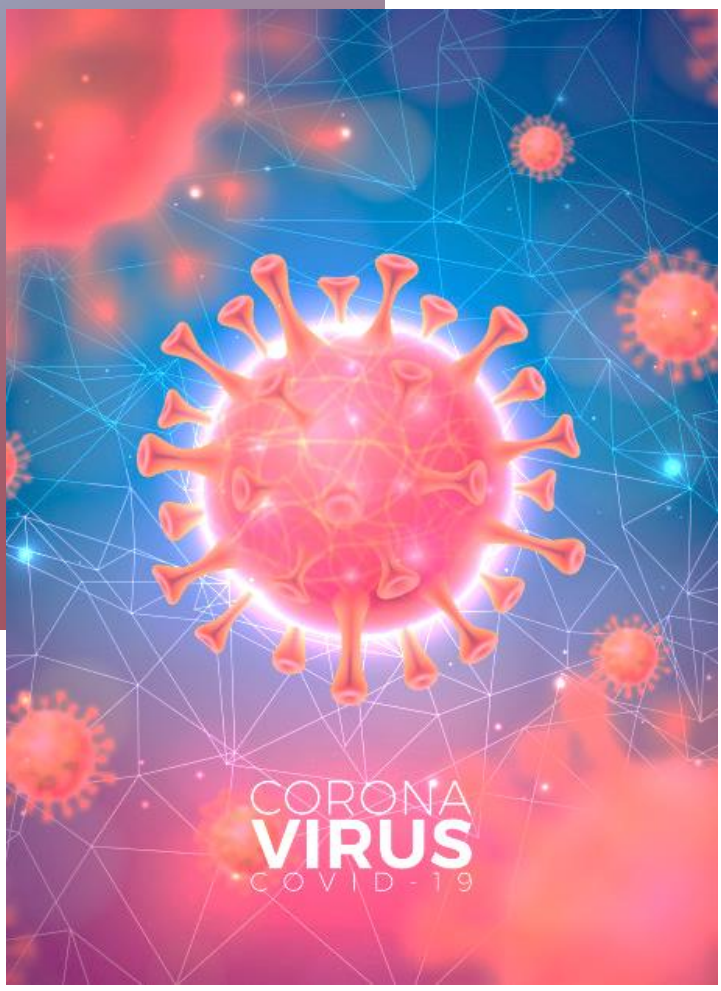
UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II



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

Si vuole realizzare una base di dati, a partire dal dataset fornito dalla specifica e reperibile sul profilo GitHub della Croce Rossa, 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

1. 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.

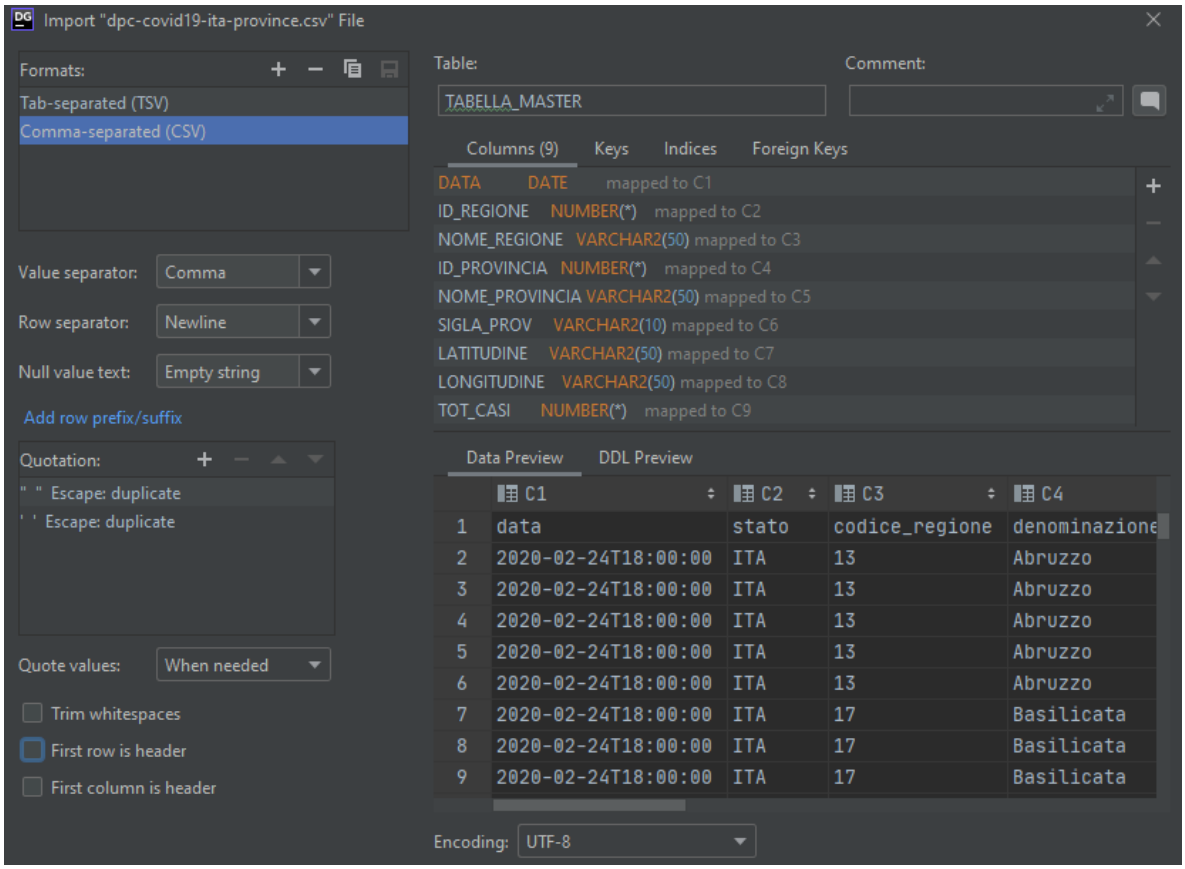
1.3. Creazione della tabella master e relativo popolamento

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
```

2. 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)
);
```

3. 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) 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:

DATA	CODICE_REGIONE	DEN_REGIONE	RICOVERATI	TER_INTENSIVA	TOTALE_OSPEDALIZZATI	ISOLAMENTO	GUARITI	MORTI	TAMPONI
02/03/2020 18:00	11	Marche	17	6	23	11	0	1	137
02/03/2020 18:00	14	Molise	0	0	0	0	0	0	13
02/03/2020 18:00	1	Piemonte	12	2	14	37	0	0	434
02/03/2020 18:00	16	Puglia	2	0	2	2	0	0	278
02/03/2020 18:00	20	Sardegna	0	0	0	0	0	0	29
02/03/2020 18:00	19	Sicilia	2	0	2	3	2	0	307

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), 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 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 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,
);
```

4. 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

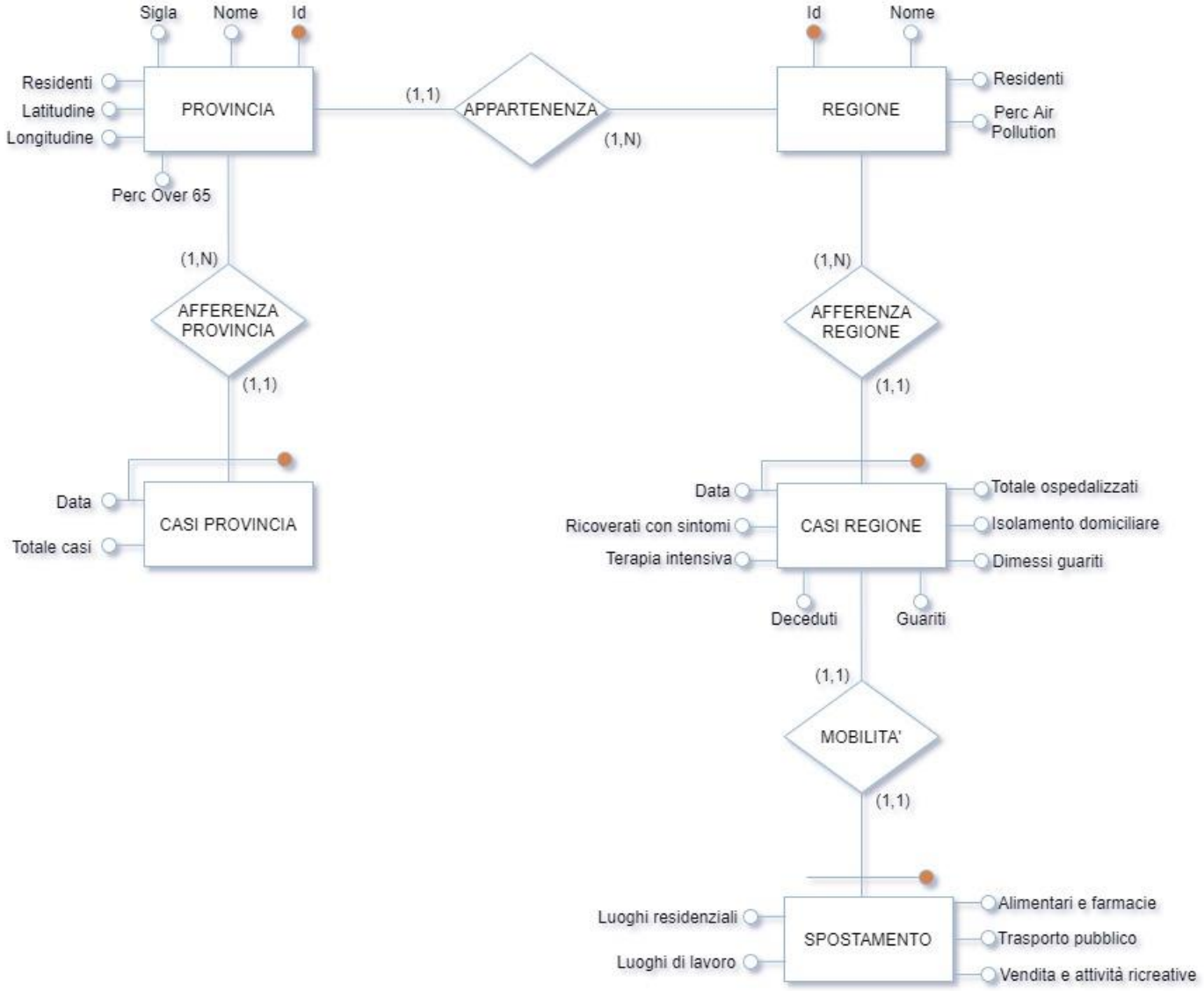
Dalle relazioni si individuano le entità presenti nel modello: REGIONE, PROVINCIA, CASI REGIONE, CASI PROVINCIA e SPOSTAMENTO.

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:



5. 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;
- media dei contagiati per ogni mese in ciascuna provincia; G☺
- giorno in cui c'è stato il picco di contagi per una specifica provincia;
- giorno in cui c'è stato il picco di contagi per tutte le province;
- per ogni regione, la provincia con più contagiati nel periodo temporale;
- la percentuale dei casi in rapporto al numero di residenti per provincia nella finestra temporale; G☺
- la percentuale dei casi in rapporto al numero di over 65 per provincia nella finestra temporale;
- media, per ogni regione, di tutti i valori relativi agli spostamenti in tutto il periodo; G☺
- valor medio degli spostamenti nei periodi pre, post e durante lockdown; G☺
- casi per regione e correlazione con l'inquinamento dell'aria da polveri sottili.

- La prima delle query proposte indaga sulla media di persone decedute in ogni regione durante l'intervallo di tempo studiato:

```
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:

	MEDIA_DECEDUTI 1	NOME_REGIONE
1	6534.9	Lombardia
2	1590.9	Emilia-Romagna
3	1107.1	Piemonte
4	547.1	Veneto
5	469.2	Liguria
6	420.1	Marche
7	308.5	Toscana
8	175.9	Lazio
9	171.1	P.A. Trento
10	156.3	Puglia
11	148	Campania
12	125.9	Abruzzo
13	119.3	Friuli Venezia Giulia
14	114.9	P.A. Bolzano
15	92.6	Sicilia
16	60.1	Valle d'Aosta
17	42.4	Sardegna
18	36.3	Calabria
19	30.4	Umbria
20	9.9	Basilicata
21	9.3	Molise

- 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.
  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:

utput | MEDIA_CONTAGIATI_MESE_PROVINCIA

428 rows

NOME_PROVINCIA ▲ 1	MESE	MEDIA_CONTAGIATI_MESE
Bergamo	2	18.4
Bergamo	3	280.42
Bergamo	4	83.67
Bergamo	5	46.67
Biella	2	0
Biella	3	15.84
Biella	4	15.87

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:

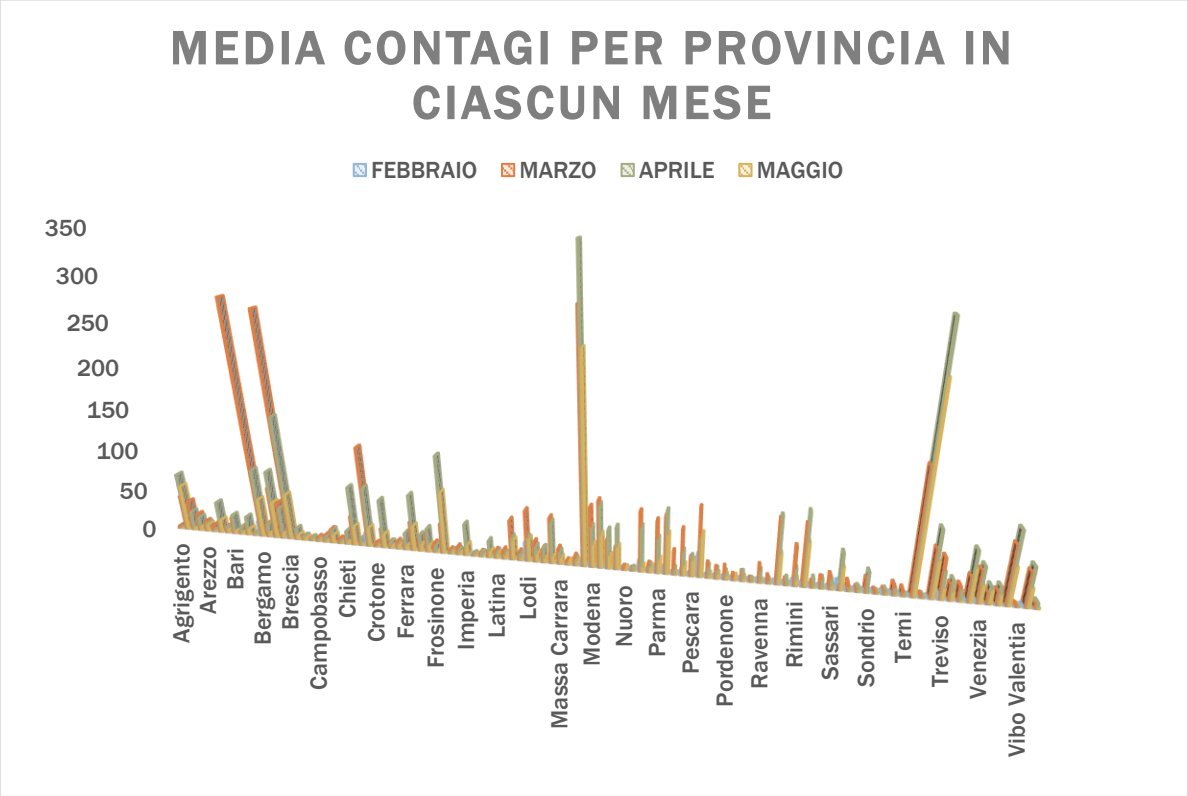
```
--creato vista con visualizzazione mensile
CREATE MATERIALIZED VIEW MEDIA_CONT_FEB AS
SELECT NOME_PROVINCIA, MEDIA_CONTAGIATI_MESE AS FEBBRAIO
FROM PROGETTO.MEDIA_CONTAGIATI_MESE_PROVINCIA
WHERE MESE=2
```

```
--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
```

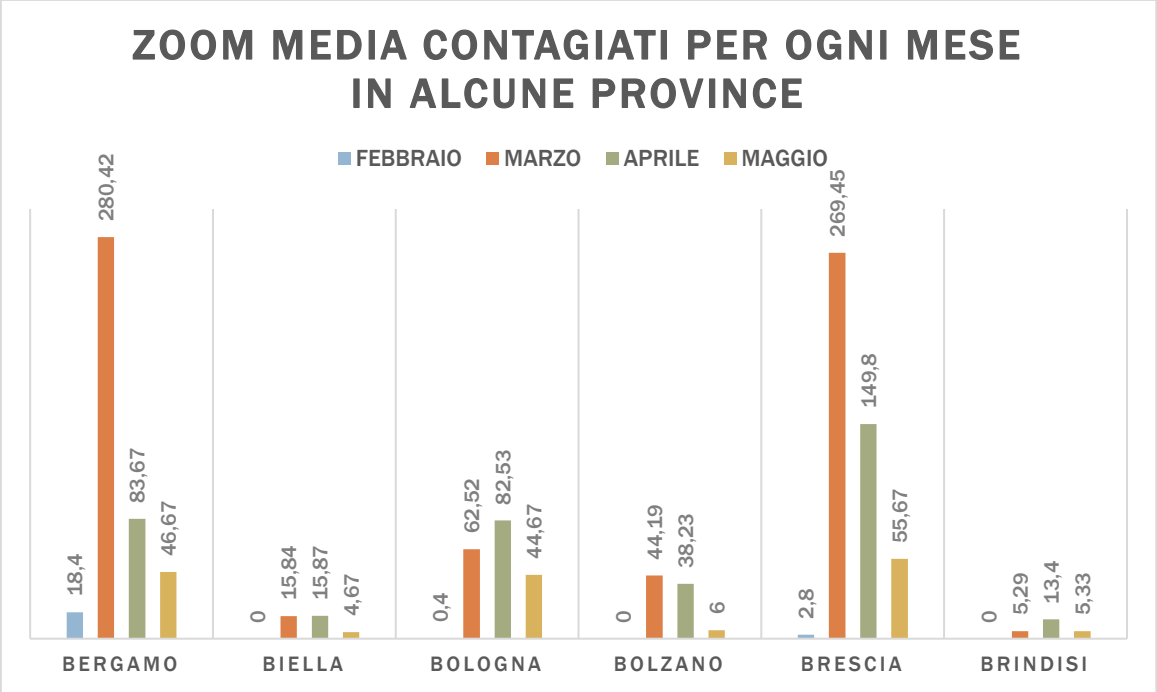
Il risultato della join di seguito:

NOME_PROVINCIA	FEBBRAIO	MARZO	APRILE	MAGGIO
Agrigento	0	3.03	1.37	0
Alessandria	0	42.48	70	55.67
Ancona	0	36	23.13	2.33
Aosta	0	20.26	16.67	4.67
Arezzo	0	11.65	9.73	8
Ascoli Piceno	0	7.16	1.97	0.33
Asti	0	14	37.93	15.67
Avellino	0	7.29	7.73	4.33
Bari	0.2	19.87	23.23	3
Barletta-Andria-Trani	0	3.58	8.73	2
Belluno	0	13.87	22.57	7.33
Benevento	0	2.52	3.53	0.33

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



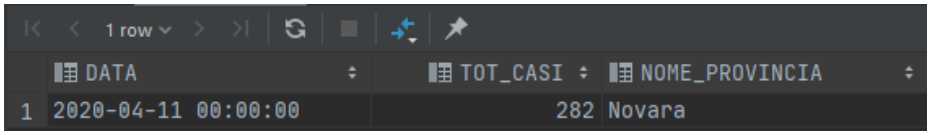
Di seguito si propone un dettaglio dello stesso per agevolare la visione:



➲ 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.

```
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.

```
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:

	DATA	TOT_CASI	NOME_PROVINCIA
16	2020-04-12 00:00:00	141	Bolzano
17	2020-03-23 00:00:00	588	Brescia
18	2020-04-01 00:00:00	27	Brindisi
19	2020-04-19 00:00:00	27	Brindisi
20	2020-04-10 00:00:00	18	Cagliari
21	2020-03-24 00:00:00	11	Caltanissetta
22	2020-03-29 00:00:00	11	Caltanissetta

- Questa query propone di selezionare, per ogni regione, la provincia con più contagiati.
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):

	COD_REGIONE	NOME_PROVINCIA	CASI_TOTALI
1	1	Alessandria	3584
2	1	Novara	2371
3	1	Biella	981
4	1	Cuneo	2535
5	1	Asti	1619
6	1	Vercelli	1148
7	1	Verbano-Cusio-Ossola	1055

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:

	NOME_REGIONE ▲ 1	NOME_PROVINCIA	CASI_TOTALI
1	Abruzzo	Pescara	1352
2	Basilicata	Matera	199
3	Calabria	Cosenza	460
4	Campania	Napoli	2481
5	Emilia-Romagna	Reggio nell'Emilia	4765
6	Friuli Venezia Giulia	Trieste	1296
7	Lazio	Roma	4923
8	Liguria	Genova	4860
9	Lombardia	Milano	20060
10	Marche	Pesaro e Urbino	2540
11	Molise	Campobasso	227
12	P.A. Bolzano	Bolzano	2535
13	P.A. Trento	Trento	4247
14	Piemonte	Torino	13796
15	Puglia	Bari	1323
16	Sardegna	Sassari	848
17	Sicilia	Catania	1002
18	Toscana	Firenze	3213
19	Umbria	Perugia	996
20	Valle d'Aosta	Aosta	1142
21	Veneto	Verona	4800

- 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 coperta dal dataset. Utilizzando ancora la vista materializzata della precedente interrogazione (qui), 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:

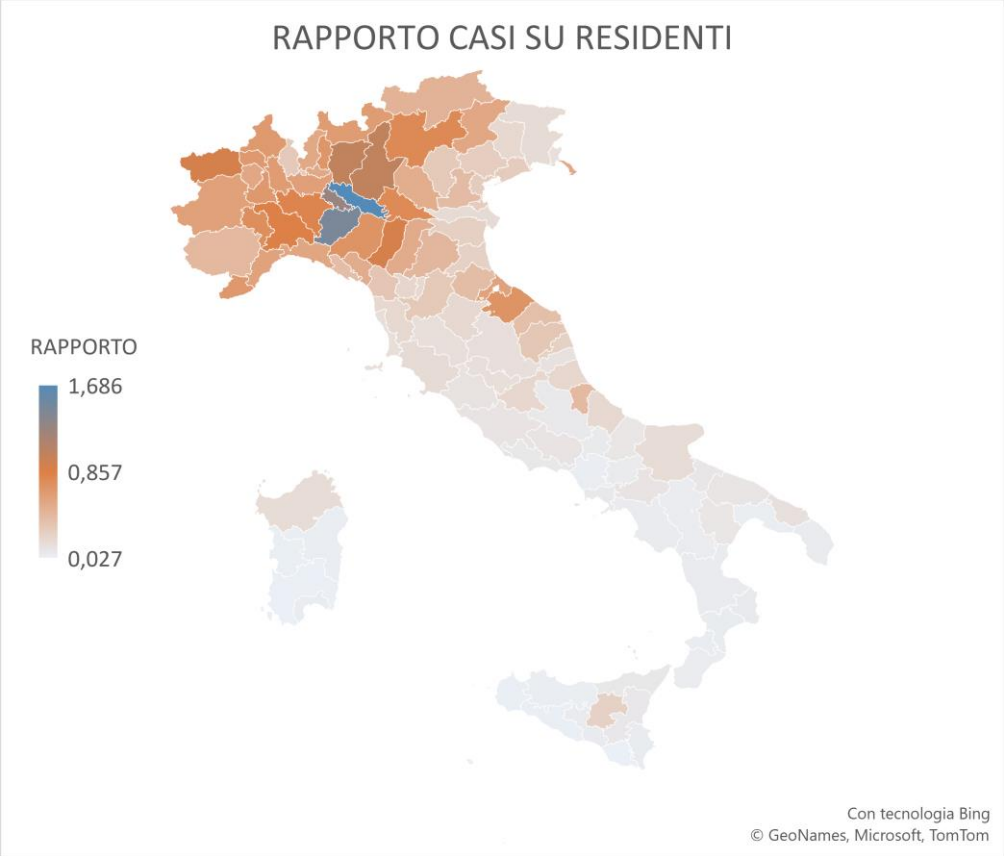
```
SELECT P.NOME_PROVINCIA, ROUND(((CASI_TOTALI*100)/RESIDENTI),3) AS RAPPORTO
FROM   PROVINCE P, PROGETTO.TOTALE_COSTANTE T
WHERE NOME_PROVINCIA= T.NOME_PROVINCIA
ORDER BY NOME_PROVINCIA, RAPPORTO
```

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

107 rows

	NOME_PROVINCIA	RAPPORTO_PERCENTUALE 1
1	Cremona	1.686
2	Piacenza	1.461
3	Lodi	1.269
4	Brescia	1.029
5	Bergamo	1.026
6	Aosta	0.909
7	Reggio nell'Emilia	0.896
8	Alessandria	0.851
9	Pavia	0.818
10	Trento	0.785
11	Mantova	0.776
12	Asti	0.754
13	Parma	0.709
14	Pesaro e Urbino	0.708
15	Lecco	0.695

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, 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:

	NOME_PROVINCIA	RAPPORTO	PERC_OVER_65
1	Sud Sardegna	0.027	22.7
2	Ragusa	0.029	25.4
3	Agrigento	0.031	22
4	Trapani	0.032	22.1
5	Oristano	0.035	22.4
6	Nuoro	0.039	26.5
7	Palermo	0.041	23.2
8	Taranto	0.046	23.1
9	Caserta	0.047	19.7
10	Reggio di Calabria	0.053	21.6
11	Potenza	0.053	22.2
12	Vibo Valentia	0.054	21.8
13	Cagliari	0.057	21

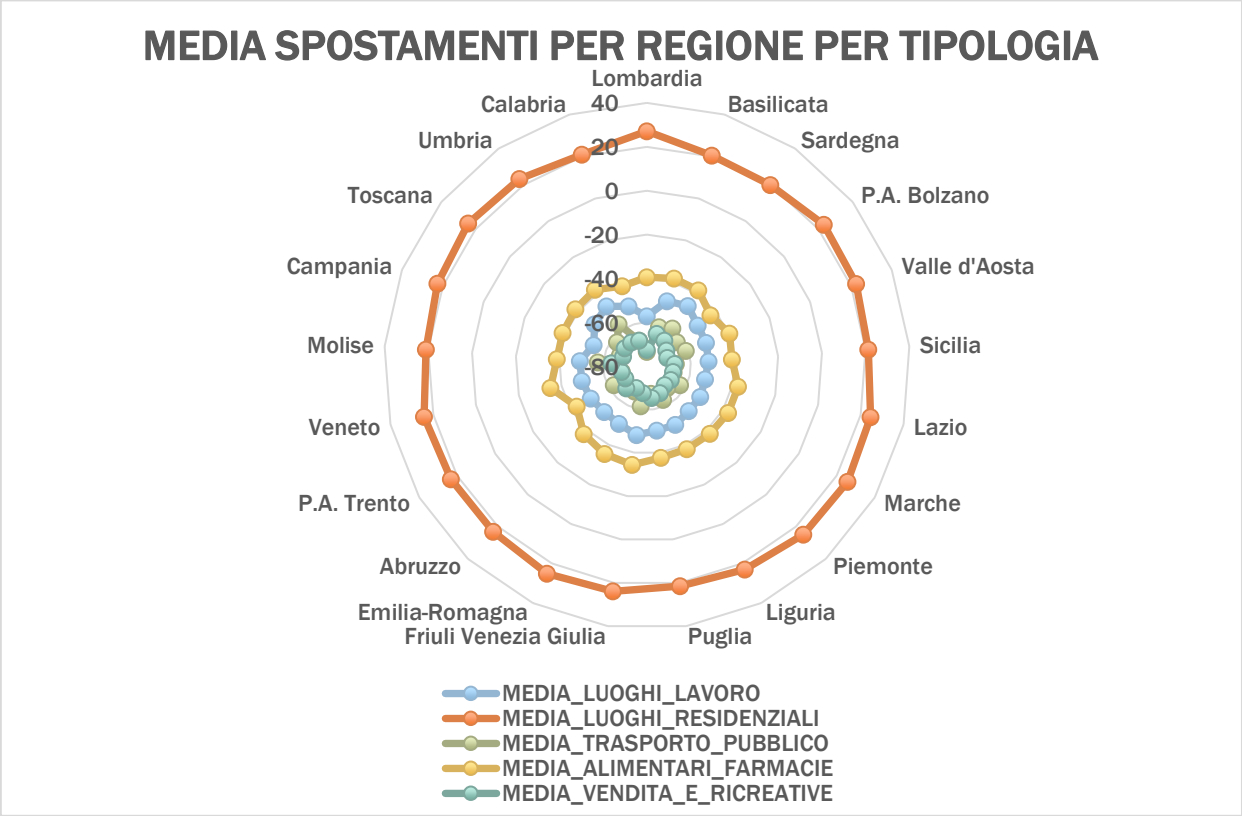
- Altro valore interessante da ricavare è quello della media, per ogni regione, di tutti i valori relativi agli spostamenti in tutta la finestra temporale.

```
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:

	NOME_REGIONE	MEDIA_LUOGHI_LAVORO	MEDIA_LUOGHI_RESIDENZIALI	MEDIA_TRASPORTO_PUBBLICO
1	Lombardia	-57.33	27.1	-73.57
2	Basilicata	-49	20.43	-61.49
3	Sardegna	-46.88	19.96	-59.33
4	P.A. Bolzano	-50.36	23.28	-62.61
5	Valle d'Aosta	-51.06	22.45	-60.96
6	Sicilia	-51.84	21.29	-66.7

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 (di 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.



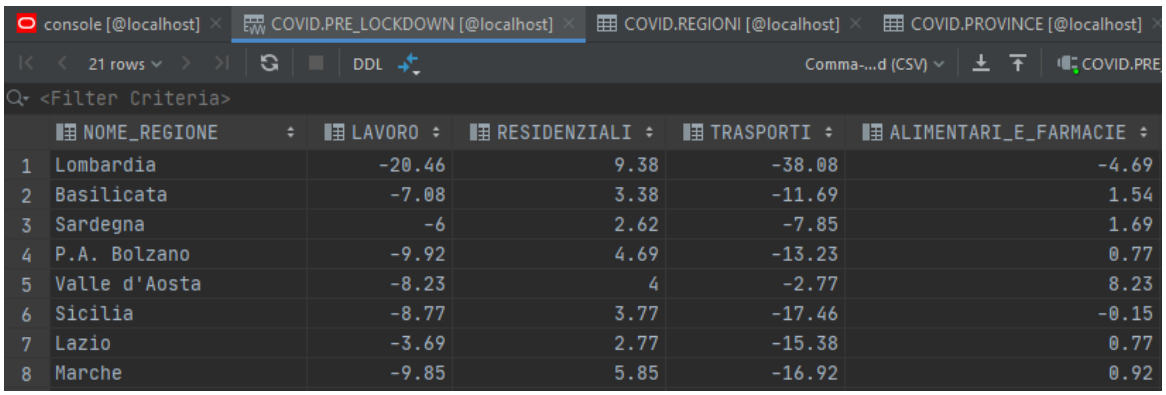
- Di particolare interesse è anche ricavare i valori medi dei valori della tabella SPOSTAMENTI in determinate sezioni dell'intervallo di interesse. Si considera quindi la media relativa ai valori:

  - 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:

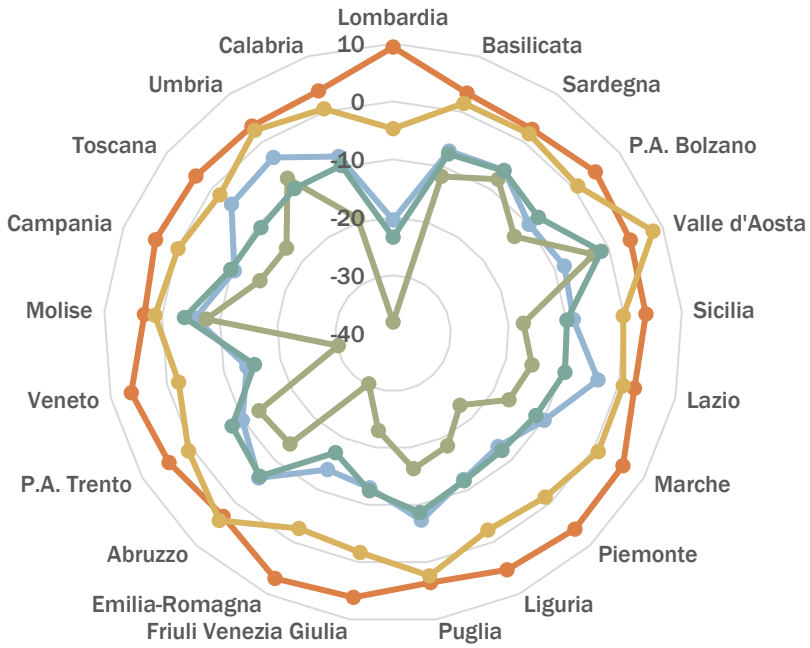

console [@localhost] | COVID.PRE_LOCKDOWN [@localhost] | COVID.REGIONI [@localhost] | COVID.PROVINCE [@localhost]

21 rows | DDL | Comma-...d (CSV) | COVID.PRE

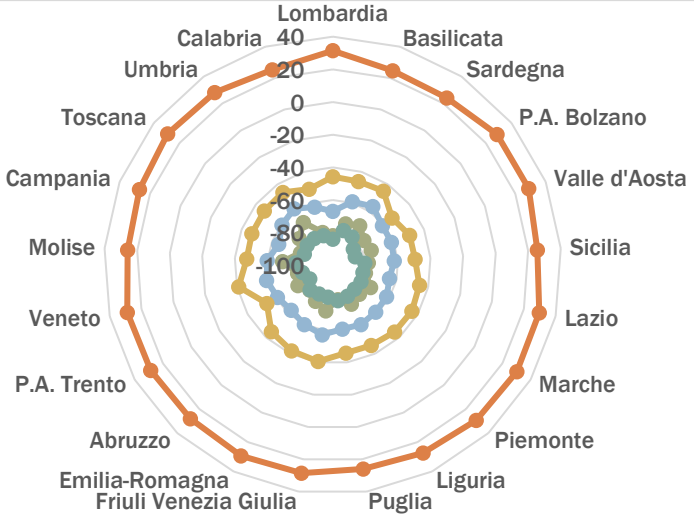
<Filter Criteria>

	NOME_REGIONE	LAVORO	RESIDENZIALI	TRASPORTI	ALIMENTARI_E_FARMACIE
1	Lombardia	-20.46	9.38	-38.08	-4.69
2	Basilicata	-7.08	3.38	-11.69	1.54
3	Sardegna	-6	2.62	-7.85	1.69
4	P.A. Bolzano	-9.92	4.69	-13.23	0.77
5	Valle d'Aosta	-8.23	4	-2.77	8.23
6	Sicilia	-8.77	3.77	-17.46	-0.15
7	Lazio	-3.69	2.77	-15.38	0.77
8	Marche	-9.85	5.85	-16.92	0.92

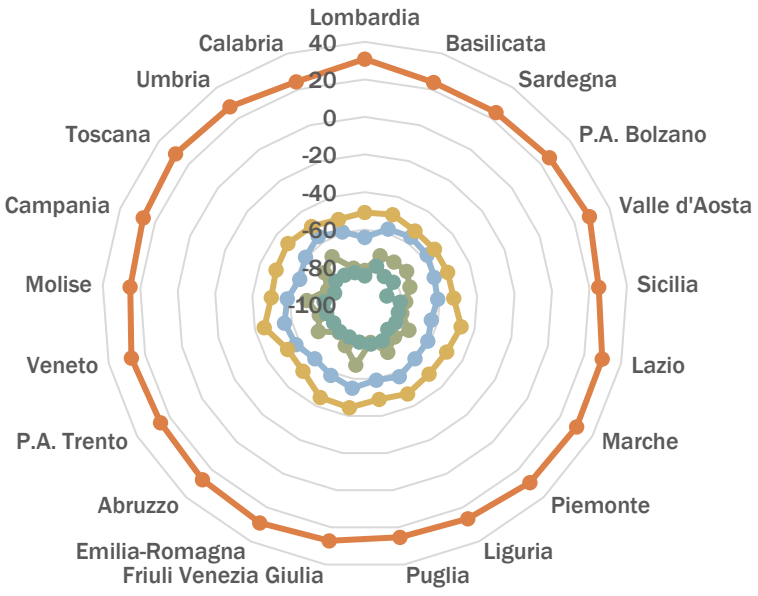
PRE-LOCKDOWN



LOCKDOWN



SITUAZIONE DAL 14 APRILE



- 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 (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):

21 rows

NOME_REGIONE	SOMMACASIPERREG	PERC_INQUINAMENTO_ARIA
Lombardia	75605	189
Piemonte	27089	104
Emilia-Romagna	25992	75
Veneto	17956	226
Toscana	9596	121
Liguria	8823	94
Lazio	6735	61
Marche	6120	29
P.A. Trento	4247	20

6. 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;
```

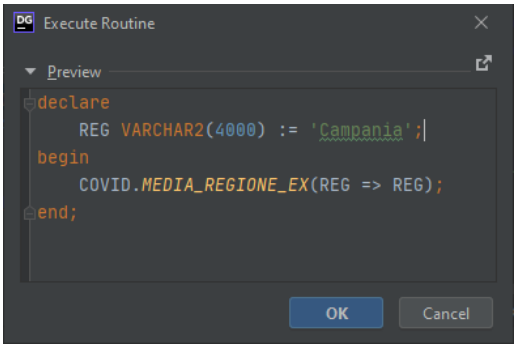
La funzione si occupa di caricare il risultato della query in un cursore e, attraverso un ciclo, esporre in maniera più chiara i risultati.
Si noti che per permettere il corretto funzionamento di DBMS_OUTPUT.PUT_LINE(), è stato necessario utilizzare il comando *Enable DBMSOUTPUT* sulla console.
Si presenta,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

```
Lombardia                 Media per regione: 6534.9
Basilicata                Media per regione: 9.86
Sardegna                  Media per regione: 42.43
P.A. Bolzano              Media per regione: 114.94
Valle d'Aosta             Media per regione: 60.07
Sicilia                   Media per regione: 92.59
Lazio                     Media per regione: 175.9
Marche                    Media per regione: 420.12
```

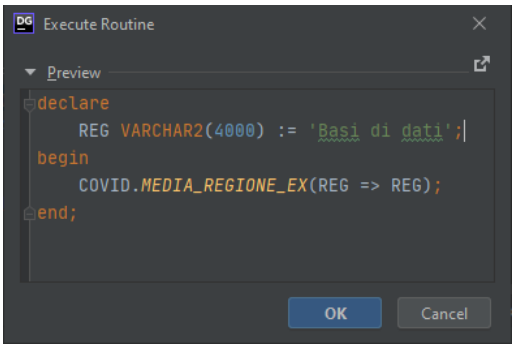
⇨ Questa procedura, basata sempre sulla stessa query, permette di selezionare la particolare regione di cui si è interessati a conoscere la media dei deceduti.

```
  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.




```
Nome regione:Campania  Media per regione:148.03
```




```
Si prega di inserire un argomento valido
```

⇨ 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.

```
Puglia                Data: 04-APR-20         Percentuale: 19.62%
Puglia                Data: 12-APR-20         Percentuale: 10.44%
Puglia                Data: 14-APR-20         Percentuale: 9.21%
Puglia                Data: 22-APR-20         Percentuale: 9.62%
Puglia                Data: 25-APR-20         Percentuale: 9.28%
Basilicata            Data: 27-FEB-20         Non è possibile calcolare la percentuale
Basilicata            Data: 09-MAR-20         Percentuale: 0%
Basilicata            Data: 10-APR-20         Percentuale: 20.27%
Basilicata            Data: 23-APR-20         Percentuale: 10.29%
Calabria              Data: 27-FEB-20         Non è possibile calcolare la percentuale
Calabria              Data: 01-MAR-20         Non è possibile calcolare la percentuale
Calabria              Data: 03-MAR-20         Non è possibile calcolare la percentuale
Calabria              Data: 04-MAR-20         Non è possibile calcolare la percentuale
Calabria              Data: 13-MAR-20         Percentuale: 14.29%
Calabria              Data: 19-MAR-20         Percentuale: 17.81%
```

6.2 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) che rendono possibile un corretto inserimento di una nuova tupla in tali tabelle, 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 codici provincia diversi da quelli già presenti nella base di dati 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), mentre gli altri dati semplicemente non sono letti dal trigger (e dunque qualsiasi incongruenza che li riguardi viene ignorata), visto che i vincoli legati alla tabella CASI_PROVINCE forniscono automaticamente i valori associati al CODICE_PROVINCIA inserito;
- 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.