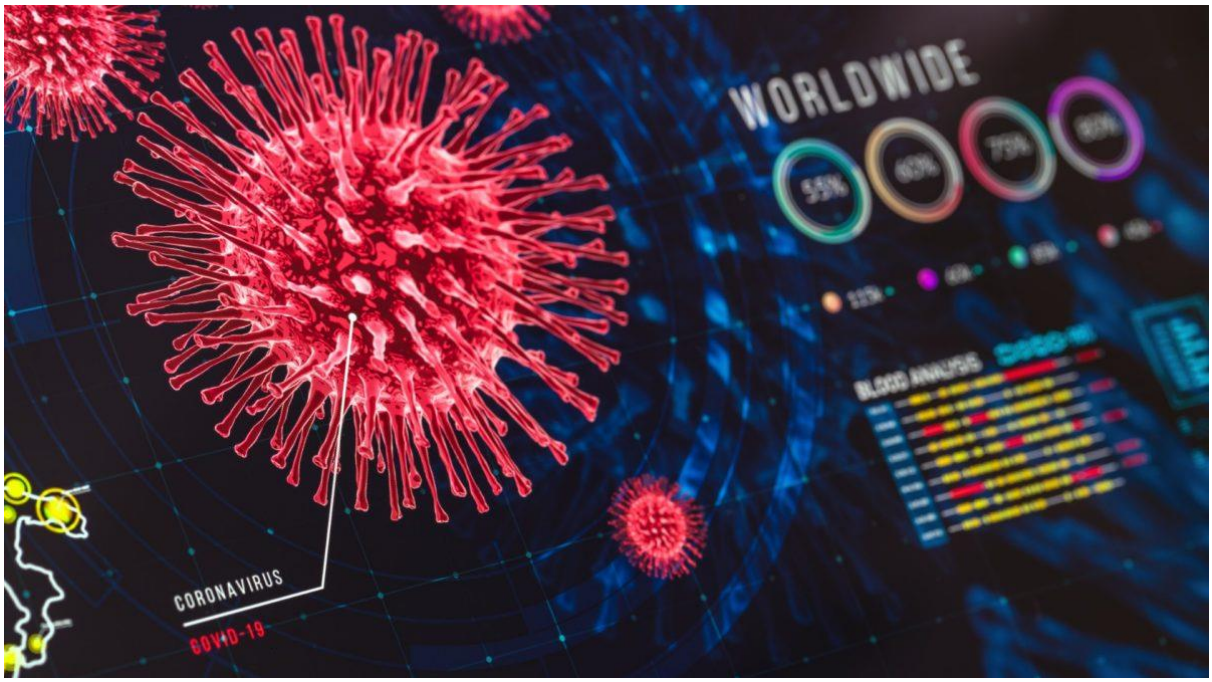


Progettazione Base Dati

per

un'analisi e gestione della situazione epidemiologica relativa alla pandemia di Covid-19 in Italia

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

Corso di Basi di Dati
Anno Accademico 2019-2020
Prof. Vincenzo Moscato

Realizzato da:

Aniello D'Abundo	**N43001436**
Aniello Pio De Stefano	**N43001440**
Emanuele d'Aiello	**N46005346**

Indice

Specifiche di progetto

1. La creazione di una o più tabelle master (con lo stesso schema dei file in formato CSV della protezione civile) ed il relativo popolamento con i dati del contagio per tutte le province/regioni italiane dal 25/02/2020 al 03/05/2020;

2. La verifica della 3NF per lo schema della tabella precedentemente istanziata e l'eventuale decomposizione con la definizione di tutti i vincoli (mediante comandi DDL);

3. L'arricchimento dello schema ottenuto con ulteriori informazioni utili all'analisi del fenomeno;

4. L'individuazione, attraverso un processo di reverse engineering, di un possibile schema concettuale E/R della base di dati;

5. La specifica in SQL di una serie di query utili all'analisi dell'andamento del contagio del COVID-19, con un'eventuale visualizzazione grafica dei risultati;

6. La specifica in PL/SQL di una serie di procedure/trigger che possano consentire un'analisi più flessibile ed eventuali aggiornamenti automatici della base di dati qualora vogliano essere importati dati successivi al 03/5/2020;

7. La definizione di viste sui dati ed indici che possono essere utili per migliorare i tempi di esecuzione delle query.

STEP 1: Creazione tabelle *Master*

Lo schema relazionale denominato "***COVID***", contenente le informazioni necessarie per l'analisi successiva, è il seguente:

> ***COVID*** (Data, Stato, Codice_regione, Denominazione_regione, Codice_provincia, Denominazione_provincia, Sigla_provincia, Latitudine, Longitudine, Totale_casi, Note_it, Note_en).

I vincoli definiti sullo schema sono:

- Primary Key: (Data, Codice_provincia)

E' chiaro che conseguentemente, in tale *Schema Relazionale* va poi inserita la parte *estensionale* che contiene tutte le tuple con le informazioni necessarie.

Passiamo ora al linguaggio SQL e creiamo la tabella ***COVID*** mediante l'utilizzo della applicazione *Oracle XE 18c*:

```
CREATE TABLE COVID (
data DATE,
stato CHAR(3),
codice_regione NUMBER (2),
denominazione_regione VARCHAR2 (200),
codice_provincia NUMBER(3) NOT NULL,
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)
);
```

Caricati gli **script** della protezione civile, provvediamo ad aggiornare i codici regioni delle province autonome di *Bolzano* e *Trento*. Conseguentemente vengono cancellate alcune informazioni non necessarie ai fini delle specifiche come ad esempio la dicitura *"in fase di definizione/aggiornamento";*

```
-- Problema Bolzano
update covid set codice_regione=98
where codice_regione=4 and denominazione_regione= 'P.A. Bolzano';

-- Problema Trento
update covid set codice_regione=99
where codice_regione=4 and denominazione_regione= 'P.A. Trento';

--Dati eccessivi cancellati
delete from covid
where denominazione_provincia = 'In fase di definizione/aggiornamento';
```

STEP 2: Normalizzazione

A questo punto essendo state inserite tutte le informazioni nella tabella COVID è possibile procedere con la normalizzazione. Dato il seguente schema relazionale:

> ***COVID*** (Data, Stato, Codice_Regione, Denominazione_Regione, Codice_Provincia, Denominazione_Provincia, Sigla_Provincia, Latitudine, Longitudine, Totale_Casi, Note_It, Note_En)

-*Verificare se la relazione è in **3NF** ed in caso contrario effettuarne un'opportuna decomposizione.*

Per accertarsi se la relazione è in Terza Forma Normale (3NF) bisogna effettuare le seguenti verifiche:

1. Verifica della Prima Forma Normale (1NF);
2. Fissare la chiave e calcolare le dipendenze funzionali (FD);
3. Verifica della Seconda Forma Normale (2NF);
4. Verifica della Terza Forma Normale (3NF);

FASE DI VERIFICA

1. Verifica della Prima Forma Normale (1NF):

Lo schema relazionale ***COVID*** (Data, …, Note_En) sarà in 1NF se ogni attributo appartenente ad X (con X={Data,Stato,…, Note_En} insieme degli attributi) è un attributo semplice, ovvero se per ogni attributo il relativo dominio è atomico.

Detto in soldoni, lo schema relazionale non deve contenere attributi multi-valore -ovvero quegli attributi i cui valori sono un insieme di valori- e attributi strutturati -ovvero quegli attributi i cui valori sono ennuple di valori.

Ciò detto, si deduce che lo schema COVID (Data, …, Note_En) è in Prima Forma Normale in quanto tutti gli attributi di X sono attributi semplici. Nel caso così non fosse stato, era necessario *Normalizzare* ovvero '***sviluppare'*** gli attributi multi-valore, ed '***estrarre'*** gli attributi strutturati delle relazioni originarie.

2. Fissare la chiave e calcolare le dipendenze funzionali (FD)

(A) PK={Data, Codice_Provincia} , chiave fissata;

(B) Dipendenze funzionali (non banali):

(I): Sulla scorta del *Teorema(Generalizzazione del vincolo di chiave)* tutti gli attributi *non primi* dipendono dalla chiave della relazione, quindi:

(Data, Codice_Provincia) -> Stato, Codice_Regione, Denominazione_Regione, Denominazione_Provincia,Sigla_Provincia,Latitudine,Longitudine,Totale_Casi,Note_it, Note_En

(II):

(Codice_Regione) ->Denominazione_Regione, Stato

(III):

(Codice_Provincia) -> Stato, Codice_Regione, Denominazione_Regione Denominazione_Provincia, Sigla_Provincia, Latitudine Longitudine

3. Verifica della Seconda Forma Normale (2NF)

Lo schema di relazione ***COVID*** (Data, … , Note_En) è in 2NF se:

- E' in Prima Forma Normale (Verificato)
- Ogni attributo non primo di COVID (Data, … , Note_En) è in dipendenza funzionale completa dalla PK di COVID(Data, … , Note_En).

Il Test per la 2NF prevede l'esame delle dipendenze funzionali degli attributi non primi dalla chiave della relazione, con conseguente verifica della completezza o meno di succitata ***FD***; una *dipendenza funzionale* è **completa** se ogni attributo non primo dipende da **tutta** la chiave e non parzialmente. Resta inteso che, se ogni chiave di una relazione è formata da un solo attributo ed inoltre essa è in 1NF, allora tale relazione è in 2NF.

A questo punto considerando le FD possiamo dire che :

- *"Stato"*, *"Codice_Regione"*, *"Denominazione_Regione"*, *"Denominazione_Provincia"*, *"Sigla_Provincia", "Latitudine", "Longitudine"* non sono in dipendenza funzionale completa dalla chiave *(Data, Codice_Provincia)* in quanto dipendono da parte di essa (*Codice_Provincia*).

- *"Totale_casi", "Note_It"*, *"Note_En"* sono in dipendenza funzionale completa con la chiave *(Data, Codice_Provincia)*.

In definitiva, la relazione *COVID* (Data, …, Note_En) NON è in 2NF e quindi va decomposta.

Le dipendenze che causano la non completezza vanno a comporre la tabella "*PROVINCE"*, la cui *primary key* risulterà essere *Codice_Provincia* e i cui attributi sono tutti quegli attributi in dipendenza funzionale non completa dalla chiave *(Data,Codice_Provincia).*

***PROVINCE* (Codice_Provincia, Denominazione_Provincia, Sigla_Provincia, Latitudine, Longitudine, Codice_Regione, Denominazione_Regione, Stato)**

Le dipendenze che causano la completezza compongono la tabella "*COVID_PROVINCE".* Tale tabella ha come *primary key (Data,Codice_Provincia)* e come attributi, gli attributi che sono in dipendenza funzionale completa dalla chiave *(Data,Codice_Provincia).*

***COVID_PROVINCE* (Data, Codice_Provincia, Totali_Casi, Note_It, Note_En)**

4. Verifica della Terza Forma Normale (3NF)

Lo schema di relazione ***COVID*** (*Data, …, Note_En*) è in **3NF** se:

- E' in Seconda Forma Normale (***Verificato***);
- Tutti gli attributi non primi dipendono in maniera non transitiva dalla chiave;

La verifica va fatta sulla tabella "*PROVINCE" e "COVID_PROVINCE".* La tabella *"PROVINCE"* risulta non essere in 3NF in quanto "*Denominazione_Regione*" e "*Stato*" dipendono *transitivamente* dalla chiave "*Codice_Provincia*" a causa di "*Codice_Regione*".

La tabella ***"COVID_PROVINCE"*** è in **3NF** in quanto gli attributi non primi dipendono in maniera non transitiva dalla chiave. Pertanto si procede alla decomposizione della tabella "Province" per renderla in 3NF ovvero si crea la tabella ***REGIONI*** per le dipendenze transitive:

***REGIONI* (Codice_Regione, Denominazione_Regione, Stato)**

Alla fine del processo, le tabelle ottenute sono:

-***REGIONI*** *(Codice_Regione, Denominazione_Regione, Stato)*

-***PROVINCE*** *(Codice_Provincia, Denominazione_Provincia, Sigla_Provincia, Latitudine, Longitudine, Codice_Regione:REGIONI)*

-***COVID_PROVINCE****(Data,Codice_Provincia:PROVINCE,Totali_Casi,Note_It,Note_En)*

- **CREAZIONE RELAZIONI IN SQL**

Si parte dalle 3 tabelle normalizzate: “*Regioni*”, “*Province*”, “*COVID_Province*” con annessi vincoli *intra-relazionali*:

```
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_provincia 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)
);

alter table province add constraint fk_province_regione foreign key (codice_regione) references regioni(codice_regione);
alter table covid_province add constraint fk_covid_prov_prov foreign key (codice_provincia) references province (codice_provincia);
```

Si procede con il riempimento delle tabelle, grazie all’ausilio di “*Insert*” con Query SQL:

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

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

insert into COVID_PROVINCE
select data, codice_provincia, totale_casi, note_it, note_en
from COVID;
```

STEP 3: Arricchimento dello schema

Arricchiamo lo schema mediante la creazione di due ulteriori relazioni.

```
CREATE TABLE COVID_REGIONI (
data DATE,
stato CHAR(3),
codice_regione NUMBER(2),
denominazione_regione VARCHAR2(200),
latitudine NUMBER,
longitudine NUMBER,
ricoverati_con_sintomi NUMBER,
terapia_intensiva NUMBER,
totale_ospedalizzati NUMBER,
isolamento_domiciliare NUMBER,
totale_positivi NUMBER,
variazione_totale_positivi NUMBER,
nuovi_positivi NUMBER,
dimessi_guariti NUMBER,
deceduti NUMBER,
totale_casi NUMBER,
tamponi NUMBER,
casi_testati NUMBER,
note_it CLOB,
note_en CLOB
);

alter table COVID_REGIONI add constraint pk_covid_regioni primary key (data, codice_regione);
alter table COVID_REGIONI add constraint fk_covid_regioni foreign key (codice_regione) references REGIONI (codice_regione);

update COVID_REGIONI set codice_regione=98 where (codice_regione=4 and denominazione_regione= 'P.A. Bolzano');
update COVID_REGIONI set codice_regione=99 where (codice_regione=4 and denominazione_regione= 'P.A. Trento');
```

Di fatto, con quest'ultima è possibile effettuare una analisi dei dati a livello ***regionale*** più accurata cosa non ugualmente possibile con la disponibilità di soli dati provinciali.

Ripetiamo il tutto a livello ***nazionale***:

```
CREATE TABLE COVID_ITALIA (
data DATE,
stato CHAR(3),
ricoverati_con_sintomi NUMBER,
terapia_intensiva NUMBER,
totale_ospedalizzati NUMBER,
isolamento_domiciliare NUMBER,
totale_positivi NUMBER,
variazione_totale_positivi NUMBER,
nuovi_positivi NUMBER,
dimessi_guariti NUMBER,
deceduti NUMBER,
totale_casi NUMBER,
tamponi NUMBER,
casi_testati NUMBER,
note_it CLOB,
note_en CLOB
constraint pk_covid_italia primary key (data)
);
```

Alla tabella “***Province”*** vengono aggiunti i seguenti dati: *popolazione provincia, superficie provincia, densità abitativa province, numero scuole;*

Alla tabella “***Regioni”*** vengono aggiunti i seguenti dati: *popolazione regione, superficie, densità abitanti kmq, numero aeroporti, numero porti, età media.*

Si procede alla verifica della 3NF per le relazioni “*Covid_regioni*” e “*Covid_italia*”.

- ***Normalizzazione tabella COVID_REGIONI***

K= {data, codice regione};

FD:

(I): (data, codice_regione) *-> stato, denominazione_regione, latitudine, longitudine, ricoverati_con_sintomi, terapia_intensiva, totale_ospedalizzati, isolamento_domiciliare, totale_positivi, variazione_tabelle_positivi, nuovi_positivi, dimessi_guariti, deceduti, totale_casi, tamponi, casi_testati, note_it, note_en*

(II): codice_regione *-> stato, denominazione_regione, latitudine, longitudine*

Verifica 2NF: Non è verificata poiché vi sono attributi non primi che dipendono da parte della chiave.

Creo solo una tabella per i *dati regionali* poiché quella per le regioni già esiste.

> **Dati_regionali** (data, codice_regione: *regioni*, ricoverati_con_sintomi, terapia_intensiva, totale_ospedalizzati, isolamento_domiciliare, totale_positivi, variazione_tabelle_positivi, nuovi_positivi, dimessi_guariti, deceduti, totale_casi, tamponi, casi_testati, note_it, note_en)

Verifica 3NF: OK

- ***Normalizzazione tabella COVID_ITALIA***

K={data}

FD:

(I) ***Data*** *-> stato, ricoverati_con_sintomi, terapia_intensiva, totale_ospedalizzati, isolamento_domiciliare, totale_positivi, variazione_tabelle_positivi, nuovi_positivi, dimessi_guariti, deceduti, totale_casi, tamponi, casi_testati, note_it, note_en*

- *Verifica 2NF: OK*
- *Verifica 3NF: OK*

Creazione delle nuove tabelle in virtù della decomposizione di ***Covid_regioni*** (relazione **“*Dati_regionali”***)

```
CREATE TABLE DATI_REGIONALI (
data DATE,
codice_regione NUMBER(2),
ricoverati_con_sintomi NUMBER,
terapia_intensiva NUMBER,
totale_ospedalizzati NUMBER,
isolamento_domiciliare NUMBER,
totale_positivi NUMBER,
variazione_totale_positivi NUMBER,
nuovi_positivi NUMBER,
dimessi_guariti NUMBER,
deceduti NUMBER,
totale_casi NUMBER,
tamponi NUMBER,
casi_testati NUMBER,
note_it CLOB,
note_en CLOB,
constraint pk_dati_regionali primary key (data,codice_regione)
);

alter table DATI_REGIONALI add constraint fk_DATI_REGIONALI foreign key (codice_regione) references regioni(codice_regione);

insert into DATI_REGIONALI
select data, codice_regione, ricoverati_con_sintomi, terapia_intensiva, totale_ospedalizzati,isolamento_domiciliare,
totale_positivi,variazione_totale_positivi,nuovi_positivi, dimessi_guariti, deceduti, totale_casi, tamponi, casi_testati, note_it, note_en
from COVID_REGIONI;
```

Alla tabella “***Dati_regionali”*** vengono aggiunti i seguenti dati: *posti letto terapia intensiva iniziali/aggiunti/totali, spostamenti per vendita al dettaglio e ricreazione vp, spostamenti per supermercati e farmacie vp, spostamento parchi pubblici vp, spostamenti con trasporti pubblici vp, spostamenti verso luoghi di lavoro vp, spostamenti verso le residenze.*

Alla tabella “***Covid_Italia***” vengono aggiunti i seguenti dati: *spostamenti per vendita al dettaglio e ricreazione vp, spostamenti per supermercati e farmacie vp, spostamento parchi pubblici vp, spostamenti con trasporti pubblici vp, spostamenti verso luoghi di lavoro vp, spostamenti verso le residenze.*

Vien poi aggiunta *un’ulteriore relazione* che contiene i dati delle più rilevanti strutture sanitare divise per regione (aziende ospedaliere, ospedali a gestione diretta, case di cura accreditate e non, guardie mediche etc.)

```
create table STRUTTURE_SANITARIE
(
codice_regione NUMBER,
aziende_ospedaliere NUMBER,
ospedali_a_gest_diretta NUMBER,
a_o_integrata_con_il_ssn NUMBER,
a_o_integrata_con_università NUMBER,
policlinico_universitario_privato NUMBER,
istituti_a_carattere_scientifico NUMBER,
ospedali_classificati_o_assimilati NUMBER,
istituti_presidio_delle_asl NUMBER,
enti_di_ricerca NUMBER,
totale_aziende_ospedaliere NUMBER,
case_di_cura_accreditate NUMBER,
totale_case_di_cura NUMBER,
guardia_medica NUMBER
);

alter table STRUTTURE_SANITARIE add constraint pk_strutture_sanitarie primary key (codice_regione);
alter table STRUTTURE_SANITARIE add constraint fk_strutture_sanitarie_regioni foreign key (codice_regioni) references regioni(codice_regione);
```

STEP 4: MODELLO E/R

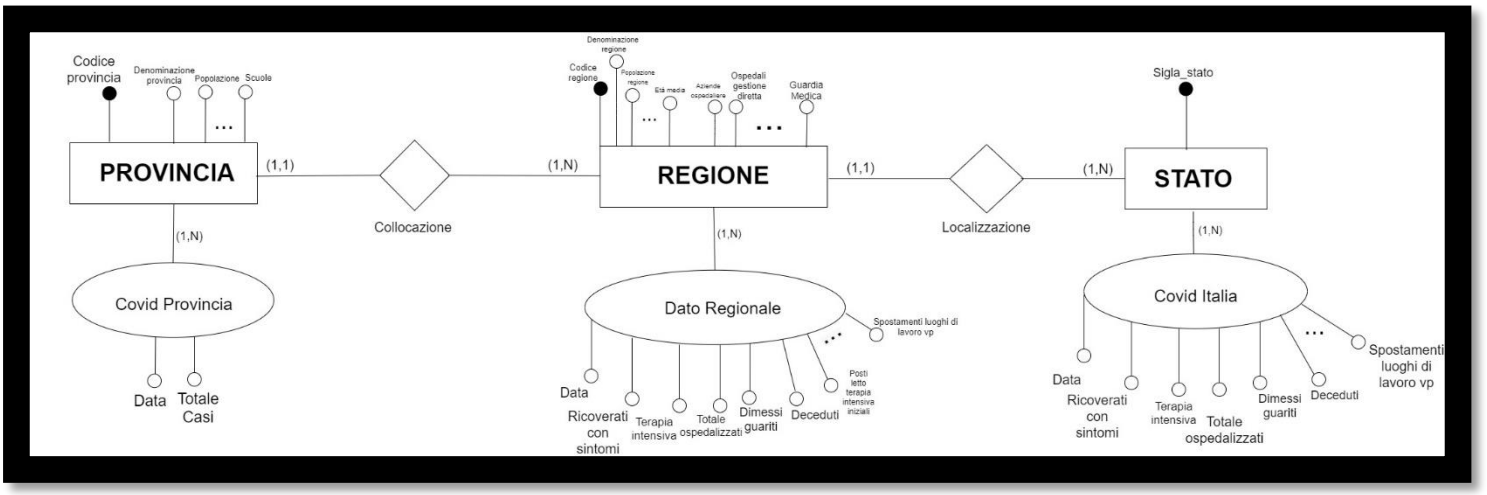
La progettazione dei componenti informatici di un sistema informativo avviene progettando da un lato le applicazioni software che interagiscono con la Base Dati e dall'altro la Base Dati stessa.
La progettazione della Base Dati si articolano in 3 fasi distinti:

- Progettazione Concettuale → Si parte dai requisiti della Base Dati espressi in linguaggio naturale e si ottiene il modello concettuale (E/R);

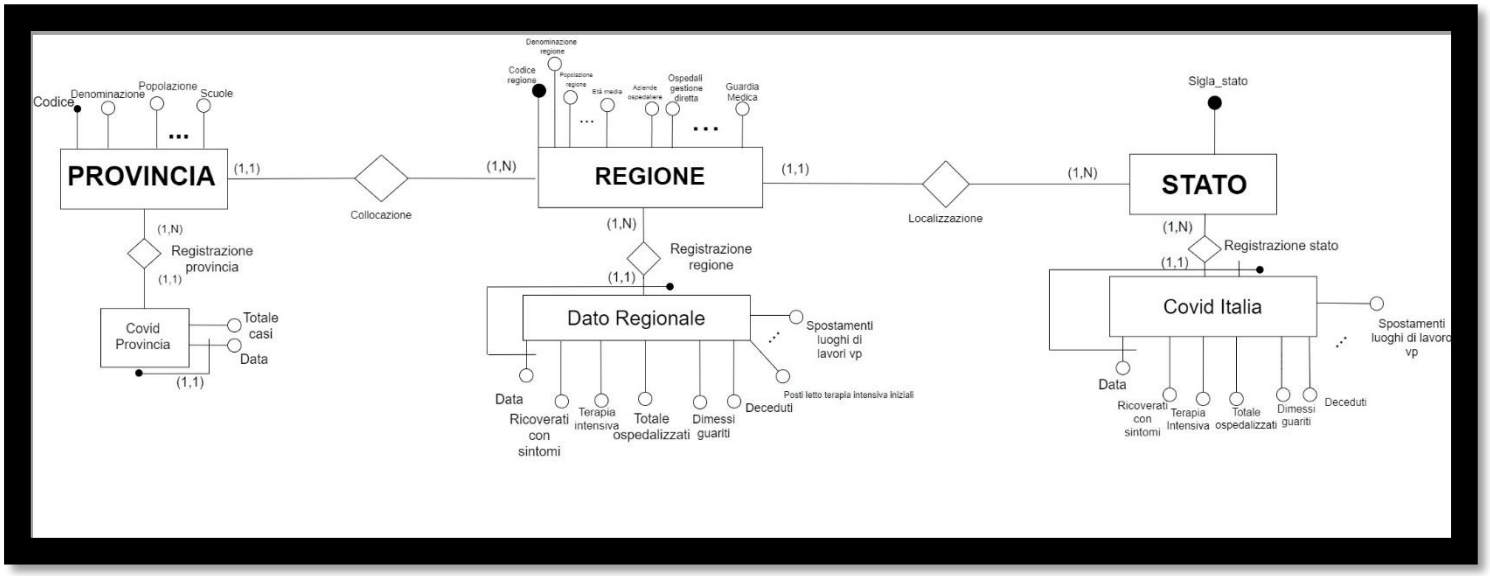
- Progettazione Logica → Il Modello concettuale viene trasformato nel modello logico attraverso l'insieme di regole;

- Progettazione Fisica → Viene creata la Base Dati fisicamente sul DBMS e si può gestire la politica di sicurezza;

In questo caso si adotta un processo di ***reverse engineering*** in quanto si parte dalla progettazione logica e si ricava il modello E/R corrispondente. All'uopo, proponiamo di seguito un possibile modello E/R del problema trattato in questo elaborato.



A questo punto, sulla scorta delle regole di trasformazione dei modelli E/R, otteniamo il seguente *modello E/R trasformato*.



STEP 5: QUERY SQL

OSS: Per quanto riguarda la relazione COVID_ITALIA sono stati inseriti i dati fino al 15 settembre 2020.

QUERY n.1: *Selezionare i nuovi positivi al giorno (02/04) della regione 8;*

```
select nuovi_positivi
from DATI_REGIONALI
where codice_regione=8 and data='02-Apr-2020';
```

NUOVI_POSITIVI
546

QUERY n.2: *Selezionare il numero di tamponi effettuati in Italia il 2 aprile*

```
select (tamponi- (
                  select tamponi
                  from covid_italia
                  where data ='01-Apr-2020'
                  )
        ) as Tamponi_Effettuati
from COVID_ITALIA
where data ='02-Apr-2020';
```

TAMPONI_EFFETTUATI
39809

QUERY n.3: *Selezionare stato, codice regione, denominazione regione delle regioni che al 7/03 non avevano ancora deceduti;*

```
select r.stato,r.codice_regione,r.denominazione_regione
from  dati_regionali d join regioni r on d.codice_regione=r.codice_regione
where d.deceduti=0 and d.data ='07-Mar-2020';
```

	STATO	CODICE_REGIONE	DENOMINAZIONE_REGIONE
1	ITA	2	Valle d'Aosta
2	ITA	6	Friuli Venezia Giulia
3	ITA	9	Toscana
4	ITA	10	Umbria
5	ITA	13	Abruzzo
6	ITA	14	Molise
7	ITA	15	Campania
8	ITA	17	Basilicata
9	ITA	18	Calabria
10	ITA	19	Sicilia
11	ITA	20	Sardegna
12	ITA	98	P.A. Bolzano
13	ITA	99	P.A. Trento

QUERY n.4: *Mostrare codice, denominazione, sigla della provincia con maggior densità abitativa e con minor densità abitativa;*

```
select codice_provincia,denominazione_provincia,sigla_provincia,densita_abitativa_prov
from PROVINCE
where densita_abitativa_prov=(select max(densita_abitativa_prov) from province)
or
      densita_abitativa_prov=(select min(densita_abitativa_prov) from province );
```

	CODICE_PROVINCIA	DENOMINAZIONE_PROVINCIA	SIGLA_PROVINCIA	DENSITA_ABITATIVA_PROV
1	63	Napoli	NA	2617
2	91	Nuoro	NU	37

QUERY n.5: *Mostrare l'età media delle regioni ed i casi totali in ordine crescente d'età;*

```
select R.codice_regione,R.denominazione_regione,R.età_media,D.totale_casi
from REGIONI R join DATI_REGIONALI D on R.codice_Regione=D.codice_regione
where D.data='03-Mag-2020'
order by età_media;
```

	CODICE_REGIONE	DENOMINAZIONE_REGIONE	ETÀ_MEDIA	TOTALE_CASI
1	15	Campania	42,15	4484
2	19	Sicilia	43,53	3240
3	18	Calabria	43,98	1114
4	16	Puglia	44,23	4144
5	12	Lazio	44,58	6809
6	99	P.A. Trento	44,6	4247
7	98	P.A. Bolzano	44,7	2536
8	3	Lombardia	44,74	77528
9	5	Veneto	45,1	18318
10	17	Basilicata	45,31	386
11	2	Valle d'Aosta	45,63	1142
12	13	Abruzzo	45,67	2996
13	8	Emilia-Romagna	45,7	26016
14	11	Marche	46,09	6319
15	14	Molise	46,28	301
16	20	Sardegna	46,33	1319
17	10	Umbria	46,49	1394
18	9	Toscana	46,52	9563
19	1	Piemonte	46,54	27430
20	6	Friuli Venezia Giulia	47	3072
21	7	Liguria	48,46	8359

QUERY n.6: *Selezionare il numero di ospedalizzati (a livello regionale) rispetto alla popolazione della regione Piemonte (in valore percentuale)*

```
select r.codice_regione,r.denominazione_regione,d.totale_ospedalizzati,
       r.popolazione_regione,round((d.totale_ospedalizzati/r.popolazione_regione),5)*100 as percentuale_ospedalizzati
from regioni r join dati_regionali d on d.codice_regione=r.codice_regione
where r.denominazione_regione='Piemonte' and d.data='03-mag-2020';
```

	CODICE_REGIONE	DENOMINAZIONE_REGIONE	TOTALE_OSPEDALIZZATI	POPOLAZIONE_REGIONE	PERCENTUALE_OSPEDALIZZATI
1	1	Piemonte	2665	4356000	0,061

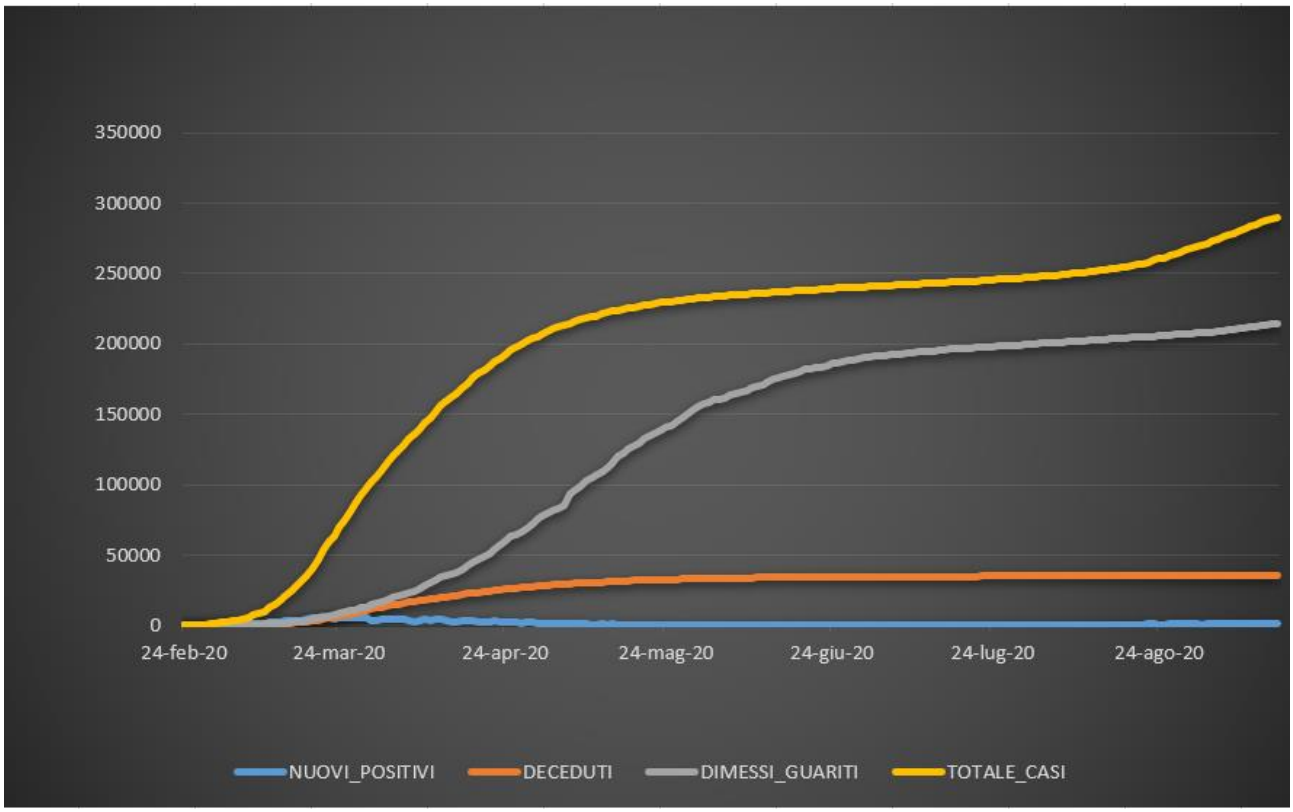
QUERY n.7: *Selezionare per ogni regione la provincia con il maggior numero di casi al 3/05.*

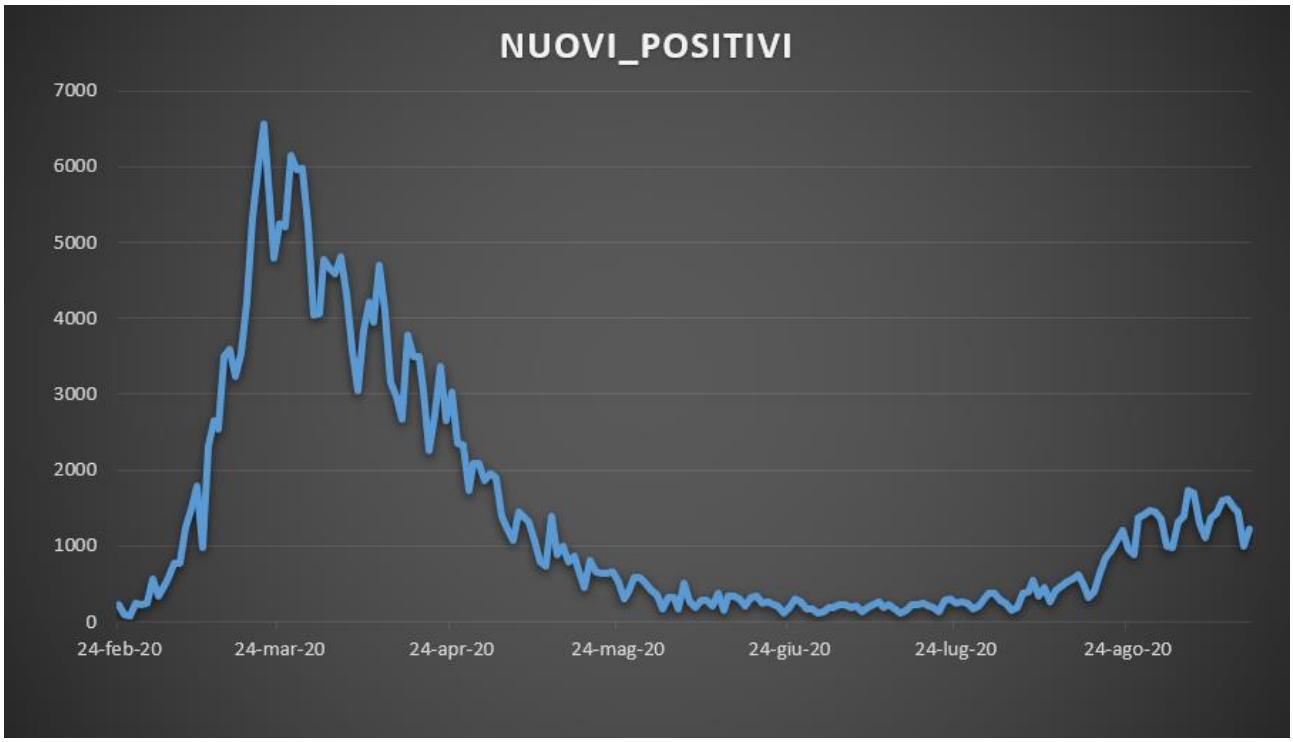
```
--Per ogni regione e per ogni provincia il totale casi è al 3 maggio (essendo cumulativo è esattamente il massimo)
--Ordino per totale casi decrescente così da riutilizzare la vista per la query 11)
create materialized view max_casi_per_prov_e_reg as
select p.codice_provincia,r.codice_regione,r.denominazione_regione,p.denominazione_provincia,c.totale_casi
from (covid_province c join province p on p.codice_provincia=c.codice_provincia) join regioni r
        on r.codice_regione=p.codice_regione
where c.data='03-mag-2020'
order by c.totale_casi desc;

--per ogni regione il totale casi massimo (cioè al 3 maggio preso dalla vista precedente)
create materialized view max_casi_per_regione as
select codice_regione,denominazione_regione,max(totale_casi) as max_totale_casi_regione
from max_casi_per_prov_e_reg
group by codice_regione,denominazione_regione;

--metto in join le due viste così da otterere per ogni regione la provincia con il maggior numero di casi
select m1.codice_regione,m1.denominazione_regione,m1.codice_provincia,m1.denominazione_provincia,m1.totale_casi
from max_casi_per_prov_e_reg m1 join max_casi_per_regione m2
    on (m1.codice_regione=m2.codice_regione and m1.totale_casi=m2.max_totale_casi_regione)
order by m1.totale_casi desc;
```

CODICE_REGIONE	DENOMINAZIONE_REGIONE	CODICE_PROVINCIA	DENOMINAZIONE_PROVINCIA	TOTALE_CASI
3	Lombardia	15	Milano	20068
1	Piemonte	1	Torino	13794
12	Lazio	58	Roma	4911
7	Liguria	10	Genova	4840
5	Veneto	23	Verona	4800
8	Emilia-Romagna	35	Reggio nell'Emilia	4765
99	P.A. Trento	22	Trento	4247
9	Toscana	48	Firenze	3210
11	Marche	41	Pesaro e Urbino	2540
98	P.A. Bolzano	21	Bolzano	2536
15	Campania	63	Napoli	2468
13	Abruzzo	68	Pescara	1347
16	Puglia	72	Bari	1322
6	Friuli Venezia Giulia	32	Trieste	1281
2	Valle d'Aosta	7	Aosta	1142
19	Sicilia	87	Catania	1002
10	Umbria	54	Perugia	992
20	Sardegna	90	Sassari	848
18	Calabria	78	Cosenza	459
14	Molise	70	Campobasso	226
17	Basilicata	77	Matera	198

QUERY n.8 *Trend giornaliero a livello nazionale di attuali positivi, guariti, morti e casi totali*

```
select data,nuovi_positivi,deceduti,dimessi_guariti,totale_casi
from covid_italia
order by data;
```

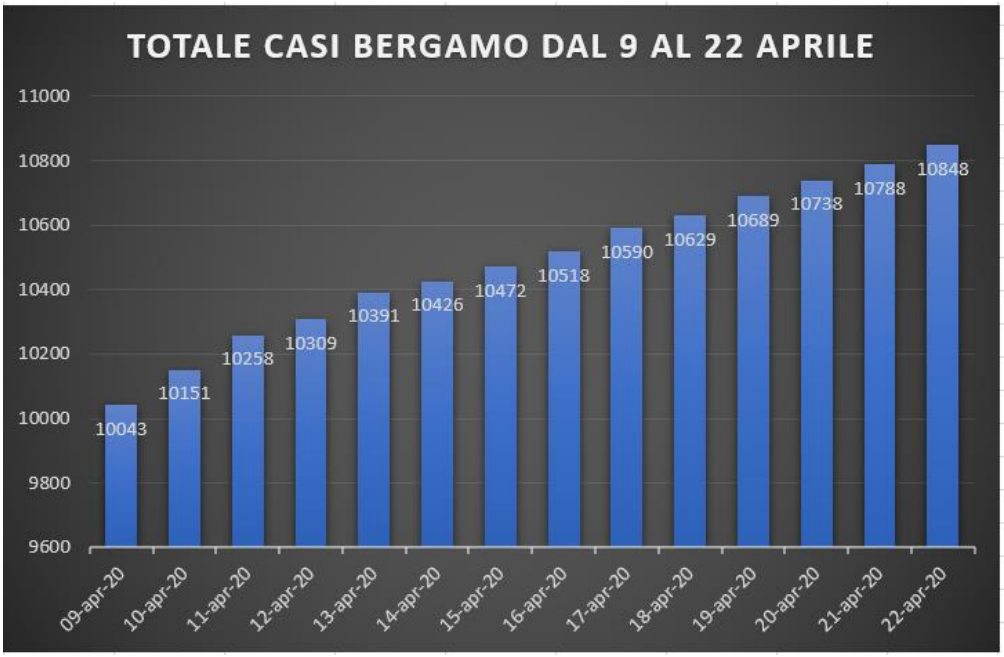




Andamento dei nuovi positivi fino al 15 settembre 2020

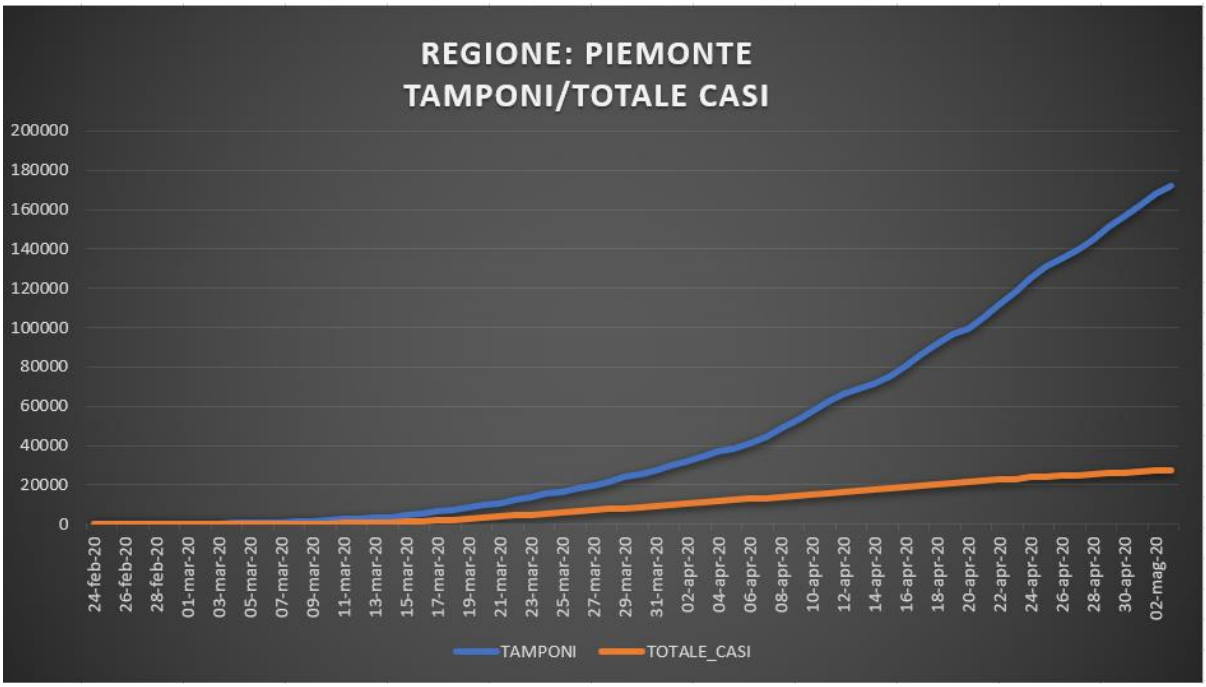
QUERY n.9: *Andamento dei casi totali nella provincia di Bergamo dal 9 al 23 aprile;*

```
 select c.data,p.denominazione_provincia,c.totale_casi
from covid_province c join province p on c.codice_provincia=p.codice_provincia
where p.denominazione_provincia='Bergamo' and(c.data>='09-Apr-2020' and c.data<'23-Apr-2020')
order by c.data;
```



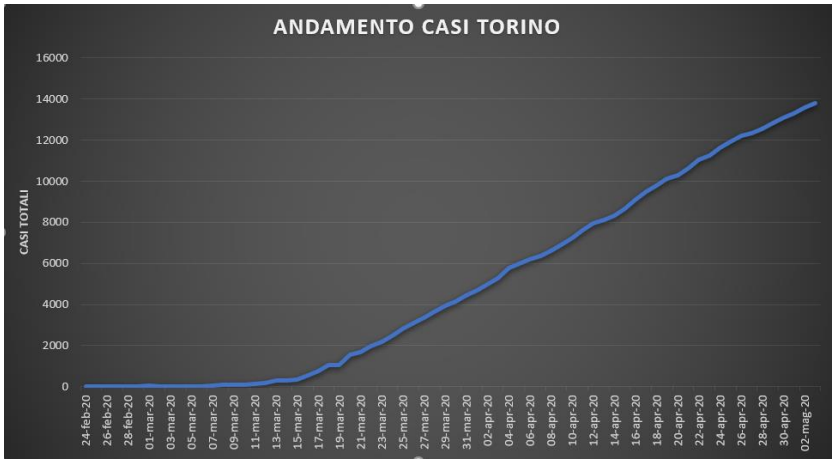
QUERY n.10**:** *Confronto tra andamento dei casi e numero tamponi totali effettuati in Piemonte;*

```
select d.data,r.denominazione_regione,d.tamponi,d.totale_casi
from dati_regionali d join regioni r on d.codice_regione=r.codice_regione
where r.denominazione_regione='Piemonte'
order by d.data;
```



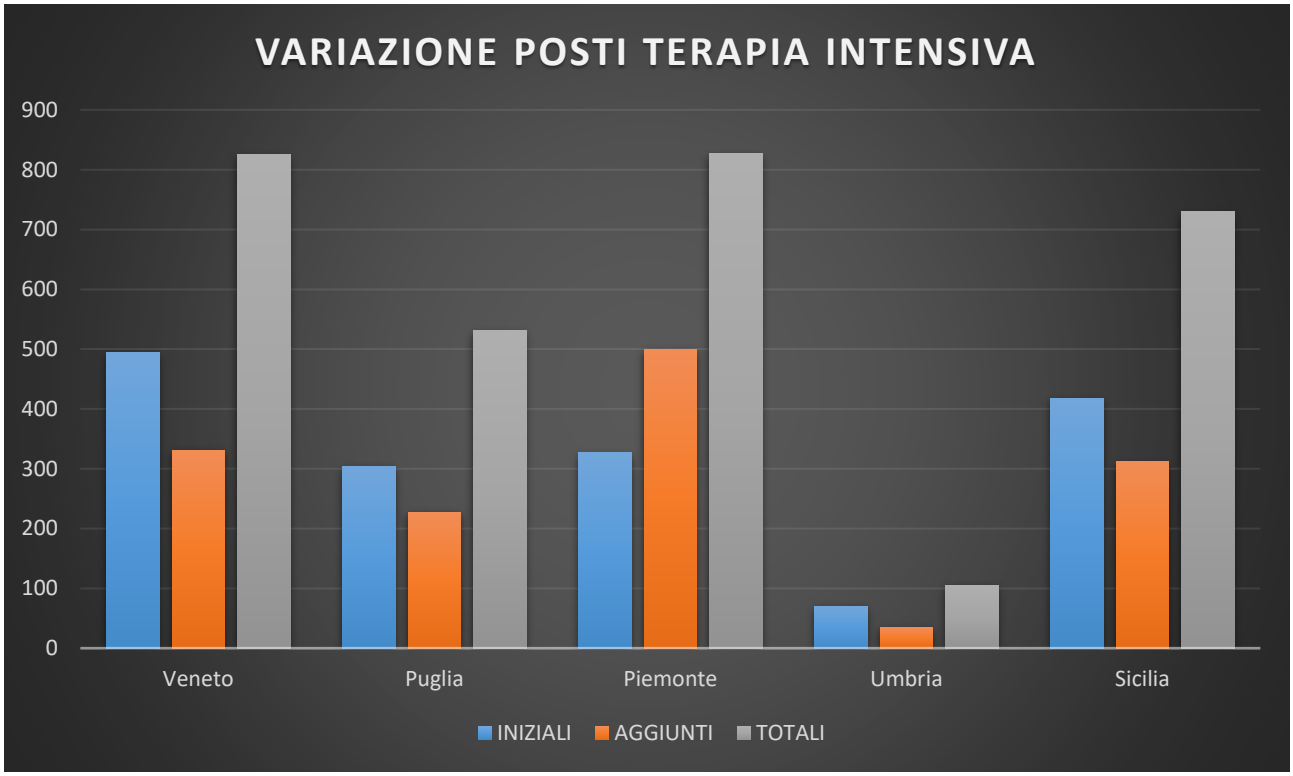
QUERY n.11 *Graficare l'andamento del totale casi delle 3 province con più casi sino al 03 Maggio*

```
select c.data,d.denominazione_provincia, c.totale_casi
from
    (
    --seleziono le 3 province con minor numero dalla vista (già ordinata) max_casi_per_prov_e_reg
    select *
    from max_casi_per_prov_e_reg
    where rownum<4) d join COVID_PROVINCE c on c.codice_provincia=d.codice_provincia
order by d.codice_provincia,c.data;
```



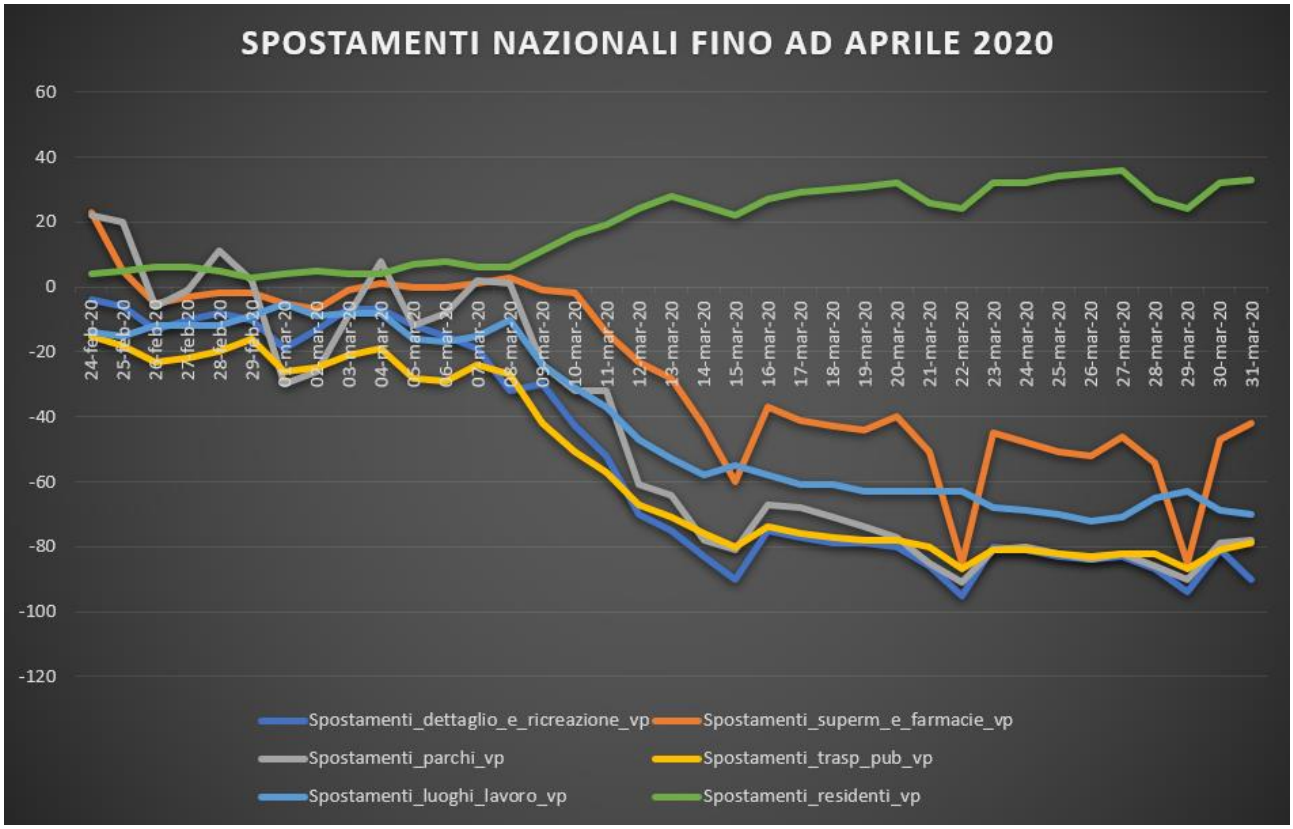
QUERY n.12: *Graficare la variazione dei posti in terapia intensiva (iniziali, totali ed aggiunti) per le seguenti regioni: Lombardia, Piemonte, Emilia-Romagna, Molise, Puglia, Campania e Lazio;*

```
select distinct r.denominazione_regione,d.posti_letto_terapia_intensiva_iniziali,d.posti_letto_terapia_intensiva_aggiunti,
                                                    d.posti_letto_terapia_intensiva_totali
from  REGIONI r join DATI_REGIONALI d  on d.codice_regione=r.codice_regione
where( r.denominazione_regione='Piemonte' or r.denominazione_regione='Veneto' or r.denominazione_regione='Umbria'
                                          or r.denominazione_regione='Puglia' or r.denominazione_regione='Sicilia') ;
```



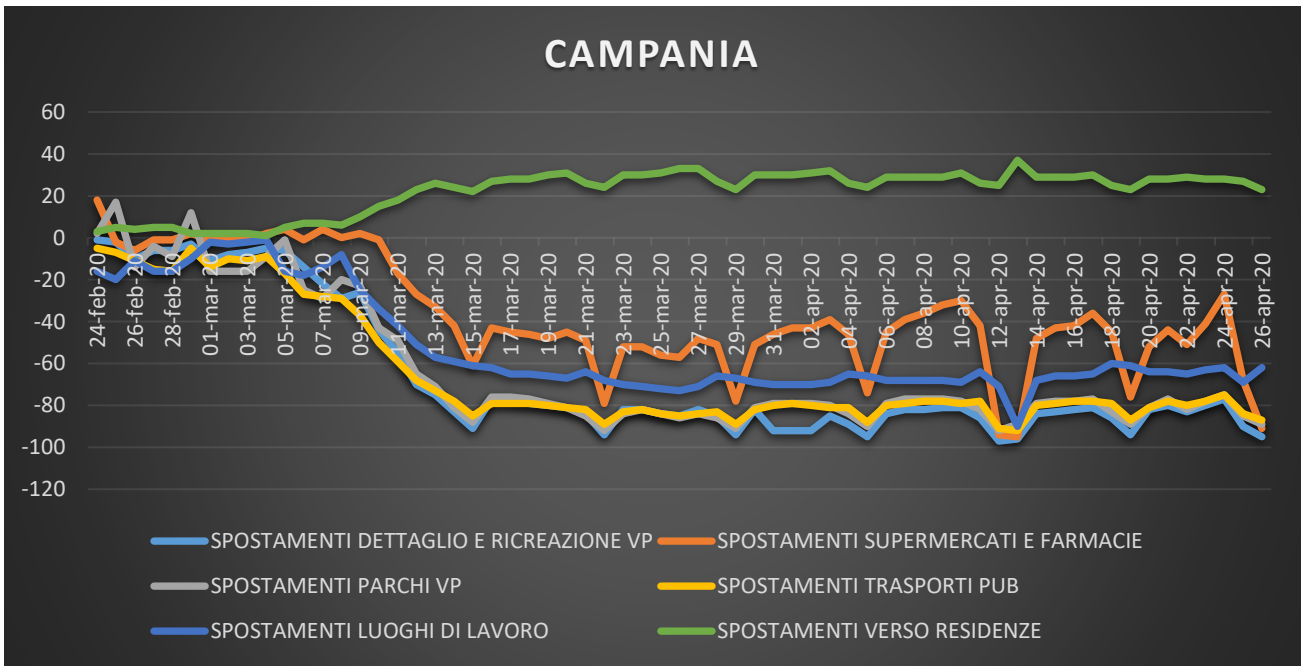
***QUERY n.13*:** *Graficare tutti gli spostamenti fino ad Aprile 2020 in Italia;*

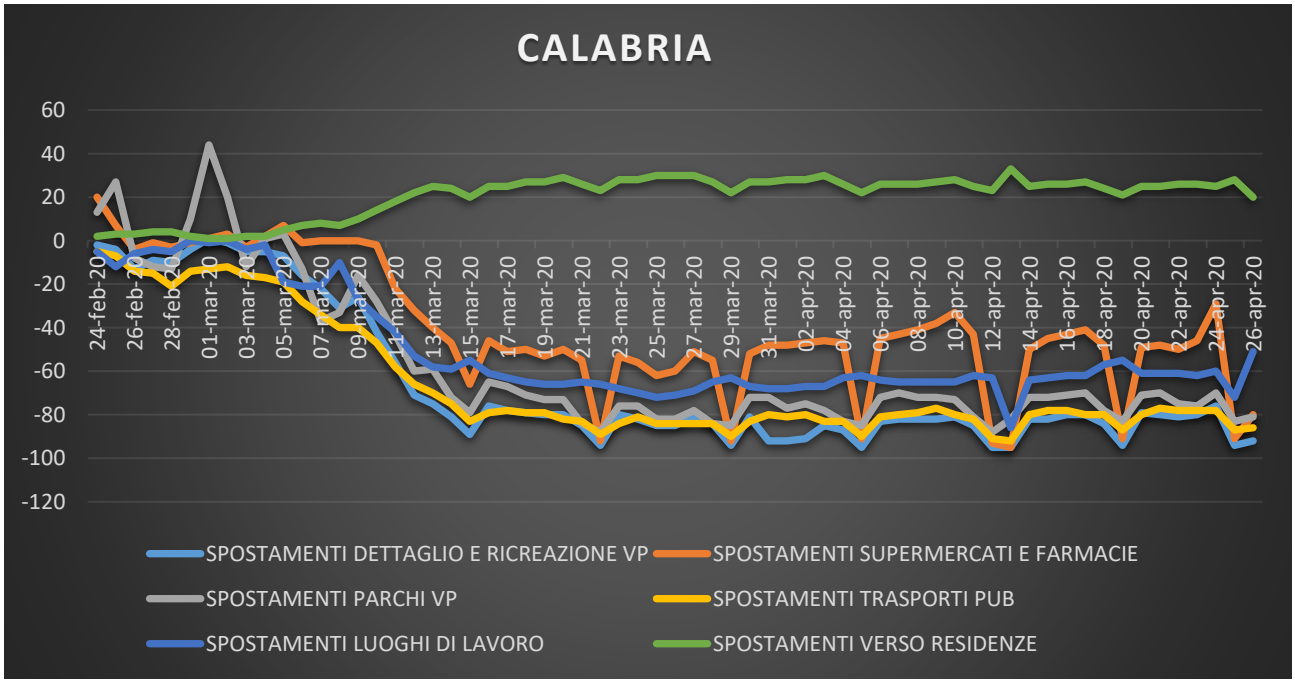
```
select data, spostamenti_dettaglio_e_ricreazione_vp,spostamenti_superm_e_farmacie_vp,
       spostamenti_parchi_vp,spostamenti_trasp_pub_vp,spostamenti_luoghi_lavoro_vp,
       spostamenti_residenti_vp
from covid_italia
where data <'01-Apr-2020'
order by data;
```



QUERY n.14: *Graficare tutti gli spostamenti delle regioni Calabria e Campania fino al 26 Aprile*

```
select  d.data,r.denominazione_regione,d.spostamenti_dettaglio_e_ricreazione_vp,
        d.spostamenti_superm_e_farmacie_vp,d.spostamenti_parchi_vp,
        d.spostamenti_trasp_pub_vp,d.spostamenti_luoghi_lavoro_vp,d.spostamenti_residenti_vp
from dati_regionali d join regioni r on d.codice_regione=r.codice_regione
where (r.denominazione_regione='Campania' or r.denominazione_regione='Calabria') and d.data <'27-Apr-2020'
order by r.codice_regione,d.data;
```





QUERY n.15: *Selezionare le province con maggior e minor densità abitativa e visualizzare il totale casi*

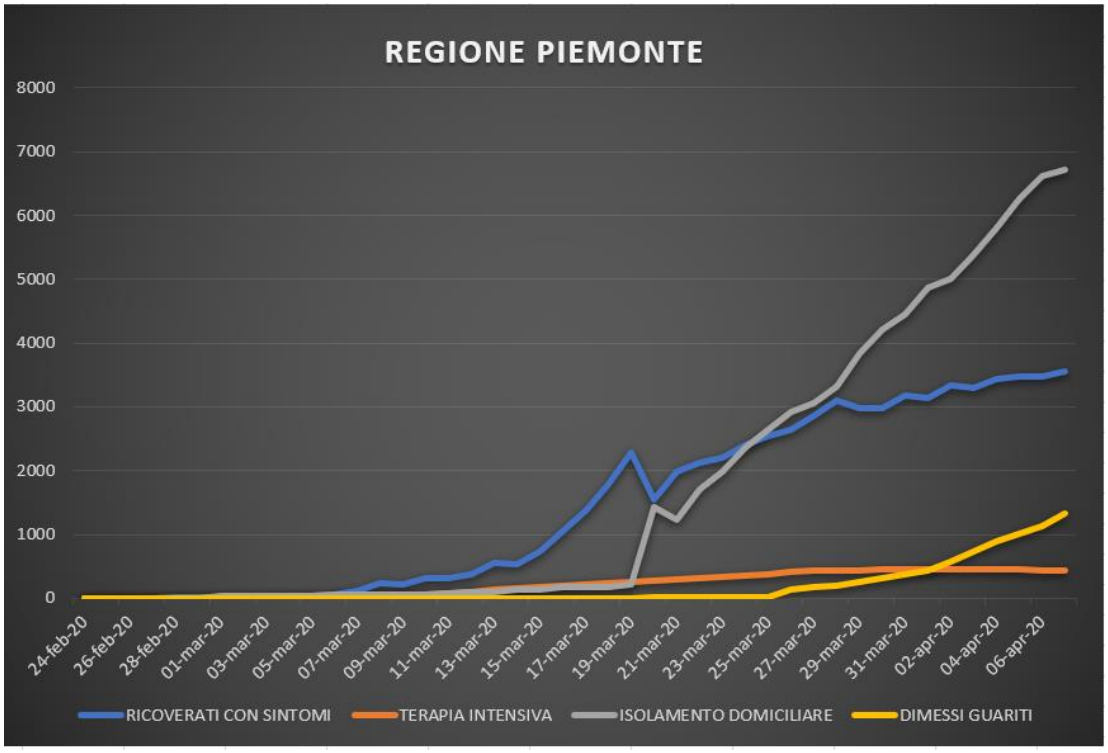
```
select p.denominazione_provincia,k.densita_abitativa,c.totale_casi
from (province p join
                    (
                    select max(densita_abitativa_prov) as densita_abitativa
                    from province
                    union
                    select min(densita_abitativa_prov) as densita_abitativa
                    from province
                    ) k on k.densita_abitativa=p.densita_abitativa_prov
                    )join covid_province c on c.codice_provincia=p.codice_provincia
where c.data='03-Mag-2020';
```

	DENOMINAZIONE_PROVINCIA	DENSITA_ABITATIVA	TOTALE_CASI
1	Nuoro	37	81
2	Napoli	2617	2468

Col primo join tra Province e K dove K contiene la massima e la minima densità abitativa delle province, stiamo escludendo tutte le province con densità abitative “intermedie”, associando di fatto ad ogni provincia la max e la min densità abitativa, dimodochè con l’ultimo join riusciamo a prelevare le informazioni riguardanti anche il totale casi di tali province.

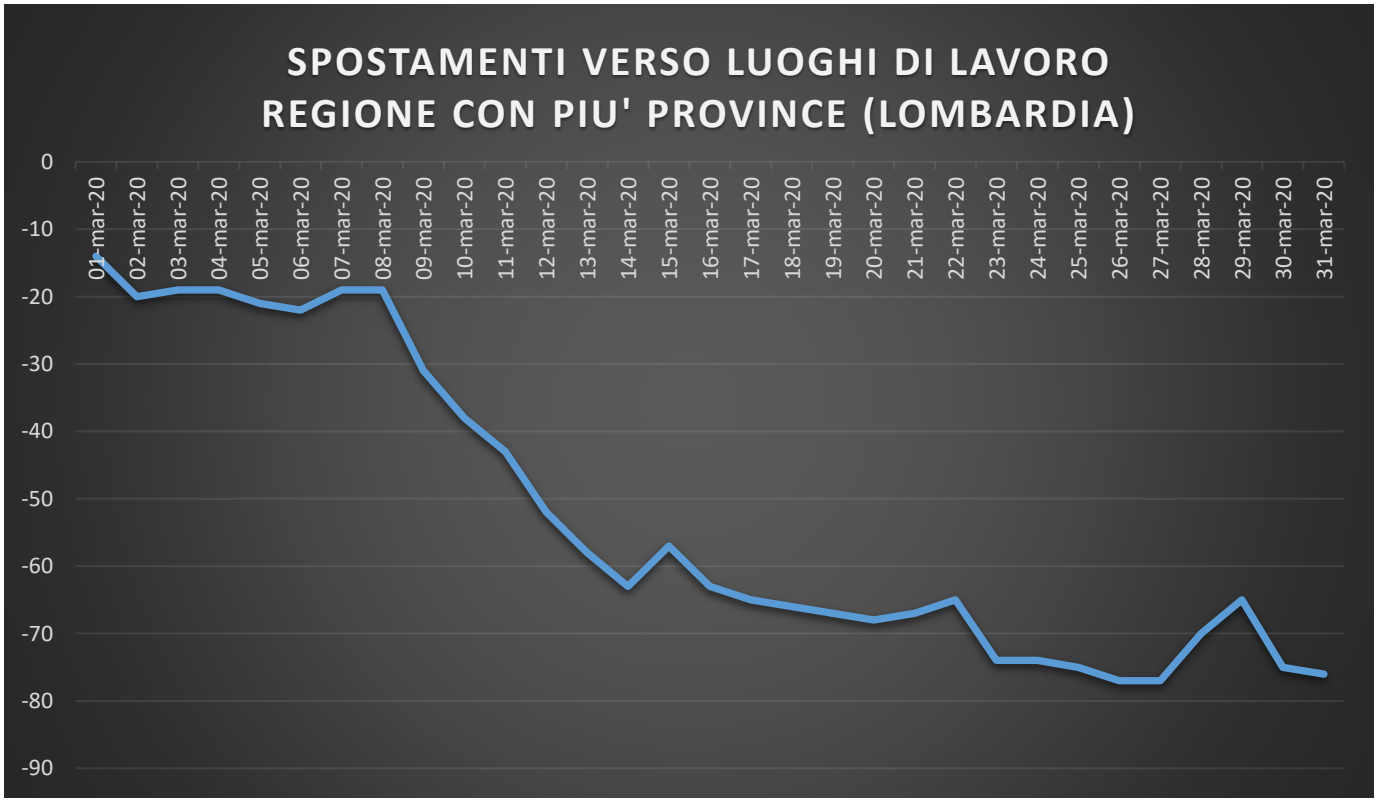
QUERY n.16: *Visualizza l’andamento per la regione Lombardia di ricoverati con sintomi, ricoverati in terapia intensiva, pazienti in isolamento domiciliare, pazienti dimessi e guariti*

```
select d.data,d.ricoverati_con_sintomi, d.terapia_intensiva, d.isolamento_domiciliare, d.dimessi_guariti
from dati_regionali d join regioni r on d.codice_regione=r.codice_regione
where r.denominazione_regione='Piemonte'
order by d.data;
```



QUERY n.17: *Graficare l'andamento degli spostamenti verso luoghi di lavoro della regione che ha il maggior numero di province nel mese di marzo*

```sql
--DEVO TROVARE LA REGIONE COL MASSIMO NUMERO DI PROVINCE
--Calcolo anzitutto il numero di province per ogni regione
select p.codice_regione,r.denominazione_regione,count(*) as numero_province
from province p join regioni r on r.codice_regione=p.codice_regione
group by p.codice_regione,r.denominazione_regione

--utilizzo la query precedente ed  una innestata per determinare la regione con il MASSIMO numero di province
--QUERY PER OTTERE LA REGIONE COL MASSIMO NUMERO DI PROVINCE
select p.codice_regione,r.denominazione_regione,count(*) as numero_province
from province p join regioni r on r.codice_regione=p.codice_regione
group by p.codice_regione,r.denominazione_regione
having count(*)>= all
(
select count(*)
from province p1
group by p1.codice_regione
)

--Determino gli spostamenti verso luoghi di lavoro utilizzando in join la query precedente

-- QUERY COMPLETA
select r.denominazione_regione, d.data, d.spostamenti_luoghi_lavoro_vp
from (dati_regionali d join regioni r on r.codice_regione=d.codice_Regione) join
(
      select p.codice_regione,r.denominazione_regione,count(*) as numero_province
      from province p join regioni r on r.codice_regione=p.codice_regione
      group by p.codice_regione,r.denominazione_regione
      having count(*)>= all
      (
      select count(*)
      from province p1
      group by p1.codice_regione
       )
) k on k.codice_regione=r.codice_regione
where extract(month from data)=3
order by d.data;
```



QUERY n.18: Selezionare la media deceduti per ogni mese e per ogni regione

```
select d.codice_regione,r.denominazione_regione,extract(month from d.data) as mese ,
                                                round(avg(d.deceduti),2)   as media_deceduti
from dati_regionali d join regioni r on r.codice_regione=d.codice_Regione
group by d.codice_regione,r.denominazione_regione,extract(month from d.data)
order by d.codice_regione;
```

QUERY n.19: *Trovare la percentuale totale_positivi/numero_tamponi della regione che ha effettuato il maggior numero di tamponi.*

```
--Anzitutto devo trovare qual è la regione che ha effettuato più tamponi
select r.denominazione_regione,r.codice_regione, max(d.tamponi) as tamponi
from dati_regionali d join regioni r on r.codice_Regione=d.codice_regione
group by r.denominazione_regione,r.codice_regione
having max(d.tamponi)>= all
(
select max(d.tamponi)
from dati_regionali d
group by d.codice_regione
)
--Regione con più tamponi effettuati: LOMBARDIA

--Uso la query precedente ed innesto
select    k.denominazione_regione,k.tamponi as tamponi_effettuati,d.totale_positivi,
          round(d.totale_positivi/k.tamponi,2)*100 as percentuale_positivi_su_tamponi
from (
      select r.denominazione_regione,r.codice_regione,max(d.tamponi) as tamponi
      from dati_regionali d join regioni r on r.codice_Regione=d.codice_Regione
      group by r.denominazione_regione,r.codice_regione
      having max(d.tamponi)>= all
        (
         select max(d.tamponi)
         from dati_regionali d
         group by d.codice_regione
         )
      ) k join dati_regionali d on k.codice_regione=d.codice_regione
where d.data='03-Mag-2020';
```

DENOMINAZIONE_REGIONE	TAMPONI_EFFETTUATI	TOTALE_POSITIVI	PERCENTUALE_POSITIVI_SU_TAMPONI
Lombardia	410857	36926	9

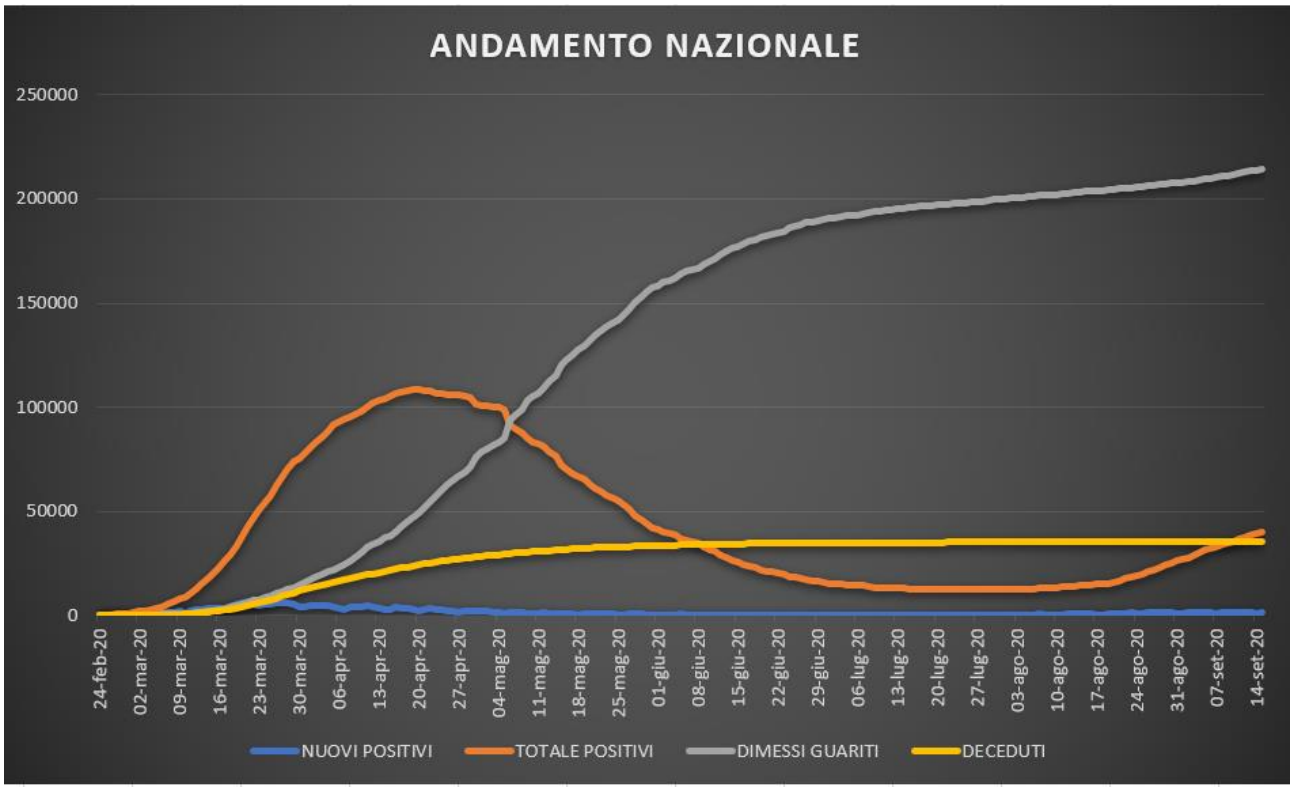
QUERY n.20: *Selezionare il totale casi delle 3 province che hanno la densità (scuole/superficie) maggiore.*

```
--Vista che calcola la densità di scuole per le province
create materialized view densità_scuole_provincia as
select codice_provincia,denominazione_provincia,superficie_prov,numero_scuole,
       round((numero_scuole/superficie_prov),2)*100 as Densita_Scuole_Sup
from province
order by densita_scuole_sup desc;

--Estraggo le prime 3 tuple dalla vista
select d.codice_provincia,d.denominazione_provincia,d.superficie_prov,
       d.numero_scuole,d.densita_scuole_sup,c.totale_casi
from (select * from densità_scuole_provincia
where rownum<4) d join covid_province c on c.codice_provincia=d.codice_provincia
where c.data='03-Mag-2020';
```

CODICE_PROVINCIA	DENOMINAZIONE_PROVINCIA	SUPERFICIE_PROV	NUMERO_SCUOLE	DENSITA_SCUOLE_SUP	TOTALE_CASI
63	Napoli	1178,93	3837	325	2468
108	Monza e della Brianza	405,41	685	169	4823
15	Milano	1575,65	2383	151	20068

QUERY n.21: *Graficare l'andamento nuovi positivi, totale positivi, dimessi_e deceduti a livello nazionale (fino al 15 settembre)*

```
select data, nuovi_positivi, totale_positivi, dimessi_guariti, deceduti
from COVID_ITALIA
order by data;
```



QUERY n.22: *Graficare l''andamento degli spostamenti verso luoghi di lavoro per la regione che ha il masssimo rapporto (porti+aeroporti)/superficie*

```
select k.denominazione_regione,d2.data,d2.spostamenti_luoghi_lavoro_vp
from(
        --query innestata (che a sua volta calcola il max con un altra query innestata)
        --in join con la quale calcolo la massima densita (aeroporti+porti)/superficie
        select r.codice_regione,r.denominazione_regione,max(round((r.numero_aeroporti+r.numero_porti)/r.superficie ,4)*100)  as massimo
        from regioni r
        group by  r.codice_regione,r.denominazione_regione
        having max(round((r.numero_aeroporti+r.numero_porti)/r.superficie ,4)*100) >=
        (
        select max(round((numero_aeroporti+numero_porti)/superficie ,4)*100) as max_densita_ap_sup
        from regioni
        )
    ) k join dati_regionali d2 on d2.codice_regione=k.codice_regione
where data <'27-apr-2020'  --GLI SPOSTAMENTI SONO FINO AL 26/04
order by data;
```

QUERY n.23: *Selezionare il numero di AZIENDE OSPEDALIERE per la regione che ha la popolazione più alta*

```
select r.codice_regione,r.denominazione_regione,r.popolazione_regione,s.aziende_ospedaliere
from regioni r join strutture_sanitarie s on s.codice_regione=r.codice_regione
where r.popolazione_regione=(select max(popolazione_regione)
                             from regioni);
```

CODICE_REGIONE	DENOMINAZIONE_REGIONE	POPOLAZIONE_REGIONE	AZIENDE_OSPEDALIERE
3	Lombardia	10060000	27

STEP 6: Stored Procedure & Trigger

*Vogliamo esporre, in linguaggio **PL/SQL** una serie di procedure/trigger al fine di esibire e consentire un'analisi più flessibile ed/o eventuali aggiornamenti automatici della base di dati qualora vogliano essere importati dati successivi al 03/5/2020.*
Il linguaggio PL/SQL è un linguaggio di programmazione utilizzato per accedere ed elaborare le informazioni gestite da un database.
Le stored procedure sono programmi che risiedono nel DBMS e che quindi posso essere elaborati più velocemente ed efficientemente.
Un trigger è una stored procedure che viene eseguita quando si verifica un particolare evento. In PL/SQL, i trigger sono blocchi di istruzioni che vengono attivati in maniera automatica a valle di operazioni DDL e DML sulla base dati.
Usando questo approccio si rende la base dati 'attiva' e non più come deposito di informazioni.
Premettiamo inoltre che le seguenti procedure sono altresì dotate di alcune eccezioni affinchè possano esser gestiti alla meglio maniera quelli che sono eventuali errori di digitazione dei parametri di input commessi dall'utente.

- ***Stored Procedure 1.***

La seguente SP calcola per una **finestra temporale di ingresso (data inizio, data fine)** il numero di dimessi guariti GIORNALIERI a livello nazionale.

```
CREATE OR REPLACE PROCEDURE DIMESSI_GUARITI_GIORNALIERI_NAZIONALE
(
DATA_IN   IN COVID_ITALIA.DATA%TYPE,
DATA_FIN  IN COVID_ITALIA.DATA%TYPE
)
AS
BEGIN
    DECLARE
    DIMESSI_GUARITI_GIORNALIERI COVID_ITALIA.DIMESSI_GUARITI%TYPE;
    DATA_A COVID_ITALIA.DATA%TYPE;
    CONTROLLO_DATA INTEGER;
    NOT_VALID EXCEPTION;

    BEGIN

            IF(DATA_IN<'24-FEB-2020' OR DATA_FIN>'03-MAG-2020' OR DATA_FIN<DATA_IN) THEN
                RAISE NOT_VALID;
            ELSE
                DATA_A:=DATA_IN;
                LOOP

                SELECT (C.DIMESSI_GUARITI-(SELECT D2.DIMESSI_GUARITI FROM COVID_ITALIA D2 WHERE D2.DATA=DATA_A-1)) INTO DIMESSI_GUARITI_GIORNALIERI
                FROM COVID_ITALIA C
                WHERE C.DATA=DATA_A;

                dbms_output.put_line(DATA_A||'|  DIMESSI_GUARITI_GIORNALIERI  :'|| DIMESSI_GUARITI_GIORNALIERI);

                DATA_A:=DATA_A+1;
                EXIT WHEN DATA_A>DATA_FIN;
                END LOOP;

            END IF;

            EXCEPTION
            WHEN NOT_VALID THEN
            dbms_output.put_line('DATA NON VALIDA ');
      END;
END DIMESSI_GUARITI_GIORNALIERI_NAZIONALE;

EXEC DIMESSI_GUARITI_GIORNALIERI_NAZIONALE ('03-MAR-2020','03-APR-2020');
```

EXEC DIMESSI_GUARITI_GIORNALIERI_NAZIONALE ('03-MAR-2020','03-APR-2020');

03-MAR-20| DIMESSI_GUARITI_GIORNALIERI :11

04-MAR-20| DIMESSI_GUARITI_GIORNALIERI :116

05-MAR-20| DIMESSI_GUARITI_GIORNALIERI :138

06-MAR-20| DIMESSI_GUARITI_GIORNALIERI :109

07-MAR-20| DIMESSI_GUARITI_GIORNALIERI :66

08-MAR-20| DIMESSI_GUARITI_GIORNALIERI :33

09-MAR-20| DIMESSI_GUARITI_GIORNALIERI :102

10-MAR-20| DIMESSI_GUARITI_GIORNALIERI :280

11-MAR-20| DIMESSI_GUARITI_GIORNALIERI :41

12-MAR-20| DIMESSI_GUARITI_GIORNALIERI :213

13-MAR-20| DIMESSI_GUARITI_GIORNALIERI :181

14-MAR-20| DIMESSI_GUARITI_GIORNALIERI :527

15-MAR-20| DIMESSI_GUARITI_GIORNALIERI :369

16-MAR-20| DIMESSI_GUARITI_GIORNALIERI :414

17-MAR-20| DIMESSI_GUARITI_GIORNALIERI :192

18-MAR-20| DIMESSI_GUARITI_GIORNALIERI :1084

19-MAR-20| DIMESSI_GUARITI_GIORNALIERI :415

20-MAR-20| DIMESSI_GUARITI_GIORNALIERI :689

21-MAR-20| DIMESSI_GUARITI_GIORNALIERI :943

22-MAR-20| DIMESSI_GUARITI_GIORNALIERI :952

23-MAR-20| DIMESSI_GUARITI_GIORNALIERI :408

24-MAR-20| DIMESSI_GUARITI_GIORNALIERI :894

25-MAR-20| DIMESSI_GUARITI_GIORNALIERI :1036

26-MAR-20| DIMESSI_GUARITI_GIORNALIERI :999

- ***Stored Procedure 2.***

La seguente SP prende **in ingresso una REGIONE e una DATA** e restituisce in uscita il numero di posti in terapia intensiva rimanenti.

```
CREATE OR REPLACE PROCEDURE POSTI_TERAPIA_INTENSIVA(
REGIONE IN REGIONI.DENOMINAZIONE_REGIONE%TYPE,
GIORNO IN DATI_REGIONALI.DATA%TYPE
)
AS
    BEGIN
        DECLARE
            POSTI_RIMANENTI INTEGER;
            COSTANTE INTEGER := 0;
            ESAURIMENTO_POSTI EXCEPTION;
        BEGIN
        IF (GIORNO < '25-FEB-2020'OR GIORNO > '03-MAG-2020') THEN
            dbms_output.put_line ('DATI NON PRESENTI PER IL GIORNO: '||GIORNO);
        ELSE
            SELECT (D.POSTI_LETTO_TERAPIA_INTENSIVA_TOTALI - D.TERAPIA_INTENSIVA) INTO POSTI_RIMANENTI
            FROM DATI_REGIONALI D JOIN REGIONI R ON D.CODICE_REGIONE=R.CODICE_REGIONE
            WHERE R.DENOMINAZIONE_REGIONE=REGIONE AND D.DATA =GIORNO;
                IF(POSTI_RIMANENTI>COSTANTE) THEN
                    dbms_output.put_line('IL NUMERO DI POSTI RIMANENTI IN TERAPIA INTENSIVA NELLA REGIONE ' ||REGIONE||' IL GIORNO' ||GIORNO||' è:' ||POSTI_RIMANENTI||'.');
                ELSE
                    RAISE ESAURIMENTO_POSTI;
                END IF;
        END IF;

        EXCEPTION WHEN ESAURIMENTO_POSTI THEN
        dbms_output.put_line ('IL NUMERO DI POSTI RIMANENTI IN TERAPIA INTENSIVA NELLA REGIONE '||REGIONE||' IL GIORNO ' ||GIORNO|| 'è ESAURITO');

        END;
END POSTI_TERAPIA_INTENSIVA;


EXEC POSTI_TERAPIA_INTENSIVA ('Campania','03-FEB-2020');
EXEC POSTI_TERAPIA_INTENSIVA ('Campania','06-APR-2020');
```

*La procedura inizia con un controllo sul corretto inserimento della data. Se la verifica produce successo, si entra nel blocco **else** nel quale si calcola il numero di posti in terapia intensiva per la regione e la data specificate in ingresso.*

EXEC POSTI_TERAPIA_INTENSIVA ('Campania','03-FEB-2020');
EXEC POSTI_TERAPIA_INTENSIVA ('Campania','06-APR-2020');

```
DATI NON PRESENTI PER IL GIORNO: 03-FEB-20

Procedura PL/SQL completata correttamente.
IL NUMERO DI POSTI RIMANENTI IN TERAPIA INTENSIVA NELLA REGIONE Campania IL GIORNO06-APR-20 è:339.

Procedura PL/SQL completata correttamente.
```

- **TRIGGER**

Si è proceduti all'implementazione dei seguenti trigger:

1. Tale trigger automatizza l'inserimento dei dati sulle tabelle normalizzate a partire da inserimenti sulla tabella master COVID;

2. Tale trigger automatizza l'inserimento dei dati sulle tabelle normalizzate a partire da inserimenti sulla tabella master COVID_REGIONI;

3. Tale trigger aggiorna automaticamente il codice delle province in COVID_PROVINCE a valle di un aggiornamento su PROVINCE;

4. Tale trigger aggiorna automaticamente il codice delle province in DATI_REGIONALI e STRUTTURE_SANITARIE a valle di un aggiornamento su REGIONI;

```
--TRIGGER 1 PER INSERT SU TABELLA COVID
CREATE OR REPLACE TRIGGER TAB_COVID
AFTER INSERT ON COVID
FOR EACH ROW
BEGIN
INSERT INTO COVID_PROVINCE (DATA,CODICE_PROVINCIA,TOTALE_CASI) VALUES (:NEW.DATA,:NEW.CODICE_PROVINCIA,:NEW.TOTALE_CASI);
END TAB_COVID;

--TRIGGER 2 PER INSERT SU TABELLA COVID REGIONI
CREATE OR REPLACE TRIGGER TAB_COVID_REGIONI
AFTER INSERT ON COVID_REGIONI
FOR EACH ROW
BEGIN
INSERT INTO DATI_REGIONALI (DATA,CODICE_REGIONE,RICOVERATI_CON_SINTOMI,TERAPIA_INTENSIVA,TOTALE_OSPEDALIZZATI,
                            ISOLAMENTO_DOMICILIARE,TOTALE_POSITIVI,VARIAZIONE_TOTALE_POSITIVI,NUOVI_POSITIVI,
                            DIMESSI_GUARITI,DECEDUTI,TOTALE_CASI,TAMPONI,CASI_TESTATI,NOTE_IT,NOTE_EN)
VALUES (:NEW.DATA,:NEW.CODICE_REGIONE,:NEW.RICOVERATI_CON_SINTOMI,:NEW.TERAPIA_INTENSIVA,:NEW.TOTALE_OSPEDALIZZATI,
        :NEW.ISOLAMENTO_DOMICILIARE,:NEW.TOTALE_POSITIVI,:NEW.VARIAZIONE_TOTALE_POSITIVI,:NEW.NUOVI_POSITIVI,:NEW.DIMESSI_GUARITI,:NEW.DECEDUTI,
        :NEW.TOTALE_CASI,:NEW.TAMPONI,:NEW.CASI_TESTATI,:NEW.NOTE_IT,:NEW.NOTE_EN);
END TAB_COVID_REGIONI;

--TRIGGER 3 AGGIORNAMENTO DEL CODICE DELLE PROVINCE
CREATE OR REPLACE TRIGGER COD_PROV
AFTER UPDATE ON PROVINCE
FOR EACH ROW
BEGIN
UPDATE COVID_PROVINCE SET CODICE_PROVINCIA=:NEW.CODICE_PROVINCIA WHERE CODICE_PROVINCIA=:OLD.CODICE_PROVINCIA;
END COD_PROV;

--TRIGGER 4 AGGIORNAMENTO DEL CODICE DELLE REGIONI
CREATE OR REPLACE TRIGGER COD_REG
AFTER UPDATE ON REGIONI
FOR EACH ROW
BEGIN
UPDATE DATI_REGIONALI SET CODICE_REGIONE=:NEW.CODICE_REGIONE WHERE CODICE_REGIONE=:OLD.CODICE_REGIONE;
UPDATE STRUTTURE_SANITARIE SET CODICE_REGIONE=:NEW.CODICE_REGIONE WHERE CODICE_REGIONE=:OLD.CODICE_REGIONE;
END COD_REG;
```

STEP 7: Viste e Indici

Siccome Oracle, di default crea degli indici definiti sulle *primary key*, se ne aggiungono ora degli altri relativi a campi frequentemente utilizzati per le query, cioè la "*denominazione della provincia*" e la "*denominazione della regione*", "*totale casi*" per "*covid_province*" e "*totale casi*" per "*dati_regionali*".

```
CREATE INDEX INDICE_DENOMINAZIONE_REGIONE    ON REGIONI (DENOMINAZIONE_REGIONE);
CREATE INDEX INDICE_DENOMINAZIONE_PROVINCIA  ON PROVINCE (DENOMINAZIONE_PROVINCIA);
CREATE INDEX INDICE_TOTALE_CASI_REGIONE      ON DATI_REGIONALI(TOTALE_CASI);
CREATE INDEX INDICE_TOTALE_CASI_PROVINCIA    ON COVID_PROVINCE (TOTALE_CASI);
```

Ordunque, per ottimizzare i tempi di esecuzione delle query riportiamo di seguito alcune viste riportate già nelle pagine precedenti per alcune query.

```
--LE VISTE SONO GIA' STATE CREATE ED UTILIZZATE NELLE QUERY

--PER OGNI PROVINCIA MOSTRA IL MASSIMO TOTALE CASI (OSSIA IL TOTALE CASI AL 3 MAGGIO)
create materialized view max_casi_per_prov_e_reg as
select p.codice_provincia,r.codice_regione,r.denominazione_regione,p.denominazione_provincia,c.totale_casi
from (covid_province c join province p on p.codice_provincia=c.codice_provincia) join regioni r
      on r.codice_Regione=p.codice_regione
where c.data='03-Mag-2020'
order by c.totale_casi desc;

--PER OGNI REGIONE MOSTRA IL TOTALE CASI MASSIMO (AL 3 MAGGIO PRESO DALLA VISTA PRECEDENTE)
create materialized view max_casi_per_regione as
select codice_regione,denominazione_regione,max(totale_casi) as Max_totale_casi_regione
from max_casi_per_prov_e_reg
group by codice_regione,denominazione_Regione;


--DENSITA' SCUOLE/PROVINCE
create materialized view densità_scuole_provincia as
select codice_provincia,denominazione_provincia,superficie_prov,numero_scuole,
      round((numero_scuole/superficie_prov),2)*100 as Densita_Scuole_Sup
from province
order by densita_scuole_sup desc;
```