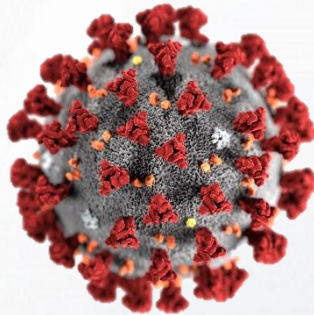




UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

Tesina COVID-19



Conte Alfonso | N46004496

De Iasio Vittorio | N46004331

Corso: Basi di Dati

Professore: Vincenzo Moscato

Anno Accademico 2019/2020

Indice

1.	Introduzione	1
2.	Normalizzazione	2
2.1	Prima Forma Normale.	2
2.2	Chiave e Calcolo delle Dipendenze Funzionali.	2
2.3	Secondo Forma Normale.	3
2.4	Terza Forma Normale.	3
3.	Reverse Engineering	4
3.1	Dal modello-logico al modello E/R	4
3.2	Dal modello E/R classico al modello E/R avanzato	5
4.	Creazione Della Base Dati	7
4.1	Creazione Tabelle Normalizzate	7
4.2	Vincoli di Foreign Key	7
4.3.	Arricchimento dello schema	8
5.	Programmazione DataBase.	9
5.1	Trigger.	9
5.2	Stored Procedure.	12
6.	Popolamento	16
7.	Data Analytics	18
7.1	Query.	18
8.	Ottimizzazione	26
8.1	Viste Materializzate.	26
8.2	Indici.	26
8.3	Esempi di ottimizzazione.	28

Appendice A

❖	Tabella Master	29
---	---------------------	----

1.Introduzione

Si vuole progettare una base di dati per la registrazione dei dati relativi al contagio da COVID-19 in Italia , in particolare, a livello provinciale e regionale.

La base di dati è stata pensata e, infine, progettata per permettere la gestione ed in particolare l'analisi dei dati sul contagio mediante:

- Informazioni raccolte al fine di individuare particolari fattori che potrebbero aver contribuito allo scoppio e alla diffusione della pandemia in Italia.
- Visualizzazione di dati e grafici rappresentanti l'andamento del contagio

I dati contenuti nella base dati fanno riferimento a:

- Caratteristiche e strutture proprie delle singole regioni/province
- Informazioni a livello provinciale/regionale sul contagio e la sua diffusione
- Informazioni statistiche relative al comportamento della popolazione e loro abitudini

In particolare:

I dati relativi al contagio sono stati ricavati dal progetto github al seguente link

- <https://github.com/pcm-dpc/COVID-19/tree/master/dati-province>

da file .csv sono stati successivamente convertiti in istruzioni DML

(Insert,Update) tramite il convertitore al link

- <https://www.convertcsv.com/csv-to-sql.htm>

(NOTA:Essi fanno riferimento al periodo compreso tra il 25/02/2020 e il 03/05/2020)

Mentre, per tutti gli altri dati, sono state prese in considerazione diverse fonti tra le quali :

- <https://www.istat.it/it/>
- <https://lab24.ilsole24ore.com/qualita-della-vita/classifiche-complete.php>

2. Normalizzazione

Con riferimento alla tabella Master (Appendice A) è facile osservare come questa sia una relazione che rappresenti informazioni eterogenee che di norma dovrebbero essere distribuite in più relazioni al fine di ottenere l'immunità dalle diverse anomalie che potrebbero verificarsi all'atto di aggiornamenti, cancellazioni o inserimenti sulla tabella stessa; da qui l'esigenza di effettuare una normalizzazione della suddetta Relazione.

2.1 Prima Forma Normale

I DBMS relazionali mettono a disposizione per la definizione dei domini solo tipi atomici. Sappiamo che il modello relazionale è per sua natura in prima forma normale che dunque risulta automaticamente verificata anche per il nostro schema di relazione:

COVID(data,stato,codice_regione,denominazione_regione,codice_provincia,denominazione_provincia,sigla_provincia, latitudine , longitudine , totale_casi , note_it, note_en)

2.2 Chiave e Calcolo delle dipendenze Funzionali

Come scelta della chiave primaria della Tabella in esame conviene considerare la coppia composta dagli attributi (data,codice_provincia) che consente di individuare in maniera univoca le singole tuple;

$K=\{data,codice_provincia\}$

Non ci resta che individuare le dipendenze funzionali:

- {data, codice_provincia }-> stato , codice_regione, denominazione_regione , denominazione_provincia, sigla_provincia, latitudine , longitudine , totale_casi , note_it, note_en
- codice_regione-> denominazione_regione, Stato
- codice_provincia-> denominazione_provincia, sigla_provincia, latitudine, longitudine, codice_regione, denominazione_regione, Stato

2.3 Seconda Forma Normale

È osservabile dalle dipendenze funzionali trovate che la relazione non verifichi la seconda forma normale data la presenza di attributi non primi che dipendono da parte della chiave stessa per cui occorre operare una decomposizione nelle tabelle:

- *PROVINCE*(codice_provincia, denominazione_provincia, sigla_provincia, latitudine, longitudine, codice_regione, denominazione_regione, Stato)
- *COVID_PROVINCE*(data, codice_provincia:PROVINCE, totale_casi, note_it, note_en)

2.4 Terza Forma Normale

La tabella PROVINCE ottenuta dalla decomposizione in seconda forma normale non verifica la terza forma normale a causa della dipendenza transitiva tra codice_provincia e denominazione_regione,stato:

codice_provincia->codice_regione

codice_regione->denominazione_regione,stato

Occorre allora ancora effettuare una decomposizione ottenendo:

- *COVID_PROVINCE*(data, codice_provincia:PROVINCE, totale_casi, note_it, note_en)
- *REGIONI*(codice_regione, denominazione_regione, Stato)
- *PROVINCE*(codice_provincia, denominazione_provincia, sigla_provincia, latitudine, longitudine, codice_regione:REGIONI)

Al fine di semplificare l'analisi dei Dati sul contagio a livello regionale si è poi deciso di aggiungere un'ulteriore relazione:

- *COVID_REGIONI*(data, codice_regione:REGIONI, totale_casi)

3. Reverse Engineering

3.1 Dal modello-logico al modello E/R

A partire dalle tabelle normalizzate si vuole ricavare con un processo inverso un opportuno schema concettuale(E/R) in grado di rappresentare la realtà in esame.

È immediato ricavare che le Tabelle **REGIONI** e **PROVINCE** derivino da due entità autonome e significative che possiamo chiamare rispettivamente **REGIONE** e **PROVINCIA** i cui identificatori sono le chiavi primarie delle tabelle (rispettivamente *codice_regione* e *codice_provincia*).

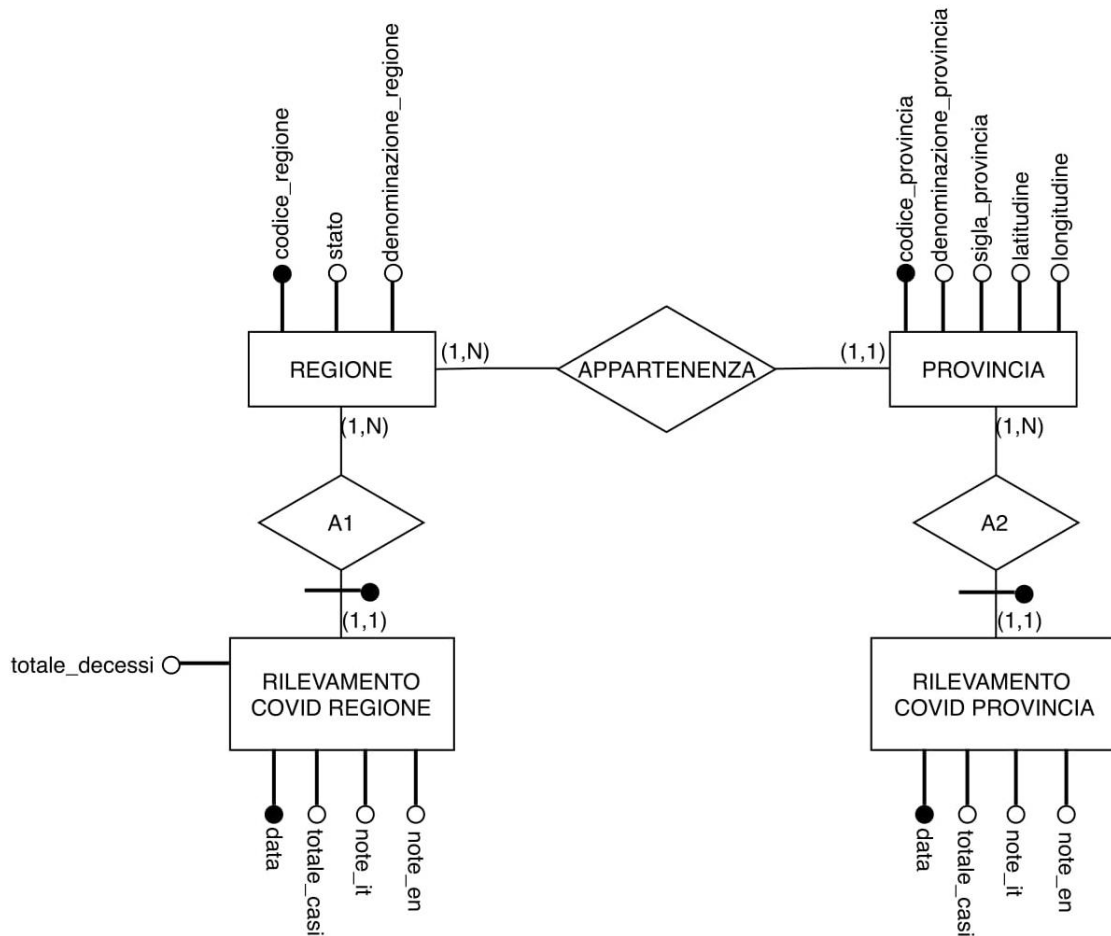
Il vincolo di chiave esterna sull'attributo della Tabella **PROVINCE** (*codice_regione:REGIONI*) ci suggerisce che le due entità corrispondenti siano legate concettualmente da un'associazione(**APPARTENENZA**) con rapporto di cardinalità uno a molti, essendo tra l'altro una regione costituita da una serie di province e una provincia appartenente ad una ed una sola regione.

Le Tabelle **COVID_REGIONI** e **COVID_PROVINCE** contengono al loro interno i rilevamenti dei dati giornalieri sul contagio. Nella nostra realtà di interesse, un rilevamento è a tutti gli effetti un'entità. Da queste considerazioni ricaviamo che le tabelle **COVID_REGIONI** e **COVID_PROVINCE** derivano da due entità che abbiamo chiamato rispettivamente **RILEVAMENTO CONTAGI REGIONE** e **RILEVAMENTO CONTAGI PROVINCIA** i cui identificatori sono le chiavi primarie delle tabelle corrispondenti (rispettivamente *data,codice_regione* e *data,codice_provincia*).

Il vincolo di chiave esterna sull'attributo della Tabella **COVID_PROVINCE** (*codice_provincia:PROVINCE*) ci suggerisce che le due entità corrispondenti siano legate concettualmente da un'associazione (**A1**) con rapporto di cardinalità uno a molti, dato che un rilevamento è associato ad una ed una sola provincia e ad una provincia sono associati da 1 ad N rilevamenti (uno per ogni giorno). Inoltre *codice_provincia* fa anche parte della chiave primaria di **COVID_PROVINCE** (insieme alla *data*) e ciò ci fa capire che l'entità **RILEVAMENTO CONTAGI PROVINCIA** è identificata, oltre che dalla *data*, esternamente dall'attributo *codice_provincia* di **PROVINCIA**.

Analogo ragionamento va fatto per l'associazione **A2** tra **REGIONE** e **RILEVAMENTO CONTAGI REGIONE**.

Da queste considerazioni ricaviamo il seguente schema E/R:

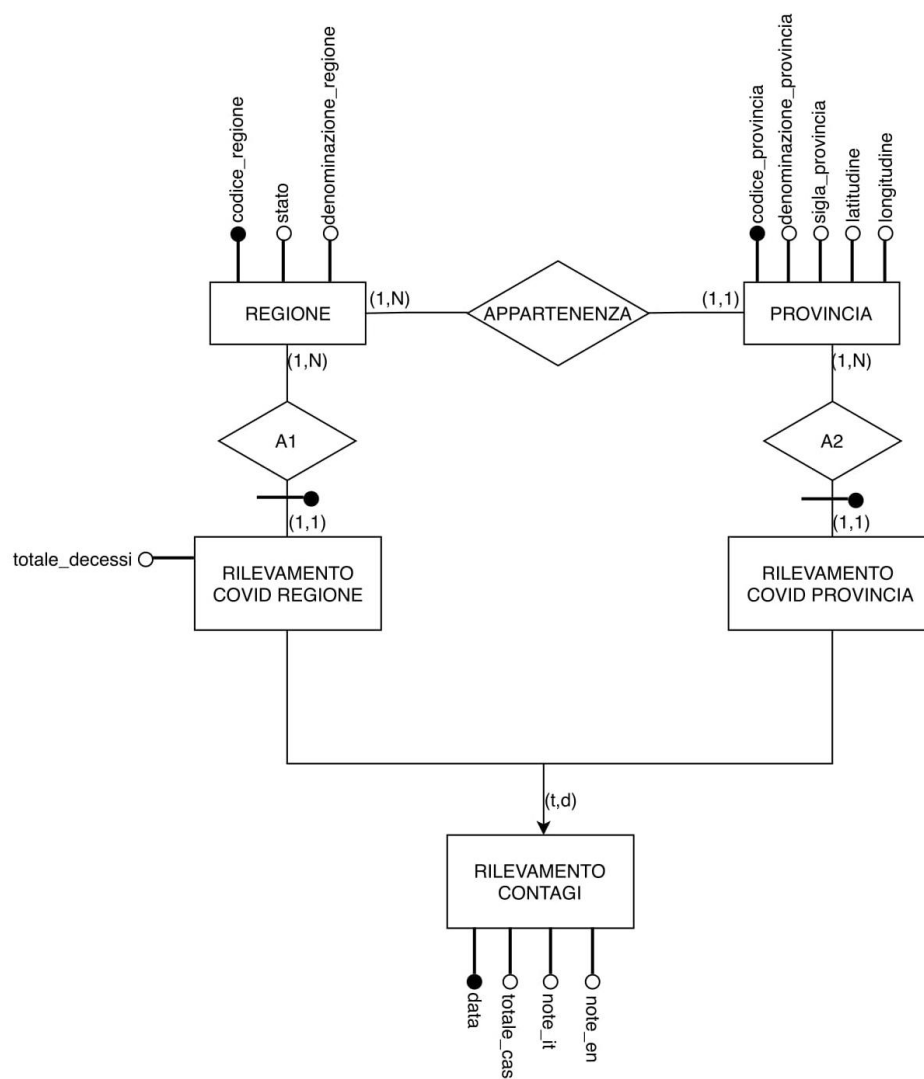


3.2 Dal modello E/R classico al modello E/R avanzato

Notiamo che le entità **RILEVAMENTO CONTAGI REGIONE** e **RILEVAMENTO CONTAGI PROVINCIA** hanno alcuni attributi in

comune (data,totale_casi, ...) e per questo possono essere viste come specializzazioni di una singola entità padre che abbiamo chiamato **RILEVAMENTO CONTAGI**. La gerarchia è inoltre di tipo totale (non esistono rilevamenti di contagi che non siano relativi ad una provincia o ad una regione nella nostra realtà di interesse) e disgiunta (un rilevamento di contagi non può essere contemporaneamente relativo ad una regione e ad una provincia).

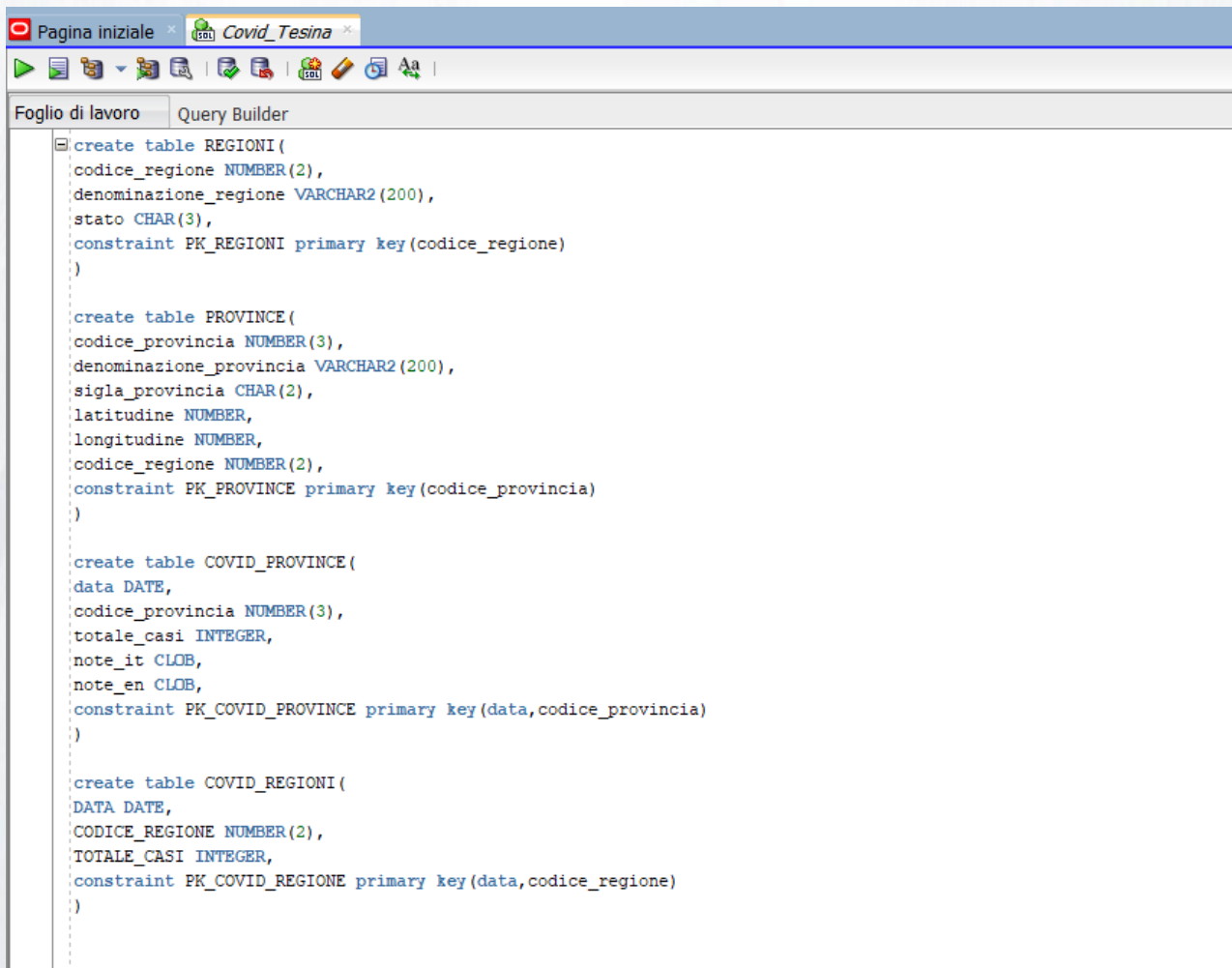
Da ciò deriva il seguente schema E/R avanzato:



4. Creazione della Base Dati

4.1 Creazione delle Tabelle Normalizzate

Di seguito sono riportati i comandi DDL per la creazione delle tabelle PROVINCE, REGIONI, COVID_PROVINCE, COVID_REGIONI



```
create table REGIONI (
  codice_regione NUMBER(2),
  denominazione_regione VARCHAR2(200),
  stato CHAR(3),
  constraint PK_REGIONI primary key (codice_regione)
)

create table PROVINCE (
  codice_provincia NUMBER(3),
  denominazione_provincia VARCHAR2(200),
  sigla_provincia CHAR(2),
  latitudine NUMBER,
  longitudine NUMBER,
  codice_regione NUMBER(2),
  constraint PK_PROVINCE primary key (codice_provincia)
)

create table COVID_PROVINCE (
  data DATE,
  codice_provincia NUMBER(3),
  totale_casi INTEGER,
  note_it CLOB,
  note_en CLOB,
  constraint PK_COVID_PROVINCE primary key (data, codice_provincia)
)

create table COVID_REGIONI (
  data DATE,
  CODICE_REGIONE NUMBER(2),
  TOTALE_CASI INTEGER,
  constraint PK_COVID_REGIONE primary key (data, codice_regione)
)
```

4.2 Vincoli di Foreign Key

A seguito della creazione delle tabelle possiamo aggiungere i vincoli di chiave esterna, secondo quanto ricavato dalla normalizzazione, mediante l'apposito comando DDL: ALTER TABLE

```
alter table COVID_PROVINCE
add constraint FK_COVIDPROV_PROVINCE
foreign key (codice_provincia) references PROVINCE(codice_provincia) on delete cascade ;

alter table PROVINCE
add constraint FK_PROVINCE_REGIONI
foreign key (codice_regioni) references REGIONI(codice_regioni) on delete cascade ;

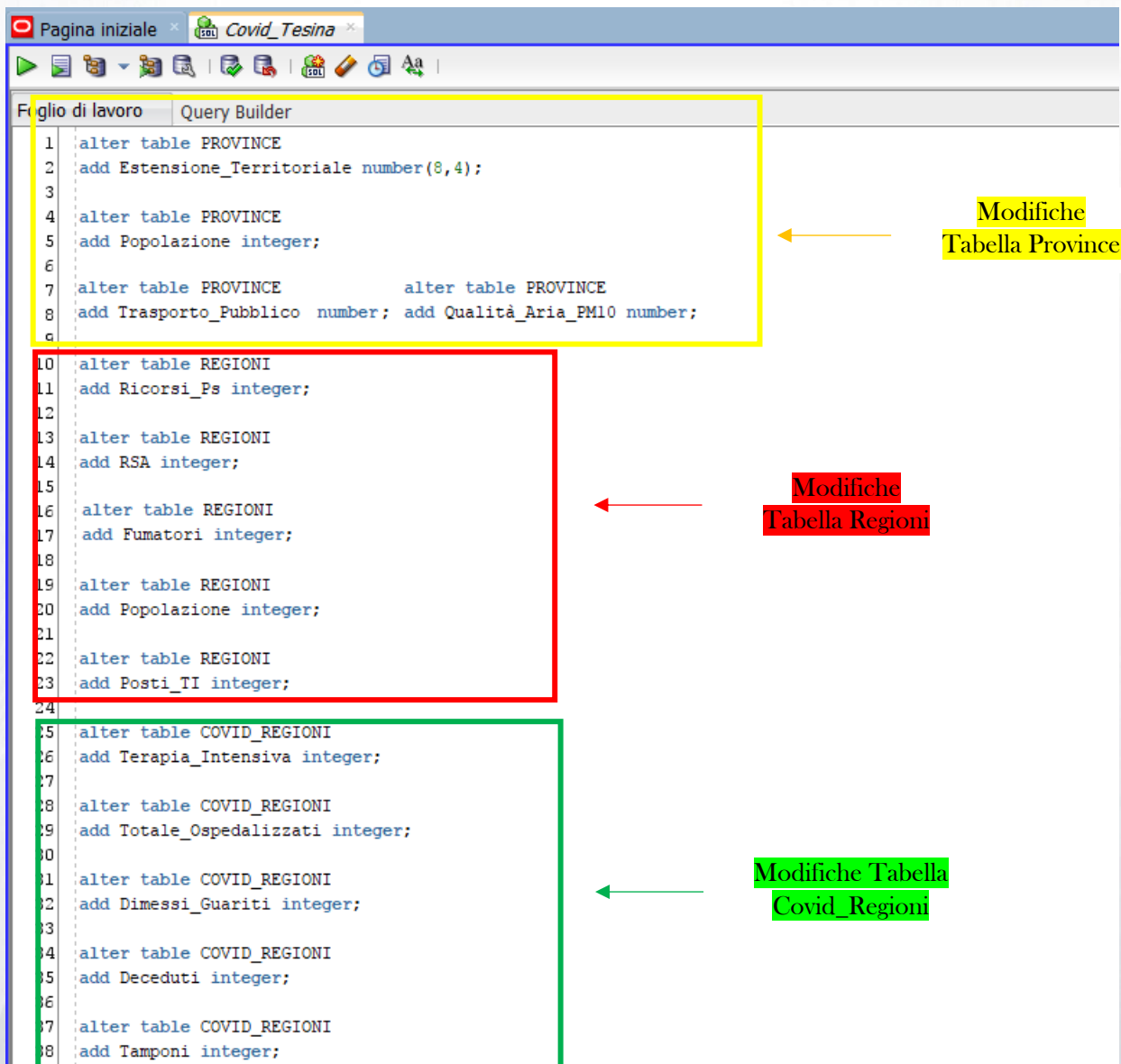
alter table COVID_REGIONI
add constraint FK_COVIDREG_REGIONI
foreign key (codice_regioni) references REGIONI(codice_regioni) on delete cascade;
```

Si noti come l'aggiunta dei vincoli di chiave esterna sia avvenuta solo in seguito alla creazione di tutte le tabelle al fine di evitare la preventiva creazione delle tabelle referenziate a quella delle tabelle referenzianti; Inoltre vengono aggiunte le politiche di reazione *'on delete cascade'* a valle di operazioni di cancellazione sulle tabelle referenziate di modo che l'operazione si propaghi sulle tabelle referenzianti.

4.3 Arricchimento dello schema

Per un'analisi più dettagliata del fenomeno del contagio si è deciso di arricchire le Tabelle con ulteriori dati provenienti da diverse fonti riguardanti sia l'andamento del contagio (deceduti, guariti ...) sia informazioni proprie di regioni e province (popolazione, estensione_territoriale, numero_RSA ...)

Si riportano di seguito i comandi di **ALTER TABLE** necessari all'aggiunta degli attributi alle diverse tabelle:



5. Programmazione DataBase

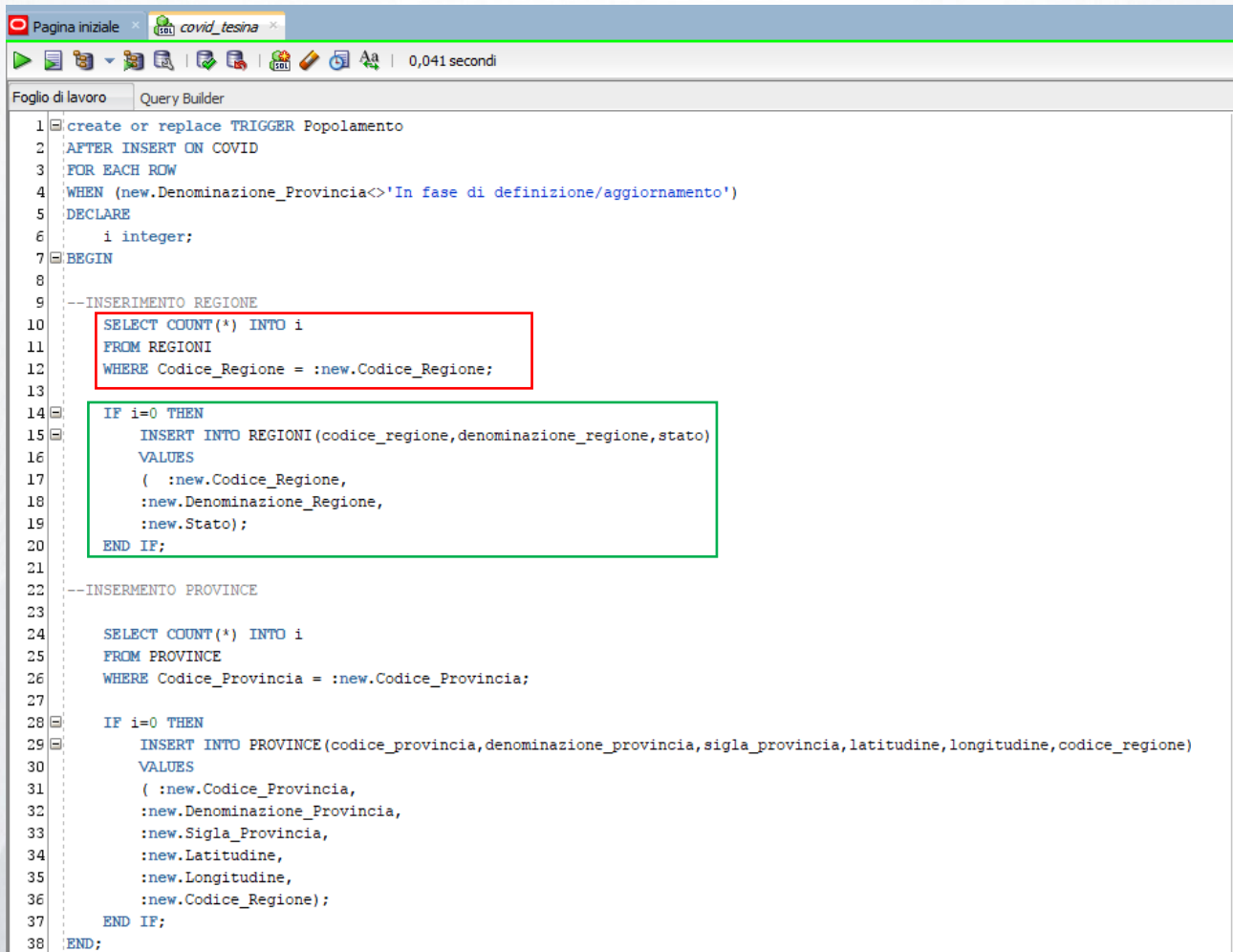
Una volta creata la base dati, è possibile definire procedure, funzioni e meccanismi automatici in grado di operare su di essa restituendo risultati utili all'analisi dei dati sul contagio, o effettuando vere e proprie operazioni di manipolazione sui dati così come programmato.

5.1 Trigger

I trigger utilizzati nella nostra base dati hanno tutti come evento di innesco un inserimento nella Tabella Master. In particolare due di essi hanno una

modalità di attivazione subito dopo l'inserimento mentre l'ultimo viene eseguito poco prima della stessa operazione.

Trigger 'POPOLAMENTO'



```
1 create or replace TRIGGER Popolamento
2 AFTER INSERT ON COVID
3 FOR EACH ROW
4 WHEN (new.Denominazione_Provincia<>'In fase di definizione/aggiornamento')
5 DECLARE
6     i integer;
7 BEGIN
8
9 --INSERIMENTO REGIONE
10 SELECT COUNT(*) INTO i
11 FROM REGIONI
12 WHERE Codice_Regione = :new.Codice_Regione;
13
14 IF i=0 THEN
15     INSERT INTO REGIONI(codice_regione,denominazione_regione,stato)
16     VALUES
17     ( :new.Codice_Regione,
18      :new.Denominazione_Regione,
19      :new.Stato);
20 END IF;
21
22 --INSERIMENTO PROVINCE
23
24 SELECT COUNT(*) INTO i
25 FROM PROVINCE
26 WHERE Codice_Provincia = :new.Codice_Provincia;
27
28 IF i=0 THEN
29     INSERT INTO PROVINCE(codice_provincia,denominazione_provincia,sigla_provincia,latitudine,longitudine,codice_regione)
30     VALUES
31     ( :new.Codice_Provincia,
32      :new.Denominazione_Provincia,
33      :new.Sigla_Provincia,
34      :new.Latitudine,
35      :new.Longitudine,
36      :new.Codice_Regione);
37 END IF;
38 END;
```

Il seguente trigger è stato realizzato per permettere il popolamento automatico delle tabelle **REGIONI** e **PROVINCE** a valle delle insert sulla Tabella target **COVID**. In particolare il trigger, di granularità *row level*, controlla tramite la clausola **WHEN** che la tupla inserita abbia un valore dell'attributo *denominazione_provincia* valido (controllo necessario per come sono stati forniti i dati).

Il primo blocco di istruzioni (rettangolo **rosso** in figura) controlla se la regione nella tupla inserita sia già presente nella tabella **REGIONI**. Se ciò non avviene, il valore della variabile *i* sarà posto uguale a 0 ed a 1 in caso contrario.

Il secondo blocco di istruzioni (rettangolo **verde** in figura) controlla il valore della variabile *i* e, se pari a 0, effettua l'inserimento della regione nella corrispondente tabella.

Analogo ragionamento va fatto per l'inserimento delle province nella tabella PROVINCE.

Trigger 'INSERT_COVID'



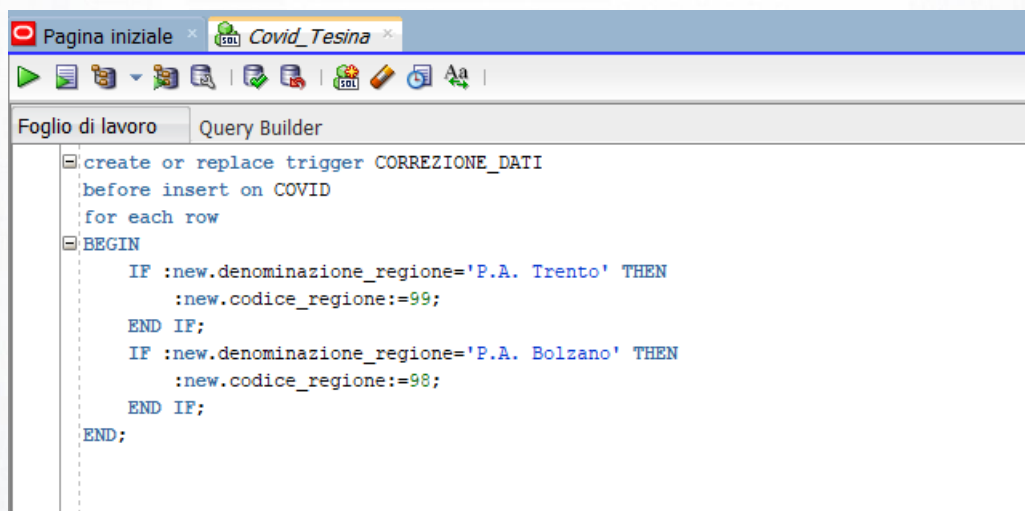
```
1 create or replace trigger insert_covid
2 after insert on COVID
3 for each row
4 when (new.Denominazione_Provincia<>'In fase di definizione/aggiornamento')
5 DECLARE
6     conta integer;
7 BEGIN
8     insert into covid_province values (:new.data, :new.codice_provincia, :new.totale_casi, :new.note_it, :new.note_en);
9     select COUNT(*) into conta
10    from COVID_REGIONI
11   where data=:new.data and codice_regione=:new.codice_regione;
12 IF conta=0 THEN
13     insert into covid_regioni(data,codice_regione,totale_casi) values (:new.data, :new.codice_regione, :new.totale_casi);
14 ELSE
15     update COVID_REGIONI
16    set totale_casi = totale_casi + :new.totale_casi
17   where data=:new.data and codice_regione=:new.codice_regione;
18 END IF;
19 END;
```

Il seguente trigger è stato realizzato per permettere il popolamento automatico delle tabelle COVID_REGIONI e COVID_PROVINCE a valle delle insert sulla Tabella target COVID.

La prima istruzione che esegue il trigger è un inserimento nella tabella COVID_PROVINCE dei dati appena inseriti relativi al contagio a livello provinciale.

Dopodiché il trigger controlla, attraverso il valore della variabile *conta*, se è già presente nella tabella COVID_REGIONI una tupla, avente lo stesso valore di quella inserita, sugli attributi *codice_regione* e *data*; in caso negativo, il valore della variabile *conta* sarà posto a 0 e sarà effettuata una insert sulla tabella COVID_REGIONI; altrimenti sarà effettuato un'aggiornamento di tale tupla incrementando l'attributo *totale_casi*.

Trigger 'CORREZIONE_DATI'

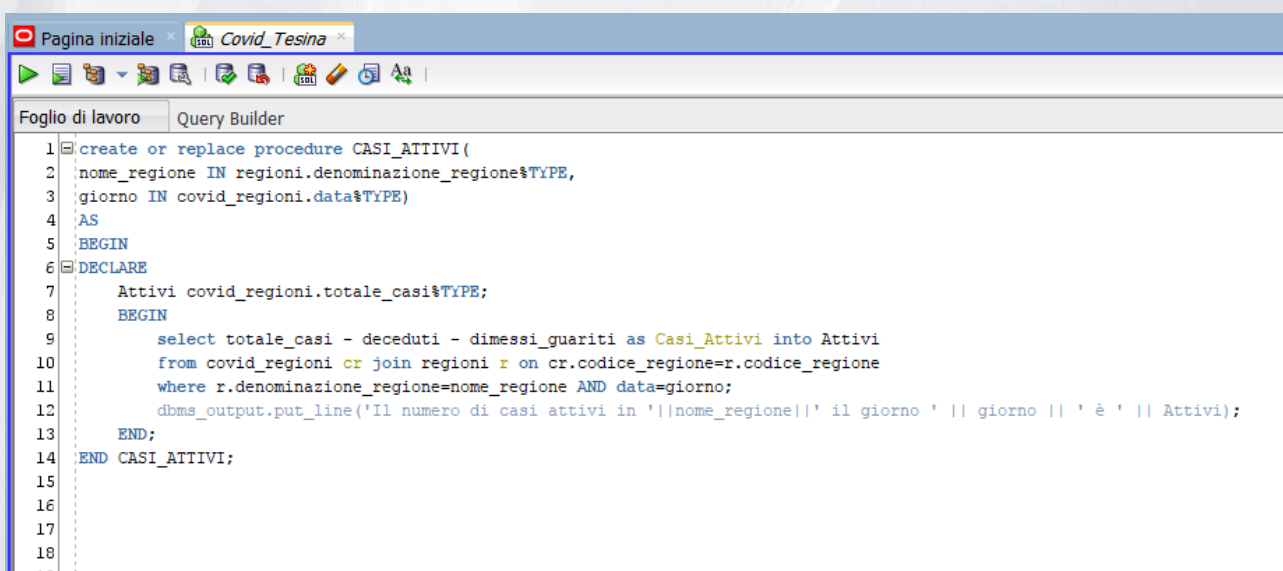


```
create or replace trigger CORREZIONE_DATI
before insert on COVID
for each row
begin
    IF :new.denominazione_regione='P.A. Trento' THEN
        :new.codice_region:=99;
    END IF;
    IF :new.denominazione_regione='P.A. Bolzano' THEN
        :new.codice_region:=98;
    END IF;
end;
```

Il seguente trigger è stato realizzato al fine di evitare l'incongruenza dei dati che si sarebbe riscontrata per la presenza di due regioni aventi denominazioni diverse ma lo stesso *codice_region* violando il vincolo di chiave primaria nella tabella REGIONI.

5.2 Stored Procedure

Procedure 'CASI_ATTIVI'



```
1 create or replace procedure CASI_ATTIVI(
2   nome_regione IN regioni.denominazione_regione%TYPE,
3   giorno IN covid_regioni.data%TYPE)
4 AS
5 BEGIN
6 DECLARE
7   Attivi covid_regioni.totale_casi%TYPE;
8 BEGIN
9   select totale_casi - deceduti - dimessi_guariti as Casi_Activi into Attivi
10  from covid_regioni cr join regioni r on cr.codice_region=r.codice_region
11  where r.denominazione_regione=nome_regione AND data=giorno;
12  dbms_output.put_line('Il numero di casi attivi in '||nome_regione||' il giorno ' || giorno || ' è ' || Attivi);
13 END;
14 END CASI_ATTIVI;
```

La seguente procedura, presa in ingresso la denominazione della regione ed una particolare data, non fa altro che restituire, stampandolo a video, il numero di casi positivi attivi in quella specifica data.

Procedure 'PICCO_ATTIVI'



```
1 create or replace procedure PICCO_ATTIVI(  
2 regione IN regioni.denominazione_regione%TYPE)  
3 AS  
4 BEGIN  
5 DECLARE  
6 PICCO covid_regioni.totale_casi%TYPE;  
7 GIORNO covid_regioni.data%TYPE;  
8 BEGIN  
9 select max(cr.totale_casi-cr.dimessi_guariti-cr.deceduti) into PICCO  
10 from covid_regioni cr join regioni r on cr.codice_regione=r.codice_regione  
11 group by r.denominazione_regione  
12 having r.denominazione_regione=regione;  
13 select data into GIORNO  
14 from covid_regioni  
15 where totale_casi-dimessi_guariti-deceduti = PICCO;  
16 dbms_output.put_line('Il picco di casi attivi per la regione '||regione||' è '||PICCO|| ', registrato in data '||GIORNO);  
17 END;  
18 END PICCO_ATTIVI;
```

Tale procedura accetta come parametro di input `denominazione_regione` e si struttura in due query. La prima ricava il massimo dei Casi_attivi per la regione specificata ponendone il valore nella variabile `PICCO`. La seconda, sulla base di tale valore, ricerca la data effettiva in cui si è avuto il maggior numero di positivi e ne pone il valore in `GIORNO`. Infine i risultati ottenuti vengono stampati a video.

Procedure 'PICCO_ATTIVI' (Altra Versione)

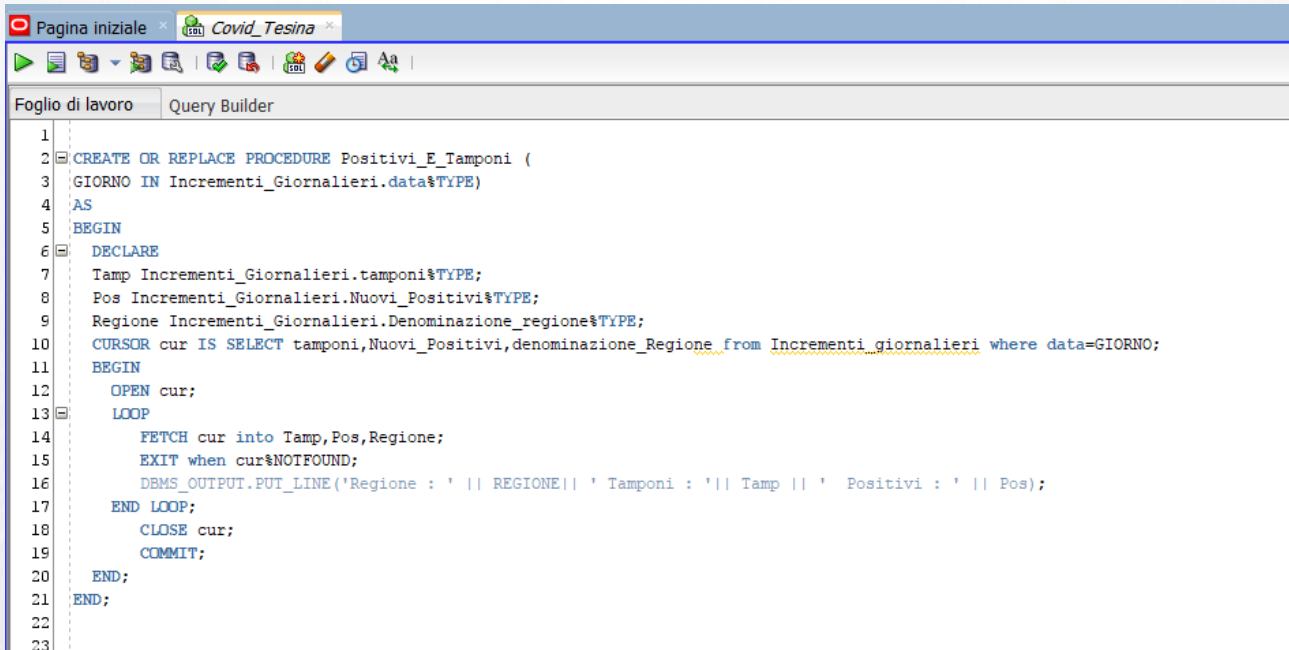


```
1 create or replace procedure PICCO_ATTIVI_REGIONE(  
2 regione IN regioni.denominazione_regione%TYPE)  
3 AS  
4 BEGIN  
5 DECLARE  
6 PICCO covid_regioni.totale_casi%TYPE;  
7 GIORNO covid_regioni.data%TYPE;  
8 BEGIN  
9 select cr.data, cr.totale_casi-cr.dimessi_guariti-cr.deceduti INTO GIORNO,PICCO  
10 from covid_regioni cr join regioni r on cr.codice_regione=r.codice_regione  
11 where r.denominazione_regione=regione and cr.totale_casi-cr.dimessi_guariti-cr.deceduti = (  
12 select max(crl.totale_casi-crl.dimessi_guariti-crl.deceduti)  
13 from covid_regioni crl join regioni rl on crl.codice_regione=rl.codice_regione  
14 group by rl.denominazione_regione  
15 having rl.denominazione_regione=regione);  
16 dbms_output.put_line('Il picco di casi attivi per la regione '||regione||' è '||PICCO|| ', registrato in data '||GIORNO);  
17 END;  
18 END PICCO_ATTIVI_REGIONE;
```

La procedura precedente può essere realizzata anche in un'altra versione.

È infatti possibile utilizzare, equivalentemente, una query nidificata per esprimere la condizione che consente di selezionare, per la regione specificata, il Picco dei casi attivi e il Giorno in cui esso si è verificato.

Procedure 'POSITIVI_E_TAMPONI'



```
1
2 CREATE OR REPLACE PROCEDURE Positivi_E_Tamponi (
3   GIORNO IN Incrementi_Giornalieri.data%TYPE)
4 AS
5 BEGIN
6   DECLARE
7     Tamp Incrementi_Giornalieri.tamponi%TYPE;
8     Pos Incrementi_Giornalieri.Nuovi_Positivi%TYPE;
9     Regione Incrementi_Giornalieri.Denominazione_regione%TYPE;
10    CURSOR cur IS SELECT tamponi,Nuovi_Positivi,denominazione_Regione from Incrementi_giornalieri where data=GIORNO;
11  BEGIN
12    OPEN cur;
13    LOOP
14      FETCH cur into Tamp,Pos,Regione;
15      EXIT when cur%NOTFOUND;
16      DBMS_OUTPUT.PUT_LINE('Regione : ' || REGIONE|| ' Tamponi : '|| Tamp || ' Positivi : ' || Pos);
17    END LOOP;
18    CLOSE cur;
19    COMMIT;
20  END;
21 END;
```

La seguente procedura accetta come parametri di input una particolare Data, sulla base della quale stampa a video la situazione delle regioni italiane relativamente al numero di tamponi fatti e ai positivi registrati in quel particolare giorno.

Da notare l'utilizzo di un cursore per gestire l'insieme di tuple ritornate dalla query impiegata.

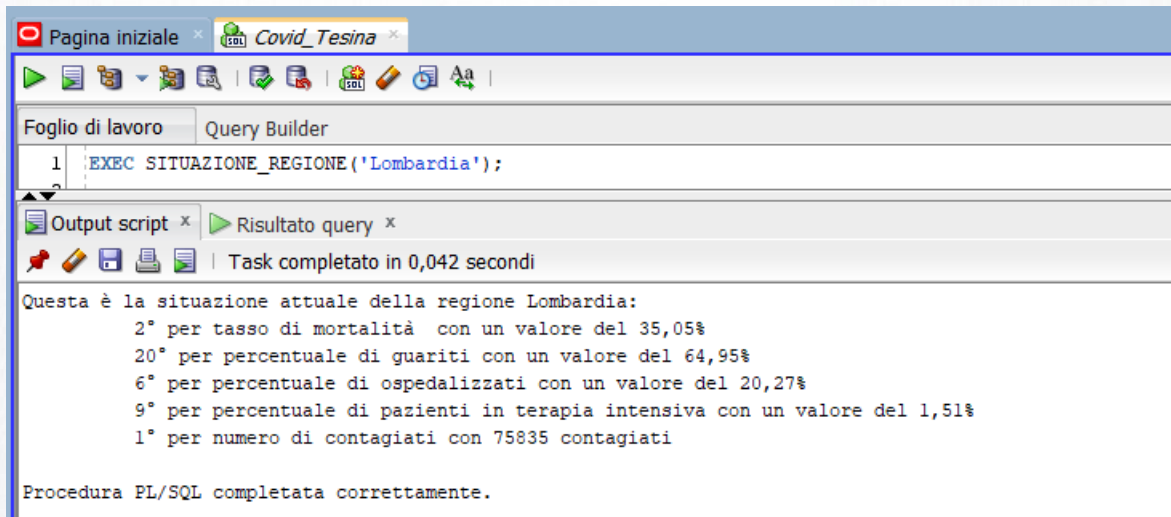
Procedure 'SITUAZIONE_REGIONE'

```
Pagina iniziale covid_tesina
Foglio di lavoro Query Builder

1 create or replace PROCEDURE SITUAZIONE_REGIONE(
2 nome_regione regioni.denominazione_regione%TYPE
3 )
4 AS
5 BEGIN
6     DECLARE
7         nome_regioni.denominazione_regione%TYPE;
8         i1 integer:=0; i2 integer:=0; i3 integer:=0; i4 integer:=0; i5 integer:=0;
9         cursor c1 is
10             select denominazione_regione,ROUND((deceduti/(deceduti+dimessi_guariti))*100,2)
11             from VISTA_REGIONI
12             order by ROUND((deceduti/(deceduti+dimessi_guariti))*100,2) DESC;
13         cursor c2 is
14             select denominazione_regione,ROUND((dimessi_guariti/(deceduti+dimessi_guariti))*100,2)
15             from VISTA_REGIONI
16             order by ROUND((dimessi_guariti/(deceduti+dimessi_guariti))*100,2) DESC;
17         cursor c3 is
18             select denominazione_regione,ROUND((totale_ospedalizzati/(totale_casi-dimessi_guariti-deceduti))*100,2)
19             from VISTA_REGIONI
20             order by ROUND((totale_ospedalizzati/(totale_casi-dimessi_guariti-deceduti))*100,2) DESC;
21         cursor c4 is
22             select denominazione_regione,ROUND((terapia_intensiva/(totale_casi-dimessi_guariti-deceduti))*100,2)
23             from VISTA_REGIONI
24             order by ROUND((terapia_intensiva/(totale_casi-dimessi_guariti-deceduti))*100,2) DESC;
25         cursor c5 is
26             select denominazione_regione,totale_casi
27             from VISTA_REGIONI
28             order by totale_casi DESC;
29         dato1 number; dato2 number; dato3 number; dato4 number; dato5 number;
30         nome_sbagliato exception;
31         controllo integer;
32     BEGIN
33         --controllo valore valido regione
34         select COUNT(*) into controllo from regioni where denominazione_regione=nome_regione;
35         IF controllo=0 THEN
36             RAISE nome_sbagliato;
37         END IF;
38         --Percentuale deceduti
39         OPEN c1;
40         LOOP
41             FETCH c1 INTO nome,dato1;
42             i1 := i1 + 1;
43             EXIT WHEN nome=nome_regione;
44             END LOOP;
45         close c1;
46         --Percentuale guariti
47         OPEN c2;
48         LOOP
49             FETCH c2 INTO nome,dato2;
50             i2 := i2 + 1;
51             EXIT WHEN nome=nome_regione;
52             END LOOP;
53         close c2;
54         --Percentuale ospedalizzati
55         OPEN c3;
56         LOOP
57             FETCH c3 INTO nome,dato3;
58             i3 := i3 + 1;
59             EXIT WHEN nome=nome_regione;
60             END LOOP;
61         close c3;
62         --Percentuale terapia intensiva
63         OPEN c4;
64         LOOP
65             FETCH c4 INTO nome,dato4;
66             i4 := i4 + 1;
67             EXIT WHEN nome=nome_regione;
68             END LOOP;
69         close c4;
70         --Totale casi
71         OPEN c5;
72         LOOP
73             FETCH c5 INTO nome,dato5;
74             i5 := i5 + 1;
75             EXIT WHEN nome=nome_regione;
76             END LOOP;
77         close c5;
78         dbms_output.put_line('Questa è la situazione attuale della regione '||nome_regione||': ');
79         dbms_output.put_line('    ||i1||' per tasso di mortalità con un valore del '||dato1||'&');
80         dbms_output.put_line('    ||i2||' per percentuale di guariti con un valore del '||dato2||'&');
81         dbms_output.put_line('    ||i3||' per percentuale di ospedalizzati con un valore del '||dato3||'&');
82         dbms_output.put_line('    ||i4||' per percentuale di pazienti in terapia intensiva con un valore del '||dato4||'&');
83         dbms_output.put_line('    ||i5||' per numero di contagiati con '||dato5||' contagiati');
84     EXCEPTION WHEN nome_sbagliato THEN
85         dbms_output.put_line('Nome regione inserito non valido. ');
86     END;
87 END SITUAZIONE_REGIONE;
```


La seguente procedura prende in ingresso la denominazione della regione . Essa fornisce un quadro complessivo della regione specificata sul fenomeno del contagio confrontando i valori di alcuni parametri con quelli di tutte le altre regioni.

Di seguito viene riportato un esempio di output per la regione 'Lombardia'



The screenshot shows the 'Covid_Tesina' application window. The 'Query Builder' tab is active, displaying the SQL command: `EXEC SITUAZIONE_REGIONE('Lombardia');`. Below the query, the 'Output script' tab shows the results of the procedure execution. The output text is as follows:

```
Questa è la situazione attuale della regione Lombardia:
2° per tasso di mortalità con un valore del 35,05%
20° per percentuale di guariti con un valore del 64,95%
6° per percentuale di ospedalizzati con un valore del 20,27%
9° per percentuale di pazienti in terapia intensiva con un valore del 1,51%
1° per numero di contagiati con 75835 contagiati

Procedura PL/SQL completata correttamente.
```

6. Popolamento

Sulla base di quanto esposto fino ad ora, possiamo finalmente mostrare come la nostra base dati è stata popolata in maniera del tutto automatizzata a partire dagli inserimenti nella tabella Master e facendo ricorso in particolar modo alle procedure automatiche definite (paragrafo 5.1).

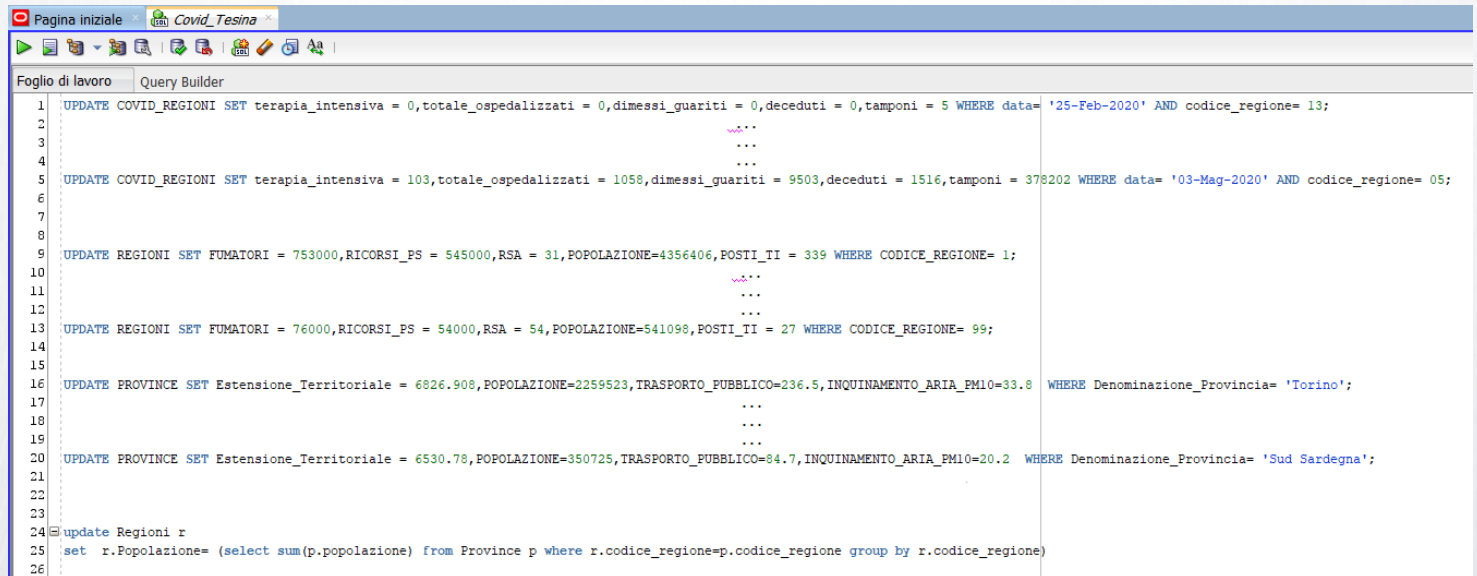


The screenshot shows the 'Covid_Tesina' application window with the 'Query Builder' tab active. The SQL script being entered is as follows:

```
INSERT INTO COVID VALUES ('25-Feb-2020','ITA',13,'Abruzzo',069,'Chieti','CH',42.35103167,14.16754574,0,NULL,NULL);
INSERT INTO COVID VALUES ('25-Feb-2020','ITA',13,'Abruzzo',066,'L'Aquila','AQ',42.35122196,13.39843823,0,NULL,NULL);
...
INSERT INTO COVID VALUES ('03-mag-2020','ITA',05,'Veneto',024,'Vicenza','VI',45.547497,11.54597109,2713,NULL,NULL);
INSERT INTO COVID VALUES ('03-mag-2020','ITA',05,'Veneto',999,'In fase di definizione/aggiornamento',NULL,0,0,325,NULL,NULL);
COMMIT;
```

A seguito di queste insert, le tabelle PROVINCE, REGIONI, COVID_PROVINCE e COVID_REGIONI sono state automaticamente riempite grazie all'innesco dei trigger POPOLAMENTO e INSERT_COVID.

Infine non resta che inserire i valori degli attributi aggiunti tramite i comandi di alter table nella fase di arricchimento della base dati (paragrafo 4.3).



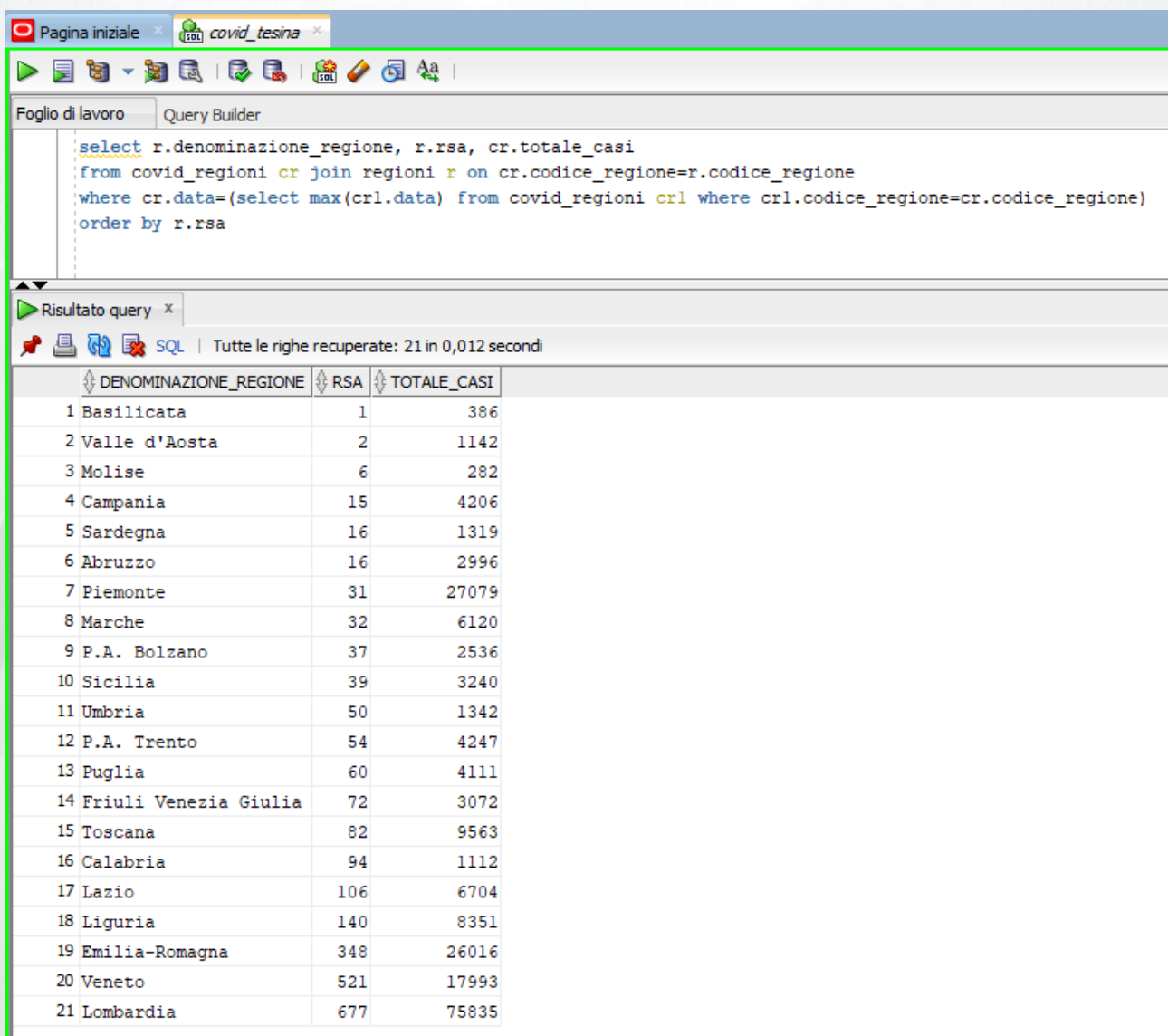
```
1 UPDATE COVID_REGIONI SET terapia_intensiva = 0,totale_ospedalizzati = 0,dimessi_guariti = 0,deceduti = 0,tamponi = 5 WHERE data= '25-Feb-2020' AND codice_regione= 13;
2
3
4
5 UPDATE COVID_REGIONI SET terapia_intensiva = 103,totale_ospedalizzati = 1058,dimessi_guariti = 9503,deceduti = 1516,tamponi = 378202 WHERE data= '03-Mag-2020' AND codice_regione= 05;
6
7
8
9 UPDATE REGIONI SET FUMATORI = 753000,RICORSI_PS = 545000,RSA = 31,POPOLAZIONE=4356406,POSTI_TI = 339 WHERE CODICE_REGIONE= 1;
10
11
12
13 UPDATE REGIONI SET FUMATORI = 76000,RICORSI_PS = 54000,RSA = 54,POPOLAZIONE=541098,POSTI_TI = 27 WHERE CODICE_REGIONE= 99;
14
15
16 UPDATE PROVINCE SET Estensione_Territoriale = 6826.908,POPOLAZIONE=2259523,TRASPORTO_PUBBLICO=236.5,INQUINAMENTO_ARIA_PM10=33.8 WHERE Denominazione_Provincia= 'Torino';
17
18
19
20 UPDATE PROVINCE SET Estensione_Territoriale = 6530.78,POPOLAZIONE=350725,TRASPORTO_PUBBLICO=84.7,INQUINAMENTO_ARIA_PM10=20.2 WHERE Denominazione_Provincia= 'Sud Sardegna';
21
22
23
24 update Regioni r
25 set r.Popolazione= (select sum(p.popolazione) from Province p where r.codice_regione=p.codice_regione group by r.codice_regione)
26
```

7.Data Analytics

Arrivati a questo punto, con una base dati correttamente popolata, disponiamo di tutti gli strumenti che ci consentono di ricavare informazioni utili ad analizzare, in maniera dettagliata, gli aspetti del contagio attraverso opportune query.

7.1 Query

Query N.1



Query Builder

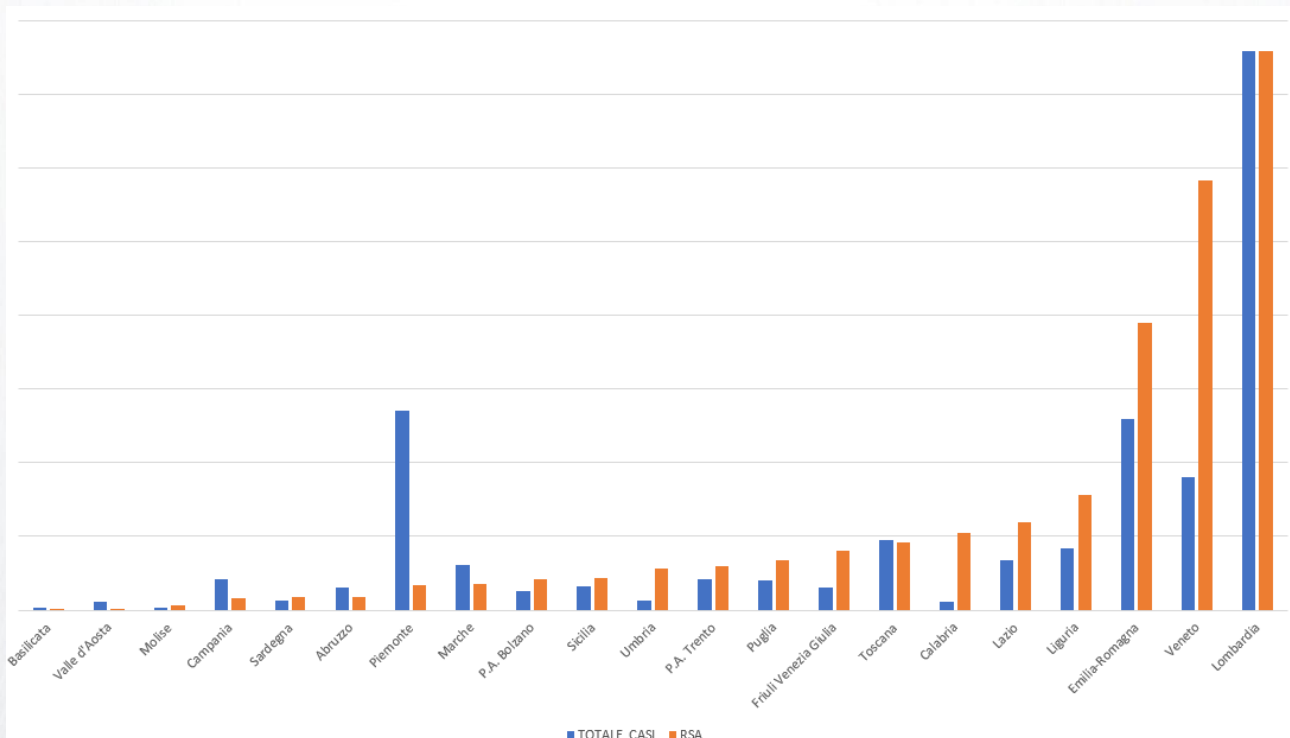
```
select r.denominazione_regione, r.rsa, cr.totale_casi
from covid_regions cr join regioni r on cr.codice_regione=r.codice_regione
where cr.data=(select max(crl.data) from covid_regions crl where crl.codice_regione=cr.codice_regione)
order by r.rsa
```

Risultato query x

Tutte le righe recuperate: 21 in 0,012 secondi

	DENOMINAZIONE_REGIONE	RSA	TOTALE_CASI
1	Basilicata	1	386
2	Valle d'Aosta	2	1142
3	Molise	6	282
4	Campania	15	4206
5	Sardegna	16	1319
6	Abruzzo	16	2996
7	Piemonte	31	27079
8	Marche	32	6120
9	P.A. Bolzano	37	2536
10	Sicilia	39	3240
11	Umbria	50	1342
12	P.A. Trento	54	4247
13	Puglia	60	4111
14	Friuli Venezia Giulia	72	3072
15	Toscana	82	9563
16	Calabria	94	1112
17	Lazio	106	6704
18	Liguria	140	8351
19	Emilia-Romagna	348	26016
20	Veneto	521	17993
21	Lombardia	677	75835

La Query seguente visualizza per ogni regione il numero di RSA presenti e il totale dei casi di COVID-19.



Sulla base del seguente grafico, sembrerebbe che l'ipotesi delle RSA come luoghi di diffusione del contagio possa essere in parte confermata in quanto le regioni con un maggior numero di RSA hanno presentato un numero di casi totali superiore.

Query N.2

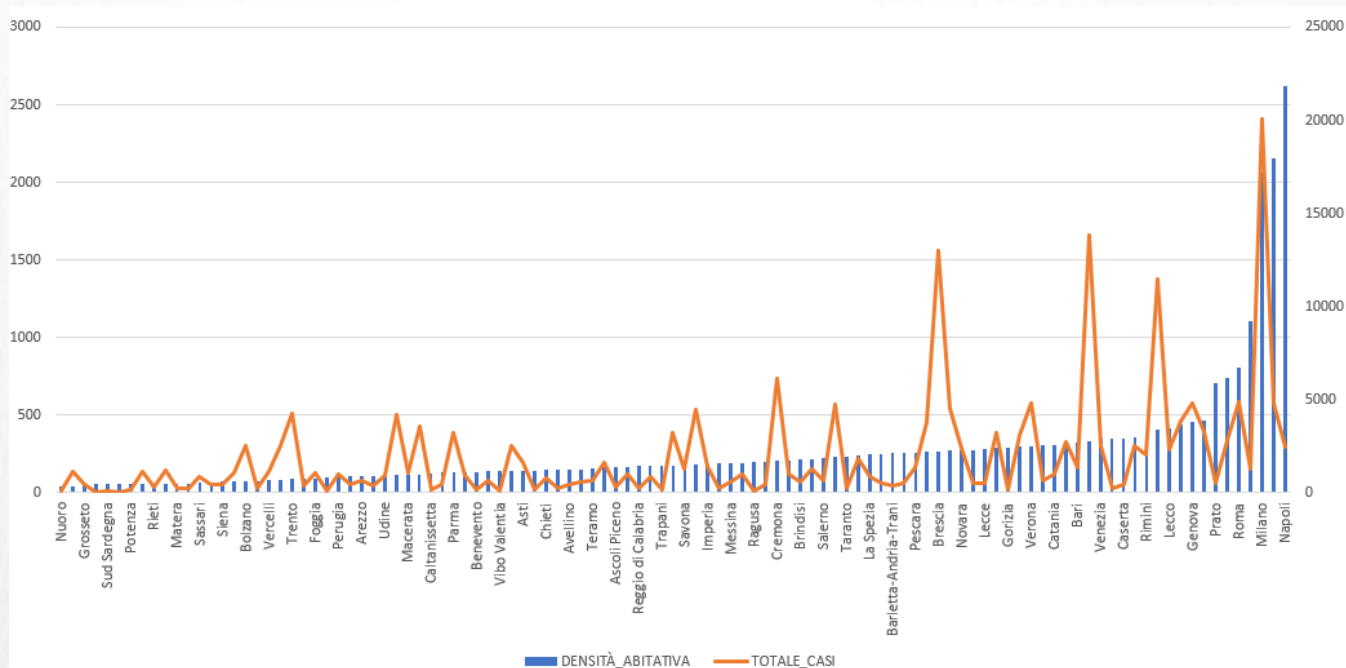
```

Pagina iniziale covid_tesina
Foglio di lavoro Query Builder
select p.denominazione_provincia, ROUND(p.popolazione/p.estensione_territoriale,2) as Densità_Abitativa , cp.totale_casi
from covid_province cp join province p on cp.codice_provincia=p.codice_provincia
where cp.data=(select max(cpl.data) from covid_province cpl where cpl.codice_provincia=cp.codice_provincia)
order by p.popolazione/p.estensione_territoriale

```

La seguente query visualizza, per ogni provincia, la densità abitativa e il totale dei contagiati al fine di verificare un'eventuale correlazione tra i due valori.

Dal grafico che segue (ottenuto dall'esportazione dei risultati dell'interrogazione) non risulta che la densità abitativa, a livello provinciale, abbia in qualche modo impattato sulla diffusione del virus.



Query N.3

Pagina iniziale Covid_Tesina

Foglio di lavoro Query Builder

```
select r.denominazione_regione, ROUND((cr.deceduti/r.popolazione)*1000000,2) as Deceduti_per_milione_Abitanti
from covid_regioni cr join regioni r on cr.codice_regione=r.codice_regione
where ROUND((cr.deceduti/r.popolazione)*1000000,2) >= ALL (select ROUND((cr.deceduti/r.popolazione)*1000000,2)
from covid_regioni cr join regioni r on cr.codice_regione=r.codice_regione)
```

Risultato... x

Tutte le righe recuperate: 1 in 0,057 secondi

DENOMINAZIONE_REGIONE	DECEDUTI_PER_MILIONE_ABITANTI
1 Lombardia	1414,53

Questa query è stata realizzata per individuare la regione più colpita dal COVID-19 in termini di deceduti per milione di abitanti. In particolare, per selezionare la regione avente il valore massimo di tale parametro, ci siamo serviti di una query nidificata.

Query N.4

Pagina iniziale covid_tesina

Foglio di lavoro Query Builder

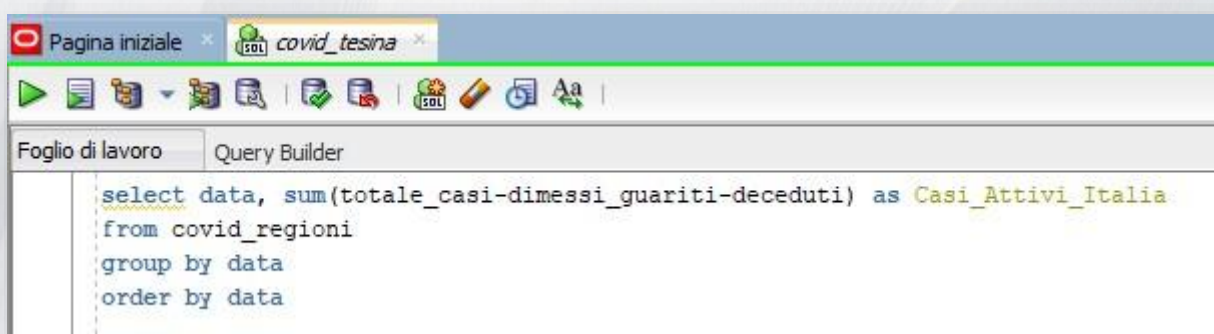
```
select r.denominazione_regione, ROUND((cr.deceduti/(cr.deceduti+cr.dimessi_guariti))*100,2) as Tasso_di_Mortalità
from covid_regioni cr join regioni r on cr.codice_regione=r.codice_regione
where cr.data=(select max(crl.data) from covid_regioni crl)
order by (cr.deceduti/(cr.deceduti+cr.dimessi_guariti))*100
```

DENOMINAZIONE_REGIONE	TASSO_DI_MORTALITÀ
Umbria	5,62
Basilicata	13,02
Valle d'Aosta	13,36
Veneto	13,76
P.A. Trento	14,3
Friuli Venezia Giulia	14,96
P.A. Bolzano	15,02
Molise	18,33
Sardegna	18,89
Toscana	20,59
Campania	20,71
Lazio	20,96
Calabria	21,36
Emilia-Romagna	21,46
Sicilia	23,34
Liguria	25,15
Piemonte	26,73
Abruzzo	29,26
Marche	29,7
Lombardia	35,05
Puglia	35,66

La query restituisce l'elenco delle regioni e dei relativi tassi di mortalità ordinato in ordine crescente in base a tale dato.

Da tali risultati possiamo notare che la regione col più alto tasso di mortalità, contro ogni aspettativa, non sia una regione del Nord Italia, nonostante l'enorme diffusione del virus in quelle zone, ma bensì la Puglia con un valore del 35,66 % .

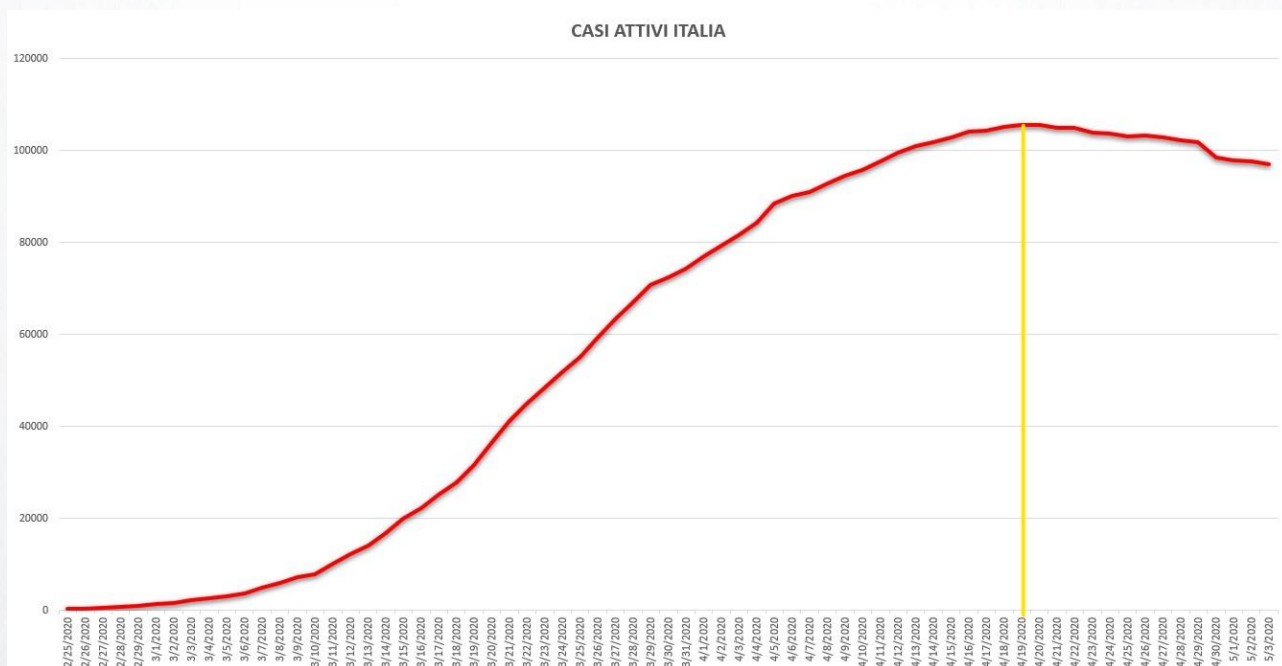
Query N.5



DATA	CASI_ATTIVI_ITALIA
15-APR-20	102903
16-APR-20	104053
17-APR-20	104330
18-APR-20	105116
19-APR-20	105563
20-APR-20	105533
21-APR-20	104990
22-APR-20	104926
23-APR-20	104003
24-APR-20	103685
25-APR-20	102993

La query è fatta con lo scopo di ottenere per ogni giorno il numero di casi attivi in Italia ed utilizzare tale risultato per rappresentare la curva del contagio.

In particolare possiamo osservare anche che la data in cui si è registrato il picco dei casi attivi in Italia è stata il 19 Aprile 2020 con 105563 casi attivi di COVID-19.



Query N.6

Pagina iniziale covid_tesina

Foglio di lavoro Query Builder

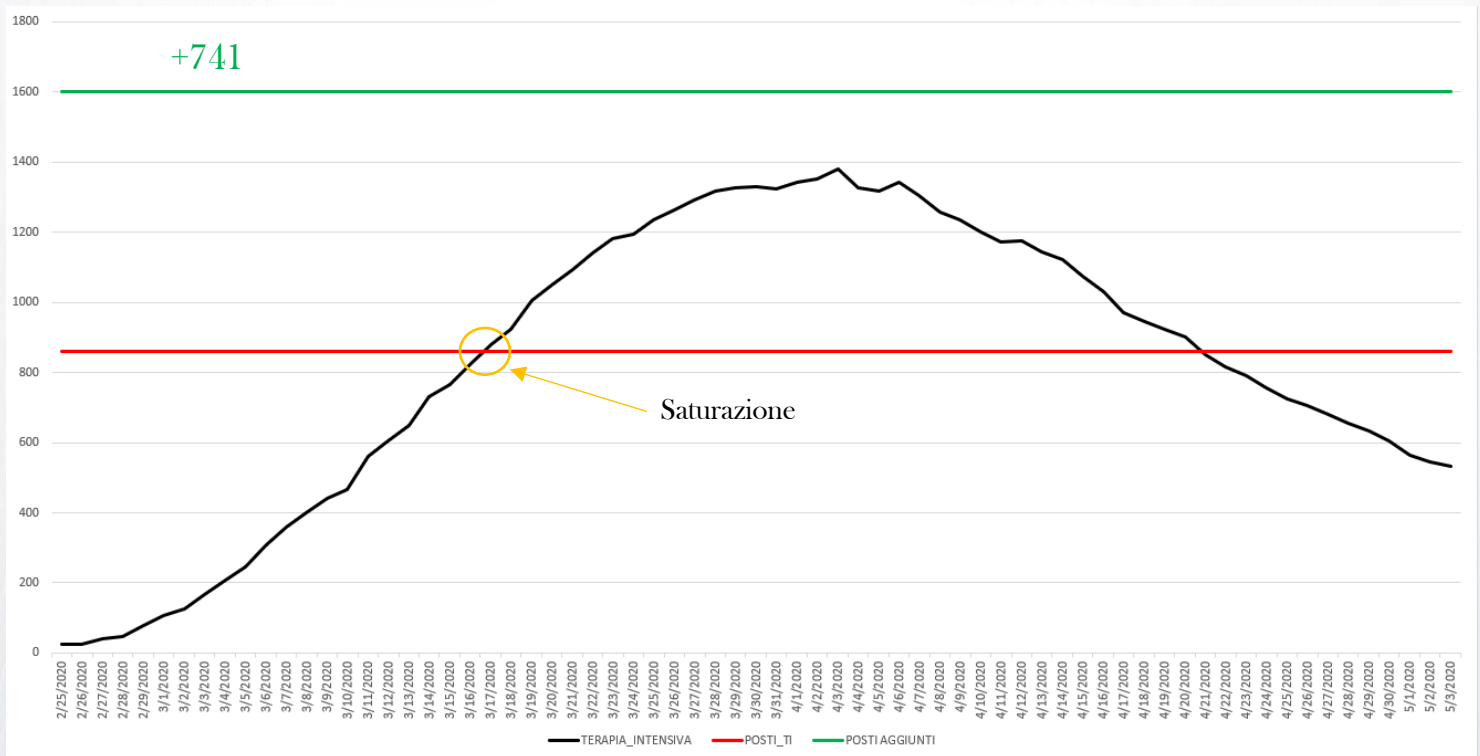
```
select r.denominazione_regione, MIN(cr.data) as Data
from covid_regioni cr join regioni r on cr.codice_regione=r.codice_regione
where cr.terapia_intensiva>r.posti_ti
group by r.denominazione_regione
```

DENOMINAZIONE_REGIONE	DATA
Lombardia	17-MAR-20
Valle d'Aosta	21-MAR-20
Marche	18-MAR-20
Piemonte	23-MAR-20
P.A. Trento	19-MAR-20
P.A. Bolzano	21-MAR-20

La query restituisce le regioni italiane che sarebbero arrivate alla saturazione dei posti in terapia intensiva e inoltre riporta le date in cui ciò sarebbe avvenuto.

Osserviamo che la prima regione a rimanere a corto di letti in terapia intensiva sarebbe stata la Lombardia.

Infatti il grafico che segue dimostra come la soglia dei posti disponibili in Lombardia sarebbe stata ampiamente superata se la regione non avesse messo in atto dei piani per aumentare la capacità di tale reparto.



Il grafico è stato realizzato con la query sotto riportata.

```

Pagina iniziale covid_tesina
Foglio di lavoro Query Builder
select cr.data,cr.terapia_intensiva,r.posti_ti
from covid_regioni cr join regioni r on cr.codice_regione=r.codice_regione
where r.denominazione_regione='Lombardia'
order by data

```

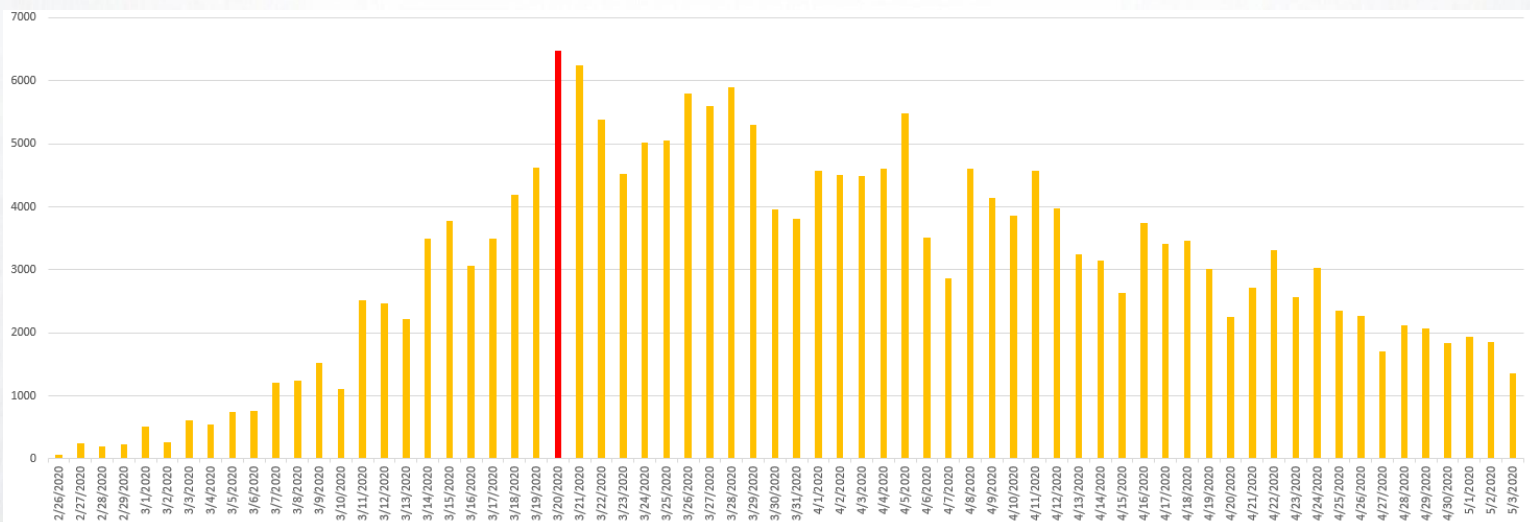
Query N.7

```

Pagina iniziale covid_tesina
Foglio di lavoro Query Builder
select cr.data,
       sum(cr.totale_casi - (select crl.totale_casi
                             from covid_regioni crl
                             where crl.data=cr.data-1 and crl.codice_regione=cr.codice_regione)) as nuovi_positivi
from covid_regioni cr
where data>'25-Feb-2020'
group by cr.data
order by cr.data

```

Questa query ci permette di visualizzare, per ogni giorno, l'incremento di casi positivi al COVID-19 a livello nazionale.

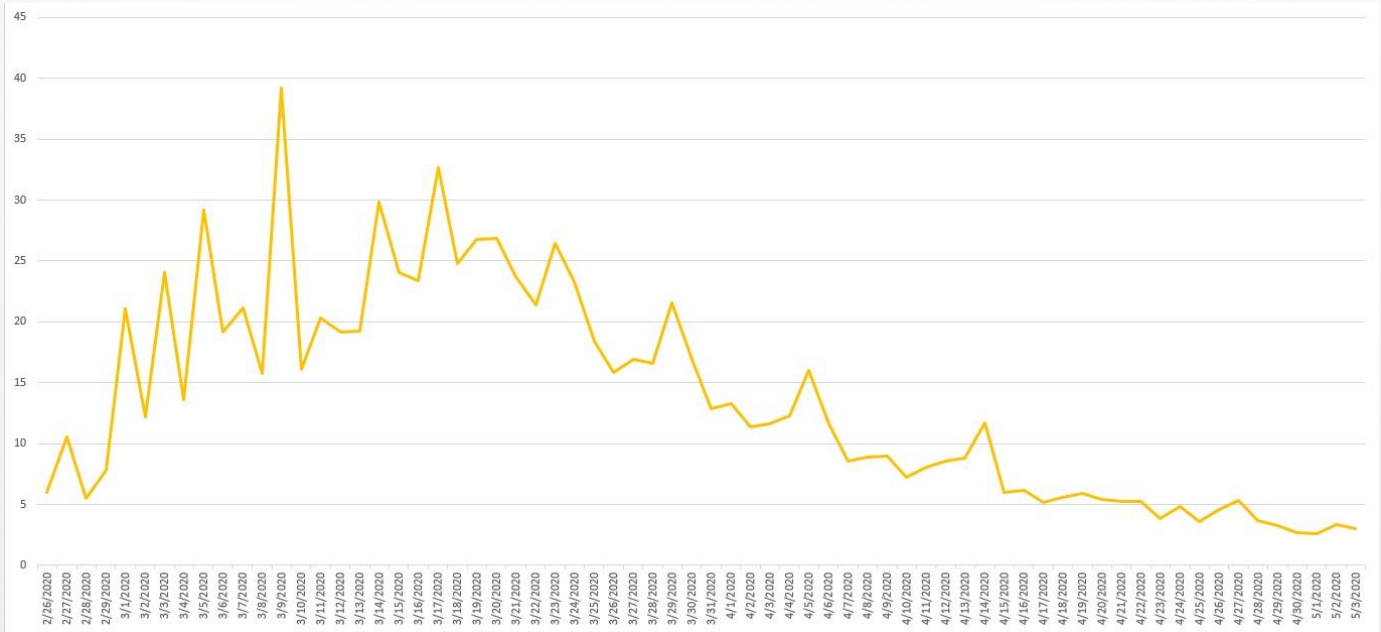


Il grafico è stato ricavato da un'esportazione dei risultati della query. Evidenziato in rosso abbiamo il giorno in cui si sono registrati più nuovi contagi in assoluto, ovvero il 20 Marzo 2020 con ben 6472 nuovi positivi.

Query N.8

```
Pagina iniziale covid_tesina
Foglio di lavoro Query Builder
1 select cr.data, ROUND (sum(cr.totale_casi - (select crl.totale_casi
2 from covid_regioni crl
3 where crl.data=cr.data-1 and crl.codice_regione=cr.codice_regione))
4 / sum(cr.tamponi - (select crl.tamponi
5 from covid_regioni crl
6 where crl.data=cr.data-1 and crl.codice_regione=cr.codice_regione))*100,2) as Percentuale_Tamponi_Positivi
7 from covid_regioni cr
8 where data>'25-Feb-2020'
9 group by cr.data
10 order by cr.data
Output script Risultato query
Recuperate 50 righe in 0,009 secondi
```

La query visualizza giorno per giorno la percentuale di tamponi positivi tra quelli effettuati nella penisola.

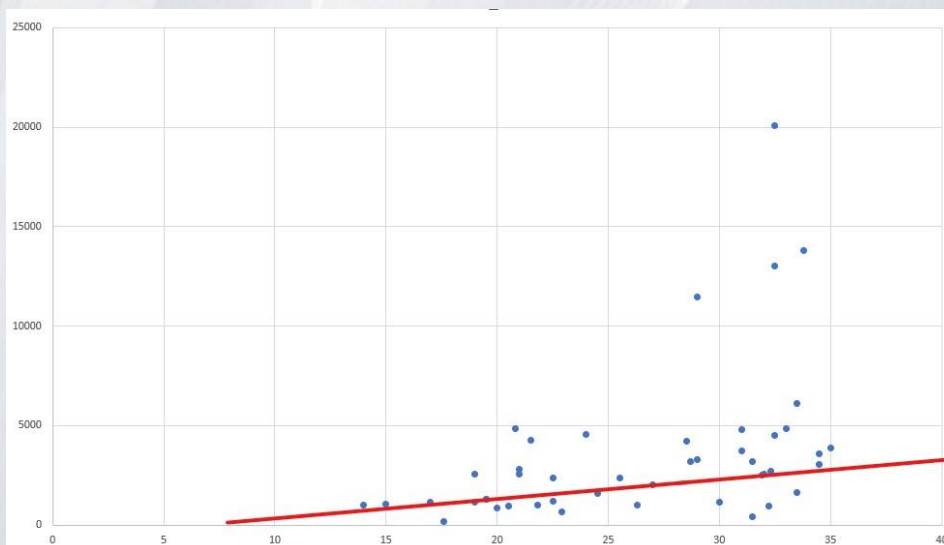


Query N.9

```

1 select p.denominazione_provincia, p.inquinamento_aria_pml0 , cp.totale_casi
2 from covid_province cp join province p on cp.codice_provincia=p.codice_provincia
3 where p.latitudine>44 AND p.inquinamento_aria_pml0 IS NOT NULL and cp.data=(select max(cpl.data)
4                                     from covid_province cpl
5                                     where cpl.codice_provincia=cp.codice_provincia)
6 order by p.inquinamento_aria_pml0

```



L'interrogazione mette in relazione il totale casi delle province del Nord Italia con il fattore di inquinamento PM10 in esse presente; dalla stessa sembrerebbe esserci, come risulta dal grafico, una possibile

correlazione dimostrata dall'andamento pseudo-lineare dei valori.

8. Ottimizzazione

La maggior parte delle stored procedure e delle query fino ad ora mostrate necessitano di informazioni contenute in più tabelle che richiedono dunque l'utilizzo di operazioni di JOIN che sappiamo essere quelle computazionalmente più onerose.

In più le query stesse effettuano molto frequentemente operazioni di selezione che hanno come argomento l'attributo *data*.

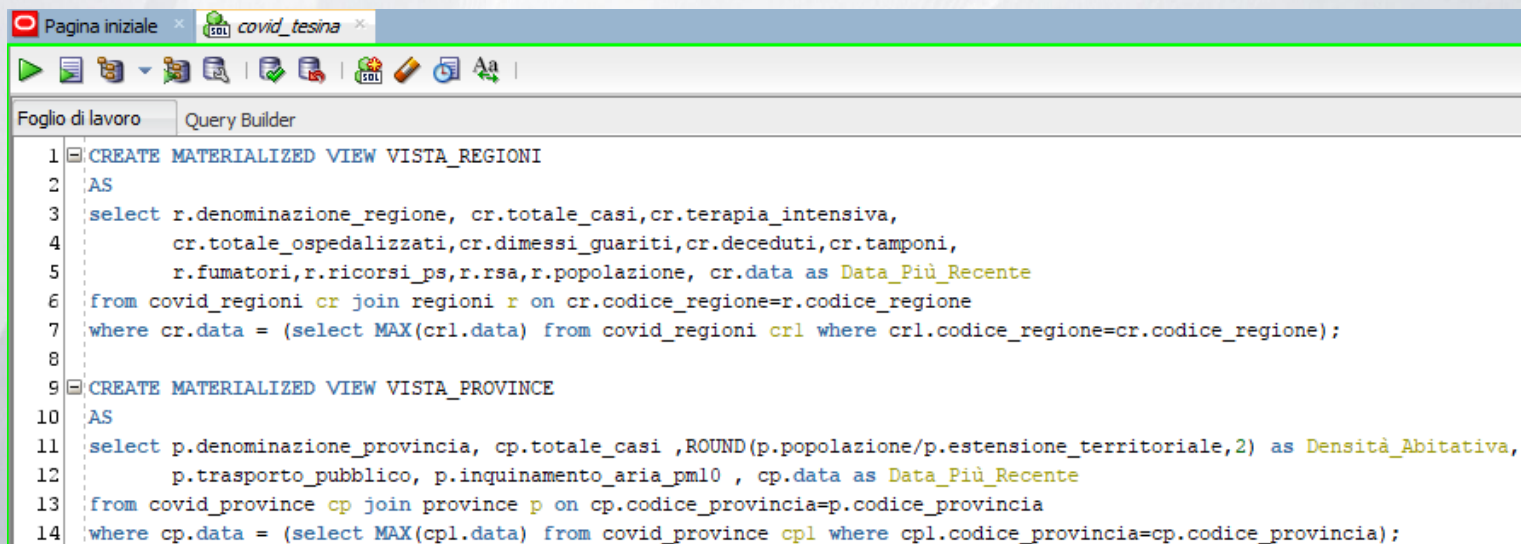
Per cui, al fine di rendere più efficienti le operazioni sulla nostra base dati è possibile definire oggetti quali indici o viste materializzate.

8.1 Viste materializzate

Il meccanismo delle viste ci viene incontro per risolvere il problema dei numerosi JOIN da effettuare.

Inoltre essendo le viste “materializzate”, costituiscono dei veri e propri oggetti della base dati e dunque pronti all'uso in ogni momento.

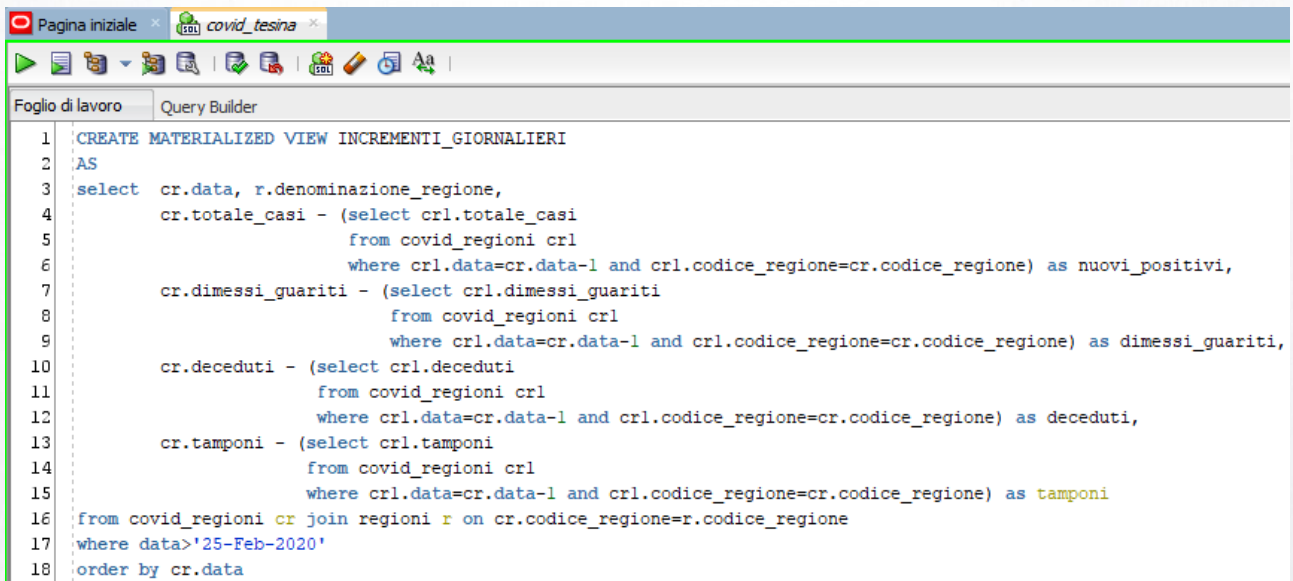
'VISTA_REGIONI' e 'VISTA_PROVINCE'



```
1 CREATE MATERIALIZED VIEW VISTA_REGIONI
2 AS
3 select r.denominazione_regione, cr.totale_casi, cr.terapia_intensiva,
4 cr.totale_ospedalizzati, cr.dimessi_guariti, cr.deceduti, cr.tamponi,
5 r.fumatori, r.ricorsi_ps, r.rsa, r.popolazione, cr.data as Data_Più_Recente
6 from covid_regioni cr join regioni r on cr.codice_regione=r.codice_regione
7 where cr.data = (select MAX(crl.data) from covid_regioni crl where crl.codice_regione=cr.codice_regione);
8
9 CREATE MATERIALIZED VIEW VISTA_PROVINCE
10 AS
11 select p.denominazione_provincia, cp.totale_casi, ROUND(p.popolazione/p.estensione_territoriale,2) as Densità_Abitativa,
12 p.trasporto_pubblico, p.inquinamento_aria_pml0, cp.data as Data_Più_Recente
13 from covid_province cp join province p on cp.codice_provincia=p.codice_provincia
14 where cp.data = (select MAX(cpl.data) from covid_province cpl where cpl.codice_provincia=cp.codice_provincia);
```

Tali viste sono state pensate a valle dei numerosi JOIN effettuati tra le tabelle COVID_REGIONI, REGIONI e COVID_PROVINCE, PROVINCE. In più contengono i dati più recenti sul contagio richiesti molto di frequente dalle nostre query.

'INCREMENTI_GIORNALIERI'



```
1 CREATE MATERIALIZED VIEW INCREMENTI_GIORNALIERI
2 AS
3 select cr.data, r.denominazione_regione,
4        cr.totale_casi - (select crl.totale_casi
5                          from covid_regioni crl
6                          where crl.data=cr.data-1 and crl.codice_regione=cr.codice_regione) as nuovi_positivi,
7        cr.dimessi_guariti - (select crl.dimessi_guariti
8                              from covid_regioni crl
9                              where crl.data=cr.data-1 and crl.codice_regione=cr.codice_regione) as dimessi_guariti,
10       cr.deceduti - (select crl.deceduti
11                     from covid_regioni crl
12                     where crl.data=cr.data-1 and crl.codice_regione=cr.codice_regione) as deceduti,
13       cr.tamponi - (select crl.tamponi
14                    from covid_regioni crl
15                    where crl.data=cr.data-1 and crl.codice_regione=cr.codice_regione) as tamponi
16 from covid_regioni cr join regioni r on cr.codice_regione=r.codice_regione
17 where data>'25-Feb-2020'
18 order by cr.data
```

Dato che gran parte dei dati ricavati dalla nostra fonte erano cumulativi, è nata la necessità di creare una vista di questo tipo che contenesse, invece, i dati dei parametri del contagio giorno per giorno che, in caso contrario, si sarebbero dovuti calcolare ogni volta che ce ne fosse stata necessità attraverso numerosi JOIN e subquery.

8.2 Indici

Un altro tipo di oggetto in grado di velocizzare l'esecuzione delle operazioni previste nella base dati è l'**indice**.

In particolare si è scelto di definire un indice sull'attributo *data* della tabella COVID_REGIONI in quanto la maggior parte delle operazioni di selezione che filtravano le varie tuple avevano come argomento proprio tale attributo.

```
CREATE INDEX COVID_REGIONI_DATA_IND
ON COVID_REGIONI (data);
```

Un indice sull'attributo *data* è stato definito anche per la vista INCREMENTI_GIORNALIERI perché più interessante effettuare su tale vista operazioni di selezione sul campo *data*.

```
CREATE INDEX INCREMENTI_GIORNALIERI_DATA_IND
ON INCREMENTI_GIORNALIERI (data);
```


Infine altri due indici sono stati definiti per le viste 'VISTA_REGIONI' e 'VISTA_PROVINCE', rispettivamente sui campi *denominazione_regione* e *denominazione_provincia* utilizzabili di frequente per selezionare dati relativi ad una particolare regione o provincia presenti nelle due viste.

```
CREATE INDEX VISTA_REGIONI_IND
ON VISTA_REGIONI(denominazione_regione);
```

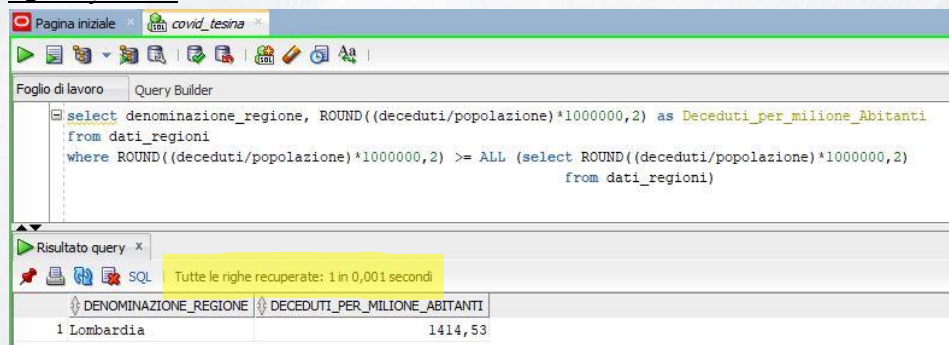
```
CREATE INDEX VISTA_PROVINCE_IND
ON VISTA_PROVINCE(denominazione_provincia);
```

8.3 Esempi di ottimizzazione

Di seguito mostreremo come, grazie agli oggetti definiti in questo capitolo, sia possibile alleggerire alcune query sia dal punto di vista sintattico che dal punto di vista computazionale.

Facendo riferimento al paragrafo 7.1 possiamo riscrivere alcune query nel seguente modo:

Query N.3:



Query Builder

```
select denominazione_regione, ROUND((deceduti/popolazione)*1000000,2) as Deceduti_per_milione_Abitanti
from dati_regioni
where ROUND((deceduti/popolazione)*1000000,2) >= ALL (select ROUND((deceduti/popolazione)*1000000,2)
from dati_regioni)
```

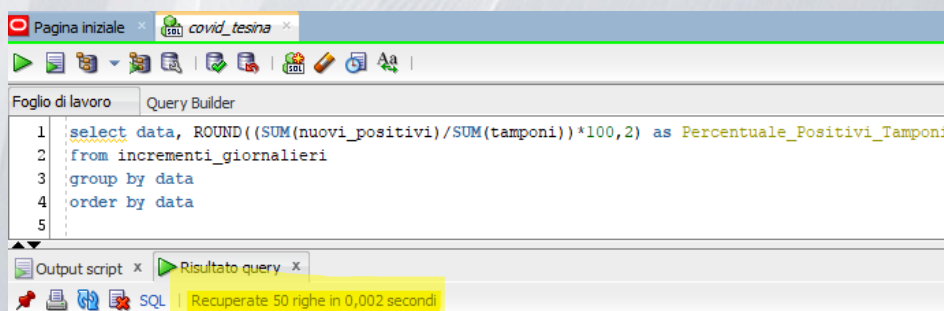
Risultato query x

Tutte le righe recuperate: 1 in 0,001 secondi

DENOMINAZIONE_REGIONE	DECEDUTI_PER_MILIONE_ABITANTI
1 Lombardia	1414,53

Da notare che i tempi di esecuzione sono notevolmente ridotti rispetto alla versione precedente non ottimizzata (da 0,057 secondi a 0,001 secondi).

Query N.8:



Query Builder

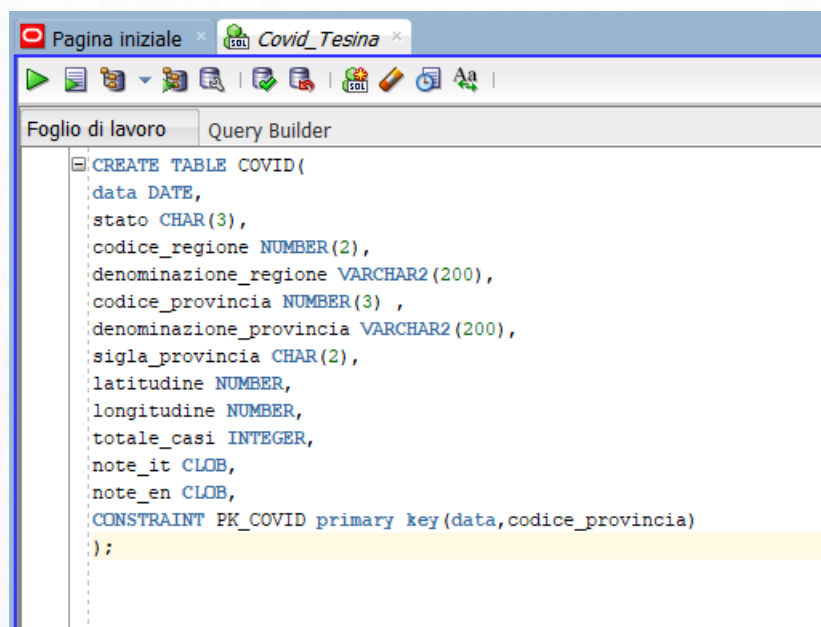
```
1 select data, ROUND((SUM(nuovi_positivi)/SUM(tamponi))*100,2) as Percentuale_Positivi_Tamponi
2 from incrementi_giornalieri
3 group by data
4 order by data
5
```

Output script x Risultato query x

Recuperate 50 righe in 0,002 secondi

Appendice A

Comando DDL per la creazione della tabella Master COVID.



```
CREATE TABLE COVID(  
  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)  
);
```

La tabella Master è stata creata con i seguenti attributi:

- data : attributo di tipo DATE
- stato : stringa di caratteri di lunghezza 3
- codice_regione : numero composto al massimo da 2 cifre
- denominazione_regione, denominazione_provincia: stringhe di caratteri di lunghezza variabile (max 200)
- codice_provincia : numero composto al massimo da 3 cifre
- sigla_provincia: stringa di caratteri di lunghezza 2
- latitudine, longitudine : numeri reali
- totale_casi : numero intero
- note_it, note_en: attributo di tipo CLOB (stringhe fino a dimensioni massime di 4 GB)

La chiave primaria scelta per la relazione è la coppia (*data,codice_provincia*).

La struttura di questa tabella è stata condizionata dalla necessità di contenere le informazioni così come pubblicate e raccolte nel progetto github.