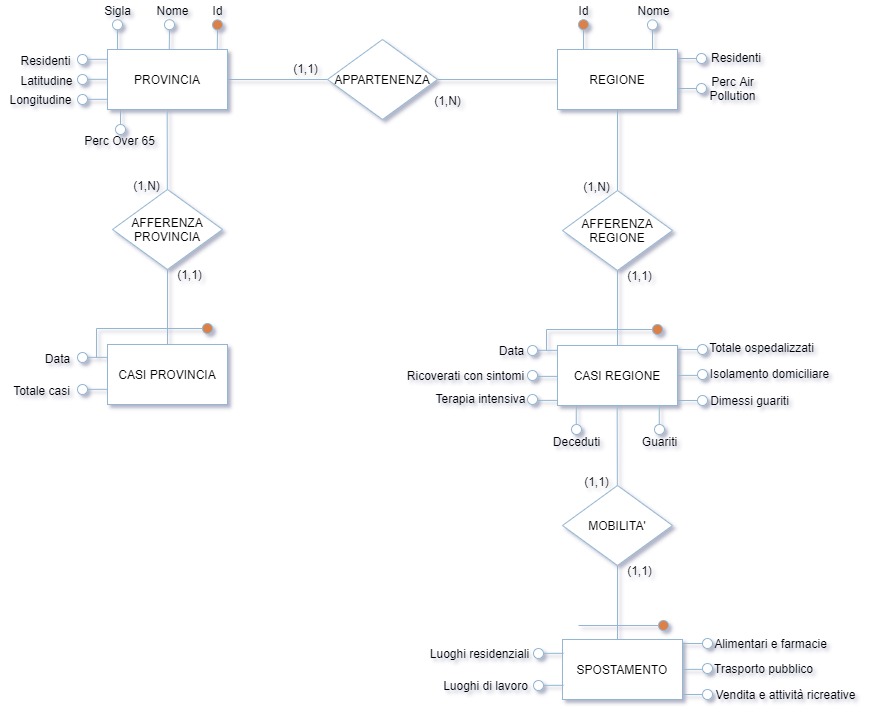
Dalle relazioni si individuano le entità presenti nel modello: REGIONE, PROVINCIA, CASI REGIONE, CASI PROVINCIA e SPOSTAMENTO.

Il rapporto tra PROVINCIA e REGIONE è esprimibile come una relazione 1:M: ad una singola regione possono essere infatti associate più province. Questa associazione permette facilmente lo sviluppo nel modello relazionale che si evince dalle tabelle, in cui l'associazione è incorporata nella relazione PROVINCE per mezzo dell'aggiunta dell'attributo cod\_regione (che si riferisce a id\_regione di REGIONI).

Anche per la resa nel modello ER del rapporto tra CASI PROVINCIA e PROVINCIA la scelta migliore è quella di collegarle mediante un'associazione 1:M, AFFERENZA PROVINCIA, considerando però Id di PROVINCIA come identificatore esterno per CASI PROVINCIA. Questo permette l’assorbimento dell'associazione dalla relazione CASI PROVINCIA, che ha come chiavi primarie proprio data e id\_provincia.    
Speculare è il rapporto tra CASI REGIONE, REGIONE e l’associazione che li lega, AFFERENZA PROVINCIA, per cui anche la loro traduzione segue la procedura appena descritta.

REGIONE, però, non è l’unica relazione collegata a CASI REGIONE: la tabella SPOSTAMENTI, infatti, condivide per intero la sua chiave primaria con CASI\_REGIONI. Questo particolare rapporto, per cui le due tabelle potrebbero praticamente essere unite in una tabella unica (già si è visto che la scelta di non farlo è data puramente da una necessità di semplificazione della lettura dei dati), può essere descritto con un’associazione 1:1 tra CASI REGIONE e SPOSTAMENTO, chiamata MOBILITA’, e con l’assenza di identificatori propri in SPOSTAMENTO, che ha però come identificatori esterni tutti quelli di CASI REGIONE.

Si arriva, dunque, ad un plausibile schema ER, che viene qui mostrato nella sua veste grafica:



# Query per l’analisi del contagio

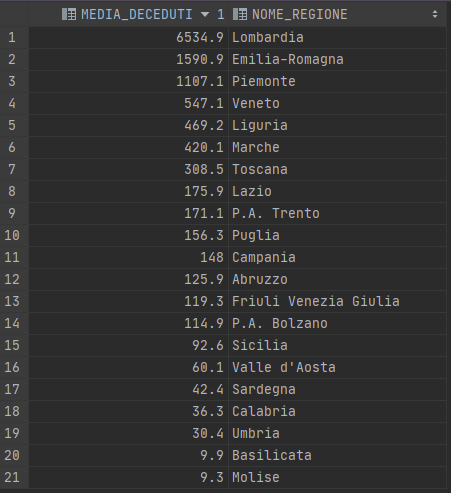
I dati così raccolti e organizzati sono stati utilizzati per creare query utili all’analisi del fenomeno. In particolare, si è scelto di indagare su:

* [media di persone decedute in ogni regione durante l’intervallo di tempo studiato;](#PRIMA_Q)
* [media dei contagiati per ogni mese in ciascuna provincia;](#SECONDA_Q) [G☺](#grafico_media_contagi_mese)
* [giorno in cui c’è stato il picco di contagi per una specifica provincia;](#TERZA_Q)
* [giorno in cui c’è stato il picco di contagi per tutte le province;](#QUARTA_Q)
* [per ogni regione, la provincia con più contagiati nel periodo temporale;](#QUINTA_Q)
* [la percentuale dei casi in rapporto al numero di residenti per provincia nella finestra temporale;](#SESTA_Q) G☺
* [la percentuale dei casi in rapporto al numero di over 65 per provincia nella finestra temporale;](#vecchi)
* [media, per ogni regione, di tutti i valori relativi agli spostamenti in tutto il periodo;](#SETTIMA_Q) G☺
* [valor medio degli spostamenti nei periodi pre, post e durante lockdown;](#lockdown) G☺
* [casi per regione e correlazione con l’inquinamento dell’aria da polveri sottili.](#polveri_sottili)
* La prima delle query proposte indaga sulla [media di persone decedute in ogni regione durante l’intervallo di tempo studiato:](#query)

SELECT *ROUND*(*AVG*(DECEDUTI), 2) AS MEDIA\_DECEDUTI, NOME\_REGIONE   
FROM CASI\_REGIONI JOIN REGIONE R on CASI\_REGIONI.CODICE\_REGIONE = ID\_REGIONE   
GROUP BY NOME\_REGIONE

Questa consiste in una selezione del nome di ciascuna regione e della media dei deceduti (calcolata mediante la funzione di raggruppamento AVG()) sul risultato della join tra CASI\_REGIONI e REGIONI. Le tuple sono poi raggruppate per valore dell’attributo NOME\_REGIONE mediante la clausola GROUP BY.  
Per aumentare la leggibilità del risultato, si è inoltre fatto uso di un’altra funzione, ROUND(num\_da\_arrotondare, num\_dopo\_la\_virgola), scegliendo di arrotondare il risultato della media in modo che presentasse solo due numeri dopo la virgola.

Il risultato della query è quello mostrato dalla tabella sottostante. Come è possibile notare, il numero di morti maggiori si è verificato in Lombardia:



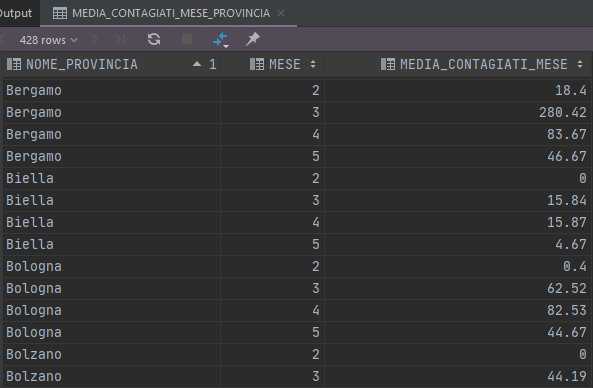
* Così come è stata calcolata la media dei deceduti sull’intera durata dell’intervallo di tempo considerato, è possibile anche selezionarla per altri intervalli: in questa query in particolare si va a considerare [la media dei contagiati per ogni mese in ciascuna provincia.](#query)

Sarà possibile, così, osservare per ogni regione quale sia stato il mese “peggiore”.

SELECT *ROUND*(*AVG*(TOT\_CASI), 2) AS MEDIA\_DECEDUTI\_MESE, *EXTRACT*(MONTH FROM DATA) AS MESE, NOME\_PROVINCIA  
FROM CASI\_PROVINCE JOIN PROVINCE R on CASI\_PROVINCE.ID\_PROV = R. ID\_PROVINCIA  
GROUP BY NOME\_PROVINCIA, *EXTRACT*(MONTH FROM DATA)   
ORDER BY *EXTRACT*(MONTH FROM DATA);

Questa query ha un’unica sostanziale differenza rispetto alla precedente (oltre al fatto che si basa sulle relazioni riferenti le province e non le regioni): nella clausola di raggruppamento va incluso anche il mese, in modo che la media possa essere calcolata singolarmente per ogni suo valore.  
L’attributo MESE è direttamente derivato da DATA, e viene ricavato attraverso la funzione EXTRACT(MONTH FROM attributo\_data), che ritorna il numero in int corrispondente alla posizione del mese nell’anno (1-12). I mesi di cui sarà possibile calcolare la media saranno quelli da Febbraio a Maggio, ma è importante considerare che proprio per questi ultimi due i dati non riguardano l’intero mese, ma solo la parte di questo compresa nell’intervallo selezionato.

Parte del risultato della query (7/428 rows) è mostrato dalla tabella sottostante:



Al fine di agevolare la creazione del grafico in excel, si è pensato di razionalizzare la tabella e i suoi valori. Per fare ciò, è stato innanzitutto necessario creare quattro viste materializzate, una per ogni mese, che sono poi state unite attraverso un natural join a 4 tabelle. Di seguito lo script di una vista e della join:

--creato vista con visualizzazione mensile  
CREATE MATERIALIZED VIEW MEDIA\_CONT\_FEB AS  
SELECT NOME\_PROVINCIA, MEDIA\_CONTAGIATI\_MESE AS FEBBRAIO  
FROM PROGETTO.MEDIA\_CONTAGIATI\_MESE\_PROVINCIA  
WHERE MESE=2

--join delle quattro viste mensili  
CREATE MATERIALIZED VIEW MEDIA\_CONTAGIATI\_PROVINCIA\_MESE AS  
SELECT \*  
FROM PROGETTO.MEDIA\_CONT\_FEB  
NATURAL JOIN PROGETTO.MEDIA\_CONT\_MAR NATURAL JOIN PROGETTO.MEDIA\_CONT\_APR NATURAL JOIN PROGETTO.MEDIA\_CONT\_MAG

Il risultato della join di seguito:



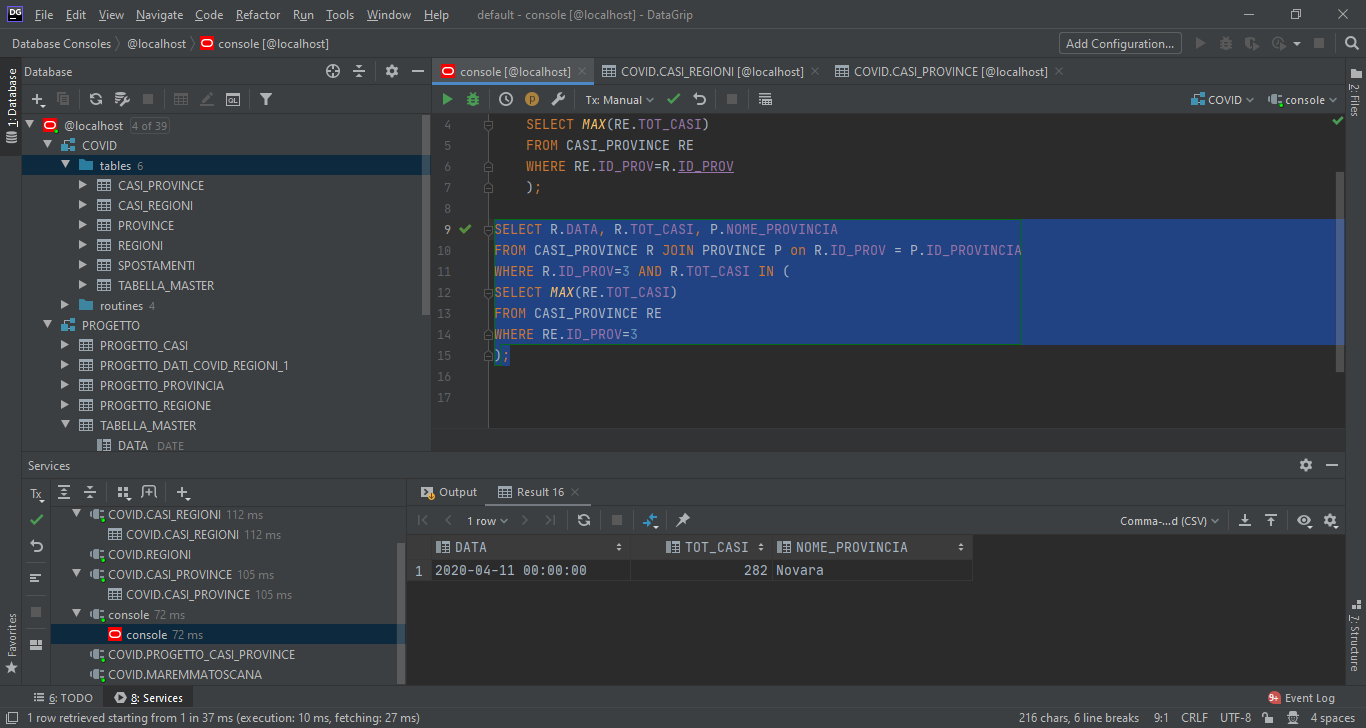
Questa è la visualizzazione grafica d’insieme del risultato della query creata esportando i dati su Excel:

Di seguito si propone un dettaglio dello stesso per agevolare la visione:

* La seguente query ha l’obiettivo di [selezionare il giorno in cui, in una data provincia, vi è stato il picco di contagi e il numero dei contagiati](#query).

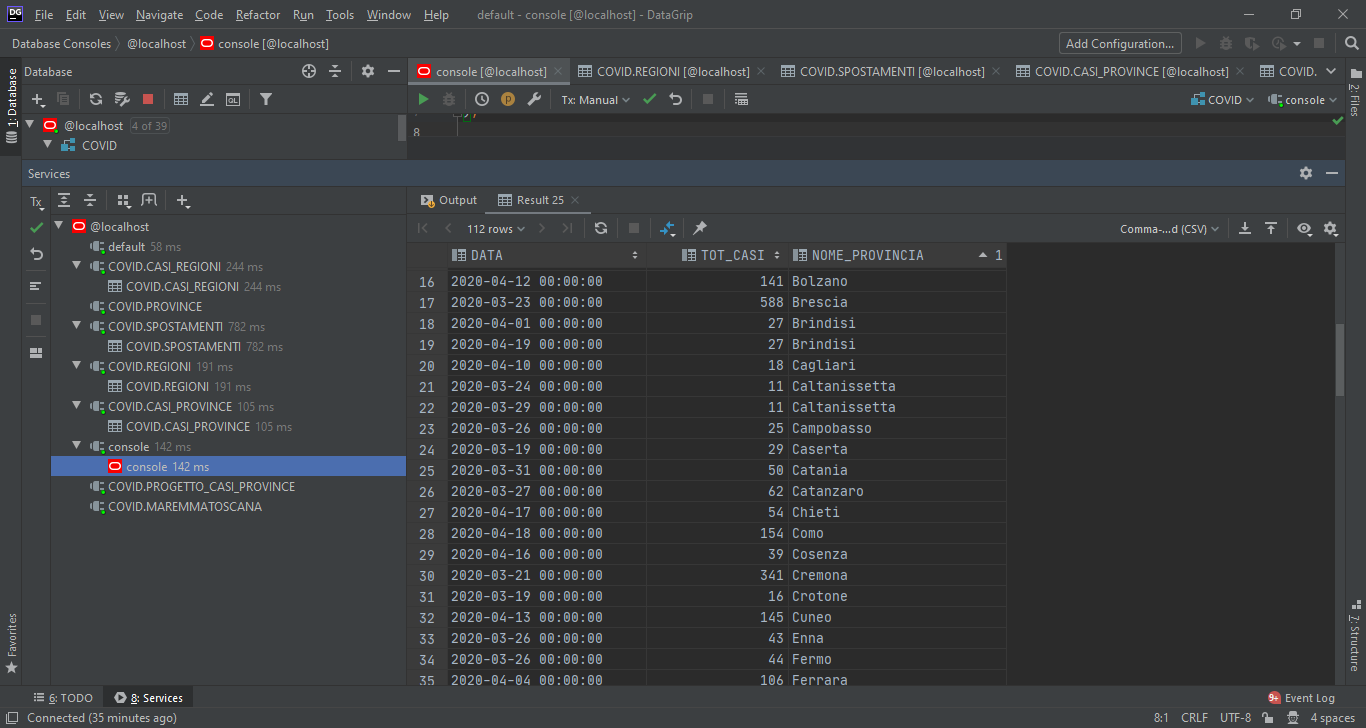
SELECT R.DATA, R.TOT\_CASI, P. NOME\_PROVINCIA  
FROM CASI\_PROVINCE R JOIN PROVINCE P on R.ID\_PROV = P.ID\_PROVINCIA   
WHERE R.ID\_PROV=3 AND R.TOT\_CASI IN (   
SELECT *MAX*(RE.TOT\_CASI)   
FROM CASI\_PROVINCE RE   
WHERE RE.ID\_PROV=3   
);

E’ possibile notare che si tratta di una query innestata: si seleziona infatti prima il valore massimo di TOT\_CASI rispetto alla provincia scelta (in questo caso si sceglie la provincia con id pari a 3, che corrisponde a Novara), e solo dopo si seleziona (attraverso join esplicito tra CASI\_PROVINCE e PROVINCE) la data che corrisponde al valore trovato, il valore di TOT\_CASI, ed il nome della provincia.  
Il risultato della query è quello mostrato dalla tabella sottostante:



* In maniera molto simile a quella vista sopra, è possibile sviluppare una query che selezioni [le date in cui, in tutte le province, vi è stato il picco di contagi e il relativo numero di contagiati](#query).

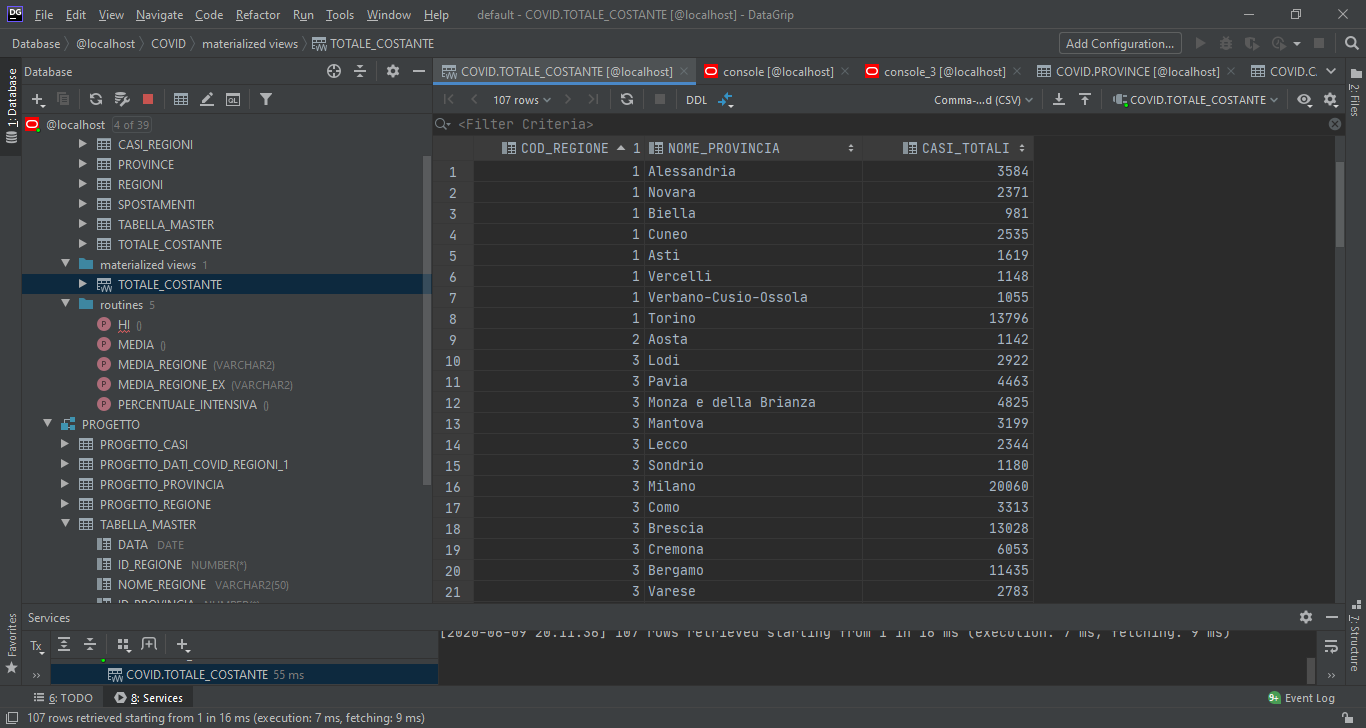
SELECT R.DATA, R.TOT\_CASI, P.NOME\_PROVINCIA   
FROM CASI\_PROVINCE R JOIN PROVINCE P on R.ID\_PROV = P.ID\_PROVINCIA   
WHERE R.TOT\_CASI IN (   
SELECT *MAX*(RE.TOT\_CASI)   
FROM CASI\_PROVINCE RE   
WHERE RE.ID\_PROV=R.ID\_PROV   
);

La struttura di questa query è quasi identica a quella della precedente, con la differenza che, non essendo indicata una specifica provincia, è necessario verificare che il massimo trovato corrisponda effettivamente a quello della provincia presa in questione imponendo la condizione che gli ID\_PROVINCIA delle due query combacino.  
E’ inoltre importante considerare che è possibile, così come nella query precedente, che venga trovata più di una sola data per ogni provincia: ciò accade se il massimo dei contagi è un valore che si ripete in più giorni.  
Parte del risultato della query (7/121 rows) è mostrato dalla tabella sottostante:

* Questa query [propone di selezionare, per ogni regione, la provincia con più contagiati](#query).  
  La prima operazione svolta è stata quella di creare una vista materializzata contenente la somma di tutti i valori di TOT\_CASI nell’intervallo di tempo preso in esame, raggruppati per ciascuna provincia, di cui è indicata la regione di provenienza: questa permetterà di conoscere, per ogni provincia, il numero totale dei contagiati (anche se già guariti o deceduti) in data 03/05/2020.

CREATE MATERIALIZED VIEW TOTALE\_COSTANTE AS   
SELECT P.COD\_REGIONE, P.NOME\_PROVINCIA, *SUM*(C.TOT\_CASI) AS CASI\_TOTALI   
FROM CASI\_PROVINCE C JOIN PROVINCE P on C.ID\_PROV = P.ID\_PROVINCIA   
GROUP BY NOME\_PROVINCIA, COD\_REGIONE;

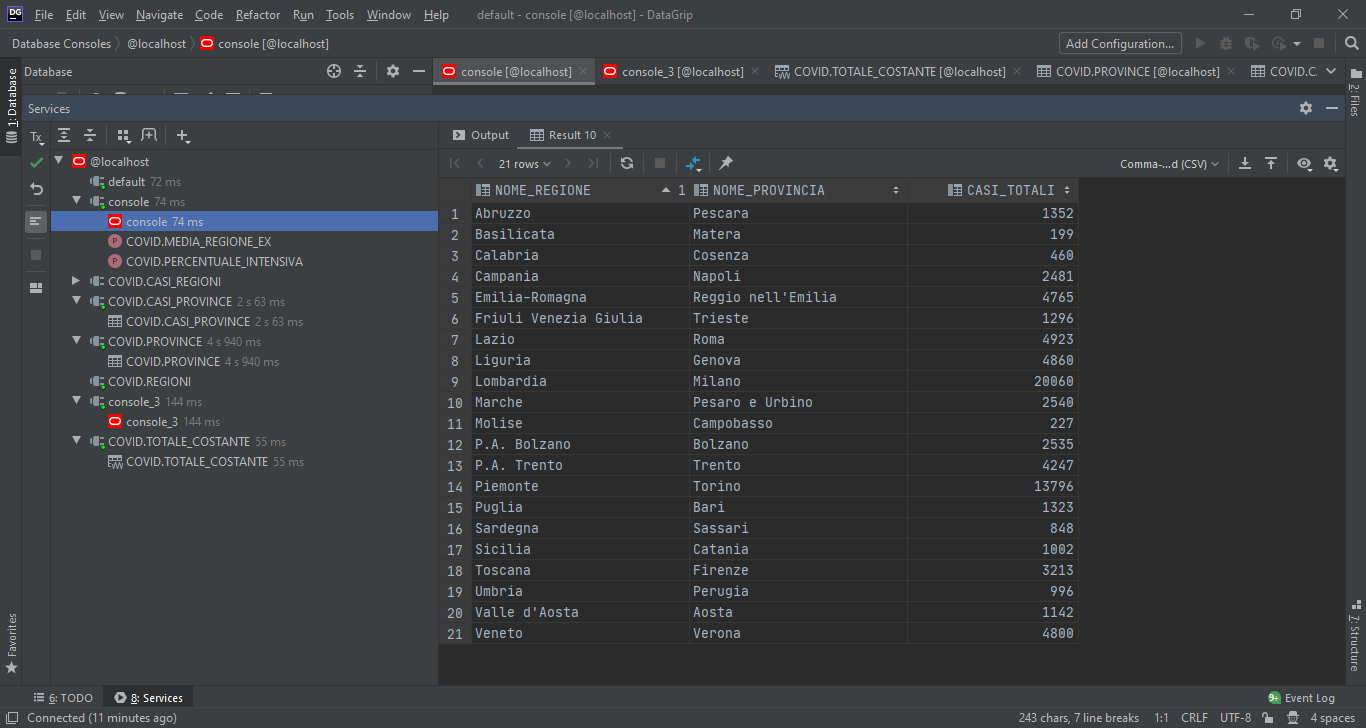
La vista da qui ricavata sarà la seguente (nell’immagine vengono proposte 7/107 rows):



Questa vista sarà utile nella creazione della effettiva query richiesta, in cui si cerca il massimo di CASI\_TOTALI per ciascuna regione, verificando che il valore trovato corrisponda effettivamente al massimo rispetto alla regione presa in considerazione imponendo la condizione che gli ID\_REGIONE delle due query combacino.

SELECT R.NOME\_REGIONE, T.NOME\_PROVINCIA, T.CASI\_TOTALI  
FROM REGIONE R JOIN COVID.TOTALE\_COSTANTE T ON T.COD\_REGIONE=R.ID\_REGIONE  
WHERE T.CASI\_TOTALI IN (  
SELECT *MAX*(TC.CASI\_TOTALI)  
FROM TOTALE\_COSTANTE TC  
WHERE TC.COD\_REGIONE=R.ID\_REGIONE  
);

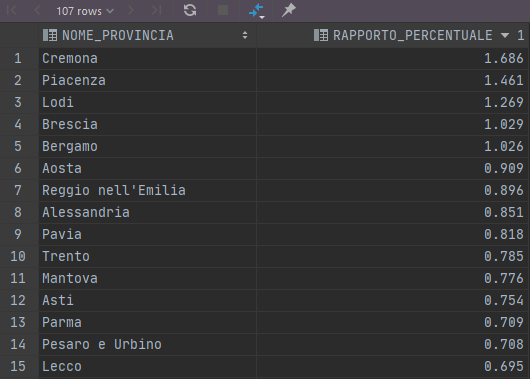
Il risultato della query è quello mostrato dalla tabella sottostante:



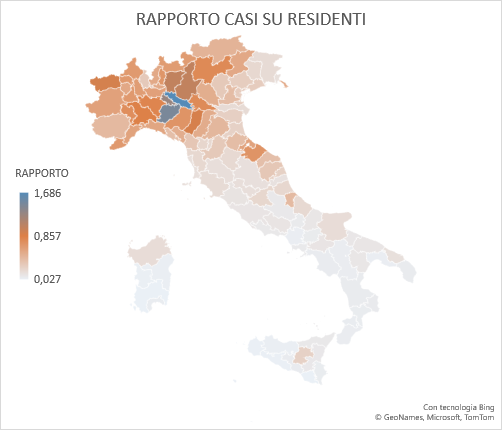
* Un’altra query di particolare interesse è quella che calcola [la percentuale dei casi in rapporto al numero di residenti per provincia nella finestra temporale](#query) coperta dal dataset. Utilizzando ancora la vista materializzata della precedente interrogazione ([qui](#vista_casi_complessivi)), si procede ad utilizzare questo dato per generare il rapporto, in percentuale, dei casi per residenti per regione semplicemente facendo una join su nome provincia della vista materializzata e della tabella province:

SELECT P.NOME\_PROVINCIA, *ROUND*(((CASI\_TOTALI\*100)/RESIDENTI),3) AS RAPPORTO  
FROM PROVINCE P, PROGETTO.TOTALE\_COSTANTE T  
WHERE NOME\_PROVINCIA= T.NOME\_PROVINCIA  
ORDER BY NOME\_PROVINCIA, RAPPORTO

Dai risultati si evince che la provincia con il maggior numero di casi in rapporto al numero di abitanti è Cremona:



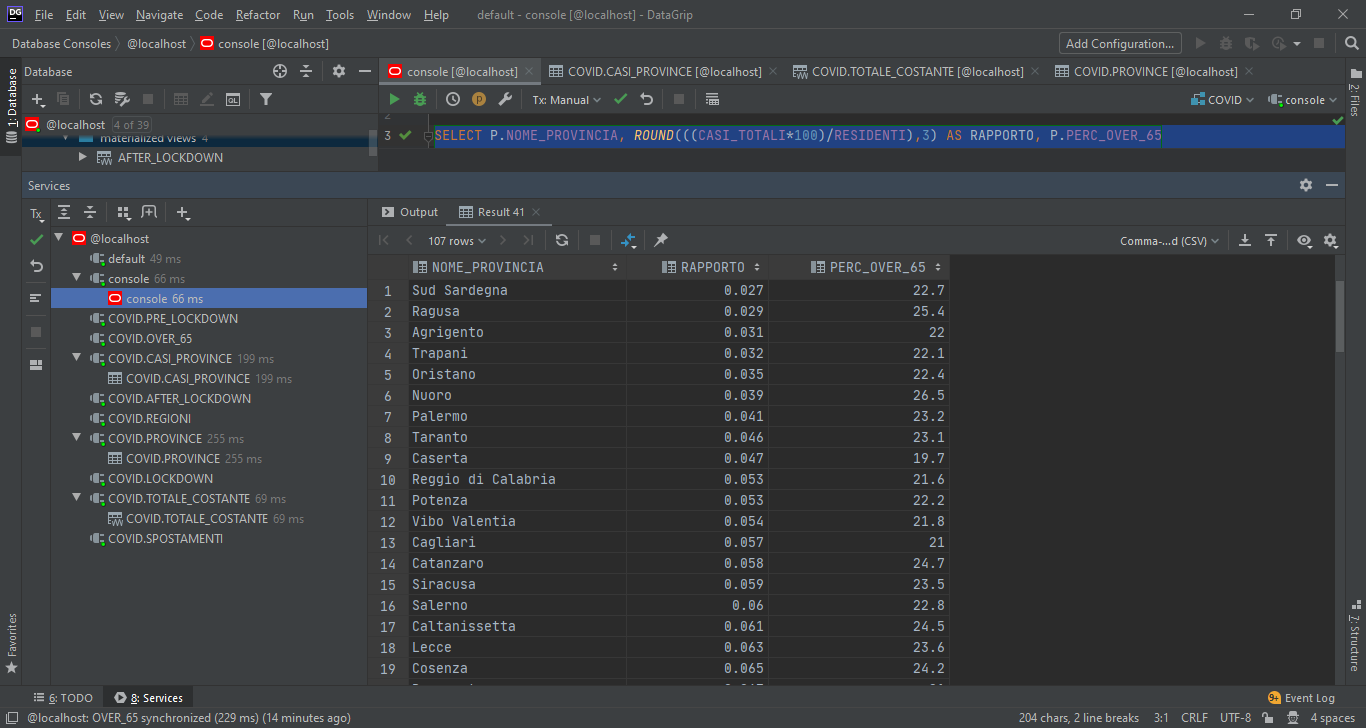
Di seguito un grafico a mappa per illustrare meglio la situazione d’insieme dal quale è possibile trarre la conclusione che, a parità di residenti, le province più colpite sono state quelle del Nord Italia:



* In questa query, rispetto alla precedente, viene aggiunta anche [la percentuale di popolazione che ha un’età superiore ai 65 anni](#query), in modo che ad occhio sia possibile osservare se è presente una relazione tra questa e la percentuale dei casi rispetto ai residenti.  
  L’ipotesi che possa esservi una correlazione tra le due, però, anche solo da un primo sguardo, non sembra avere fondamenta.

SELECT P.NOME\_PROVINCIA, *ROUND*(((CASI\_TOTALI\*100)/RESIDENTI),3) AS RAPPORTO, P.PERC\_OVER\_65  
FROM PROVINCE P JOIN TOTALE\_COSTANTE TC ON P.NOME\_PROVINCIA=TC.NOME\_PROVINCIA  
ORDER BY RAPPORTO, PERC\_OVER\_65;

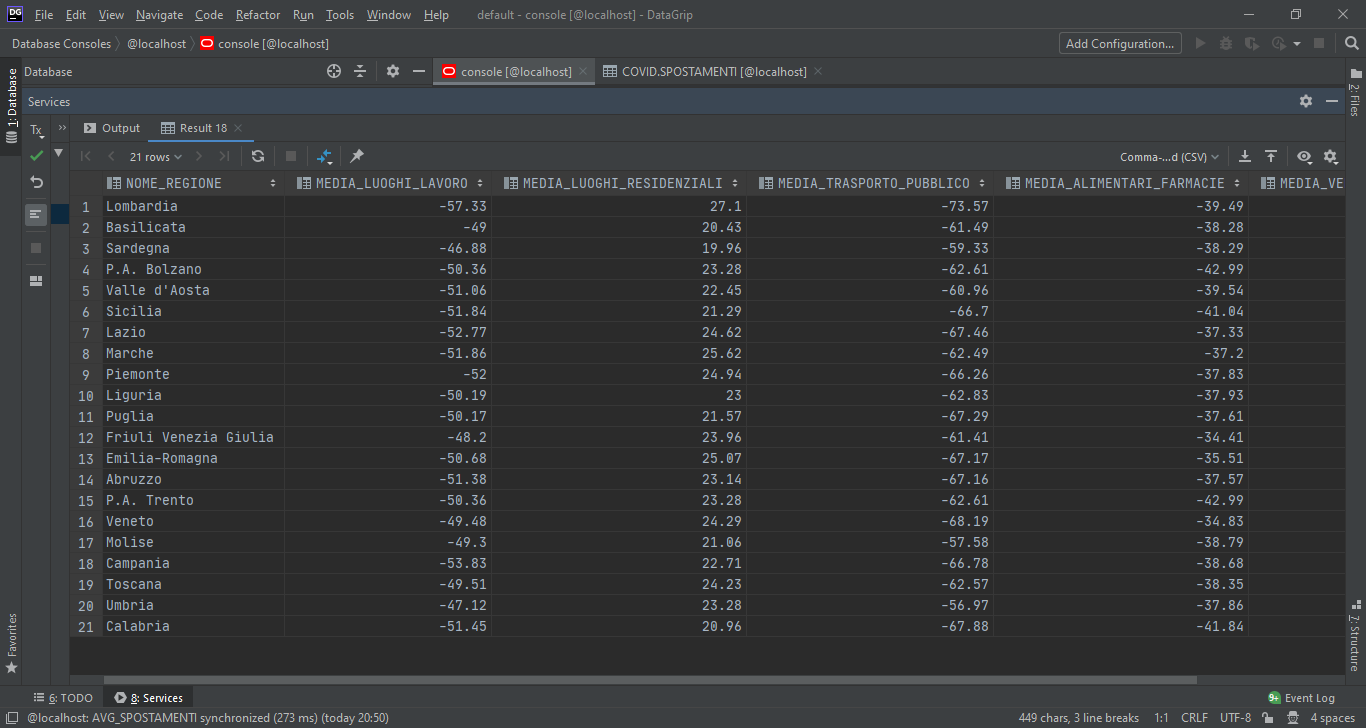
Come è possibile vedere, è bastato aggiungere il campo PERC\_OVER\_65 alla SELECT query precedente per arrivare al risultato, in parte mostrato sotto:



* Altro valore interessante da ricavare è quello della [media, per ogni regione, di tutti i valori relativi agli spostamenti in tutta la finestra temporale](#query).

SELECT R.NOME\_REGIONE, *ROUND*(*AVG*(LUOGHI\_LAVORO),2) AS MEDIA\_LUOGHI\_LAVORO, *ROUND*(*AVG*(LUOGHI\_RESIDENZIALI),2) AS MEDIA\_LUOGHI\_RESIDENZIALI, *ROUND*(*AVG*(TRASPORTO\_PUBBLICO),2) AS MEDIA\_TRASPORTO\_PUBBLICO,   
*ROUND*(*AVG*(ALIMENTARI\_FARMACIE),2) AS MEDIA\_ALIMENTARI\_FARMACIE, *ROUND*(*AVG*(VENDITA\_E\_ATTIVITA\_RICREATIVE),2) AS MEDIA\_VENDITA\_E\_RICREATIVE   
FROM SPOSTAMENTI S JOIN REGIONI R ON S.CODICE\_REGIONE=R.ID\_REGIONE   
GROUP BY R.NOME\_REGIONE;

La query per ricavare i dati richiesti è piuttosto semplice: basta selezionare le medie desiderate e applicare la clausola GROUP BY rispetto al campo NOME\_REGIONE in modo da suddividere i valori interessati dalla funzione per regione. Parte del risultato della query (manca, per questione di spazio, le colonne riferite a MEDIA\_ALIMENTARI\_FARMACIE e MEDIA\_VENDITA\_E\_RICREATIVE) è mostrato nella tabella sottostante:

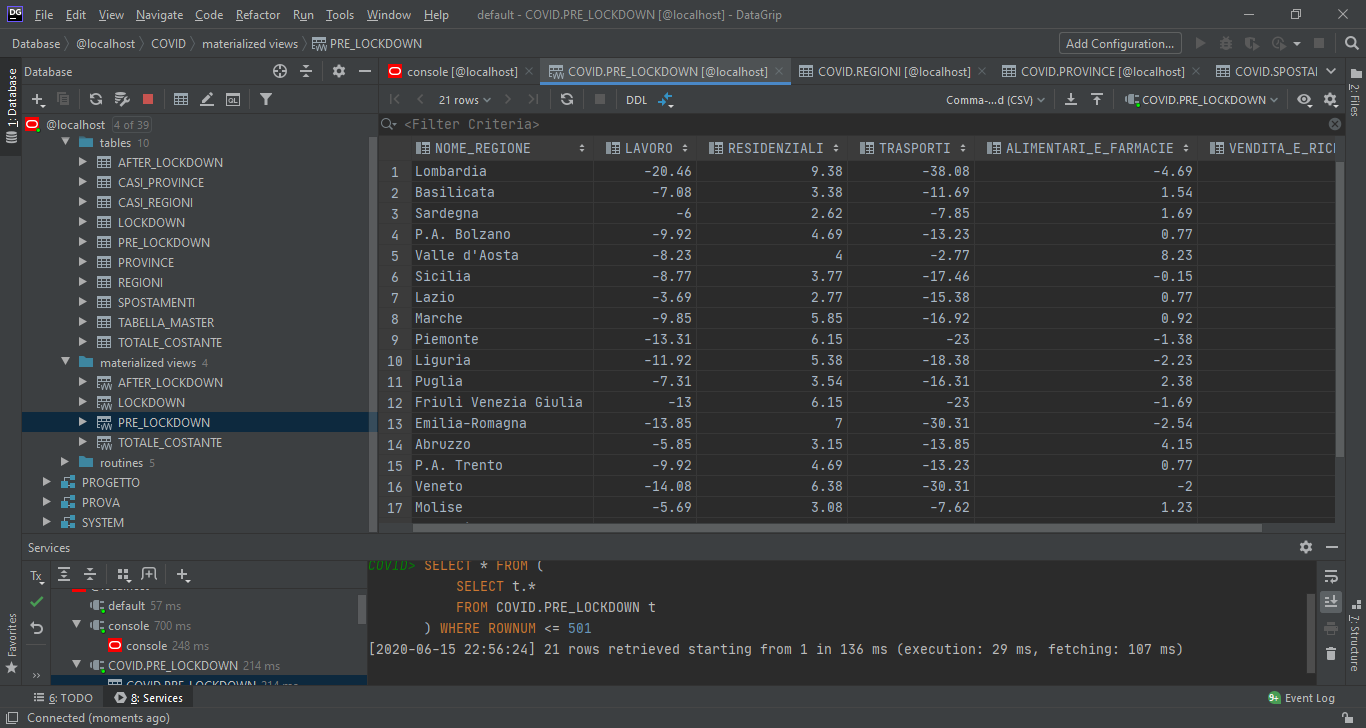


Di peculiare impatto è il grafico associato a questi dati. Da esso è possibile notare a colpo d’occhio come gli spostamenti nelle zone residenziali siano l’unico valore positivo. Si ricorda che questi dati forniscono la variazione percentuale rispetto al valore di riferimento standard (cioè un paio di mesi prima della comparsa dei primi contagi) degli spostamenti in luoghi residenziali, di lavoro, di trasporto pubblico, di alimentari e farmacie e di vendita al dettaglio di beni e di strutture per attività ricreative.

* Di particolare interesse è anche [ricavare i valori medi dei valori della tabella SPOSTAMENTI in determinate sezioni dell’intervallo di interesse](#query). Si considera quindi la media relativa ai valori:
* dal 25/02 al 08/03, la cui corrispettiva vista è riferita come PRE\_LOCKDOWN;
* dal 09/03 al 13/04, cioè dall’inizio del lockdown, inseriti nella vista chiamata LOCKDOWN;
* dal 14/04 al 03/05 cioè dal giorno di ripresa di alcune delle attività commerciali e la fine della raccolta dei dati, inseriti nella vista POST\_LOCKDOWN, nome scelto più per affinità rispetto agli altri che per accuratezza.

I valori trovati permetteranno di poter effettuare un confronto veloce della situazione delle attività (economiche e non) in ogni regione sotto i vari decreti e restrizioni imposte nei periodi selezionati.

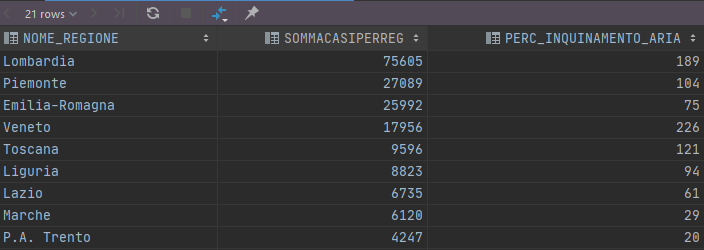
CREATE MATERIALIZED VIEW PRE\_LOCKDOWN AS  
SELECT R.NOME\_REGIONE, *ROUND*(*AVG*(LUOGHI\_LAVORO),2) AS LAVORO, *ROUND*(*AVG*(LUOGHI\_RESIDENZIALI),2) AS RESIDENZIALI,  
 *ROUND*(*AVG*(TRASPORTO\_PUBBLICO),2) AS TRASPORTI, *ROUND*(*AVG*(ALIMENTARI\_FARMACIE),2) AS ALIMENTARI\_E\_FARMACIE,  
 *ROUND*(*AVG*(VENDITA\_E\_ATTIVITA\_RICREATIVE),2) AS VENDITA\_E\_RICREATIVE  
FROM REGIONI R JOIN SPOSTAMENTI S ON CODICE\_REGIONE=R.ID\_REGIONE  
WHERE DATA BETWEEN DATE'2020-02-25' AND DATE'2020-03-08'  
GROUP BY R.NOME\_REGIONE;

Si sceglie, per evitare ridondanze, di mostrare la creazione di solo una delle viste proposte: l’unico elemento a variare sono infatti le date indicate nella condizione WHERE.  
Si noti che la struttura di quest’ultima è molto simile a quella della query indicata nel punto precedente, ad eccezione dell’aggiunta proprio della condizione WHERE che specifica l’arco temporale da considerare nel momento delle media. Parte del risultato della vista (manca, per questione di spazio, la colonna riferita a VENDITA\_E\_RICREATIVE) è mostrato di seguito:

* In questa query è stato oggetto di interesse [la correlazione tra i casi di covid registrati nel consueto arco temporale e l’inquinamento medio dell’aria da polveri sottili](#query) (PM 2.5). Infatti, si stanno conducendo studi secondo i quali l’elevata esposizione cronica al particolato atmosferico rende le persone più vulnerabili nei confronti del virus.

SELECT NOME\_REGIONE, *SUM*(TOT\_CASI) SommaCasiPerReg, INQUINAMENTO\_ARIA  
from REGIONI join PROVINCE P on REGIONI.ID\_REGIONE = P.COD\_REGIONE join CASI\_PROVINCE CP on P.ID\_PROVINCIA = CP.ID\_PROV  
group by NOME\_REGIONE, INQUINAMENTO\_ARIA  
order by SommaCasiPerReg desc

A causa dell’utilizzo della funzione di raggruppamento SUM(TOT\_CASI) è necessaria la presenza della clausola group by che raggruppa i dati per Regione. Per mettere in evidenza quanto detto sopra, si è deciso di ordinare le tuple per i casi complessivi per Regione grazie anche all’uso del comando desc che li ordina in modalità decrescente. Questo è parte del risultato della query (che parrebbe confermare la tesi):



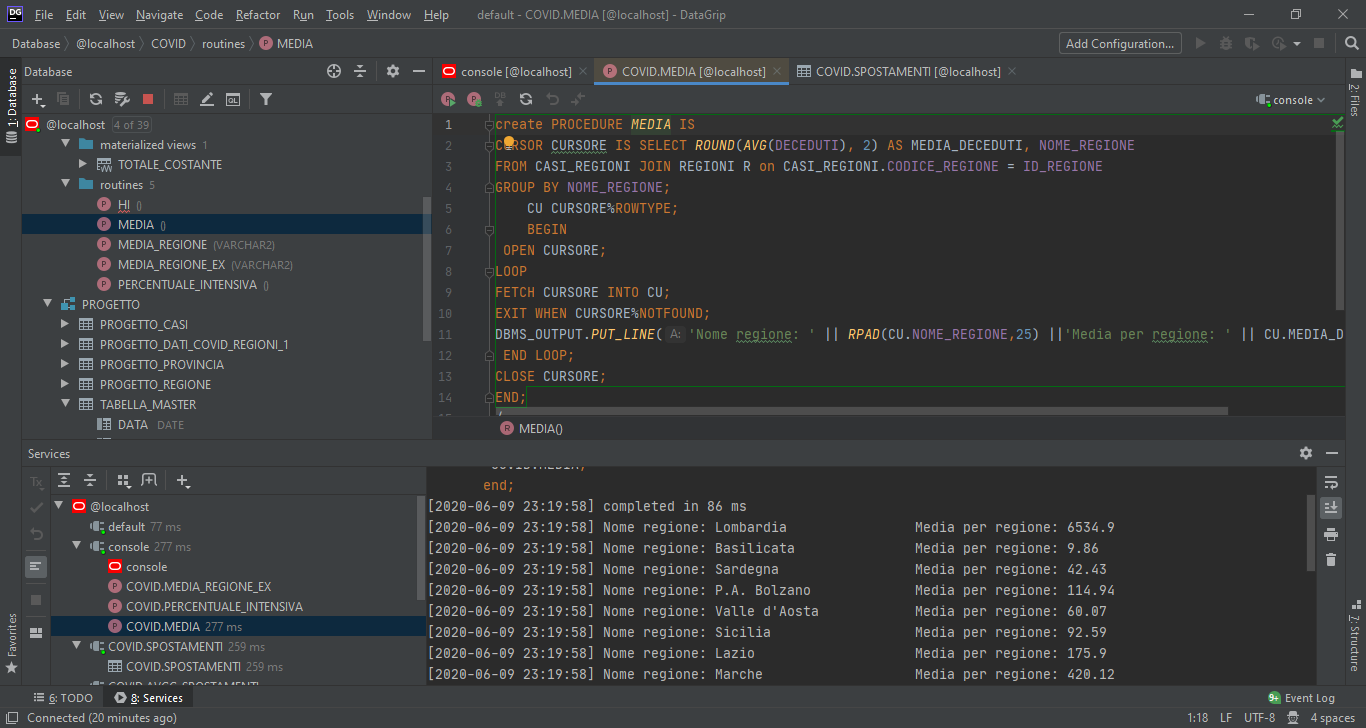
# PL/SQL

## 6.1. Specifica in PL/SQL di Stored Procedure

* La prima procedura proposta stampa semplicemente a video i risultati trovati nella prima query indicata nel paragrafo 5.

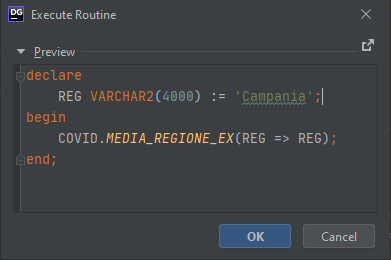
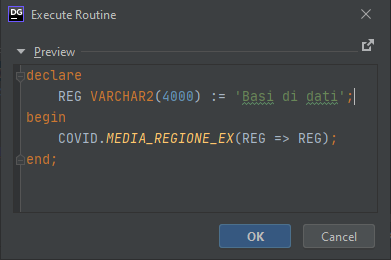
CREATE PROCEDURE *MEDIA* IS   
CURSOR CURSORE IS SELECT *ROUND*(*AVG*(DECEDUTI), 2) AS MEDIA\_DECEDUTI, NOME\_REGIONE   
FROM CASI\_REGIONI JOIN REGIONI R on CASI\_REGIONI.CODICE\_REGIONE = ID\_REGIONE   
GROUP BY NOME\_REGIONE;   
 CU CURSORE%ROWTYPE;   
 BEGIN   
OPEN CURSORE;   
LOOP   
FETCH CURSORE INTO CU;   
EXIT WHEN CURSORE%NOTFOUND;   
DBMS\_OUTPUT.*PUT\_LINE*('Nome regione: ' || *RPAD*(CU.NOME\_REGIONE,25) ||'Media per regione: ' || CU.MEDIA\_DECEDUTI);   
END LOOP;   
CLOSE CURSORE;   
END;

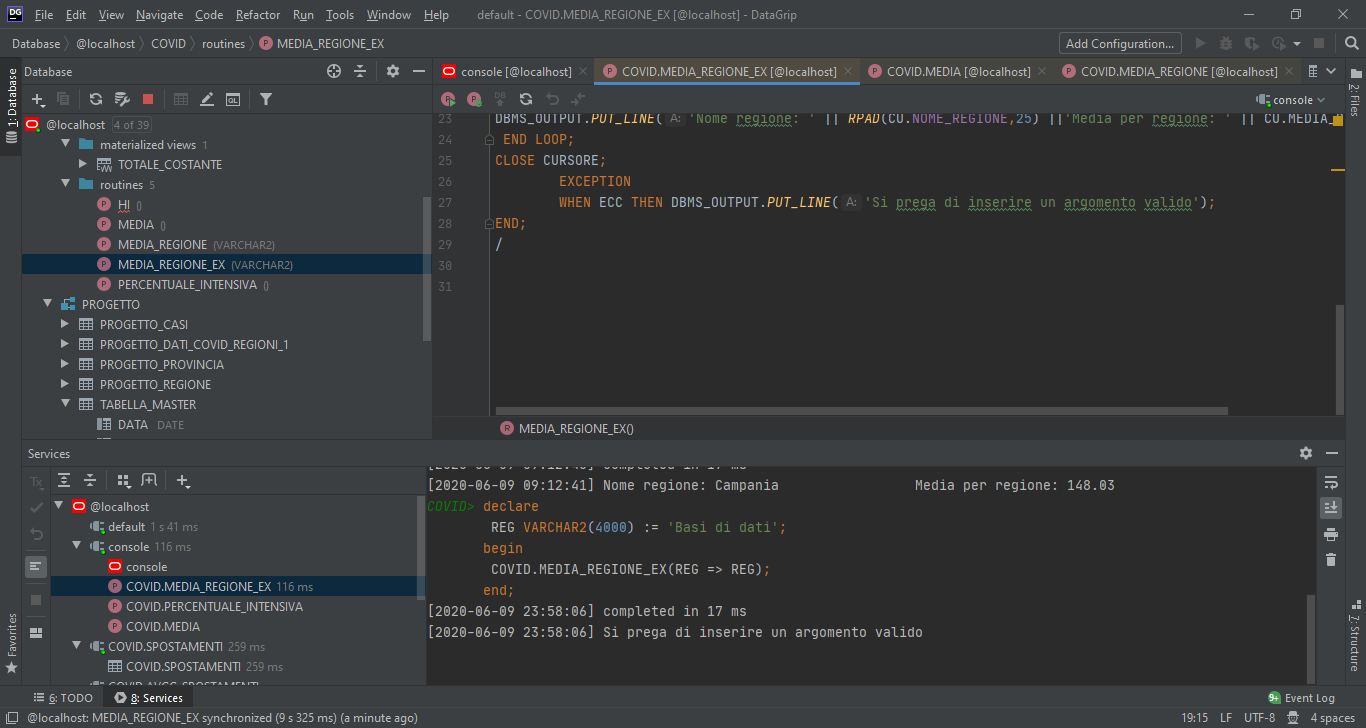
La funzione si occupa di caricare il risultato della query in un cursore e, attraverso un ciclo, esporre in maniera più chiara i risultati.  
Si noti che per permettere il corretto funzionamento di DBMS\_OUTPUT.PUT\_LINE(), è stato necessario utilizzare il comando *Enable DBMSOUTPUT* sulla console.  
Si presenta, di seguito, parte del risultato ottenuto.



* Questa procedura, basata, sempre sulla stessa query, permette di selezionare la particolare regione di cui si è interessati a conoscere la media dei deceduti.

CREATE PROCEDURE *MEDIA\_REGIONE\_EX*(REG IN REGIONI.NOME\_REGIONE%TYPE) IS   
 ECC EXCEPTION;   
CURSOR CURSORE IS SELECT *ROUND*(*AVG*(DECEDUTI), 2) AS MEDIA\_DECEDUTI, NOME\_REGIONE   
FROM CASI\_REGIONI JOIN REGIONI R on CASI\_REGIONI.CODICE\_REGIONE = ID\_REGIONE   
GROUP BY NOME\_REGIONE   
HAVING NOME\_REGIONE=REG;   
 CU CURSORE%ROWTYPE;   
 BEGIN   
 IF (   
 (REG != 'Campania') AND (REG != 'Piemonte') AND (REG !='Valle d''Aosta') AND (REG !='Lombardia')   
 AND (REG !='Veneto') AND (REG !='Friuli Venezia Giulia') AND (REG !='Liguria') AND   
 (REG !='Emilia-Romagna') AND (REG !='Toscana') AND (REG !='Umbria') AND (REG !='Marche')   
 AND (REG !='Lazio') AND (REG !='Abruzzo') AND (REG !='Molise') AND (REG !='Puglia')   
 AND (REG !='Basilicata') AND (REG !='Calabria') AND (REG !='Sicilia')   
 AND (REG !='Sardegna') AND (REG !='P.A. Bolzano') AND (REG !='P.A. Trento')   
 )   
 THEN RAISE ECC;   
 END IF;   
OPEN CURSORE;   
LOOP   
FETCH CURSORE INTO CU;   
EXIT WHEN CURSORE%NOTFOUND;   
DBMS\_OUTPUT.*PUT\_LINE*('Nome regione: ' || *RPAD*(CU.NOME\_REGIONE,25) ||'Media per regione: ' || CU.MEDIA\_DECEDUTI);   
END LOOP;   
CLOSE CURSORE;   
 EXCEPTION   
 WHEN ECC THEN DBMS\_OUTPUT.*PUT\_LINE*('Si prega di inserire un argomento valido');   
END;

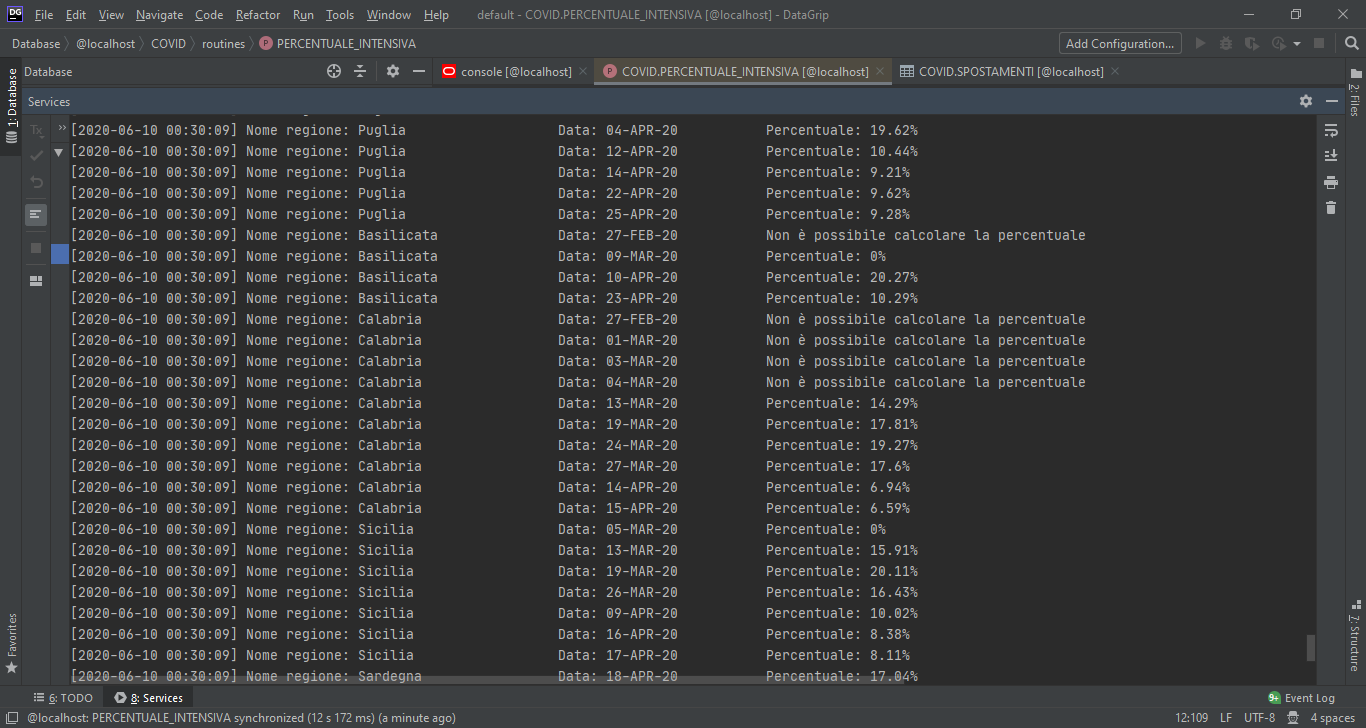
La procedura richiede quindi la presenza di un parametro di ingresso, il cui riferimento formale è REG, che permette l’inserimento da parte dell’utente della regione.  
La possibilità di inserire manualmente un parametro rende però necessario che sia possibile controllare cosa viene inserito ed agire di conseguenza: viene dunque definita e gestita un’eccezione, ECC, che si attiva nel momento in cui la stringa inserita non corrisponda a nessuna di quelle che indichino i nomi delle regioni e che mette in output un messaggio di errore.  
Si presenta, di seguito, il risultato ottenuto nel caso in cui la stringa in input sia valida, e nel caso in cui non lo sia.



* La procedura presentata indica, per ogni giorno e per ogni provincia, la percentuale, tra tutti gli ospedalizzati a causa del COVID19, dei pazienti in terapia intensiva, il che potrebbe fornire dati interessanti sulla gravità dei casi per ogni provincia.

CREATE PROCEDURE *PERCENTUALE\_INTENSIVA* IS  
CURSOR CURSORE IS SELECT R.NOME\_REGIONE, C.DATA, C.TOTALE\_OSPEDALIZZATI, C.TERAPIA\_INTENSIVA  
 FROM CASI\_REGIONI C JOIN REGIONI R on C.CODICE\_REGIONE = R.ID\_REGIONE  
 GROUP BY R.NOME\_REGIONE, C.DATA, C.TERAPIA\_INTENSIVA, C.TOTALE\_OSPEDALIZZATI;  
 CU CURSORE%ROWTYPE;  
 BEGIN  
OPEN CURSORE;  
LOOP  
FETCH CURSORE INTO CU;  
EXIT WHEN CURSORE%NOTFOUND;  
IF (CU.TOTALE\_OSPEDALIZZATI=0)  
 THEN DBMS\_OUTPUT.*PUT\_LINE*('Nome regione:'  
|| *RPAD*(CU.NOME\_REGIONE,25) ||'Data: ' || *RPAD*(CU.DATA,20) ||  
 'Persone in terapia intensiva: '|| *RPAD*(CU.TERAPIA\_INTENSIVA,10) ||  
 'Totale persone ospedalizzate: ' || *RPAD*(CU.TOTALE\_OSPEDALIZZATI,10) ||  
 'Non è possibile calcolare la percentuale');  
  
 ELSE DBMS\_OUTPUT.*PUT\_LINE*('Nome regione: ' || *RPAD*(CU.NOME\_REGIONE,25) ||'Data: ' || *RPAD*(CU.DATA,20)||  
 'Persone in terapia intensiva: '|| *RPAD*(CU.TERAPIA\_INTENSIVA,10) ||  
 'Totale persone ospedalizzate: ' || *RPAD*(CU.TOTALE\_OSPEDALIZZATI,10) ||  
 'Percentuale in terapia intensiva degli ospedalizzati: '|| *ROUND*(((100\*CU.TERAPIA\_INTENSIVA)/CU.TOTALE\_OSPEDALIZZATI),2) || '%');  
END IF;  
END LOOP;  
CLOSE CURSORE;  
END;

Anche in questo caso, per analizzare i dati è necessario l’uso di un cursore, che recupera i valori di NOME\_REGIONE, DATA, TERAPIA\_INTENSIVA, TOTALE\_OSPEDALIZZATI e li manipola per calcolare la percentuale desiderata.   
La variazione giornaliera del TOTALE\_OSPEDALIZZATI fa si che sia necessario implementare un costrutto *if* che gestisca il caso in cui TOTALE\_OSPEDALIZZATI sia nullo, in modo da non avere istanze in cui capiti una divisione per zero. L’*if* si assicura, quindi, che nel caso TOTALE\_OSPEDALIZZATI=0 venga presentato il messaggio 'Non è possibile calcolare la percentuale' al posto del valore percentuale. Si presenta, di seguito, parte del risultato ottenuto.



## Specifica in PL/SQL di Trigger

Il trigger inserisce i valori presi dall’insert nella Master nella tabella CASI\_PROVINCE.

create trigger TRIG   
    after insert   
    on TABELLA\_MASTER   
    for each row   
BEGIN   
        INSERT INTO CASI\_PROVINCE (CASI\_PROVINCE.DATA, CASI\_PROVINCE.ID\_PROV, CASI\_PROVINCE.TOT\_CASI)   
        VALUES (:NEW.DATA, :NEW.CODICE\_PROVINCIA, :NEW.TOT\_CASI);   
    end;

La scelta effettuata è stata quella di dare a questo trigger una struttura relativamente semplice: la sua azione è infatti quella di inserire i dati aggiunti alla tabella master nella tabella CASI\_PROVINCE.    
Si è scelto di non apportare modifiche alle tabelle CASI\_REGIONI e SPOSTAMENTI in quanto, pur avendo disponibili DATA e CODICE\_REGIONE (le chiavi primarie di entrambe le tabelle) e quindi ottenendo un corretto inserimento di una nuova tupla, la tabella master non contiene nessun ulteriore dato che possa concernere queste due tabelle. Ogni aggiunta proveniente dalla tabella master fatta a una di queste due tabelle sarebbe quindi al meglio incompleta, e potrebbe causare errori nell’esecuzione di query.   
È poi sorto il problema di come gestire errori riguardanti l’inserimento di campi non conformi nella tabella master, e le ripercussioni che causerebbero nello schema normalizzato (es: inserimento in CODICE\_REGIONE di un valore che non corrisponde a nessuno di quelli già presenti in tabella e che porterebbe alla necessità di inserire una nuova tupla in REGIONI). A riguardo, si è deciso di non prendere alcuna iniziativa per due motivi principali: in prima istanza, qualsiasi aggiunta di tuple con eventuali errori di codici provincia verrebbe rifiutata dal DBMS che in automatico controlla la consistenza dei dati (come del resto già successo durante l’inserimento di dati con refusi del file d’origine); in seconda istanza, data la natura di questa base di dati,  ogni tipo di informazione aggiunta non è inserita manualmente, ma importata da file affidabili (provengono direttamente dalla Protezione Civile), il che permette di essere sicuri della conformità dei dati inseriti senza ulteriori controlli.