

Università di Napoli Federico II

Ingegneria informatica

# TESINA DI BASI DI DATI

COVID19: La situazione in Italia

A cura di

CONTI MATTEO  
FAZZARI DANIELE



# Indice

Introduzione	3
Normalizzazione	4
Creazione delle base dati	5
Descrizione dei comandi DDL	5
Popolamento delle tabelle normalizzate	6
Arricchimento dello schema	7
Reverse engineering	10
Modello Portante	10
Modello Entity Relationship	10
Analytics	12
Programmazione della base dati	25
Trigger	25
Procedure	29
Funzioni	32
Ottimizzazione analytics	35
Viste	35
Indici	36
Appendice	37
Oracle Database 18c (Express edition) e Oracle SQL developer	37
Popolamento della tabella master COVID19	40

## Introduzione

Il seguente progetto ha il fine di mostrare, attraverso l'utilizzo di una base dati relazionale, l'andamento epidemico del virus Covid-19 in Italia. L'informazione e la sua fruizione rappresentano la colonna semantica portante dell'intera tesina: innanzitutto, raccogliere i dati relativi alla realtà di interesse, successivamente assegnarne un significato ben specifico e infine manipolarla per analizzare statistiche e dati analitici, come risultato di un percorso strutturato e spiegato dettagliatamente.

La tesina, realizzata in occasione della prova finale del corso di Basi di Dati, tenuto dal professore Vincenzo Moscato presso l'Università Federico II di Napoli, si pone anche l'obiettivo di spiegare il processo di progettazione e realizzazione di una base dati relazionale e dunque far comprendere al lettore l'importanza di tale strumento di raccolta dati e di gestione dell'informazione, sperando che egli possa altresì interessarsi ed approfondire il tema trattato.

Per ospitare i dati relativi al contagio, forniti dalla Protezione Civile, è stata creata una base dati relazionale. Nell'elaborato vengono illustrati i processi di normalizzazione dello schema concettuale, del suo arricchimento e di reverse engineering per poi realizzare una serie di Query SQL utili all'analisi del fenomeno studiato. Infine, viene introdotta la componente attiva della base dati, costruita attraverso una serie di procedure/trigger scritti in linguaggio PL-SQL, e la creazione di viste e indici sui dati per ottimizzare l'accesso alle informazioni.

## Normalizzazione

### Verifica della prima forma normale (1NF)

*I campi sono tutti atomici, dunque la 1FN è automaticamente verificata.*

### Chiave e Relazioni funzionali

1) Chiave = {data, codice\_provincia}

2) Relazioni funzionali:

A) {data, codice\_provincia} -> stato, codice\_regione, denominazione\_regione, denominazione\_provincia, sigla\_provincia, latitudine, longitudine, totale\_casi, note\_it, note\_en

B) codice\_regione -> denominazione\_regione, Stato

C) codice\_provincia -> denominazione\_provincia, sigla\_provincia, latitudine, longitudine, codice\_regione, denominazione\_regione, Stato

### Verifica della seconda forma normale (2NF)

*Esistono delle dipendenze parziali dalla chiave, quindi la 2NF non è verificata.*

A) **PROVINCE** (codice\_provincia, denominazione\_provincia, sigla\_provincia, latitudine, longitudine, codice\_regione, denominazione\_regione, Stato)

B) **COVID19\_PROVINCE** (data, codice\_provincia:PROVINCE, totale\_casi, note\_it, note\_en)

### Verifica della terza forma normale (3NF)

*Sono presenti delle dipendenze transitive, quindi anche la 3NF non è verificata.*

1) **COVID19\_PROVINCE** (data, codice\_provincia:PROVINCE, totale\_casi, note\_it, note\_en)

2) **REGIONI** (codice\_regione, denominazione\_regione, Stato)

3) **PROVINCE** (codice\_provincia, denominazione\_provincia, sigla\_provincia, latitudine, longitudine, codice\_regione:REGIONI)

## Creazione delle base dati

Attraverso la normalizzazione della tabella Master, si giunge quindi a tre tabelle, che verificano correttamente la 3NF e che sono state create attraverso le seguenti istruzioni DDL (*Data Definition Language*):

```
1) CREATE TABLE REGIONI
(
  codice_regione NUMBER(2),
  denominazione_regione VARCHAR2(200) NOT NULL,
  stato CHAR(3) DEFAULT 'ITA',
  CONSTRAINT PK_REGIONI primary key(codice_regione)
);

2) CREATE TABLE PROVINCE
(
  codice_provincia NUMBER(3),
  denominazione_provincia VARCHAR2(200) NOT NULL,
  sigla_provincia CHAR(2) NOT NULL UNIQUE,
  latitudine NUMBER NOT NULL,
  longitudine NUMBER NOT NULL,
  codice_regione NUMBER(2),
  CONSTRAINT PK_PROVINCE primary key(codice_provincia),
  CONSTRAINT FK_PROVINCE foreign key(codice_regione) REFERENCES
  REGIONI(codice_regione)
);

3) CREATE TABLE COVID19_PROVINCE
(
  data DATE,
  codice_provincia NUMBER(3),
  totale_casi INTEGER NOT NULL,
  note_it CLOB,
  note_en CLOB,
  CONSTRAINT PK_COVID_P primary key(data, codice_provincia),
  CONSTRAINT FK_COVID_P foreign key(codice_provincia) REFERENCES
  PROVINCE(codice_provincia)
);
```

## Descrizione dei comandi DDL

Lo statement #1 crea la tabella **REGIONI**, la quale è costituita da 3 campi (*codice\_regione*, *denominazione\_regione* e *stato*) che descrivono rispettivamente il codice numerico associato ad una specifica regione (istanza dell'entità **REGIONI**, individuata univocamente da un codice a due cifre il quale ha ruolo di chiave primaria), il nome di essa (con lunghezza massima pari a 200 caratteri ASCII, e vincolo di integrità intra-relazionale NOT NULL che impedisce al record di assumere valore NULL in corrispondenza di tale campo) e la sigla dello stato di appartenenza. La base dati che stiamo costruendo contiene i dati relativi al solo stato italiano e dunque alle regioni che sicuramente vi appartengono. 'ITA' è un valore che è sicuramente assunto da qualsiasi record sul campo *stato*, ergo ha senso impostarlo come default value.

Lo statement #2 crea la tabella **PROVINCE**, costituita da 6 campi (*codice\_provincia*, *denominazione\_provincia*, *sigla\_provincia*, *latitudine*, *longitudine*, *codice\_regione*) che descrivono rispettivamente il codice numerico associato ad una specifica provincia (istanza di **PROVINCE**, individuata univocamente da un codice numerico a 3 cifre che ha ruolo di chiave primaria), il nome di essa (con lunghezza massima pari a 200 caratteri ASCII e vincolo di integrità NOT NULL che impedisce al record di assumere valore NULL in corrispondenza di tale campo), una sigla composta da due caratteri ASCII (univoca per ogni provincia, deve rispettare il vincolo di dominio UNIQUE), le coordinate geografiche che la localizzano e il codice di 2 cifre rappresentante la regione nella quale è compresa. Il campo *codice\_regione* presenta un vincolo di integrità referenziale per la primary key della relazione **REGIONI**.

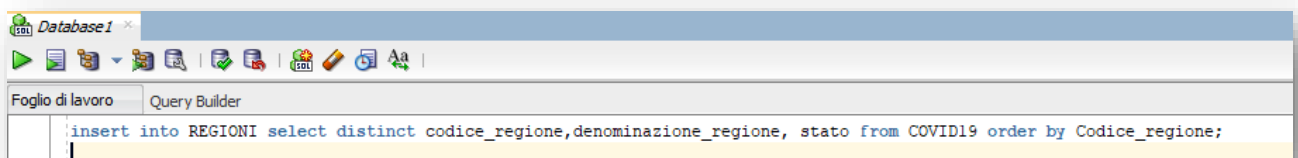
Lo statement #3 crea la tabella **COVID19\_PROVINCE**, costituita da 5 campi (*data*, *codice\_provincia*, *totale\_casi*, *note\_it*, *note\_en*) che descrivono rispettivamente la data nella quale sono stati prelevati i dati di contagio relativi ad una determinata provincia, il codice numerico a 3 cifre che individua la provincia stessa, il numero intero totale di coloro che sono risultati positivi al tampone per il COVID19 e infine lo spazio per delle note aggiuntive sia in lingua italiana che in lingua inglese (il tipo del campo è il CLOB, le cui istanze sono in grado di memorizzare fino a 4GB di testo).

## Popolamento delle tabelle normalizzate

Il popolamento delle tabelle alle quali si è giunti è stato effettuato (in via preliminare) con dei comandi DML di *INSERT* che ricevono i valori di inserimento attraverso delle proiezioni verticali sui campi di interesse della tabella Master. Ricordiamo che grazie all'utilizzo di trigger, ai quali è riservato un paragrafo, è possibile automatizzare il processo all'inserimento dei dati nella tabella Master (che con analoghi automatismi è ripulita da record non necessari o con valori non corretti). Per tale motivo, all'interno della base dati è stata conservata la tabella master, la quale può essere utilizzata da appoggio per l'inserimento di nuovi dati collezionati nel formato stabilito dalla Protezione Civile, i quali si propagheranno tramite trigger anche alla tabella **COVID19\_PROVINCE** (secondo il nostro schema le tabelle **REGIONI** e **PROVINCE** non hanno bisogno di essere aggiornate all'inserimento di nuovi dati, ma dovranno esserlo nel caso di *UPDATE* di campi di interesse nella tabella Master)

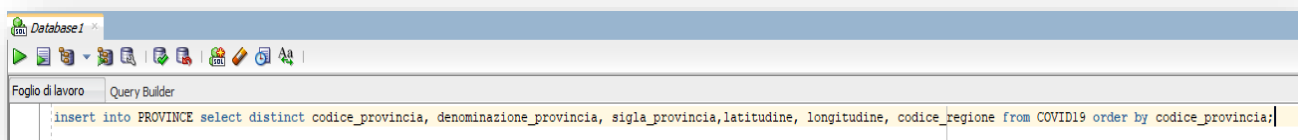
Di seguito si mostrano i costrutti che sono stati utilizzati:

### 1) Popolamento tabella REGIONI



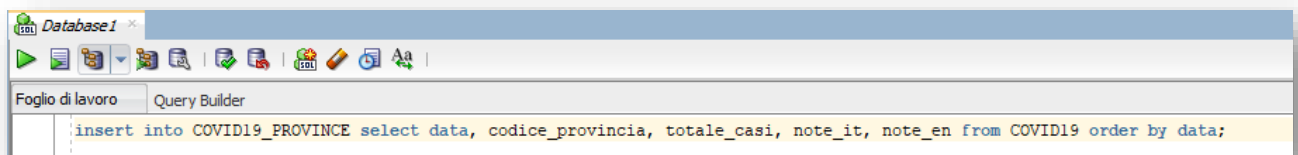
```
insert into REGIONI select distinct codice_regione, denominazione_regione, stato from COVID19 order by Codice_regione;
```

### 2) Popolamento tabella PROVINCE



```
insert into PROVINCE select distinct codice_provincia, denominazione_provincia, sigla_provincia, latitudine, longitudine, codice_regione from COVID19 order by codice_provincia;
```

### 3) Popolamento tabella COVID19\_PROVINCE



```
insert into COVID19_PROVINCE select data, codice_provincia, totale_casi, note_it, note_en from COVID19 order by data;
```

## Arricchimento dello schema

Per realizzare analytics di qualità migliore e di conseguenza ottenere una più utile analisi del fenomeno d'interesse, abbiamo deciso di estendere lo schema della base dati mediante i seguenti comandi DDL ALTER TABLE:

```
ALTER TABLE REGIONI ADD numero_rsa INTEGER;
ALTER TABLE REGIONI ADD pil NUMBER;
ALTER TABLE REGIONI ADD pil_procapite NUMBER;
ALTER TABLE REGIONI ADD eta_media NUMBER CHECK(eta_media>0);
ALTER TABLE REGIONI ADD numero_aeroporti INTEGER;
ALTER TABLE REGIONI ADD numero_anziani INTEGER;
ALTER TABLE REGIONI ADD numero_sportelli INTEGER;
ALTER TABLE REGIONI ADD numero_scuole INTEGER;
```

- Il numero di RSA (Residenza Sanitaria Assistenziale) per regione è stato prelevato dal sito dell'ISTAT ([http://dati.istat.it/Index.aspx?DataSetCode=DCIS\\_POSTILETTOPRESIDI#](http://dati.istat.it/Index.aspx?DataSetCode=DCIS_POSTILETTOPRESIDI#)).
- I dati relativi al PIL, PIL procapite e al numero di aeroporti derivano dalle pagine Wikipedia associate alle singole regioni italiane.
- I dati sull'età media per ogni regione sono stati dedotti dal sito AdminStatITALIA (<https://ugeo.urbistat.com/AdminStat/it/it/classifiche/eta-media/regioni/italia/380/1>).
- I dati sul numero di anziani (ovvero le persone con un'età maggiore o pari a 60 anni) per regione sono stati prelevati dal sito tuttitalia.it (<https://www.tuttitalia.it/statistiche/popolazione-eta-sesso-stato-civile-2019/>).
- Numeri\_sportelli rappresenta il numero totale di sportelli bancari presenti nella regione. I dati per questo campo sono disponibili sul sito tuttitalia (<https://www.tuttitalia.it/banche/>), così come quelli relativi al numero di scuole per ogni regione (<https://www.tuttitalia.it/scuole/>).

```
ALTER TABLE PROVINCE ADD superficie NUMBER CHECK(superficie>0);
ALTER TABLE PROVINCE ADD altitudine INTEGER;
ALTER TABLE PROVINCE ADD numero_abitanti INTEGER;
ALTER TABLE PROVINCE ADD densita NUMBER;
ALTER TABLE PROVINCE ADD numero_ospedali INTEGER;
```

- I dati relativi alla superficie, all'altitudine, al numero di abitanti, al numero di ospedali e alla densità abitativa sono stati ricavati dalle pagine Wikipedia associate alle singole province italiane.

Inoltre, sono state create delle nuove tabelle:

- **METEO** (*data*, *codice\_regione*:**REGIONI**, *temperatura\_media*, *umidita*)
- **DISTRIBUZIONE** (*data*, *codice\_regione*:**REGIONI**, *mascherine*, *guanti*)
- **COVID19\_REGIONI** (*data*, *codice\_regione*:**REGIONI**, *ricoverati\_con\_sintomi*, *terapia\_intensiva*, *totale\_ospedalizzati*, *isolamento\_domiciliare*, *totale\_positivi*, *nuovi\_positivi*, *guariti*, *deceduti*, *tamponi*)

Che si aggiungono al restante schema della base dati:

- **PROVINCE** (*codice\_provincia*, *denominazione\_provincia*, *sigla\_provincia*, *latitudine*, *longitudine*, *codice\_regione*:**REGIONI**)
- **REGIONI** (*codice\_regione*, *denominazione\_regione*, *stato*)
- **COVID19\_PROVINCE** (*data*, *codice\_provincia*:**PROVINCIA**, *totale\_casi*, *note\_it*, *note\_en*)

Di seguito, si mostrano le istruzioni DDL che creano le tabelle sopracitate:

```
1) CREATE TABLE METEO
(
    codice_regione NUMBER(2),
    data DATE,
    temperatura_media NUMBER NOT NULL,
    umidita NUMBER NOT NULL,
    CONSTRAINT PK_METEO primary key(data, codice_regione),
    CONSTRAINT FK_METEO foreign key(codice_regione) REFERENCES
    REGIONI(codice_regione)
);

2) CREATE TABLE DISTRIBUZIONE
(
    data DATE,
    codice_regione NUMBER(2),
    mascherine INTEGER,
    guanti INTEGER,
    CONSTRAINT PK_DISTRIBUZIONE primary key(data, codice_regione),
    CONSTRAINT FK_DISTRIBUZIONE foreign key(codice_regione) REFERENCES
    REGIONI(codice_regione)
);

3) CREATE TABLE COVID19_REGIONI
(
    data DATE,
    codice_regione NUMBER(2),
    ricoverati_con_sintomi INTEGER,
    terapia_intensiva INTEGER,
    totale_ospedalizzati INTEGER,
    isolamento_domiciliare INTEGER,
    totale_positivi INTEGER,
    nuovi_positivi INTEGER,
    dimessi_guariti INTEGER,
    deceduti INTEGER,
    tamponi INTEGER,
    CONSTRAINT PK_COVID_R primary key(data, codice_regione),
    CONSTRAINT FK_COVID_R foreign key(codice_regione) REFERENCES
    REGIONI(codice_regione)
);
```

- La tabella **METEO** è stata popolata grazie all'archivio meteorologico del sito [ilmeteo.it](https://www.ilmeteo.it/portale/archivio-meteo) (<https://www.ilmeteo.it/portale/archivio-meteo>).
- La tabella **DISTRIBUZIONE** è stata costruita attraverso l'A.D.A. (Analisi Distribuzione Aiuti) del Commissario Straordinario per l'attuazione e il coordinamento delle misure di contenimento e contrasto dell'emergenza epidemiologica COVID-19. (<https://app.powerbi.com/view?r=eyJrIjoieNTE2NWw3ZjktZGFhNi00MzYxLWJlMzEtYThmOWEzYjA1MGNhliwidCI6ImFmZDBhNzVjLTg2NzEtNGNjZS05MDYxLTJjYTlkOTJlNDIyZiIsImMiOiJh9>).
- Infine, il popolamento della tabella **COVID19\_REGIONI** proviene dall'insieme dei file in formato CSV scaricabili dal GitHub della Presidenza del Consiglio dei Ministri – Dipartimento della Protezione Civile riguardo il COVID-19 (<https://github.com/pcm-dpc/COVID-19/tree/master/dati-regioni>).



Per l'inserimento dei dati si è utilizzato il comando DML UPDATE, secondo la sintassi "UPDATE PROVINCE SET superficie=valore, altitudine=valore, ....numero\_ospedali=valore where codice\_provincia=value"

```

UPDATE PROVINCE SET superficie=1248,altitudine=23, numero_abitanti=430780, densita=345,18,numero_ospedali=2 where codice_provincia=92
UPDATE PROVINCE SET superficie=283,6,altitudine=24, numero_abitanti=114046, densita=402,14,numero_ospedali=0 where codice_provincia=93
UPDATE PROVINCE SET superficie=1535,24,altitudine=423, numero_abitanti=83490, densita=54,38,numero_ospedali=0 where codice_provincia=94
UPDATE PROVINCE SET superficie=2990,45,altitudine=9, numero_abitanti=156904, densita=52,47,numero_ospedali=0 where codice_provincia=95
UPDATE PROVINCE SET superficie=913,28,altitudine=420, numero_abitanti=174718, densita=191,31,numero_ospedali=0 where codice_provincia=96
UPDATE PROVINCE SET superficie=805,61,altitudine=214, numero_abitanti=337380, densita=418,79,numero_ospedali=1 where codice_provincia=97
UPDATE PROVINCE SET superficie=782,99,altitudine=87, numero_abitanti=230198, densita=294,numero_ospedali=3 where codice_provincia=98
UPDATE PROVINCE SET superficie=864,88,altitudine=5, numero_abitanti=339437, densita=392,47,numero_ospedali=1 where codice_provincia=99
UPDATE PROVINCE SET superficie=365,72,altitudine=61, numero_abitanti=259181, densita=708,69,numero_ospedali=0 where codice_provincia=100
UPDATE PROVINCE SET superficie=1735,68,altitudine=8, numero_abitanti=174641, densita=100,62,numero_ospedali=1 where codice_provincia=101
UPDATE PROVINCE SET superficie=1150,64,altitudine=476, numero_abitanti=160073, densita=139,12,numero_ospedali=0 where codice_provincia=102
UPDATE PROVINCE SET superficie=2260,91,altitudine=197, numero_abitanti=157782, densita=69,79,numero_ospedali=1 where codice_provincia=103
UPDATE PROVINCE SET superficie=405,49,altitudine=162, numero_abitanti=875769, densita=2159,78,numero_ospedali=2 where codice_provincia=108

```

Esempi di Update della tabella REGIONI:

```

UPDATE REGIONI SET Numero_sportelli=1927 ,numero_RSA=5010 ,PIL= 132671000000 ,PIL_procapite=30300 where codice_regione=1 ;
UPDATE REGIONI SET Numero_sportelli=76 ,numero_RSA=255 ,PIL=4453000000 ,PIL_procapite=35200 where codice_regione=2 ;
UPDATE REGIONI SET Numero_sportelli=4690 ,numero_RSA=50124 ,PIL= 380955000000 ,PIL_procapite=38500 where codice_regione=3 ;
UPDATE REGIONI SET Numero_sportelli=2393 ,numero_RSA=5006 ,PIL= 162224000000 ,PIL_procapite=33100 where codice_regione=5 ;
UPDATE REGIONI SET Numero_sportelli=660 ,numero_RSA=767 ,PIL=37642000000 ,PIL_procapite=30900 where codice_regione=6 ;
UPDATE REGIONI SET Numero_sportelli=596 ,numero_RSA=2437 ,PIL=49315000000 ,PIL_procapite=31600 where codice_regione=7 ;
UPDATE REGIONI SET Numero_sportelli=2402 ,numero_RSA=2272 ,PIL=15717700000 ,PIL_procapite=35300 where codice_regione=8 ;
UPDATE REGIONI SET Numero_sportelli=1702 ,numero_RSA=679 ,PIL= 113798000000 ,PIL_procapite=30400 where codice_regione=9 ;
UPDATE REGIONI SET Numero_sportelli=392 ,numero_RSA=1185 ,PIL=21697000000 ,PIL_procapite=24500 where codice_regione=10 ;
UPDATE REGIONI SET Numero_sportelli=788 ,numero_RSA=4533 ,PIL=41183000000 ,PIL_procapite=26800 where codice_regione=11 ;
UPDATE REGIONI SET Numero_sportelli=1975 ,numero_RSA=4533 ,PIL= 193101000000 ,PIL_procapite=32700 where codice_regione=12 ;
UPDATE REGIONI SET Numero_sportelli=514 ,numero_RSA=1280 ,PIL=32558000000 ,PIL_procapite=24700 where codice_regione=13 ;
UPDATE REGIONI SET Numero_sportelli=97 ,numero_RSA=15 ,PIL=6121000000 ,PIL_procapite=19800 where codice_regione=14 ;

```

```

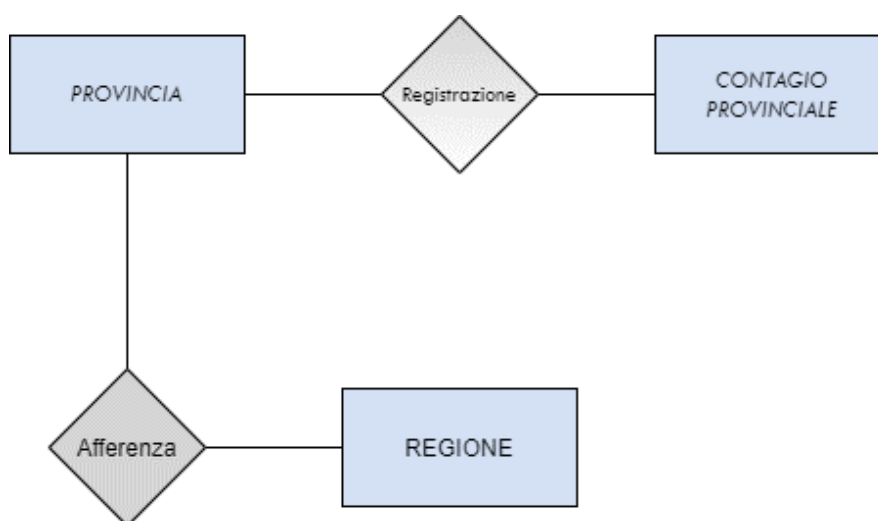
UPDATE REGIONI SET numero_aeroporti= 10 ,eta_media= 46.54 ,numero_anziani= 1401516 where codice_regione= 1 ;
UPDATE REGIONI SET numero_aeroporti= 1 ,eta_media= 45.63 ,numero_anziani= 38173 where codice_regione= 2 ;
UPDATE REGIONI SET numero_aeroporti= 16 ,eta_media= 44.74 ,numero_anziani= 2887766 where codice_regione= 3 ;
UPDATE REGIONI SET numero_aeroporti= 6 ,eta_media= 45.10 ,numero_anziani= 1434419 where codice_regione= 5 ;
UPDATE REGIONI SET numero_aeroporti= 6 ,eta_media= 47 ,numero_anziani= 397050 where codice_regione= 6 ;
UPDATE REGIONI SET numero_aeroporti= 3 ,eta_media= 48.46 ,numero_anziani= 546941 where codice_regione= 7 ;
UPDATE REGIONI SET numero_aeroporti= 14 ,eta_media= 45.70 ,numero_anziani= 1349509 where codice_regione= 8 ;
UPDATE REGIONI SET numero_aeroporti= 11 ,eta_media= 46.52 ,numero_anziani= 1187619 where codice_regione= 9 ;
UPDATE REGIONI SET numero_aeroporti= 2 ,eta_media= 46.49 ,numero_anziani= 282619 where codice_regione= 10 ;
UPDATE REGIONI SET numero_aeroporti= 3 ,eta_media= 46.09 ,numero_anziani= 477867 where codice_regione= 11 ;
UPDATE REGIONI SET numero_aeroporti= 6 ,eta_media= 44.58 ,numero_anziani= 1646365 where codice_regione= 12 ;
UPDATE REGIONI SET numero_aeroporti= 2 ,eta_media= 45.67 ,numero_anziani= 400176 where codice_regione= 13 ;

```

## Reverse engineering

### Modello Portante

Per ricavare il modello ER, è possibile utilizzare ad esempio un approccio di tipo *Top-down* secondo il quale è necessario inizialmente trovare lo schema concettuale portante della base dati (formato da entità ed associazioni) per poi procedere a raffinarlo. Il modello portante contenente le entità fondamentali del nostro processo di reverse engineering è il seguente:

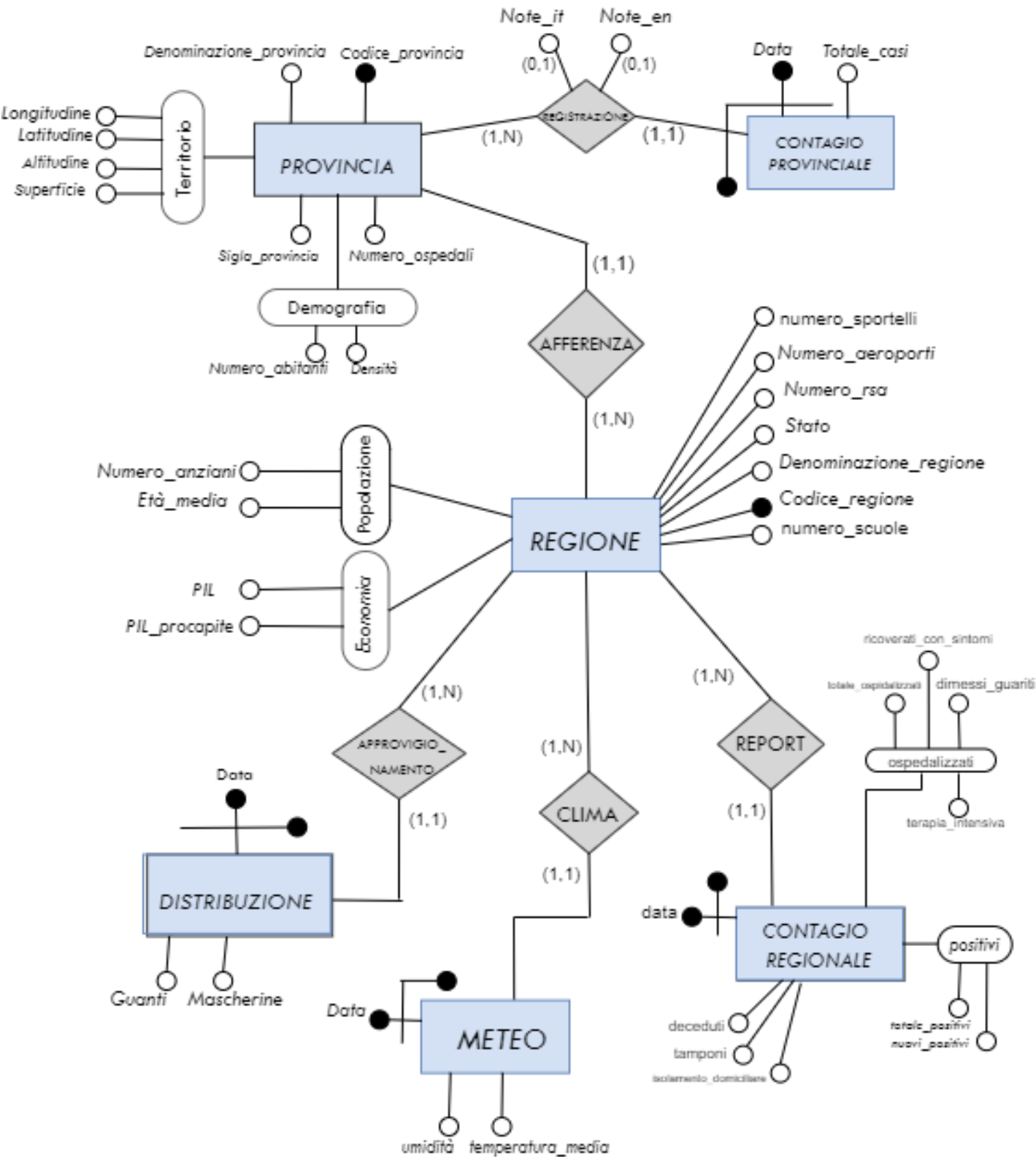


### Modello Entity Relationship

A partire dallo schema portante, è stato poi determinato il modello ER estendendolo con opportune nuove entità, relazioni (con rispettive cardinalità) e attributi. Per ogni provincia e per ogni data (tra il 25 febbraio 2020 e il 24 giugno) è stato registrato il numero complessivo dei casi di COVID19, dunque concettualmente si manifestano due entità, PROVINCIA e CONTAGIO PROVINCIALE, legate tra di esse dalla relazione “registrazione”. Una provincia appartiene ad una ed una sola regione (oppure è capoluogo della omonima provincia autonoma, considerata comunque concettualmente come regione, come accade per Bolzano e Trento). Per ogni regione e per ogni data è riportata una rassegna sul contagio relativo alla regione (il quale tiene conto di importanti dati sugli ospedalizzati, sui positivi, sui deceduti, sui tamponi effettuati, sui pazienti in isolamento domiciliare, sui ricoverati con sintomi e sui guariti), sui parametri climatici e sulla distribuzione di guanti e mascherine. È evidente che per ogni regione ci possono essere più rassegne (tante quanti sono i giorni di misurazione) mentre una misurazione può essere relativa ad una sola regione: si tratta di relazioni 1 a molti. Infine, sono stati inseriti gli attributi (atomici e composti) ed opportuni identificatori esterni per le entità Distribuzione, Meteo, Contagio regionale e Contagio provinciale.

Per giungere allo schema della base dati precedentemente descritto, è sufficiente eseguire prima una trasformazione dello schema (semplificando tutti gli attributi composti), applicare la regola n°1 di traduzione delle entità e la regola n°4 di traduzione delle associazioni uno a molti.

Il modello E/R risultante è il seguente:



## Analytics

In questa sezione, sono riportate ed eventualmente graficate delle query SQL sulla base dati. Tali istruzioni rappresentano un utile metodo di analisi del fenomeno analizzato.

### SUPERFICIE, LATITUDINE E LONGITUDINE DI OGNI PROVINCIA ITALIANA

```
select distinct denominazione_provincia as PROVINCIA, superficie,
latitudine, longitudine
from province;
```

	PROVINCIA	SUPERFICIE	LATITUDINE	LONGITUDINE
1	Aosta	3260,9	45,73750286	7,320149366
2	Imperia	1154,78	43,88570648	8,027850298
3	Savona	1546,29	44,30750461	8,481108654
4	Como	1279,04	45,8099912	9,085159546
5	Gorizia	467,14	45,94149817	13,62212502
6	Massa Carrara	1154,68	44,03674425	10,14173829
7	Viterbo	3615,24	42,4173828	12,10473416
8	Rieti	2750,52	42,40488444	12,86205939
9	Avellino	2806,07	40,91404699	14,79528803
10	Teramo	1954,38	42,6589177	13,70439971
11	Chieti	2599,58	42,35103167	14,16754574
12	Lecce	2799,07	40,35354285	18,1718973

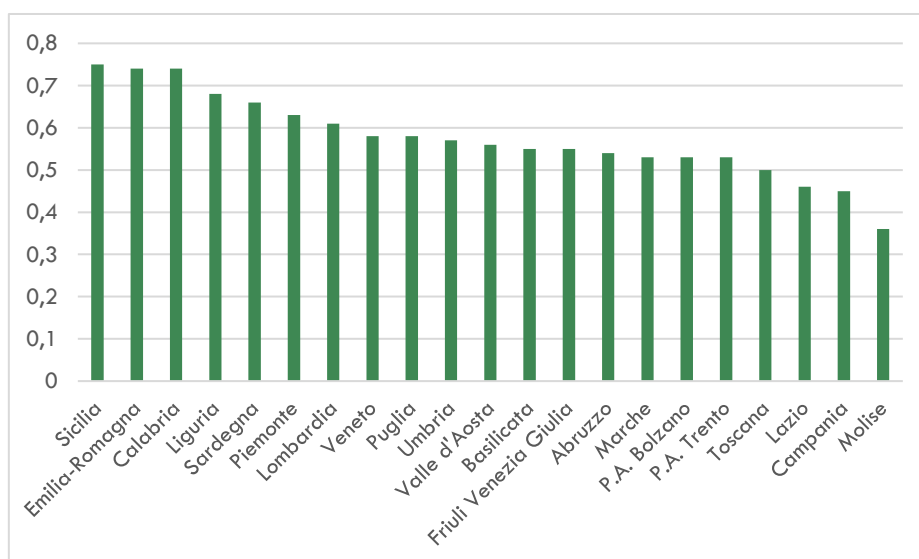
### VISUALIZZAZIONE DI TUTTE LE CARATTERISTICHE DELLA CAMPANIA

```
select * from regioni where denominazione_regioni like '_amp%a';
```

	CODICE_REGIONE	DENOMINAZIONE_REGIONE	STATO	NUMERO_SCUOLE	NUMERO_SPORTELLI	PIL	PIL_PROCAPITE	NUMERO_RSA	NUMERO_AEROPORTI	ETA_MEDIA	NL
1	15	Campania	ITA	7404	1206	106431000000	18200	1152	6	42,15	

### UMIDITÀ PERCENTUALE DEL 24 GIUGNO 2020

```
select r.denominazione_regione, m.umidita/100 as UMIDITA_IN_DECIMALE from
meteo m join regioni r on r.codice_regione=m.codice_regione where
m.data='24-Giu-2020' order by m.umidita desc;
```



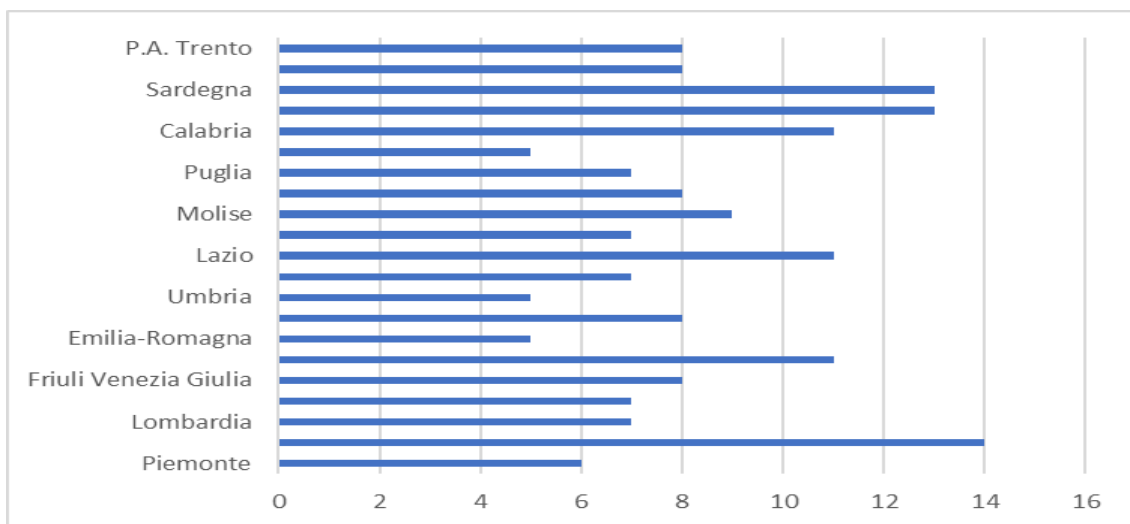
## VISUALIZZAZIONE DI TUTTE LE CARATTERISTICHE DELLE PROVINCE CON IL TOTALE DEI CASI PER DATA

```
select * from province p join covid19_province c on
p.codice_provincia=c.codice_provincia;
```

	CODICE_PROVINCIA	DENOMINAZIONE_PROVINCIA	SIGLA_PROVINCIA	LATITUDINE	LONGITUDINE	CODICE_REGIONE	NUMERO_OSPEDALI	SUPERFICIE	ALTITUDINE	NUMERO_ABITAN
1	37	Bologna	BO	44,49436681	11,3417208	8	5	3703	54	10171
2	24	Vicenza	VI	45,547497	11,54597109	5	1	2722,53	39	8625
3	65	Salerno	SA	40,67821961	14,7594026	15	1	4954,16	4	10937
4	63	Napoli	NA	40,83956555	14,25084984	15	11	1171	17	30704
5	61	Caserta	CE	41,07465878	14,33240464	15	0	2651,35	68	9204
6	62	Benevento	BN	41,12969987	14,78151683	15	2	2080,44	135	2744
7	64	Avellino	AV	40,91404699	14,79528803	15	2	2806,07	348	4146
8	102	Vibo Valentia	VV	38,67624147	16,10157414	18	0	1150,64	476	1600
9	80	Reggio di Calabria	RC	38,10922769	15,6434527	18	5	3183	31	5434
10	101	Crotone	KR	39,08036878	17,12538864	18	1	1735,68	8	1746

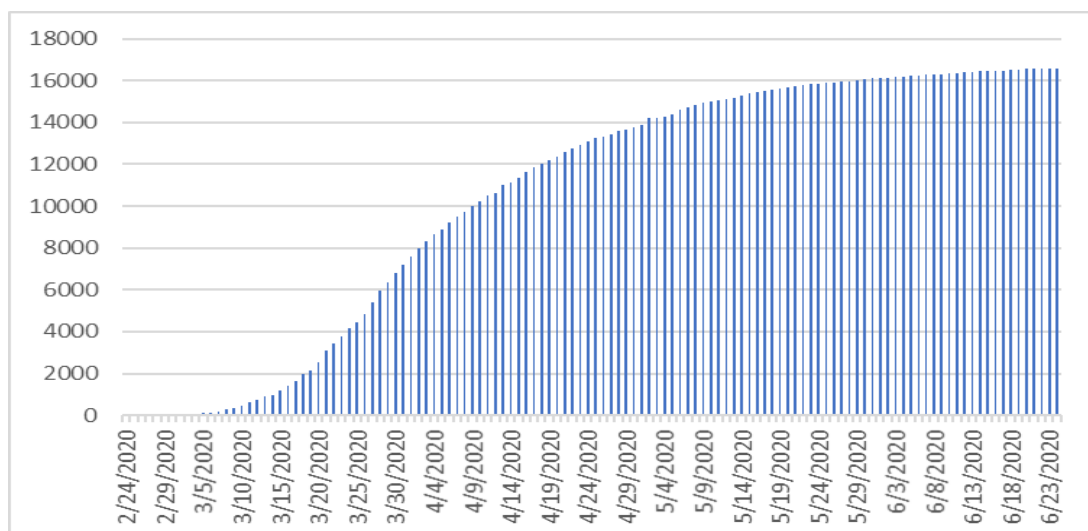
## TEMPERATURA MEDIA DI OGNI REGIONE IN DATA 01-04-2020

```
select temperatura_media, denominazione_regione from regioni r ,meteo m
where m.codice_regione= r.codice_regione and m.data='1-Apr-2020'
```



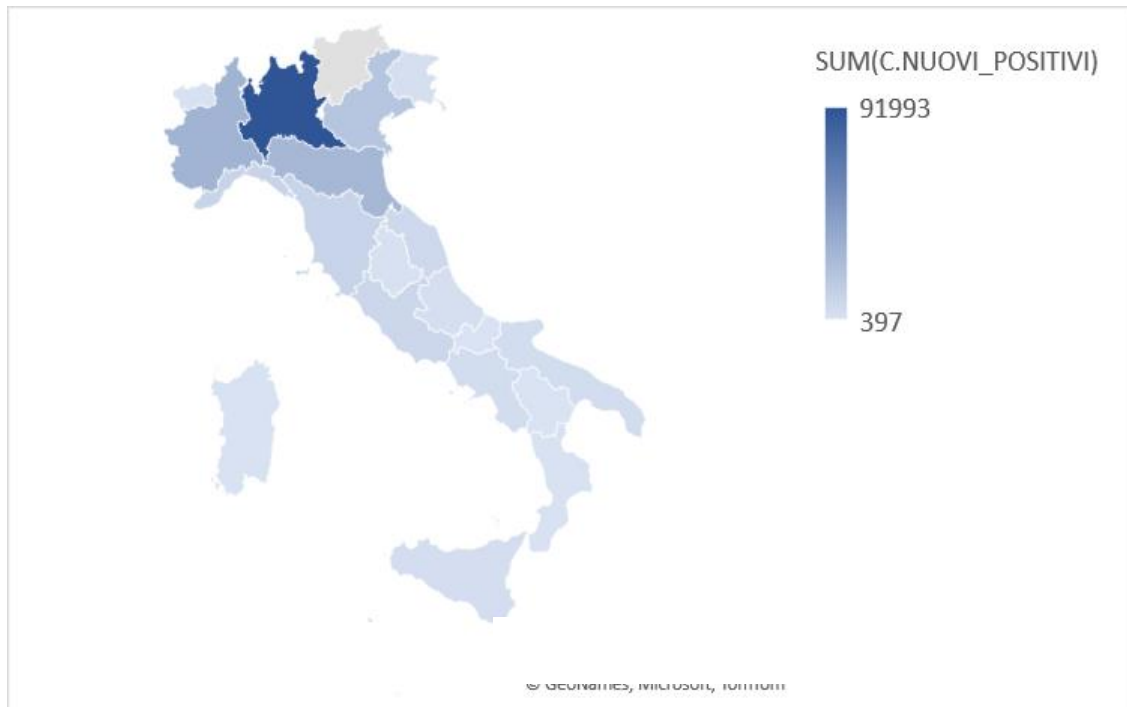
## ANDAMENTO DEL NUMERO DI DECEDUTI IN LOMBARDIA

```
select c.data, c.deceduti from covid19_regioni c join regioni r on
r.codice_regione=c.codice_regionewhere r.denominazione_regione like
'Lombardia' order by data;
```



### SOMMA DEI NUOVI POSITIVI PER REGIONE

```
select sum(c.nuovi_positivi), r.denominazione_regione from
covid19_regioni c join regioni r on r.codice_regione=c.codice_regione
group by denominazione_regione;
```



### MASSIMO E MINIMO DEL NUMERO DI DECEDUTI A LIVELLO REGIONALE

```
select max(deceduti), min(deceduti) from covid19_regioni;
```

	MAX(DECEDUTI)	MIN(DECEDUTI)
1	16586	0

### MEDIA DEI NUOVI POSITIVI GIORNALIERI

```
select avg(distinct nuovi_positivi) as MEDIA_NUOVI_POSITIVI from
covid19_regioni;
```

	MEDIA_NUOVI_POSITIVI
1	388,348235294117647058823529411764705882

### CONTEGGIO DELLE VOLTE IN CUI CI SONO STATI PIU' DI 300 GUARITI A LIVELLO REGIONALE

```
select count(*) as CONTEGGIO_VOLTE_CON_PIU_300_GUARITI from
covid19_regioni where dimessi_guariti>300;
```

	CONTEGGIO_VOLTE_CON_PIU_300_GUARITI
1	1552

### CONTEGGIO DELLE VOLTE IN CUI CI SONO STATI MENO DI 50 CASI DI ISOLAMENTO DOMICILIARE NELLE REGIONI DEL SUD

```
select count(CR.isolamento_domiciliare) as CONTEGGIO
from COVID19_REGIONI CR join REGIONI R on
CR.codice_regione=R.codice_regione
where CR.isolamento_domiciliare < 50 AND (R.denominazione_regione like
'Campania' or R.denominazione_regione like 'Puglia' or
R.denominazione_regione like 'Basilicata' or R.denominazione_regione like
'Calabria' or R.denominazione_regione like 'Sicilia');
```

CONTEGGIO
1 147

### REGIONI CON 0 RICOVERATI IN DATA 24 GIUGNO 2020

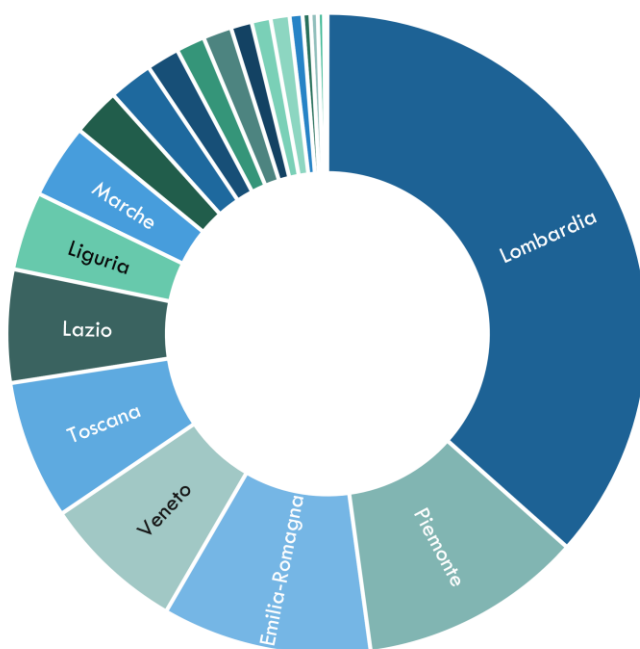
```
select distinct denominazione_regione
from regioni where codice_regione in (
select codice_regione
from covid19_regioni
where ricoverati_con_sintomi=0 and data='24-Giu-2020');
```

DENOMINAZIONE_REGIONE
1 P.A. Trento

### NUMERO DI CASI TOTALI DI TERAPIA INTENSIVA PER REGIONE

```
select distinct R.denominazione_regione, sum(CR.terapia_intensiva) as
TOTALI_TERAPIA_INTENSIVA
from COVID19_REGIONI CR join REGIONI R on
CR.codice_regione=R.codice_regione
group by R.denominazione_regione;
```

DENOMINAZIONE_REGIONE	TOTALI_TERAPIA_INTENSIVA
1 Sardegna	1175
2 Basilicata	558
3 Lombardia	66154
4 Sicilia	2997
5 Valle d'Aosta	713
6 Lazio	10293
7 Marche	6772
8 Liguria	7161
9 Piemonte	20364
10 Puglia	4031
11 P.A. Trento	2618
12 Abruzzo	2614
13 Emilia-Romagna	19070
14 Friuli Venezia Giulia	1727
15 Veneto	12938
16 P.A. Bolzano	1916
17 Campania	4386
18 Molise	308
19 Toscana	12679
20 Umbria	1732
21 Calabria	665





### REGIONI CON UN NUMERO DI SCUOLE MAGGIORE DI QUELLO DEL VENETO

```
select distinct denominazione_regione from regioni where numero_scuole >
any (select numero_scuole from regioni where denominazione_regione like
'Ven%');
```

	DENOMINAZIONE_REGIONE
1	Lombardia
2	Sicilia
3	Campania

### PROVINCIA ITALIANA CON PIU' CASI IN TOTALE

```
select p.denominazione_provincia from province p join covid19_province c
on c.codice_provincia=p.codice_provincia where c.totale_casi >= all
(select totale_casi from covid19_province);
```

	DENOMINAZIONE_PROVINCIA
1	Milano

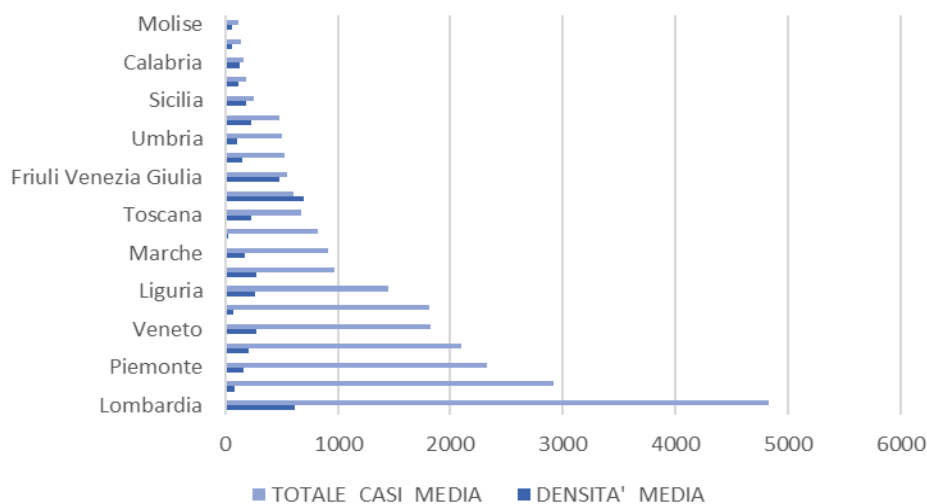
### REGIONE CON PIU' CASI DI TERAPIA INTENSIVA IN DATA 24 GIUGNO 2020

```
select r.denominazione_regione, c.terapia_intensiva from regioni r join
covid19_regioni c on c.codice_regione=r.codice_regione where c.data='24-
Giu-2020' and c.terapia_intensiva >= all(select terapia_intensiva from
covid19_regioni where data='24-Giu-2020');
```

	DENOMINAZIONE_REGIONE	TERAPIA_INTENSIVA
1	Lombardia	48

### CONFRONTO DENSITÀ MEDIA DI POPOLAZIONE PER REGIONE E CASI TOTALI PER REGIONE

```
select r.denominazione_regione, avg(p.densita) as DENSITA'_MEDIA,
avg(c.totale_casi) as TOTALE_CASI_MEDIA
from (province p join regioni r on p.codice_regione=r.codice_regione)
join covid19_province c on c.codice_provincia=p.codice_provincia
group by r.denominazione_regione
order by avg(c.totale_casi)
```





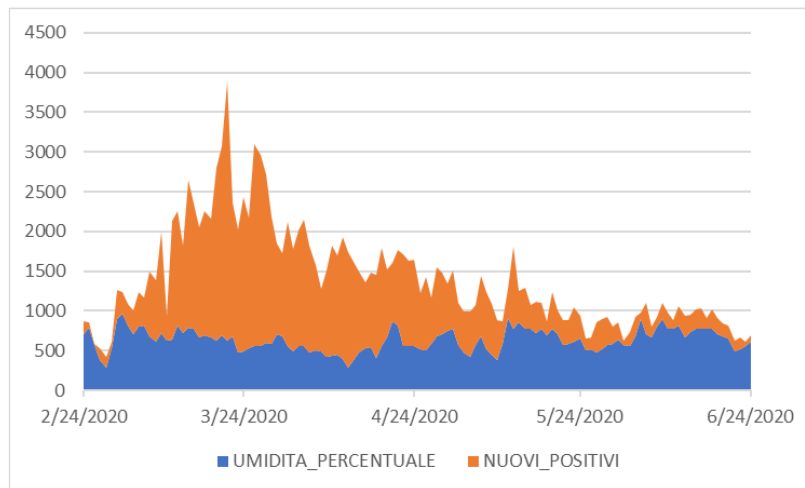
## REGIONI CON TEMPERATURA MEDIA MAX MAGGIORE DI 27 GRADI

```
select distinct R.denominazione_regione, MAX(M.temperatura_media) as
TEMP_MAX
from REGIONI R join METEO M on R.codice_regione=M.codice_regione
group by R.denominazione_regione, M.codice_regione
HAVING MAX(M.temperatura_media)> 27;
```

	DENOMINAZIONE_REGIONE	TEMP_MAX
1	Sicilia	32
2	Emilia-Romagna	28

## CONFRONTO ANDAMENTO UMIDITÀ CON IL NUMERO DI NUOVI POSITIVI IN LOMBARDIA

```
select m.data, m.umidita as UMIDITA_PERCENTUALE, c.nuovi_positivi from
(meteo m join covid19_regioni c on c.codice_regione=m.codice_regione and
c.data=m.data) join regioni r on r.codice_regione=c.codice_regione where
denominazione_regione='Lombardia' order by data;1
```



	DATA	UMIDITA_PERCENTUALE	NUOVI_POSITIVI
1	24-FEB-20	70	166
2	25-FEB-20	79	68
3	26-FEB-20	57	18
4	27-FEB-20	38	145
5	28-FEB-20	29	128
6	29-FEB-20	53	84
7	01-MAR-20	89	369
8	02-MAR-20	97	270
9	03-MAR-20	82	266
10	04-MAR-20	71	300
11	05-MAR-20	80	421

<sup>1</sup> Nel grafico sottostante la query, i valori riguardanti l'umidità sono stati moltiplicati per 10 in modo tale da osservare più facilmente le variazioni del dato tra 2 giorni successivi

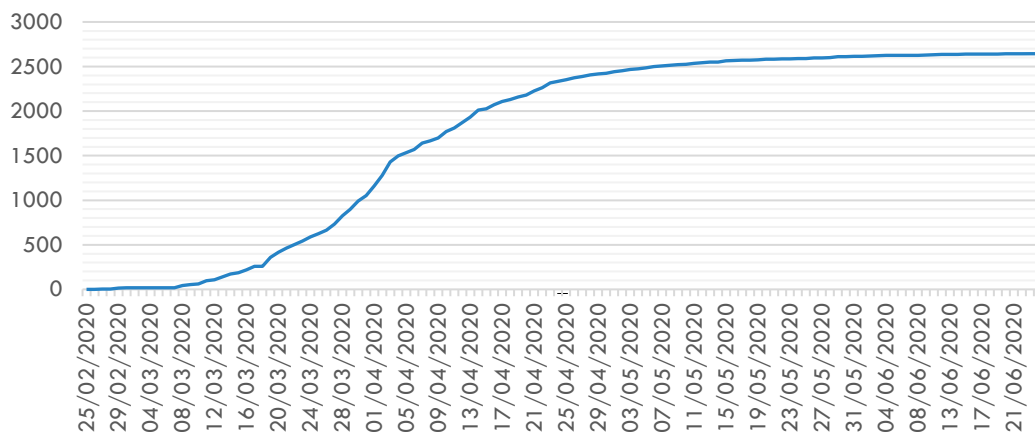
### REGIONE CON TEMPERATURA MEDIA PIÙ BASSA

```
select distinct R.denominazione_regione, MIN(M.temperatura_media) as
TEMP_MIN from REGIONI R join METEO M on R.codice_regione=M.codice_regione
group by R.denominazione_regione, M.codice_regione
having MIN(M.temperatura_media) <= ALL
(select MIN(M1.temperatura_media) as TEMP_MIN1
from REGIONI R1 join METEO M1 on R1.codice_regione=M1.codice_regione
group by R1.denominazione_regione, M1.codice_regione)
```

DENOMINAZIONE_REGIONE	TEMP_MIN
1 Emilia-Romagna	3

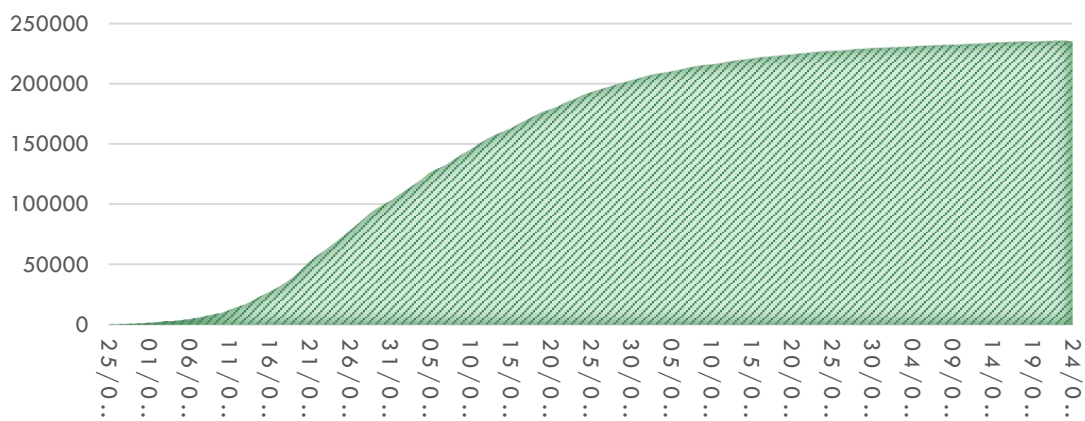
### ANDAMENTO DEI CONTAGI NELLA CITTÀ DI NAPOLI TRA IL 25 FEBBRAIO 2020 E IL 24 GIUGNO 2020

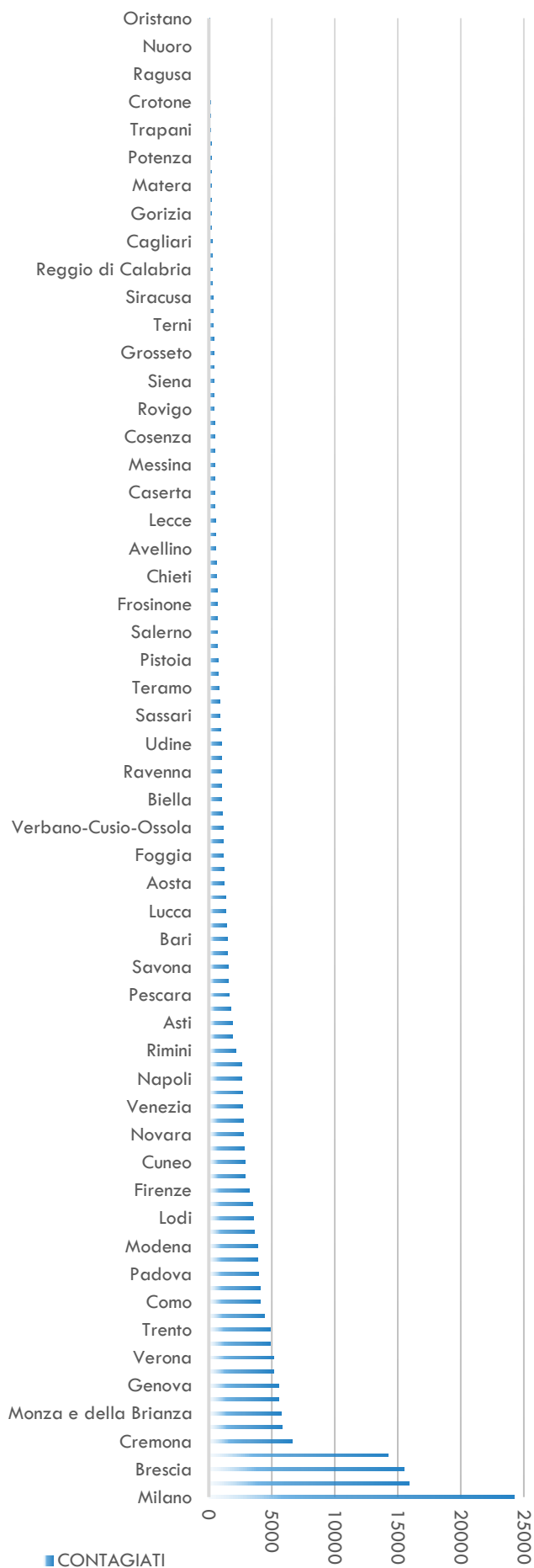
```
select CP.data, CP.totale_casi
from PROVINCE P join COVID19_PROVINCE CP on
CP.codice_provincia=P.codice_provincia
WHERE denominazione_provincia LIKE 'Napoli' AND data >= '25-Feb-2020' AND
data <= '25-Giu-2020'
order by data;
```



### ANDAMENTO DEI CONTAGI IN ITALIA TRA IL 25 FEBBRAIO E IL 24 GIUGNO

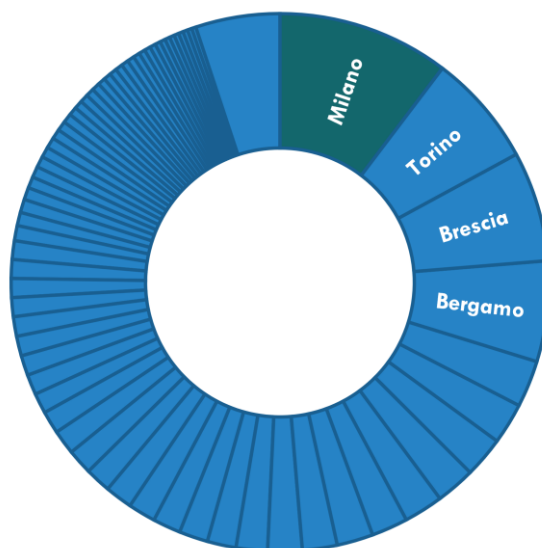
```
select CP.data, SUM(CP.totale_casi) as CASI_ITALIA
from COVID19_PROVINCE CP join PROVINCE P
on CP.codice_provincia=P.codice_provincia
group by data order by data;
```





### TOTALE CONTAGI 24 GIUGNO 2020 PER PROVINCIA

```
select P.denominazione_provincia as
PROVINCIA, CP.totale_casi as
CONTAGIATI
from COVID19_PROVINCE CP
join PROVINCE P on
CP.codice_provincia=P.codice_provincia
where data='24-Giu-2020'
order by CP.totale_casi;
```



PROVINCIA	CONTAGIATI
1 Oristano	61
2 Isernia	62
3 Nuoro	78
4 Vibo Valentia	84
5 Ragusa	87
6 Sud Sardegna	99
7 Crotone	118
8 Agrigento	135
9 Trapani	136
10 Caltanissetta	186
11 Potenza	189
12 Benevento	209
13 Matera	210
14 Catanzaro	214
15 Gorizia	216
16 L'Aquila	224
17 Cagliari	250
18 Taranto	280
19 Reggio di Calabria	289
20 Ascoli Piceno	290

#### PROVINCIA CON MAGGIOR NUMERO DI OSPEDALI

```
select P.denominazione_provincia, P.numero_ospedali
from PROVINCE P
where P.numero_ospedali >= ALL
(select P1.numero_ospedali
from PROVINCE P1 )
```

DENOMINAZIONE_PROVINCIA	NUMERO_OSPEDALI
1 Milano	32

#### REGIONE CON IL NUMERO MAGGIORE DI SPORTELLI BANCARI

```
select R.denominazione_regione, R.numero_sportelli
from REGIONI R
where R.numero_sportelli >= ALL
(select R1.numero_sportelli
from REGIONI R1)
```

DENOMINAZIONE_RE...	NUMERO_SPORTELLI
1 Lombardia	4690

#### REGIONE CON IL MAGGIOR NUMERO DI RSA

```
select R.denominazione_regione, R.numero_RSA
from REGIONI R
where R.numero_RSA >= ALL
(select R1.numero_RSA
From REGIONI R1)
```

DENOMINAZIONE_REGIONE	NUMERO_RSA
1 Lombardia	50124

#### NUMERO TOTALE DI OSPEDALI IN ITALIA

```
select sum(numero_ospedali) as NUMERO_OSPEDALI_ITALIANI
from PROVINCE;
```

NUMERO_OSPEDALI_ITALIANI
1 254

#### DENSITÀ MEDIA DI POPOLAZIONE IN ITALIA

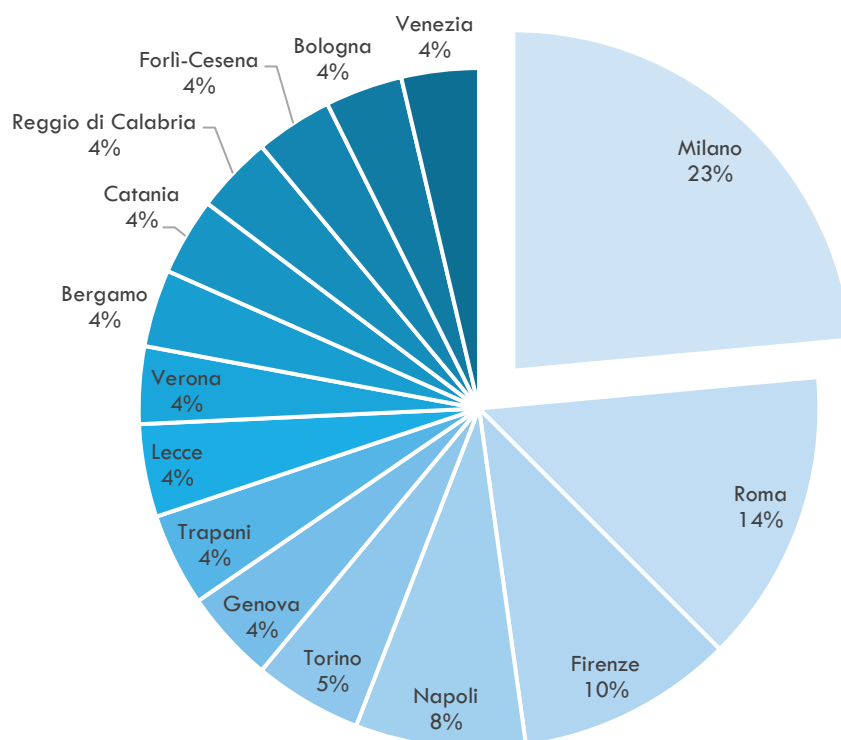
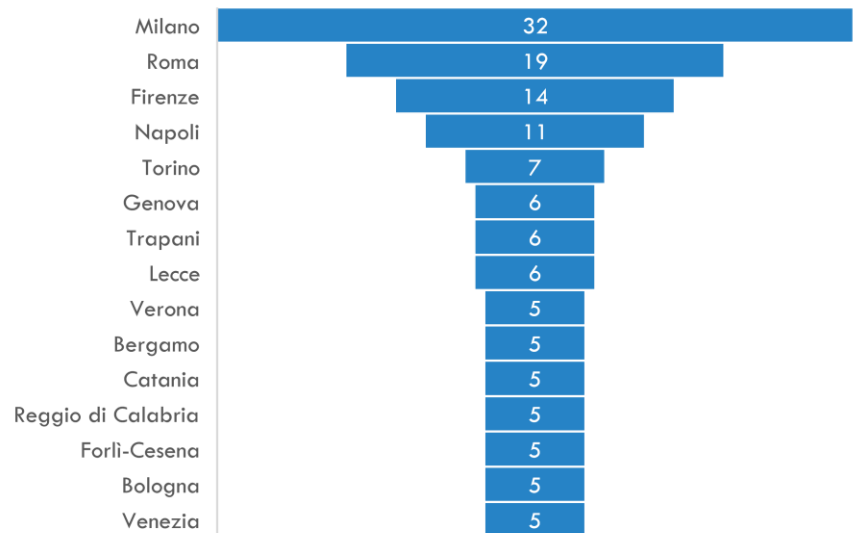
```
select AVG(densita) as DENSITA_MEDIA
from PROVINCE;
```

DENSITA_...
1 272,491...

## PROVINCE CON PIÙ DI 4 OSPEDALI PUBBLICI

```
select denominazione_provincia as PROVINCIA, numero_ospedali as OSPEDALI
from PROVINCE
where numero_ospedali >= 5
order by numero_ospedali desc;
```

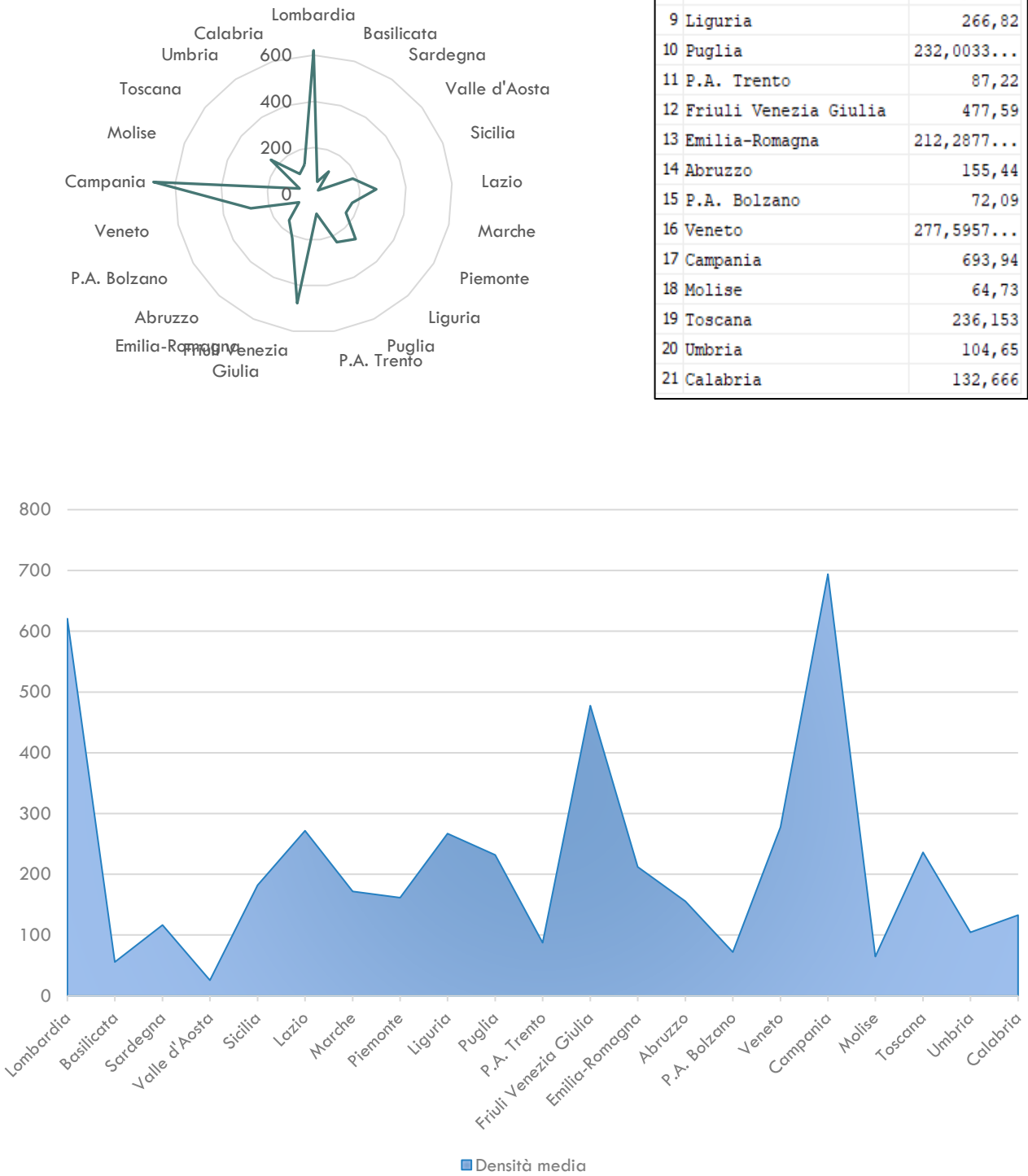
PROVINCIA	OSPEDALI
1 Firenze	38
2 Milano	32
3 Roma	19
4 Napoli	11
5 Torino	7
6 Genova	6
7 Trapani	6
8 Lecce	6
9 Verona	5
10 Bergamo	5
11 Catania	5
12 Reggio di Calabria	5
13 Forlì-Cesena	5
14 Bologna	5
15 Venezia	5



# DENSITÀ MEDIA DI POPOLAZIONE PER REGIONE

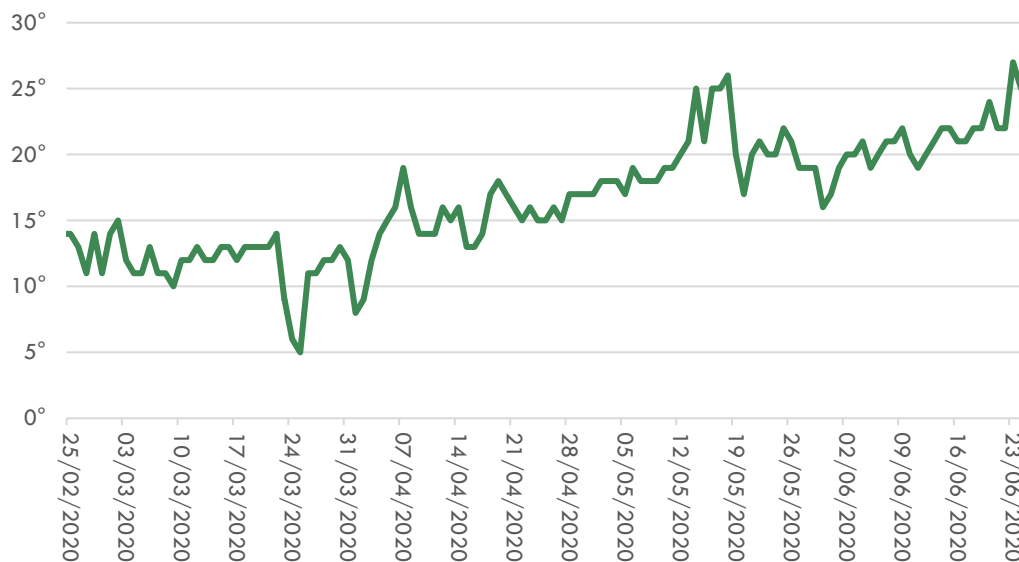
```
select R.denominazione_regione as REGIONE,
Avg(P.densita) as DENSITA_MEDIA
from PROVINCE P join REGIONI R on
P.codice_regione=R.codice_regione
group by R.denominazione_regione;
```

REGIONE	DENSITA_...
1 Lombardia	620,5883...
2 Basilicata	55,665
3 Sardegna	116,624
4 Valle d'Aosta	25,59
5 Sicilia	182,1933...
6 Lazio	271,8
7 Marche	171,914
8 Piemonte	161,66625
9 Liguria	266,82
10 Puglia	232,0033...
11 P.A. Trento	87,22
12 Friuli Venezia Giulia	477,59
13 Emilia-Romagna	212,2877...
14 Abruzzo	155,44
15 P.A. Bolzano	72,09
16 Veneto	277,5957...
17 Campania	693,94
18 Molise	64,73
19 Toscana	236,153
20 Umbria	104,65
21 Calabria	132,666



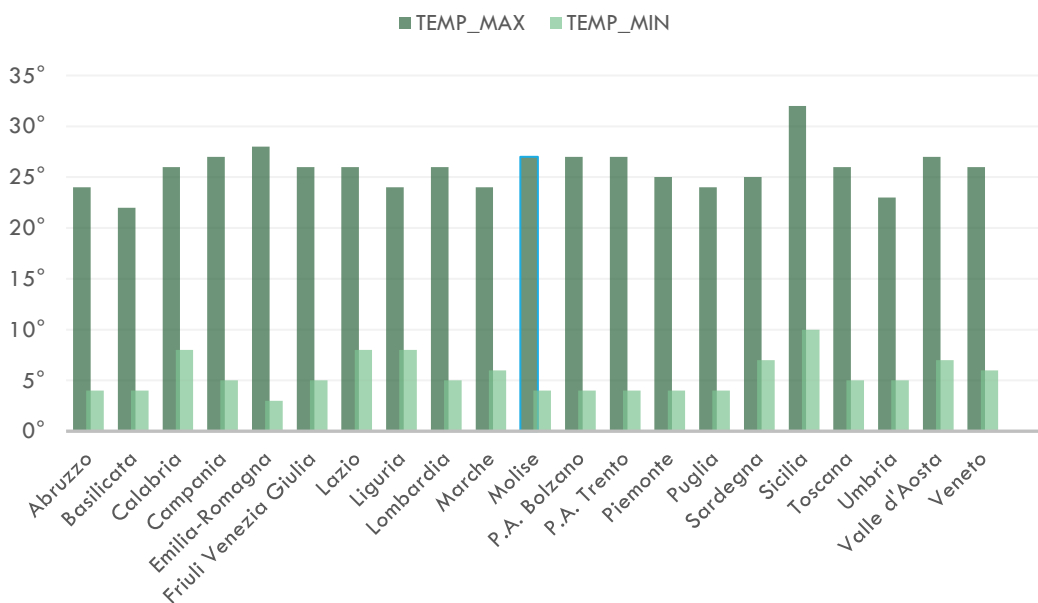
### ANDAMENTO TEMPERATURA MEDIA IN CAMPANIA

```
select M.data, M.temperatura_media
from REGIONI R join METEO M on R.codice_regione=M.codice_regione
where R.denominazione_regione='Campania'
order by data
```



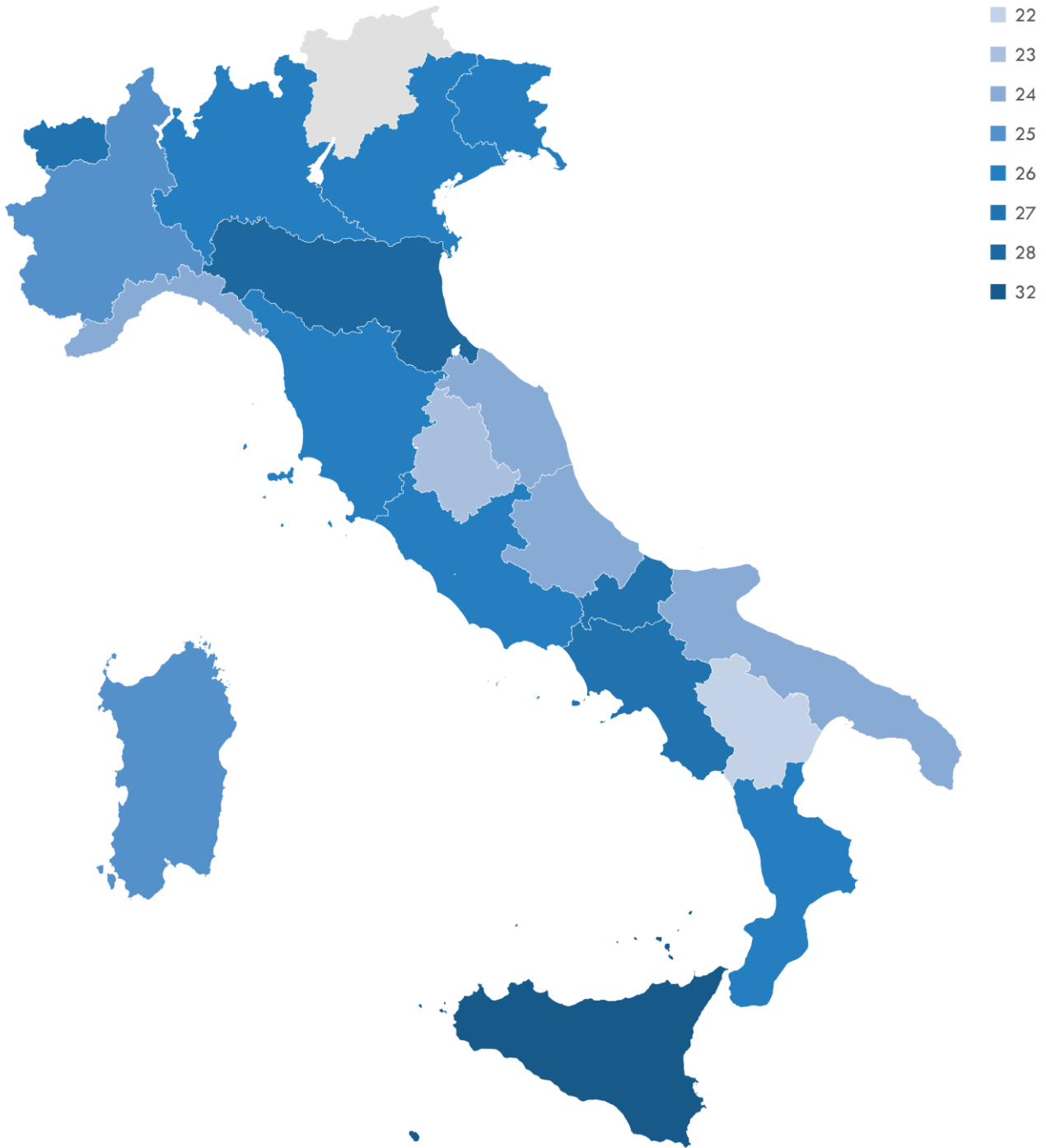
### TEMPERATURA PIÙ ALTA E PIÙ BASSA PER OGNI REGIONE

```
select distinct R.denominazione_regione, MAX(M.temperatura_media) as
TEMP_MAX, MIN(M.temperatura_media) as TEMP_MIN
from REGIONI R join METEO M on R.codice_regione=M.codice_regione
group by R.denominazione_regione, M.codice_regione order by
R.denominazione_regione;
```



### TEMPERATURE MASSIME IN ITALIA TRA IL 25 FEBBRAIO 2020 E IL 24 GIUGNO 2020

```
select distinct R.denominazione_regione, MAX(M.temperatura_media) as  
TEMP_MAX  
from REGIONI R join METEO M on R.codice_regione=M.codice_regione  
group by R.denominazione_regione, M.codice_regione  
order by TEMP_MAX;
```





# Programmazione della base dati

## Trigger

In questo paragrafo si mostrano delle procedure necessarie per automatizzare l'eliminazione dalle relazioni delle tuple prive di informazione, la modifica dei valori non corretti e lo smistamento dei dati nel caso di inserimento di tuple nella tabella MASTER. Infatti, nei paragrafi precedenti, è stato descritto il processo non automatico che porta al corretto popolamento delle tabelle normalizzate: sulla tabella master COVID19 sono stati inizialmente utilizzati dei comandi DML per l'eliminazione delle tuple "in fase di definizione" e per la modifica del valore assunto in corrispondenza del campo `codice_regione` dalle tuple relative alle province autonome Bolzano e Trento, e successivamente delle INSERT per il popolamento delle tabelle ricavate con la normalizzazione. Grazie a tali processi automatizzati, nel caso in cui un utente avente i permessi volesse inserire dati sul contagio di date successive al 24 giugno 2020 (data di fine della nostra analisi), si avrebbe la sicurezza che le tuple che si propagano nelle tabelle normalizzate siano pulite.

- **TRIGGER PER LA MODIFICA DEL CODICE\_REGIONE DI BOLZANO**

```
CREATE OR REPLACE TRIGGER Modifica_Bolzano
BEFORE insert ON COVID19
FOR EACH ROW
BEGIN
    IF :new.denominazione_provincia='Bolzano' THEN
        :new.codice_regione:=98;
    END IF;
END;
```

- **TRIGGER PER LA MODIFICA DEL CODICE\_REGIONE DI TRENTO**

```
CREATE OR REPLACE TRIGGER Modifica_Trento
BEFORE insert ON COVID19
FOR EACH ROW
BEGIN
    IF :new.denominazione_provincia='Trento' THEN
        :new.codice_regione:=99;
    END IF;
END;
```

- **TRIGGER PER IL POPOLAMENTO DI COVID19\_PROVINCE**

```
CREATE OR REPLACE TRIGGER Popolamento_covid19_province
AFTER INSERT ON COVID19
FOR EACH ROW
BEGIN
    IF :new.denominazione_provincia NOT LIKE 'In fase di
    definizione/aggiornamento' THEN
        INSERT INTO COVID19_PROVINCE (data, codice_provincia, totale_casi,
        note_it, note_en) values( :new.data, :new.codice_provincia,
        :new.totale_casi, :new.note_it, :new.note_en);
    END IF;
END;
```

Sono poi stati implementati ulteriori trigger per l'aggiornamento delle tabelle in caso di modifica della tabella MASTER

- **AGGIUNTA DI UNA NOTA IN INGLESE O IN ITALIANO AL REPORT GIORNALIERO DI UNA PROVINCIA**

```
CREATE OR REPLACE TRIGGER Aggiorna_Note
AFTER UPDATE ON COVID19
FOR EACH ROW
BEGIN
    UPDATE COVID19_PROVINCE SET note_it=:new.note_it,
    note_en=:new.note_en      where data=:new.data AND
    codice_provincia=:new.Codice_provincia;
END;
```

- **CAMBIO DEL CODICE DI UNA REGIONE (POLITICA DI UPDATE CASCADE)**

```
CREATE OR REPLACE TRIGGER Aggiorna_Regioni
BEFORE UPDATE ON REGIONI
FOR EACH ROW
BEGIN
    UPDATE PROVINCE PR SET PR.codice_regione=:new.codice_regione
    WHERE PR.codice_regione=:old.codice_regione;
    UPDATE COVID19_REGIONI CR SET CR.codice_regione=:new.codice_regione
    WHERE CR.codice_regione=:old.codice_regione;
    UPDATE METEO M SET M.codice_regione=:new.codice_regione
    WHERE M.codice_regione=:old.codice_regione;
    UPDATE DISTRIBUZIONE D SET D.codice_regione=:new.codice_regione
    WHERE D.codice_regione=:old.codice_regione;
END;
```

- **CAMBIO DEL CODICE DI UNA PROVINCIA (POLITICA DI UPDATE CASCADE)**

```
CREATE OR REPLACE TRIGGER Aggiorna_Provincia2
BEFORE UPDATE ON PROVINCE
FOR EACH ROW
BEGIN
    UPDATE COVID19_PROVINCE CP set
    CP.codice_provincia=:new.codice_provincia
    WHERE CP.codice_provincia=:old.codice_provincia;
END;
```

- **AGGIORNAMENTO SIGLA PROVINCIA**

```
CREATE OR REPLACE TRIGGER Aggiorna_Sigla
BEFORE UPDATE ON COVID19
FOR EACH ROW
BEGIN
    UPDATE PROVINCE P SET P.sigla_provincia=:new.sigla_provincia
    WHERE P.sigla_provincia=:old.sigla_provincia;
END;
```

- **CANCELLAZIONE DI UNA PROVINCIA (POLITICA DI DELETE CASCADE)**

Se si volesse fare una DELETE di una provincia, questa non risulterebbe una istruzione legale in quanto esiste un vincolo di integrità referenziale che lega la tabella COVID19\_PROVINCE a PROVINCE. È possibile implementare una politica di reazione “a cascata” tramite un trigger, il quale, nell’istante subito precedente alla DELETE, provvede ad eliminare le tuple che referenziano la provincia da eliminare (e a pulire la tabella MASTER).

```
CREATE OR REPLACE TRIGGER Elimina_Provincia
BEFORE DELETE ON PROVINCE
FOR EACH ROW
BEGIN
    DELETE from COVID19_PROVINCE CP
    WHERE CP.codice_provincia=:old.codice_provincia;
    DELETE from COVID19 C
    WHERE C.codice_provincia=:old.codice_provincia;
END;
```

- **CANCELLAZIONE DI UNA REGIONE (POLITICA DI DELETE CASCADE)**

```
CREATE OR REPLACE TRIGGER Elimina_Regione
BEFORE DELETE ON REGIONI
FOR EACH ROW
BEGIN
    DELETE from PROVINCE P
    WHERE P.codice_regione=:old.codice_regione;
    DELETE from METEO M
    WHERE M.codice_regione=:old.codice_regione;
    DELETE from DISTRIBUZIONE D
    WHERE D.codice_regione=:old.codice_regione;
    DELETE from COVID19_REGIONI CR
    WHERE CR.codice_regione=:old.codice_regione;
    DELETE from COVID19 C
    WHERE C.codice_regione =:old.codice_regione;
END;
```

Sono poi stati aggiunti trigger di check all’inserimento nella tabella MASTER.

- **LE PROVINCE DEVONO ESSERE ITALIANE**

```
CREATE OR REPLACE TRIGGER Check_stato
BEFORE INSERT ON COVID19
FOR EACH ROW
DECLARE
    STATO_ERR EXCEPTION;
BEGIN
    IF :new.stato NOT LIKE 'ITA' THEN RAISE STATO_ERR;
    END IF;
EXCEPTION
    WHEN STATO_ERR
    THEN raise_application_error(-20001,'inserimento con stato non
    corretto');
END;
```

- **LA DENOMINAZIONE DELLA REGIONE DEVE ESSERE CORRISPONDENTE CON UNA DELLE ATTUALI REGIONI ITALIANE (**

```
CREATE OR REPLACE TRIGGER Check_regione
BEFORE INSERT ON COVID19
FOR EACH ROW
BEGIN
    DECLARE
        REGIONE_ERR EXCEPTION;
        Temp INTEGER;
    BEGIN
        select count(*) into Temp from REGIONI R
        where R.denominazione_regione=:new.denominazione_regione;
        IF(Temp=0) THEN RAISE REGIONE_ERR;
        END IF;
    EXCEPTION
        WHEN REGIONE_ERR
        THEN raise_application_error(-20002, 'Regione non corretta');
    END;
END check_regione;
```

- **DENOMINAZIONE DELLA PROVINCIA CORRISPONDENTE CON UNA DELLE ATTUALI PROVINCE ITALIANE**

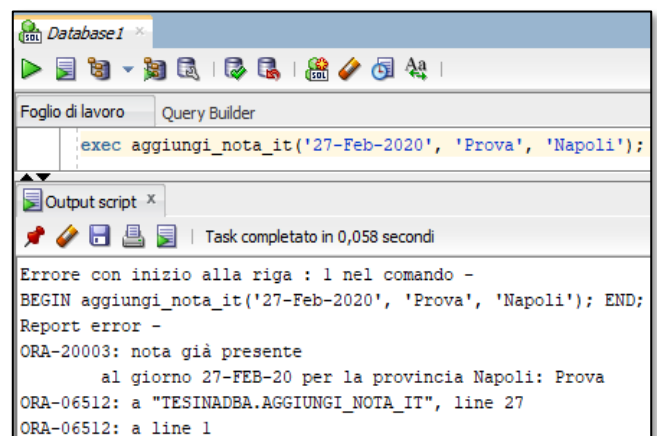
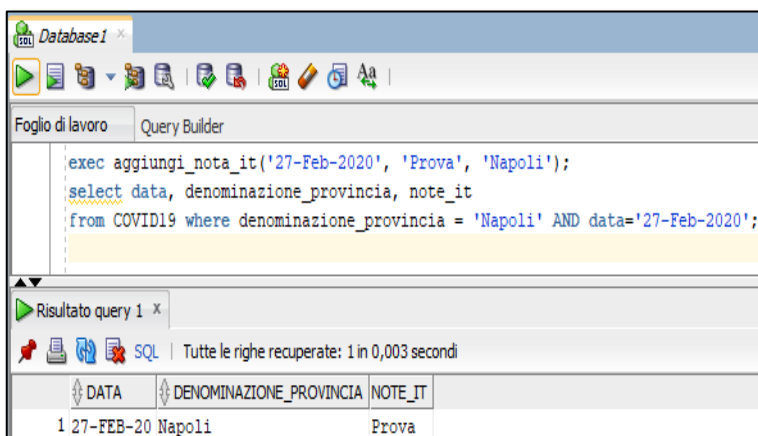
```
CREATE OR REPLACE TRIGGER Check_Provincia
BEFORE INSERT ON COVID19
FOR EACH ROW
BEGIN
    DECLARE
        PROVINCIA_ERR EXCEPTION;
        Temp INTEGER;
    BEGIN
        select count(*) into Temp from PROVINCE P
        where P.denominazione_provincia=:new.denominazione_provincia;
        IF(Temp=0) THEN RAISE PROVINCIA_ERR;
        END IF;
    EXCEPTION
        WHEN PROVINCIA_ERR
        THEN raise_application_error(-20003, 'Provincia non
corretta');
    END;
END Check_Provincia;
```

## Procedure

### • AGGIUNTA DI UNA NOTA

È stata implementata una procedura che facilita l'aggiunta di una nota alla tupla relativa ad un determinato giorno e una determinata provincia. L'aggiornamento viene fatto sulla tabella master COVID19 poiché esiste un apposito trigger, precedentemente mostrato, che propaga l'aggiornamento anche alla tabella COVID19\_PROVINCE.

```
CREATE OR REPLACE PROCEDURE Aggiungi_nota_it
(
    DataIN IN COVID19.data%TYPE,
    Nota IN COVID19.note_it%TYPE,
    Provincia IN COVID19.denominazione_provincia%TYPE
)
AS
BEGIN
    DECLARE
        Temp INTEGER;
        NOTE_EXISTS EXCEPTION;
        NoteTest COVID19.note_it%TYPE;
    BEGIN
        select count(*) into Temp from COVID19 C1
        where C1.denominazione_provincia=Provincia AND C1.data=DataIN
        AND C1.note_it IS NULL;
        IF (Temp=0) THEN RAISE NOTE_EXISTS;
        ELSE Update COVID19 C set C.note_it = Nota
        Where C.denominazione_provincia=Provincia AND C.data=DataIN;
        END IF;
    EXCEPTION
        WHEN NOTE_EXISTS THEN
            Select C2.note_it into NoteTest
            From COVID19 C2 where C2.denominazione_provincia = Provincia
            AND C2.data=DataIN;
            raise_application_error(-20003, 'nota già presente
            al giorno ' || DataIN || ' per la provincia ' || Provincia ||
            ': ' || NoteTest); ('nota già presente
            al giorno ' || DataIN || ' per la provincia' || Provincia ||
            ': ' || NoteTest);
    END;
END Aggiungi_nota_it;
```



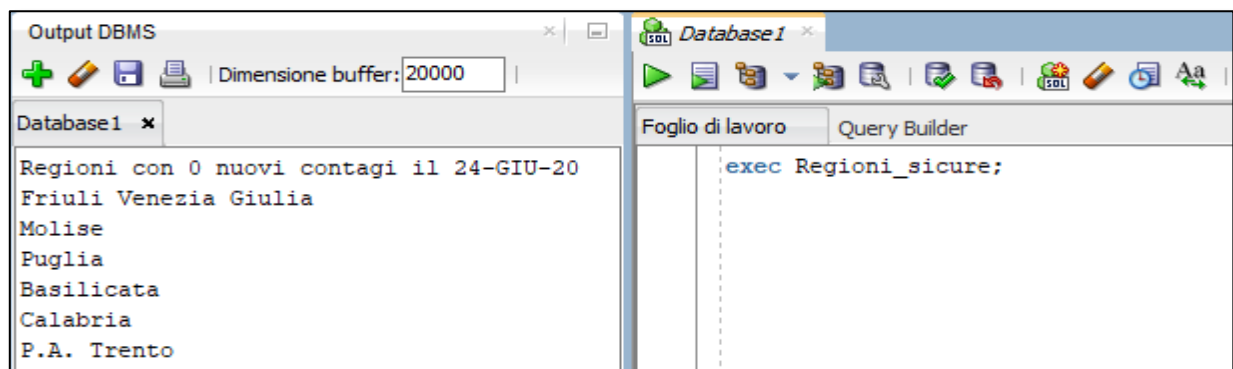
- **STAMPA A VIDEO DELLE REGIONI CON NUMERO DI NUOVI CASI PARI A 0 NELL'ULTIMO GIORNO DI ANALISI**

```

CREATE OR REPLACE PROCEDURE Regioni_sicure
AS
BEGIN
    DECLARE
        DataTemp COVID19.data%TYPE;
        RegioneN REGIONI.denominazione_regione%TYPE;
        Temp INTEGER;
        NONE exception;
        CURSOR CRSR IS select R.denominazione_regione
        From REGIONI R join COVID19_REGIONI C19 on
        R.codice_regione=C19.codice_regione
        WHERE C19.data=Ultimo_giorno() AND C19.nuovi_positivi=0;

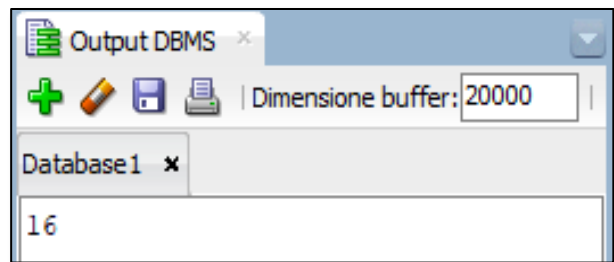
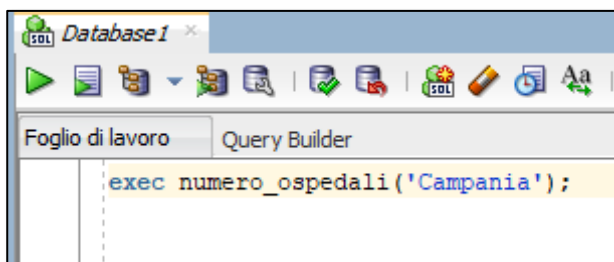
    BEGIN
        select count(*) INTO Temp
        From REGIONI R join COVID19_REGIONI C19 on
        R.codice_regione=C19.codice_regione
        Where C19.data=Ultimo_giorno() AND C19.nuovi_positivi=0;
        IF(Temp=0) THEN RAISE NONE;
        ELSE
            OPEN CRSR;
            DataTemp:=Ultimo.giorno();
            DBMS_OUTPUT.PUT_LINE('Regioni con 0 nuovi contagi
            il '|| DataTemp );
            LOOP
                FETCH CRSR INTO RegioneN;
                EXIT WHEN CRSR%NOTFOUND;
                DBMS_OUTPUT.PUT_LINE(RegioneN);
            END LOOP;
            CLOSE CRSR;
        END IF;
    EXCEPTION
        WHEN NONE THEN
            DBMS_OUTPUT.PUT_LINE('Attualmente non esistono regioni
            sicure');
    END;
END Regioni_sicure;

```



- **VISUALIZZA IL NUMERO DI OSPEDALI PRESENTI IN UNA DATA REGIONE**

```
CREATE OR REPLACE PROCEDURE Numero_Ospedali
(
    Regione IN REGIONI.denominazione_regione%TYPE
)
AS
BEGIN
    DECLARE
        NumOspedali PROVINCE.numero_ospedali%type;
    BEGIN
        select SUM(P.numero_ospedali) into NumOspedali
        From REGIONI R join PROVINCE P on
        P.codice_regione=R.codice_regione
        where R.denominazione_regione=Regione;
        DBMS_OUTPUT.PUT_LINE('Nella regione' ||
        Regione || 'sono presenti' || NumOspedali
        || 'ospedali pubblici.');
```



- **REGIONE CON PIU' TAMPONI EFFETTUATI FINO AD UNA DATA ASSEGNATA**

```
CREATE OR REPLACE PROCEDURE Calcolo_tamponi
(
    Data_1 IN DATE
)
AS
BEGIN
    DECLARE
        Nome_R REGIONI.denominazione_regione%TYPE;
    BEGIN
        select denominazione_regione into Nome_R
        from REGIONI r join COVID19_REGIONI c on
        r.codice_regione=c.codice_regione where c.data=Data_1 AND
        c.tamponi >= ALL (select tamponi from COVID19_REGIONI where
        data=Data_1);
        DBMS_OUTPUT.PUTLINE('La regione italiana con più tamponi fatti
        fino al giorno' || Data_1 || ' è ' || Nome_R || '.');
```

## Funzioni

Sono di seguito elencate delle funzioni PL/SQL che restituiscono informazioni utili riguardanti la situazione del contagio in Italia:

- **GIORNO DI FINE ANALISI (INNESTATA NELLA FUNZIONE SUCCESSIVA)**

La seguente funzione restituisce la data di fine analisi, dunque può essere utilizzata per trovare i dati più recenti memorizzati. Ne è un esempio di utilizzo la query *“select \* from covid19\_province where data=Ultimo\_giorno();”* che seleziona i dati provinciali sul contagio più recenti contenuti nella base dati.

```
CREATE OR REPLACE FUNCTION Ultimo_giorno
RETURN DATE
IS
BEGIN
    DECLARE
        Ultimo_giorno COVID19.data%type;
    BEGIN
        Select max(data) INTO Ultimo_giorno from COVID19;
        RETURN Ultimo_giorno;
    END;
END Ultimo_giorno;
```

- **PROVINCIA CON MAGGIOR NUMERO DI CONTAGIATI TOTALI ALL'ULTIMO GIORNO DI ANALISI**

La seguente query, grazie all'utilizzo innestato della funzione Ultimo\_giorno, restituisce la denominazione della provincia che, al giorno di fine analisi, ha totalizzato più casi di contagio in Italia. Ne è un esempio di utilizzo la query *“select \* from PROVINCE where denominazione\_provincia=Provincia\_colpita\_maggiormente();”*

```
CREATE OR REPLACE FUNCTION Provincia_colpita_maggiormente
RETURN Province.denominazione_provincia%TYPE
IS
BEGIN
    DECLARE
        Temp INTEGER;
        Nome_Provincia Province.denominazione_provincia%TYPE;
        FUNCTION_ERR2 EXCEPTION;
    BEGIN
        Select count(*) INTO Temp from COVID19_PROVINCE CP1 join
        PROVINCE P1 on CP1.codice_provincia=P1.codice_provincia Where
        CP1.data=Ultimo_giorno() AND CP1.totale_casi >= ALL
        (Select CP2.totale_casi from COVID19_PROVINCE CP2 where
        CP2.data=Ultimo_giorno());

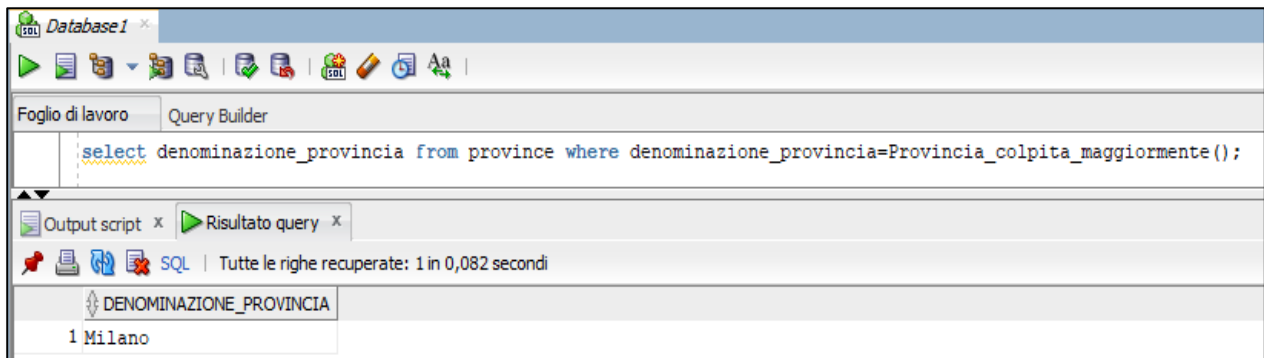
        IF(Temp=1) THEN
            Select P.denominazione_provincia INTO Nome_provincia
            from COVID19_PROVINCE CP join PROVINCE P on
            CP.codice_provincia=P.codice_provincia Where
            CP.data=Ultimo_giorno() AND CP.totale_casi >= ALL
            (Select CP1.totale_casi from COVID19_PROVINCE CP1 where
            CP1.data=Ultimo_giorno());
        END IF;
    END;
END Provincia_colpita_maggiormente;
```



```

        ELSE Raise FUNCTION_ERR2;
    END IF;
    RETURN Nome_Provincia;
EXCEPTION
    WHEN FUNCTION_ERR2 THEN
        raise_application_error(-20005, 'Piu' province hanno lo stesso
        numero di casi totali');
END;
END Provincia_colpita_maggiormente;

```



- **REGIONE CON NUMERO DI ANZIANI PIU' ALTO IN ITALIA**

```

CREATE OR REPLACE FUNCTION Regione_piu_anziani
RETURN REGIONI.denominazione_regione%TYPE
IS
BEGIN
    DECLARE
        Temp INTEGER;
        Nome_R REGIONI.denominazione_regione%TYPE;
        FUNCTION_ERR1 EXCEPTION;
    BEGIN
        SELECT count(*) INTO Temp from Regioni R1
        Where numero_anziani >= ALL( SELECT max(R2.numero_anziani)
        from REGIONI R2);
        IF(Temp=1) THEN
            SELECT denominazione_regione INTO Nome_R
            FROM REGIONI WHERE numero_anziani >= ALL(
            SELECT max(R4.numero_anziani) from REGIONI R4);
            ELSE RAISE FUNCTION_ERR1;
            END IF;
            RETURN Nome_R;
        EXCEPTION
            WHEN FUNCTION_ERR1 THEN
                raise_application_error(-20004, 'Esistono più regioni con lo
                stesso numero di anziani');
        END;
    END Regione_piu_anziani;

```

- **SOMMA DELLE PERSONE OSPEDALIZZATE IN UN INTERVALLO TEMPORALE A LIVELLO NAZIONALE**

```
CREATE OR REPLACE FUNCTION Somma_ospedalizzati
(
    Data_1 IN COVID19.data%TYPE,
    Data_2 IN COVID19.data%TYPE
)
RETURN COVID19_REGIONI.totale_ospedalizzati%TYPE
IS
BEGIN
    DECLARE
        Temp INTEGER;
    BEGIN
        SELECT sum(totale_ospedalizzati) INTO TEMP
        FROM COVID19_REGIONI
        WHERE Data_1<=data and data<=Data_2;
        DBMS_OUTPUT.PUT_LINE(Temp);
        RETURN Temp;
    END;
END Somma_ospedalizzati;
```

- **NUMERO DI GUANTI DISTRIBUITI IN UNA DATA REGIONE E IN UN DATO INTERVALLO TEMPORALE**

```
CREATE OR REPLACE FUNCTION Guanti_distribuiti
(
    Data_1 IN DATE,
    Data_2 IN DATE,
    Regione_1 IN REGIONI.denominazione_regione%TYPE
)
RETURN Distribuzione.guanti%TYPE
IS
BEGIN
    DECLARE
        Temp INTEGER;
    BEGIN
        SELECT sum(d.guanti) INTO Temp
        FROM DISTRIBUZIONE d JOIN REGIONI r ON
        d.codice_regione=r.codice_regione
        WHERE r.denominazione_regione like Regione_1 AND d.data >= Data_1
        AND d.data <= Data_2;
        RETURN Temp;
    END;
END Guanti_distribuiti;
```

## Ottimizzazione analytics

### Viste

Un primo metodo di ottimizzazione dell'esecuzione delle query è quello di definire delle viste sulla base dati. In questo caso, risulterà utile creare delle viste *materializzate*, che rappresentano delle fotografie dei dati in un certo istante di tempo. Quindi, le viste materializzate non sono allineate ai dati su cui esse vengono calcolate, nel caso in cui la tabella base venga modificata. Come si può vedere, negli esempi successivi, usare una vista materializzata permetterà di risparmiare l'utilizzo del *JOIN*, uno delle operazioni più costose a livello computazionale nelle istruzioni di interrogazione.

#### VISUALIZZAZIONE DI TUTTI I DATI DELLE REGIONI MERIDIONALI

```
CREATE MATERIALIZED VIEW REGIONI_MERIDIONALI AS SELECT * from REGIONI
where denominazione_regione LIKE 'Campania' or denominazione_regione LIKE
'Puglia' or denominazione_regione LIKE 'Basilicata' or
denominazione_regione LIKE 'Calabria' or denominazione_regione like
'Sicilia';
```

```
SELECT * FROM REGIONI_MERIDIONALI;
```

	CODICE_REGIONE	DENOMINAZIONE_REGIONE	STATO	NUMERO_SCUOLE	NUMERO_SPORTELLI	PIL	PIL_PROCAPITE	NUMERO_RSA	NUMERO_AEROPORTI	ETA_MEDIA
1	15	Campania	ITA	7404	1206	106431000000	18200	1152	6	42,15
2	16	Puglia	ITA	3825	1042	74752000000	18700	173	9	44,23
3	17	Basilicata	ITA	868	198	12023000000	21100	123	0	45,31
4	18	Calabria	ITA	3365	372	33965000000	17400	755	5	43,98
5	19	Sicilia	ITA	6344	1182	86998000000	17400	863	8	43,53

#### REGIONE CON TEMPERATURA MEDIA PIÙ BASSA

```
CREATE MATERIALIZED VIEW STATISTICHE_METEO AS select distinct
R.denominazione_regione AS NOME, MIN(M.temperatura_media) as TEMP_MIN,
MAX(M.temperatura_media) as TEMP_MAX
from REGIONI R join METEO M on R.codice_regione=M.codice_regione GROUP BY
R.denominazione_regione, M.codice_regione;
```

```
SELECT s.NOME, s.TEMP_MIN
FROM STATISTICHE_METEO s WHERE s.TEMP_MIN
ALL (
SELECT s1.TEMP_MIN
FROM STATISTICHE_METEO s1);2
```

	NOME	TEMP_MIN
1	Emilia-Romagna	3

 <=

#### REGIONE CON IL MAGGIOR NUMERO DI MASCHERINE DISTRIBUITE IN UN GIORNO

```
CREATE MATERIALIZED VIEW STATISTICHE_MASCHERINE as select distinct
R.denominazione_regione AS NOME, MAX(D.mascherine) as NUMERO_MASSIMO from
REGIONI R join DISTRIBUZIONE D on R.codice_regione=D.codice_regione GROUP
BY R.denominazione_regione, D.codice_regione;
```

```
SELECT s.NOME, s.NUMERO_MASSIMO
FROM STATISTICHE_MASCHERINE s
WHERE s.NUMERO_MASSIMO>=ALL (
SELECT s1.NUMERO_MASSIMO
FROM STATISTICHE_MASCHERINE s1);
```

	NOME	NUMERO_MASSIMO
1	Veneto	4702200

<sup>2</sup> Nella sottointerrogazione, l'uso di due differenti ridenominazioni della vista STATISTICHE\_METEO (s e s1) è stato applicato per una migliore visibilità grafica, ma non è assolutamente necessario in quanto le due query sono indipendenti tra loro

## NUMERO DI ABITANTI E ALTITUDINE DELLA PROVINCIA LOMBARDA CON PIU' CASI IN TOTALE

```
CREATE MATERIALIZED VIEW STATISTICHE_PROVINCIA AS SELECT
p.denominazione_provincia,p.numero_abitanti,p.altitudine,c.totale_casi,c.
data, r.denominazione_regione FROM (PROVINCE p JOIN COVID19_PROVINCE c ON
c.codice_provincia = p.codice_provincia) JOIN REGIONI r ON
p.codice_regione = r.codice_regione WHERE c.data='24-Giu-2020';
```

```
SELECT denominazione_provincia, numero_abitanti, altitudine, totale_casi
FROM STATISTICHE_PROVINCIA WHERE denominazione_regione LIKE 'Lombardia'
AND totale_casi>=ALL ( SELECT totale_casi FROM STATISTICHE_PROVINCIA
WHERE denominazione_regione LIKE 'Lombardia');
```

	DENOMINAZIONE_PROVINCIA	NUMERO_ABITANTI	ALTITUDINE	TOTALE_CASI
1	Milano	3267053	122	24239

## Indici

Un altro metodo utile per migliorare la velocità di esecuzione di una interrogazione sulla base dati è creare degli indici su alcune tabelle della base dati.

Nei DBMS (*Data Base Management System*) un indice è una struttura dati che permette di individuare più velocemente un particolare record in un data file, ottenendo come vantaggio una chiara diminuzione dei tempi di risposta di una query. Perciò, sono stati creati indici di tipo B-Tree (*Balanced Tree*) sui campi *denominazione\_provincia* e *denominazione\_regione* delle rispettive tabelle PROVINCE e REGIONI in quanto verranno fatte delle ricerche puntuali proprio su questi attributi:

```
CREATE INDEX index_regioni ON REGIONI(denominazione_regione);
CREATE INDEX index_province ON PROVINCE(denominazione_provincia);
```

Quando viene utilizzato lo statement *CREATE INDEX*, non c'è bisogno di specificare il tipo di indice se si vuole creare un indice B-Tree. Infatti, di default, l'indice creato sarà di tipo B-Tree.

In maniera molto simile, è stato creato un indice<sup>3</sup> B-Tree sul campo *data* della vista materializzata STATISTICHE\_PROVINCIA, in quanto anche qui sono stati analizzati dei valori per una singola data, ovvero il 24 giugno 2020:

```
CREATE INDEX index_data ON STATISTICHE_PROVINCIA (data);
```

Infine, è da sottolineare il fatto che se ci fossero state delle viste che lavoravano su intervalli di date, allora sarebbero stati definiti degli indici B+ Tree, una variante dell'albero B dove i puntatori ai dati sono memorizzati esclusivamente nei nodi foglia dell'albero.

<sup>3</sup> L'indice creato non è sicuramente primario in quanto la vista materializzata non contiene chiave primaria

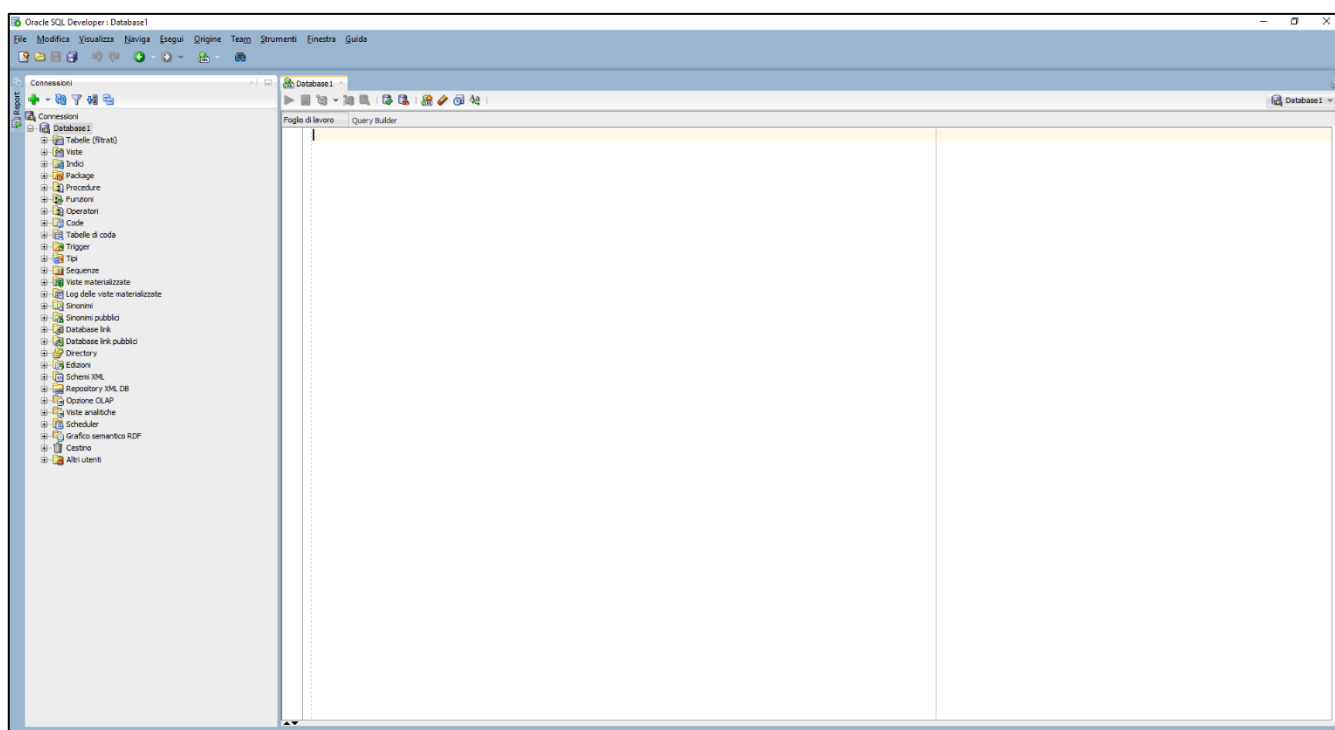
## Appendice

### Oracle Database 18c (Express edition) e Oracle SQL developer



Per il seguente progetto si è fatto uso della distribuzione Oracle Database 18c Express edition, fornita gratuitamente da Oracle Corporation sulla piattaforma online ufficiale, in sinergia con l'ambiente di sviluppo integrato SQL Developer (versione 19.2.1.247), sviluppato dalla stessa società. SQL developer permette di lavorare sui database Oracle tramite il linguaggio SQL, in alternativa all'utilizzo del tool integrato nella distribuzione (SQL Plus) il quale presenta una interfaccia a command line.

La seguente immagine è un esempio di interfaccia di gestione di una base dati (Database1) alla quale si è connessi. All'interno di un "foglio di lavoro" è possibile scrivere le istruzioni (scritte in linguaggio SQL) che si vogliono eseguire (è chiaro che le azioni possibili sono dipendenti dal particolare utente con il quale si è effettuato l'accesso).



SQL DEVELOPER

Completato il download del file Zip contenente la distribuzione ed estratto il contenuto, per avviare l'installazione sul dispositivo locale è stato sufficiente lanciare il tool di setup, scegliere la directory di destinazione ed impostare una password per il database (utile per accedervi tramite ad esempio l'utente *SYSTEM*). Al termine dell'installazione saranno automaticamente avviati i servizi Oracle necessari per connettersi ad esso ed utilizzarlo.

OracleOraDB18Home1MTSRecoveryService  
OracleOraDB18Home1TNSListener  
OracleServiceXE

In esecuzione Automatico  
In esecuzione Automatico  
In esecuzione Automatico

NT SERVICE\OracleOraDB18Home1MTSRecoveryService  
NT SERVICE\OracleOraDB18Home1TNSListener  
NT SERVICE\OracleServiceXE

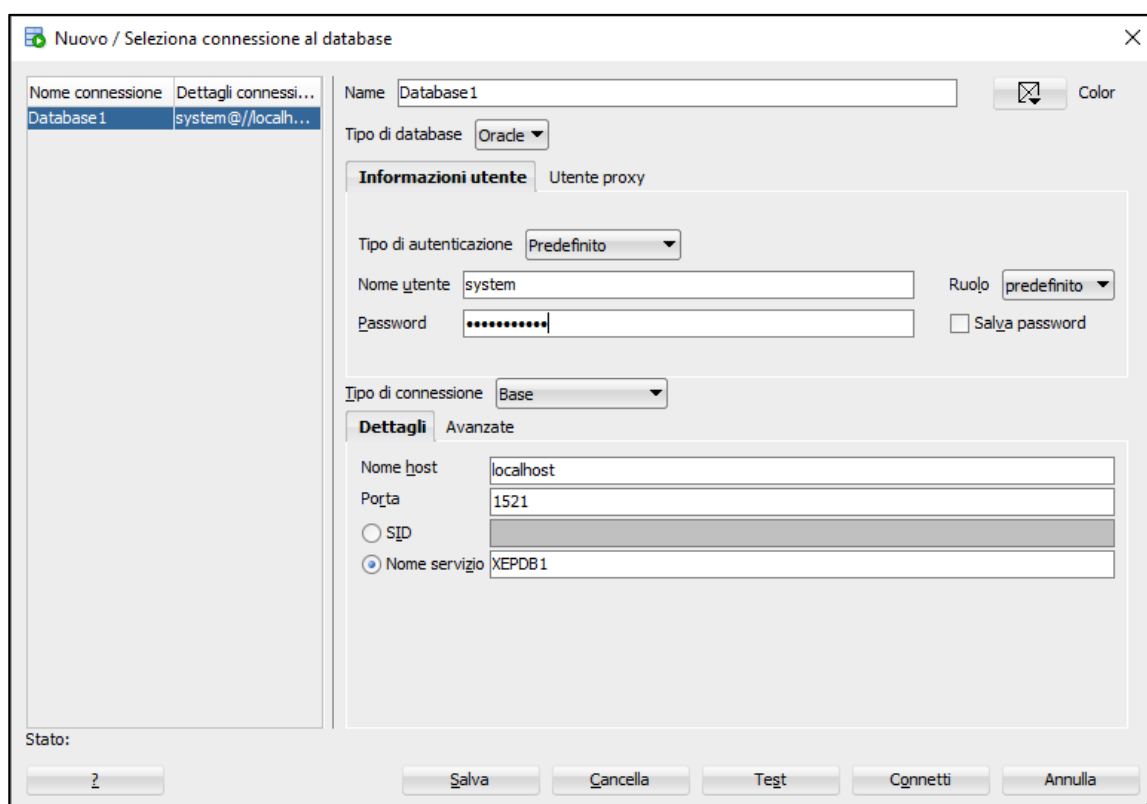
#### SERVIZI ORACLE ATTIVI

Successivamente la connessione è stata testata lanciando l'eseguibile sqlplus dalla shell integrata di Windows, in particolare accedendo al Pluggable Database di default (XEPDB1) con l'utente SYSTEM, specificando la porta (di default la 1521) e l'indirizzo IP nella rete locale della macchina (possibile anche utilizzando l'alias localhost).

```
C:\Users\Username>sqlplus system/*****@localhost:1521/XEPDB1
SQL*Plus: Release 18.0.0.0.0 - Production on ...Timestamp....
Version 18.4.0.0.0
Copyright (c) 1982, 2018, Oracle. All rights reserved.
Connesso a:
Oracle Database 18c Express Edition Release 18.0.0.0.0 - Production
Version 18.4.0.0.0
```

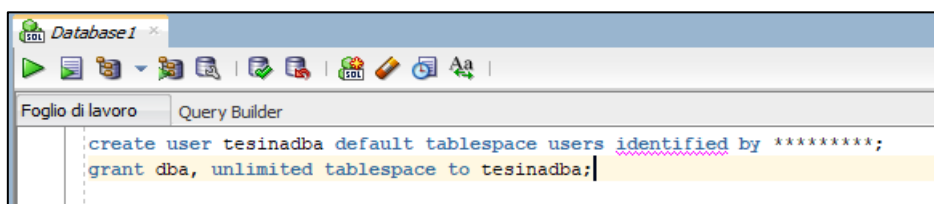
#### ACCESSO AL PLUGGABLE DATABASE XEPDB1

Una volta avviato il developer, è stata aggiunta una nuova connessione al Pluggable Database di default, accedendovi utilizzando l'utente *SYSTEM*, la cui password associata è stata specificata durante l'installazione della distribuzione (la password di accesso è comunque modificabile anche direttamente da linea di comando, così come è possibile definire una nuova coppia utente-password, ma in tal caso si è fatto uso dell'IDE per la gestione degli utenti)



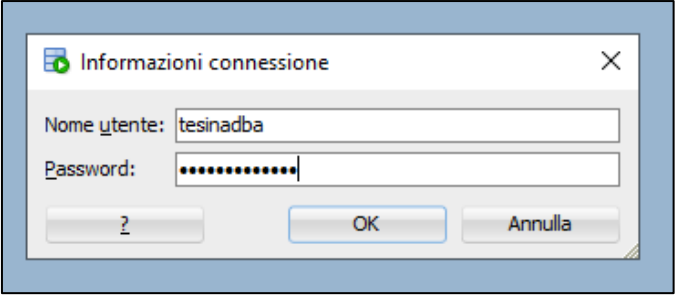
#### NUOVA CONNESSIONE SQL DEVELOPER

Successivamente, attraverso degli appositi comandi, è stato creato un utente (*tesinadba*) al quale è stato consentito l'utilizzo del tablespace "users" per la memorizzazione degli oggetti e dei dati, garantendogli inoltre il ruolo di *database administrator* e fornendogli spazio illimitato sul tablespace. Senza questa operazione, qualsiasi oggetto sarebbe stato creato nel tablespace di sistema, il quale viene automaticamente generato alla creazione della base dati. La sintassi delle istruzioni utilizzate è mostrata nella figura seguente:



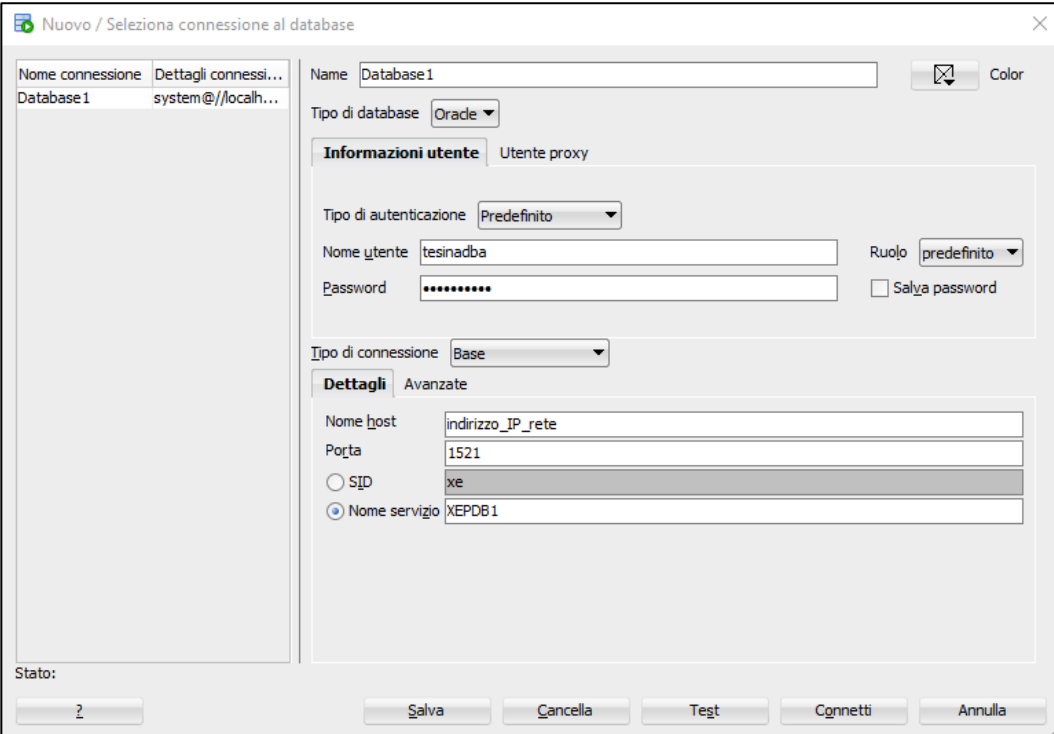
#### CREAZIONE UTENTE DBA

È ora possibile disconnettersi dalla base dati per poi riconnettersi con l'utente appena creato ed iniziare a gestirla.



A dialog box titled "Informazioni connessione" with a close button (X) in the top right corner. It contains two input fields: "Nome utente:" with the text "tesinadba" and "Password:" with masked characters (dots). At the bottom, there are three buttons: a button with a question mark icon, an "OK" button, and an "Annulla" button.

Per consentire ad entrambi i membri del team di connettersi alla base dati anche da remoto, quindi senza utilizzare la macchina sulla quale è installata la distribuzione o una macchina collegata alla rete locale, è stato necessario effettuare una operazione di port-forwarding, in tal caso della 1521. Successivamente al port-forwarding, l'accesso da remoto è stato possibile aggiungendo una nuova connessione ed inserendo come host l'indirizzo IP pubblico della rete alla quale è connessa la macchina che ospita il database.



A complex dialog box titled "Nuovo / Seleziona connessione al database" with a close button (X) in the top right corner. It features a list on the left with columns "Nome connessione" and "Dettagli connessi...", containing one entry "Database1" with details "system@//localh...". The main area is divided into sections: "Name" (Database1), "Tipo di database" (Oracle), "Informazioni utente" (Utente proxy) with fields for "Tipo di autenticazione" (Predefinito), "Nome utente" (tesinadba), "Password" (masked), "Ruolo" (predefinito), and a "Salva password" checkbox; "Tipo di connessione" (Base); and "Dettagli" (Avanzate) with fields for "Nome host" (indirizzo\_IP\_rete), "Porta" (1521), and "Nome servizio" (XE) selected from a list that also includes "SID" and "xe". At the bottom, there are buttons for "Salva", "Cancella", "Test", "Cognetti", and "Annulla". A "Stato:" label is at the bottom left.

## Popolamento della tabella master COVID19

I documenti contenenti i dati necessari al popolamento della tabella master sono stati scaricati tramite il canale GitHub ufficiale della Protezione Civile (<https://github.com/pcm-dpc/COVID-19/tree/master/dati-province>) in formato CSV. Lo schema della relazione è quello stabilito dalla loro documentazione, dunque il comando DDL per la creazione della tabella MASTER è il seguente:

```
CREATE TABLE COVID19
(
data DATE,
stato CHAR(3),
codice_regione NUMBER(2),
denominazione_regione VARCHAR2(200),
codice_provincia NUMBER(3),
denominazione_provincia VARCHAR2(200),
sigla_provincia CHAR(2),
latitudine NUMBER,
longitudine NUMBER,
totale_casi INTEGER,
note_it CLOB,
note_en CLOB,
CONSTRAINT PK_COVID primary key(data,codice_provincia)
);
```

❖ COLUMN_NAME	❖ DATA_TYPE	❖ NULLABLE	DATA_DEFAULT	❖ COLUMN_ID	❖ COMMENTS
1 DATA	DATE	No	(null)	1	(null)
2 STATO	CHAR(3 BYTE)	Yes	(null)	2	(null)
3 CODICE_REGIONE	NUMBER(2,0)	Yes	(null)	3	(null)
4 DENOMINAZIONE_REGIONE	VARCHAR2(200 BYTE)	Yes	(null)	4	(null)
5 CODICE_PROVINCIA	NUMBER(3,0)	No	(null)	5	(null)
6 DENOMINAZIONE_PROVINCIA	VARCHAR2(200 BYTE)	Yes	(null)	6	(null)
7 SIGLA_PROVINCIA	CHAR(2 BYTE)	Yes	(null)	7	(null)
8 LATITUDINE	NUMBER	Yes	(null)	8	(null)
9 LONGITUDINE	NUMBER	Yes	(null)	9	(null)
10 TOTALE_CASI	NUMBER(38,0)	Yes	(null)	10	(null)
11 NOTE_IT	CLOB	Yes	(null)	11	(null)
12 NOTE_EN	CLOB	Yes	(null)	12	(null)

**TABELLA MASTER COVID19, SQL DEVELOPER**

Successivamente è stato necessario convertire i documenti dal formato CSV a comandi DDL (per la conversione si è usufruito del tool online CONVERTCSV.COM, disponibile al link <https://www.convertcsv.com/csv-to-sql.htm>) inserendo il nome scelto per la tabella (COVID19) e un formato della data corretto per il linguaggio. Inoltre, una volta popolata la relazione, tramite l'utilizzo di appositi comandi DML, come vedremo, è stata effettuata la rimozione di alcuni record non necessari e la "pulizia" di valori non corretti su determinati campi di determinati record. (Nel paragrafo dei trigger, è stato mostrato come sia possibile realizzare lo stesso scopo attraverso l'utilizzo di trigger che rendono automatizzato il processo, la cui gestione sarebbe altrimenti compito del dba)



Per il popolamento sono stati utilizzati i dati registrati a partire dal 25 Febbraio 2020 fino al 24 Giugno 2020, i quali, una volta convertiti, sono stati memorizzati in dei file di testo a loro volta collezionati in delle directory in base al mese al quale si riferiscono.

Un esempio di transazione di inserimento in linguaggio SQL è il seguente:

```
INSERT INTO COVID19 VALUES ('25-Feb-2020', 'ITA', 13, 'Abruzzo', 069, 'Chieti', 'CH', 42.35103167, 14.16754574, 0, NULL, NULL);
```

```
INSERT INTO COVID19 VALUES ('25-Feb-2020', 'ITA', 04, 'P.A. Trento', 022, 'Trento', 'TN', 46.06893511, 11.12123097, 0, NULL, NULL);
```

```
INSERT INTO COVID19 VALUES ('25-Feb-2020', 'ITA', 13, 'Abruzzo', 068, 'Pescara', 'PE', 42.46458398, 14.21364822, 0, NULL, NULL);
```

```
INSERT INTO COVID19 VALUES ('25-Feb-2020', 'ITA', 13, 'Abruzzo', 979, 'In fase di definizione/aggiornamento', NULL, 0, 0, 0, NULL, NULL);
```

```
INSERT INTO COVID19 VALUES ('25-Feb-2020', 'ITA', 04, 'P.A. Bolzano', 021, 'Bolzano', 'BZ', 46.49933453, 11.35662422, 1, NULL, NULL);
```

```
..... commit;
```

Come anticipato, con questo piccolo gruppo di istruzioni sono inseriti nella relazione dei record che è necessario modificare o rimuovere, tramite appositi comandi DML (*Data Manipulation Language*): UPDATE e DELETE.

In realtà, all'interno dell'estratto mostrato sono presenti tutti i casi possibili da dover gestire:

- Bolzano e Trento non afferiscono a nessuna regione in quanto sono capoluoghi delle rispettive province autonome, dunque il codice identificativo di regione va modificato ad un valore che tiene conto di questa loro caratteristica. In particolare *codice\_regione* è stato aggiornato da "04" a "98" per Bolzano e da "04" a "99" per Trento.
- I record che hanno il campo *denominazione\_provincia* pari a "in fase di definizione/aggiornamento" sono evidentemente privi di alcuna informazione utile, possono essere dunque cancellati interamente dalla relazione.

Di seguito sono riportate le istruzioni DML che assolvono i seguenti compiti:

```
UPDATE COVID19 SET codice_regione='98' where denominazione_provincia='Bolzano';
UPDATE COVID19 SET codice_regione='99' where denominazione_provincia='Trento';
DELETE FROM COVID19 WHERE denominazione_provincia='In fase di definizione/aggiornamento';
```

19	11-MAR-20	ITA	3 Lombardia	19 Cremona	CR	45,13336675	10,02420865	1061 (null)	(null)
20	11-MAR-20	ITA	3 Lombardia	20 Mantova	MN	45,15726772	10,79277363	137 (null)	(null)
21	11-MAR-20	ITA	98 P.A. Bolzano	21 Bolzano	BZ	46,49933453	11,35662422	75 (null)	(null)
22	11-MAR-20	ITA	99 P.A. Trento	22 Trento	TN	46,06893511	11,12123097	77 (null)	(null)
23	11-MAR-20	ITA	5 Veneto	23 Verona	VR	45,43839046	10,99352685	110 (null)	(null)
24	11-MAR-20	ITA	5 Veneto	24 Vicenza	VI	45,547497	11,54597109	92 (null)	(null)