UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

BASE DI DATI PER LA GESTIONE DEI DATI RELATIVI

ALL'ANDAMENTO DEL CONTAGIO DEL VIRUS COVID-19

CORSO BASI DI DATI DOCENTE:MASCIARI

ANNO ACCADEMICO: 2019/2020

Progetto realizzato da:

CRISTIANO ANGELO – N46004344

DE SIMONE VINCENZO-N46004345

CAPITOLO 1

ANALISI DELLE SPECIFICHE

1.1 Informazioni generali

Si vuole progettare una base di dati che contenga informazioni relative all'andamento del contagio del virus covid-19 in Italia, in particolar modo nelle province e regioni italiane.

Tale base di dati permetterà un rapido consulto sui dati relativi ai casi di covid-19 in Italia.

1.2 Dati Generali

Informazioni sulle province

Per ogni provincia bisognerà memorizzare la sua sigla,la sua denominazione, il suo codice identificativo e la regione di appartenenza e ulteriori informazioni che verrano specificate in seguito.

Informazioni sulle regioni

Per ogni regione bisognerà conservare le informazioni relative alla sua denominazione,al suo codice,il numero di infrastrutture in quella determinata regione.

Informazioni sui contagi

La base di dati dovrà memorizzare tutti i dati relativi ai contagi giornalieri in Italia.

Dovranno essere gestite anche le informazioni relative al numero di tamponi effettuati,numero di ricoveri,numero di dimessi,numero di deceduti giornalieri.

CAPITOLO 2

Progettazione della base di dati

2.1 Progettazione fisica

In genere una base di dati viene progettata in tre fasi distinte:concettuale,logica e fisica.

Questa base di dati è stata realizzata partendo dalla progettazione fisica.

I dati relativi al contagio sono stati acquisiti da github e si è effettuata, attraverso la distribuzione Oracle XE 18c, la creazione di una tabella master e il suo popolamento.

Si è utilizzato uno strumento per convertire i file presenti su github dal formato csv a comandi sql.

2.2 SPECIFICHE SUI TIPI DI DATO

Per ottenere una minimizzazione efficiente delle dimensioni del tablespace sono state effettuate tipi di scelte di dato e dei relativi dimensionamenti a scopo di ottimizzare lo spazio occupato in memoria e i tempi di interrogazione del database.

Sono stati utilizzati i seguenti tipi di dato: NUMBER (p,s): Consente la memorizzazione di valori numerici interi, positivi e negativi, dei numeri a virgola fissa e di quelli a virgola mobile. I parametri “p” e “s” indicano rispettivamente la precisione e la scala. La precisione può avere un valore da 1 a 38, mentre la scala un valore da -84 a 127. La quantità di memoria occupata da tale tipo è ricavabile dall’equazione: [(p/2)+2].

VARCHAR2 (n): A differenza del precedente, questo tipo alfanumerico è a lunghezza variabile e può memorizzare fino a 4000 caratteri. Non esiste un valore di default per questo tipo di dati, pertanto Oracle solleverà un errore se non sarà specificato il valore di n. La quantità di memoria occupata da tale tipo è di un byte per ogni carattere

DIMENSIONAMENTO DATE: È usato per memorizzare date e orari. Un campo di questo tipo può contenere informazioni del tipo secolo, anno, mese, giorno, ora, minuti e secondi. Oracle fornisce svariate e utilissime funzioni per operare e trasformare questo tipo di dati. Il default di memorizzazione è GG-MES-AA ovvero due cifre per il giorno, tre lettere per il mese e due cifre per l'anno. CLOB: Ammette attributi di tipo "descrizione" per permettere testi anche molto lunghi (fino a 4 Gb). Il tipo CLOB è usato per la memorizzazione di testi di grandi dimensioni e in Oracle occupa solo 4 byte in quanto non contiene i dati veri e propri ma solo un puntatore all'effettivo tablespace che contiene il testo.

INT: E' l'equivalente di Integer; i dati di tipo INT rappresentano numeri interi composti da 4 byte. Puo' contenere oltre i 4 miliardi di valori.

2.3 Creazione degli oggetti della base di dati

Gli oggetti della base sono stati creati utilizzando il comando CREATE TABLE.

Di seguito è riportato il codice SQL dell'intero schema relazionale della base di dati

```
CREATE TABLE "SYSTEM"."MASTER"
  (   "DATA" VARCHAR2(19 BYTE),
  "STATO" VARCHAR2(3 BYTE),
  "CODICE_REGIONE" NUMBER(*,0),
  "DENOMINAZIONE_REGIONE" VARCHAR2(21 BYTE),
  "CODICE_PROVINCIA" NUMBER(*,0),
```

```
  "DENOMINAZIONE_PROVINCIA" VARCHAR2(36 BYTE),
  "SIGLA_PROVINCIA" VARCHAR2(2 BYTE),
  "LAT" NUMBER(11,8),
  "LONGE" NUMBER(11,9),
  "TOTALE_CASI" NUMBER(*,0),
  "NOTE_IT" VARCHAR2(11 BYTE),
  "NOTE_EN" VARCHAR2(11 BYTE)
  ) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "SYSTEM" ;
REM INSERTING into SYSTEM.MASTER
SET DEFINE OFF;
```

SQL POPOLAMENTO DELLA TABELLA MASTER

```
Insert into SYSTEM.MASTER
(DATA,STATO,CODICE_REGIONE,DENOMINAZIONE_REGIONE,CODICE_PROVINCIA,DENOMINAZIONE_PROVINCI
A,SIGLA_PROVINCIA,LAT,LONGE,TOTALE_CASI,NOTE_IT,NOTE_EN) values ('2020-02
24T18:00:00','ITA','13','Abruzzo','69','Chieti','CH','42,35103167','14,16754574','0',null,null);
Insert into SYSTEM.MASTER
(DATA,STATO,CODICE_REGIONE,DENOMINAZIONE_REGIONE,CODICE_PROVINCIA,DENOMINAZIONE_PROVINCI
A,SIGLA_PROVINCIA,LAT,LONGE,TOTALE_CASI,NOTE_IT,NOTE_EN) values ('2020-02-
24T18:00:00','ITA','13','Abruzzo','66','L''Aquila','AQ','42,35122196','13,39843823','0',null,null);
Insert into SYSTEM.MASTER
(DATA,STATO,CODICE_REGIONE,DENOMINAZIONE_REGIONE,CODICE_PROVINCIA,DENOMINAZIONE_PROVINCI
A,SIGLA_PROVINCIA,LAT,LONGE,TOTALE_CASI,NOTE_IT,NOTE_EN) values ('2020-02-
24T18:00:00','ITA','13','Abruzzo','68','Pescara','PE','42,46458398','14,21364822','0',null,null);
Insert into SYSTEM.MASTER
(DATA,STATO,CODICE_REGIONE,DENOMINAZIONE_REGIONE,CODICE_PROVINCIA,DENOMINAZIONE_PROVINCI
A,SIGLA_PROVINCIA,LAT,LONGE,TOTALE_CASI,NOTE_IT,NOTE_EN) values ('2020-02-
24T18:00:00','ITA','13','Abruzzo','67','Teramo','TE','42,6589177','13,70439971','0',null,null);
```

DEFINIZIONE DEI VINCOLI DI INTEGRITA’

Dopo la creazione delle tabelle si è deciso di implementare i vincoli di integrità utilizzando i comandi di ALTER TABLE. Si fa notare che i parametri di INITIAL e NEXT relativi allo STORAGE di ogni tabella sono calcolati arrotondando per eccesso la conversione, per sopperire all’approssimazione stabilita 1000 byte = 1 Kbyte e 1000 Kbyte = 1 Mbyte. E’ stata fatta la scelta di applicare il vincolo NOT NULL ai campi ritenuti necessari dal team per dare consistenza ai dati memorizzati. Laddove

inoltre si è presentata la necessità (per la creazione per esempio di chiavi primarie di più campi) è stato usato il campo CONSTRAINT nella creazione delle tabelle.

```
ALTER TABLE "SYSTEM"."MASTER" MODIFY ("DATA" NOT NULL ENABLE);
ALTER TABLE "SYSTEM"."MASTER" MODIFY ("STATO" NOT NULL ENABLE);
ALTER TABLE "SYSTEM"."MASTER" MODIFY ("CODICE_REGIONE" NOT NULL ENABLE);
ALTER TABLE "SYSTEM"."MASTER" MODIFY ("DENOMINAZIONE_REGIONE" NOT NULL ENABLE);
ALTER TABLE "SYSTEM"."MASTER" MODIFY ("CODICE_PROVINCIA" NOT NULL ENABLE);
ALTER TABLE "SYSTEM"."MASTER" MODIFY ("DENOMINAZIONE_PROVINCIA" NOT NULL ENABLE);
ALTER TABLE "SYSTEM"."MASTER" MODIFY ("LAT" NOT NULL ENABLE);
ALTER TABLE "SYSTEM"."MASTER" MODIFY ("LONGE" NOT NULL ENABLE);
ALTER TABLE "SYSTEM"."MASTER" MODIFY ("TOTALE_CASI" NOT NULL ENABLE);
ALTER TABLE "SYSTEM"."MASTER" ADD CONSTRAINT "PROVINCIA_PK" PRIMARY KEY ("DATA",
"CODICE_REGIONE", "CODICE_PROVINCIA")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "SYSTEM"  ENABLE;
```

SCOMPOSIZIONE

Nella tabella master vengono individuate le seguenti Dipendenze funzionali:

Codice_provincia→(denominazione_provincia,lat,longe,codice_regione)

Codice_regione→(denominazione_regione,stato)

Si ottengono ben 3 tabelle.

In particolare la 1NF è verificata in quanto , il modello relazionale è per sua stessa natura in 1NF poiché mette a disposizione ,per la definizione dei domini,solo tipi

atomici. Inoltre basterà sviluppare gli attributi multivalore ed estrarre gli attributi composti.

La 2NF è verificata in quanto poiché lo schema di relazione è in 1NF ed ogni attributo non primo dello schema di relazione è in dipendenza funzionale completa da ogni chiave dello schema.

La 3NF non è verificata .

```
CREATE TABLE "SYSTEM"."PROVINCE2"
( "COD_REGIONE" NUMBER(*,0),
"CODICE_PROVINCIA" NUMBER(*,0),
"DENOMINAZIONE_PROVINCIA" VARCHAR2(36 BYTE),
"SIGLA_PROVINCIA" VARCHAR2(2 BYTE),
"LAT" NUMBER(11,8),
"LONGE" NUMBER(11,9),
) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT
CELL_FLASH_CACHE DEFAULT)
TABLESPACE "SYSTEM" ; REM INSERTING into SYSTEM.PROVINCE2 SET DEFINE OFF
ALTER TABLE PROVINCE2
ADD (Popolazione  NUMBER(9,3)  ,Superficie FLOAT,Densità    INTEGER  ,Ncomuni  INTEGER,Scuole
INTEGER,N_ospedali INTEGER );


ALTER TABLE "SYSTEM"."PROVINCE2" MODIFY ("COD_REGIONE" NOT NULL ENABLE);
ALTER TABLE "SYSTEM"."PROVINCE2" MODIFY ("SIGLA_PROVINCIA" NOT NULL ENABLE);
ALTER TABLE "SYSTEM"."PROVINCE2" MODIFY ("LAT" NOT NULL ENABLE);
ALTER TABLE "SYSTEM"."PROVINCE2" MODIFY ("LONGE" NOT NULL ENABLE);
ALTER TABLE "SYSTEM"."PROVINCE2" ADD PRIMARY KEY ("CODICE_PROVINCIA")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "SYSTEM"  ENABLE;


-- Ref Constraints for Table PROVINCE2
------------------------------------------------------------
```

```
ALTER TABLE "SYSTEM"."PROVINCE2" ADD FOREIGN KEY ("COD_REGIONE")
 REFERENCES "SYSTEM"."REGIONI2" ("CODICE_REGIONE") ENABLE;




CREATE TABLE "SYSTEM"."REGIONI2"
   (   "STATO" VARCHAR2(3 BYTE),
   "CODICE_REGIONE" NUMBER(*,0),
   "DENOMINAZIONE_REGIONE" VARCHAR2(21 BYTE),
   "POPOLAZIONE" NUMBER(12,3),
   "SUPERFICIE" FLOAT(126),
   "DENSITA'" FLOAT(126),
   "NCOMUNI" INTEGER,
   "NPROVINCE" INTEGER,
   "AUTOSTRADE" NUMBER(*,0),
   "FERROVIE" NUMBER(*,0),
   "AEROPORTI" NUMBER(*,0)
   ) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "SYSTEM" ;
REM INSERTING into SYSTEM.REGIONI2
SET DEFINE OFF;
ALTER TABLE "SYSTEM"."REGIONI2" MODIFY ("STATO" NOT NULL ENABLE);
ALTER TABLE "SYSTEM"."REGIONI2" ADD CONSTRAINT "SYS_C007693" PRIMARY KEY ("CODICE_REGIONE")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "SYSTEM"  ENABLE;

ALTER TABLE REGIONI2 ADD  (Nprovince INTEGER,Ncomuni INTEGER,Densità FLOAT,Superficie
FLOAT,Popolazione NUMBER(9,3),autostrade INTEGER,ferrovie integer,aeroporti integer);




CREATE TABLE "SYSTEM"."CONTAGI2"
   (   "DATA" VARCHAR2(19 BYTE),
   "COD_PROVINCIA" NUMBER(*,0),
   "TOTALE_CASI" NUMBER(*,0),
   "NOTE_IT" VARCHAR2(11 BYTE),
   "NOTE_EN" VARCHAR2(11 BYTE)
   ) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "SYSTEM" ;
REM INSERTING into SYSTEM.CONTAGI2
SET DEFINE OFF;



ALTER TABLE "SYSTEM"."CONTAGI2" MODIFY ("DATA" NOT NULL ENABLE);
  ALTER TABLE "SYSTEM"."CONTAGI2" MODIFY ("COD_PROVINCIA" NOT NULL ENABLE);
```

```
 ALTER TABLE "SYSTEM"."CONTAGI2" MODIFY ("TOTALE_CASI" NOT NULL ENABLE);
 ALTER TABLE "SYSTEM"."CONTAGI2" ADD PRIMARY KEY ("DATA", "COD_PROVINCIA")
 USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
 STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
 TABLESPACE "SYSTEM"  ENABLE;
--------------------------------------------------------
--  Ref Constraints for Table CONTAGI2
--------------------------------------------------------

 ALTER TABLE "SYSTEM"."CONTAGI2" ADD FOREIGN KEY ("COD_PROVINCIA")
   REFERENCES "SYSTEM"."PROVINCE2" ("CODICE_PROVINCIA") ENABLE;
```

```
UPDATE PROVINCE2 SET Popolazione = 125666,Superficie = '3260,90',Densità = 39,Ncomuni = 74
,Scuole=247,N_ospedali=6 WHERE sigla_provincia='AO';

UPDATE PROVINCE2 SET Popolazione = 84379,Superficie = '1535,24',Densità = 55,Ncomuni =
52,Scuole=162,N_ospedali=4 WHERE sigla_provincia='IS';

UPDATE REGIONI2 SET Regione = 'Abruzzo',Popolazione = 1311580,Superficie = ‘10.831,84’,Densità =
121,NComuni = 305,NProvince = 4,Autostrade=5,Ferrovie=7,Aeroporti=1 WHERE Regione= 'Abruzzo';

UPDATE REGIONI2 SET Regione = 'Basilicata',Popolazione = 562.869,Superficie = 10.073,32,Densità =
56,NComuni = 131,NProvince = 2,Autostrade=1,Ferrovie=9,Aeroporti=0 WHERE Regione= 'Basilicata';
```

Ciascuna tabella è stata arricchita di ulteriori dati.Tuttavia non è stata possibile soddisfare la richiesta di arricchire ogni data con i dati relativi ai tamponi effettuati, ospedalizzati, dimessi, guariti etc…,

O meglio non riuscendo a trovare un dataset che contenesse i dati relativi ai tamponi,morti e guariti giornalieri, si è deciso di creare una nuova tabella contenente questi dati per regione, in quanto disponibili su github.

La creazione di questa nuova tabella ha portato ovviamente a dei problemi, nell’operazioni di join e nel soddisfare le forme normali, pertanto la progettazione della base di dati non è stata ottimizzata al meglio.

```
CREATE TABLE "SYSTEM"."CONTAGIREGIONI2"
  (   "DATA" VARCHAR2(19 BYTE),
  "CODICE_REGIONE" NUMBER(*,0),
  "RICOVERATI_CON_SINTOMI" NUMBER(*,0),
  "TERAPIA_INTENSIVA" NUMBER(*,0),
  "TOTALE_OSPEDALIZZATI" NUMBER(*,0),
  "ISOLAMENTO_DOMICILIARE" NUMBER(*,0),
  "TOTALE_POSITIVI" NUMBER(*,0),
  "VARIAZIONE_TOTALE_POSITIVI" NUMBER(*,0),
  "NUOVI_POSITIVI" NUMBER(*,0),
  "DIMESSI_GUARITI" NUMBER(*,0),
  "DECEDUTI" NUMBER(*,0),
  "TOTALE_CASI" NUMBER(*,0),
  "TAMPONI" NUMBER(*,0),
  "CASI_TESTATI" NUMBER(*,0)
  ) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "SYSTEM" ;
REM INSERTING into SYSTEM.CONTAGIREGIONI2
SET DEFINE OFF;
Insert into SYSTEM.CONTAGIREGIONI2
(DATA,CODICE_REGIONE,RICOVERATI_CON_SINTOMI,TERAPIA_INTENSIVA,TOTALE_OSPEDALIZZATI,ISOLAME
NTO_DOMICILIARE,TOTALE_POSITIVI,VARIAZIONE_TOTALE_POSITIVI,NUOVI_POSITIVI,DIMESSI_GUARITI,DEC
EDUTI,TOTALE_CASI,TAMPONI,CASI_TESTATI) values ('2020-02-
24T18:00:00','13','0','0','0','0','0','0','0','0','0','0','5',null);


INSERT INTO SYSTEM.CONTAGIREGIONI2
(DATA,CODICE_REGIONE,RICOVERATI_CON_SINTOMI,TERAPIA_INTENSIVA,TOTALE_OSPEDALIZZATI,ISOLAME
NTO_DOMICILIARE,TOTALE_POSITIVI,VARIAZIONE_TOTALE_POSITIVI,NUOVI_POSITIVI,DIMESSI_GUARITI,DEC
EDUTI,TOTALE_CASI,TAMPONI,CASI_TESTATI) values ('2020-02-
24T18:00:00','13','0','0','0','0','0','0','0','0','0','0','0','5',null);

ALTER TABLE "SYSTEM"."CONTAGIREGIONI2" MODIFY ("DATA" NOT NULL ENABLE);
  ALTER TABLE "SYSTEM"."CONTAGIREGIONI2" ADD CONSTRAINT "PK" PRIMARY KEY ("DATA",
"CODICE_REGIONE")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "SYSTEM"  ENABLE;
--------------------------------------------------------
--  Ref Constraints for Table CONTAGIREGIONI2
--------------------------------------------------------

  ALTER TABLE "SYSTEM"."CONTAGIREGIONI2" ADD CONSTRAINT "FK" FOREIGN KEY ("CODICE_REGIONE")
    REFERENCES "SYSTEM"."REGIONI2" ("CODICE_REGIONE") ENABLE;
```

CAPITOLO 3

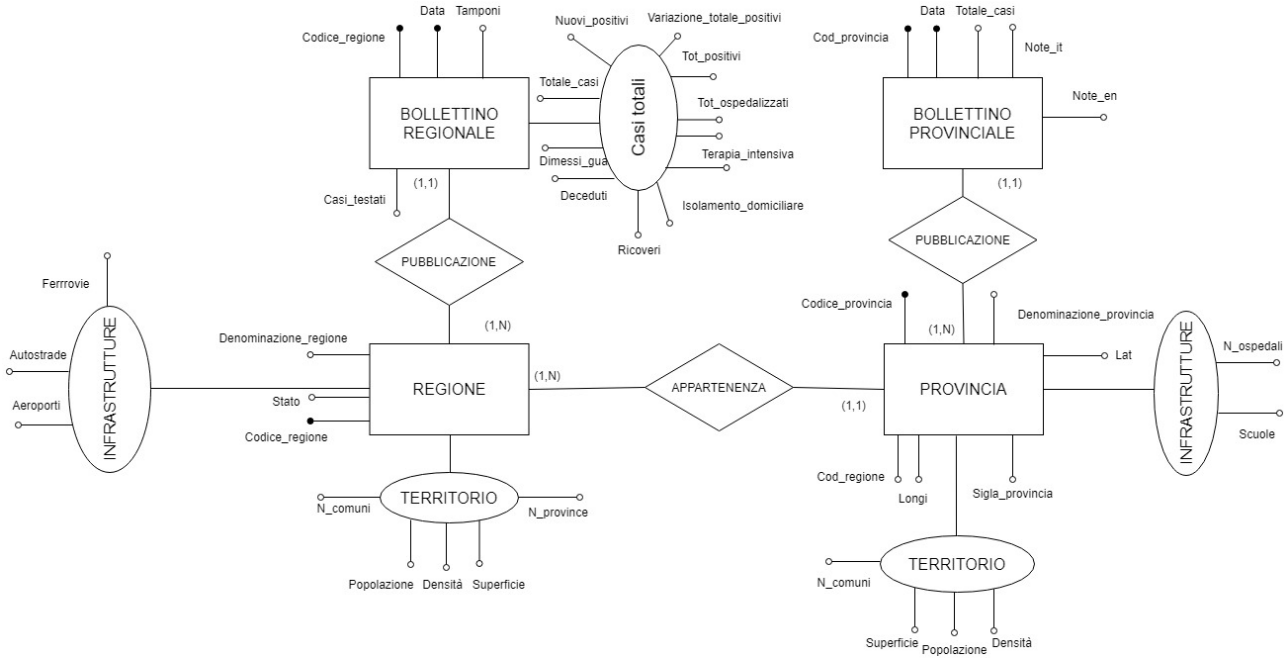
3.1 PROGETTAZIONE CONCETTUALE

Il primo passo per la progettazione della base di dati è quello di definire la struttura dello schema E/R portante, con i concetti fondamentali. Si passa poi alla sua estensione e raffinamento, inserendo via via tutte le specifiche precedentemente analizzate, per giungere a quello che è lo schema E/R finale fulcro della nostra progettazione concettuale.

3.1.2 SCHEMA E/R

Si vanno ora ad esaminare le specifiche, inserendo quelli che sono gli attributi delle nostre entità, oltre alle analisi delle cardinalità delle nostre associazioni.

1. Una PROVINCIA può appartenere ad una sola e unica REGIONE (1,1). Al contrario, ad una REGIONE possono appartenere una o piu' PROVINCE (1,N). La cardinalità dell'associazione APPARTENENZA risulta quindi (1 a N).
2. Una PROVINCIA può pubblicare piu' BOLLETTINI (1,N) , mentre un determinato BOLLETTINO ,in un determinato giorno ,è specifico ed è pubblicato da una determinata PROVINCIA(1,1). La cardinalità dell'associazione PUBBLICAZIONE è (1 a N)
3. Una REGIONE può pubblicare piu' BOLLETTINI (1,N) mentre un determinato BOLLETTINO ,in un determinato giorno, è specifico ed è pubblicato da una determinata REGIONE1,1). La cardinalità dell'associazione PUBBLICAZIONE è (1 a N)



3.2 Progettazione Logica

La fase successiva è la progettazione logica che prevede la definizione dello schema in un insieme di vincoli e relazioni.Per passare alla progettazione logica bisogna effettuare opportune semplificazioni e trasformazioni .

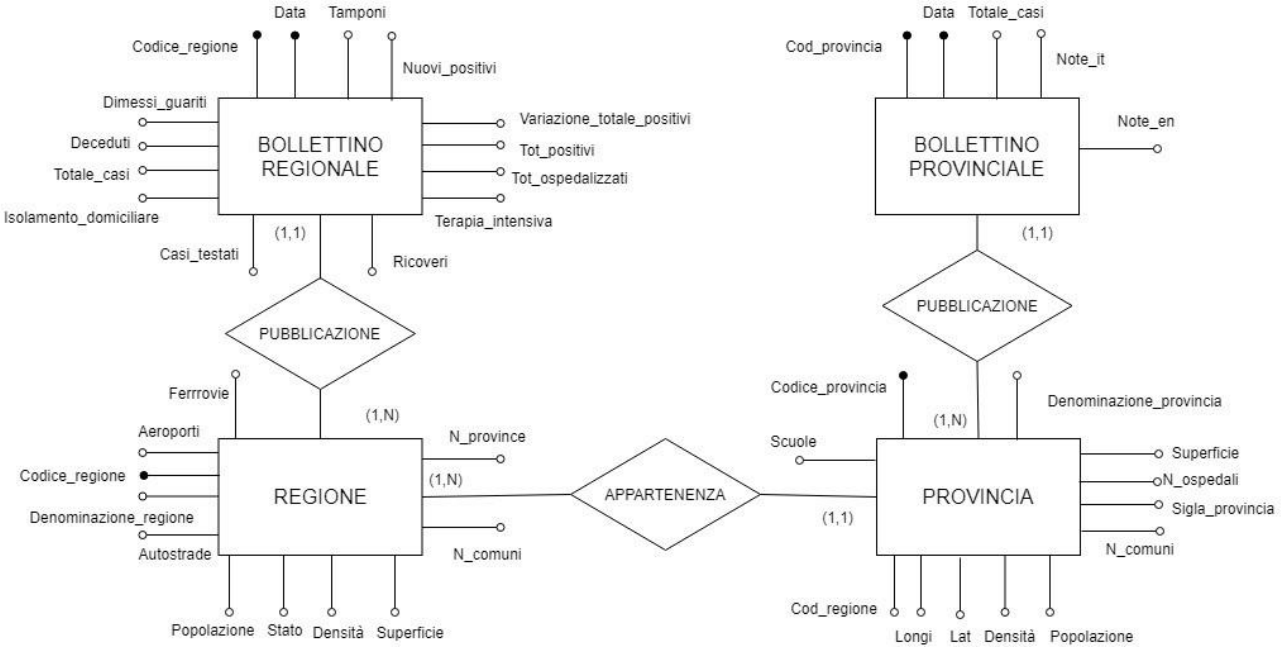
Essa si divide in due fasi:

Trasformazione:In questa fase, bisogna semplificalre alcuni costrutti del modello E/R non direttamente traducibili in quello relazionale,ossia attributi composti e multivalore e la risoluzione delle generalizzazioni qualora ci fossero.

Traduzione: Consiste nella effettiva traduzione del modello E/R semplificato nello schema relazionale e si effettua seguendo determinate regole.

Fase di trasformazione

In riferimento al modello sopra rappresentato, la trasformazione dello schema prevede la "semplificazione" degli attributi composti quali Infrastrutture(di Regione e Provincia) , Territorio(di Regione e Provincia) e Casi_totali di (Bollettino regionale), Attributi composti: come da regola, gli attributi composti si scindono, aggiungendo gli attributi componenti all'entità. (in questo caso, i componenti di Infrastrutture e Territorio vengono aggiunti a Regione e Provincia).



La traduzione è stata effettuata utilizzando le seguenti regole:

1) Un'entità dello schema concettuale si traduce in una relazione dello schema logico avente lo stesso nome (ma al plurale) e gli stessi attributi dell'entità ed avente per chiave primaria il suo identificatore.
2) Le associazioni di tipo (1 a N), scompaiono e, contemporaneamente, gli identificatori (rinominati) delle entità lato molti si aggiungono

agli attributi delle relazioni relative alle entità lato uno. Questi nuovi attributi diverranno chiavi esterne referenzianti le relazioni delle entità lato molti.

E' di seguito riportato lo schema relazionale completo risultato della progettazione logica:

PROVINCE(Codice_provincia,Cod_regione:REGIONI,Sigla_provincia,Denominazione_provincia,Lat,Longi,Densità,N_comuni,N_Scuole,N_ospedali,Superficie,Popolazione)

REGIONI(Codice_regione,Stato,Denominazione_regione,Popolazione,Superficie,Densità,N_comuni,N_province,Autostrade,Ferrovie,Aeroporti)

BOLLETTINI_PROVINCIALI
(Data,Cod_provincia:PROVINCE,Totale_casi,Note_it,Note_en)

BOLLETTINI_REGIONALI(Data,Codice_regioni:REGIONI,Ricoverati_con_sintomi,Terapia_intensiva,Totale_ospedalizzati,Totale_positivi,Variazione_totale_positivi,Nuovi_positivi,Dimessi_guariti,Deceduti,Totale_casi,TAmponi,Casi_testati)

CAPITOLO 4

4.1 Esempi di interrogazioni in SQL

Nel database le informazioni sono organizzate in una struttura logica che permette di accedere con facilità ad ogni dato. Il modo per accedere a questi dati è la query. La query viene scritta in un linguaggio di interrogazione. Ne esistono decine ed il più famoso prende il nome di SQL. Come tutti i linguaggi, l'SQL ha una sintassi e delle regole. Tramite queste regole è possibile ricercare fra i dati, applicando dei filtri ed

ordinando i dati a piacimento. Di seguito sono riportati degli esempi di interrogazione in SQL.

1)Trovare i contagi totali in ogni provincia per un determinato intervallo temporale.

```
SELECT P.DENOMINAZIONE_PROVINCIA,C.totale_casi,C.data
FROM PROVINCE2 P join CONTAGI2 C ON P.codice_provincia=C.cod_provincia
where data>'2020-03-14T17:00:00' and data<'2020-03-30T17:00:00'
```

Nella query successiva viene utilizzata una vista. Una vista è una relazione virtuale, nel senso che le sue tuple non sono effettivamente memorizzate nella base di dati, quanto piuttosto ricavabili attraverso interrogazioni, date da altre tuple presenti nella base di dati. Una vista costituisce dunque una interfaccia da mettere a disposizione di utenti o applicazioni per le interrogazioni: si tratta di dati usati di frequente che possono anche non esistere fisicamente. Abbiamo dunque creato delle viste per rendere più rapito l'accesso ad alcune informazioni a cui si accede sovente

2)Selezionare la regione con il maggior numero di contagi per rapporto densità abitativa

```
CREATE VIEW VISTA (DEN,COD,MAS,DENSITA) AS
SELECT  R.DENOMINAZIONE_REGIONE,C.codice_regione,max(C.totale_casi),R.densità
FROM REGIONI2 R join CONTAGIREGIONI2 C ON R.codice_regione=C.codice_regione
 GROUP BY R.DENOMINAZIONE_REGIONE,C.codice_regione,R.densità

 SELECT (MAS/DENSITA) AS RAPPORTO,DEN
 FROM VISTA
 WHERE MAS/DENSITA=(SELECT MAX(MAS/DENSITA)
 FROM VISTA)
```

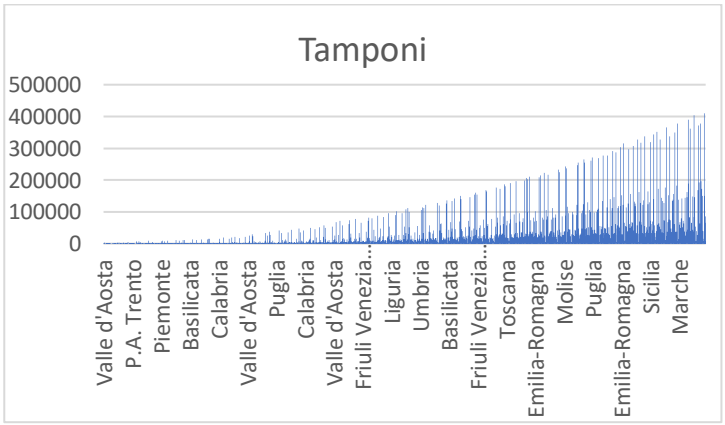
3) Trovare regione con il maggior numero di province, e la provincia con il maggior numero di comuni nella regione lombardia

```
SELECT DISTINCT R.denominazione_regione,P.denominazione_provincia
```

```
FROM REGIONI2 R join PROVINCE2 P ON R.codice_regione=P.cod_regione
WHERE R.NPROVINCE>=ALL(SELECT R1.NPROVINCE
FROM REGIONI2 R1) AND P.Ncomuni>=ALL(
SELECT P1.Ncomuni
FROM PROVINCE2 P1
WHERE P1.cod_regione=3
)

UNION
SELECT DISTINCT R.denominazione_regione,P.denominazione_provincia
FROM REGIONI2 R join PROVINCE2 P ON R.codice_regione=P.cod_regione
WHERE P.NCOMUNI>=ALL(SELECT P1.NCOMUNI
FROM PROVINCE2 P1)
select R1.denominazione_regione,sum(R.variazione_totale_positivi) as variazione
from contagiregioni2 R join regioni2 R1 ON R.codice_regione=R1.codice_REGIONE
group by R1.denominazione_regione
```

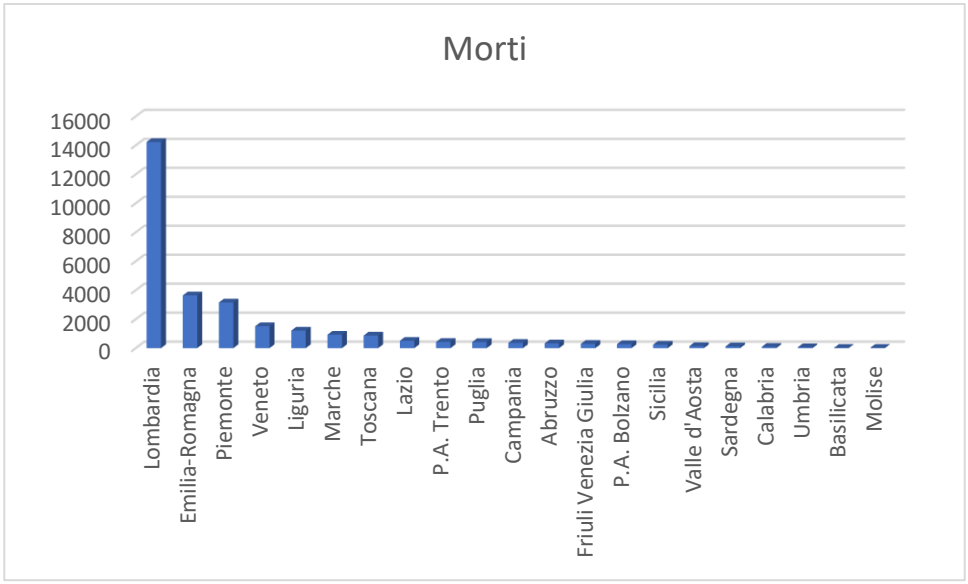
4)Numero tamponi effettuati per ogni regione

```
select R1.denominazione_regione,R.tamponi ,R.data
from contagiregioni2 R join regioni2 R1 ON R.codice_regione=R1.codice_REGIONE
group by R1.denominazione_regione,R.tamponi,R.data
order by R.data
```



4)Numero di morti per ogni regione

```
select max(R.deceduti) as n_deceduti,R1.denominazione_regione
from regioni2 R1 join contagiregioni2 R on R1.codice_regione=R.codice_regione
group by R1.denominazione_regione
order by n_deceduti desc
```

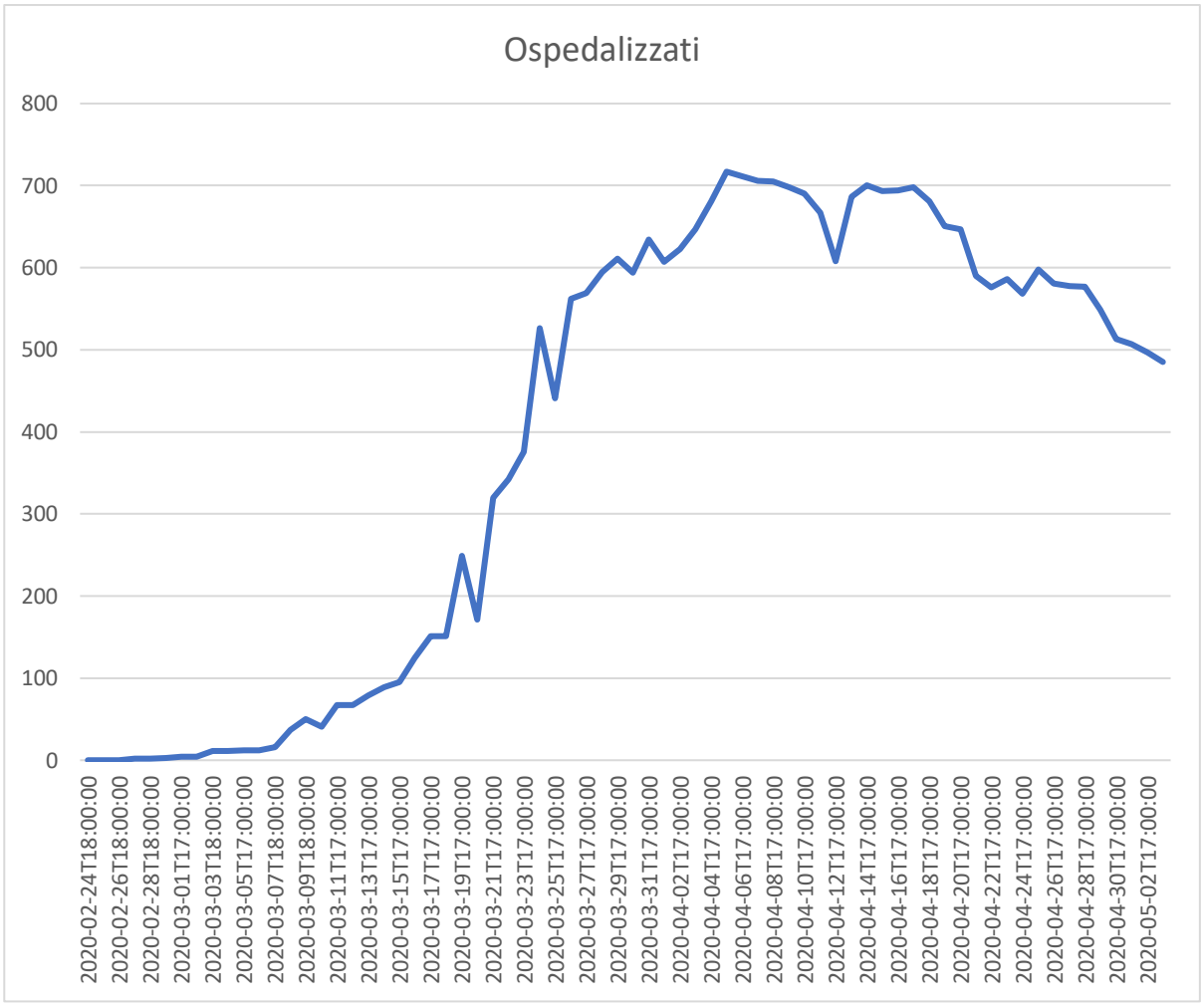


5)Variazione totale positivi per regione

```
select R1.denominazione_regione,sum(R.variazione_totale_positivi) as variazione
from contagiregioni2 R join regioni2 R1 ON R.codice_regione=R1.codice_REGIONE
group by R1.denominazione_regione
```

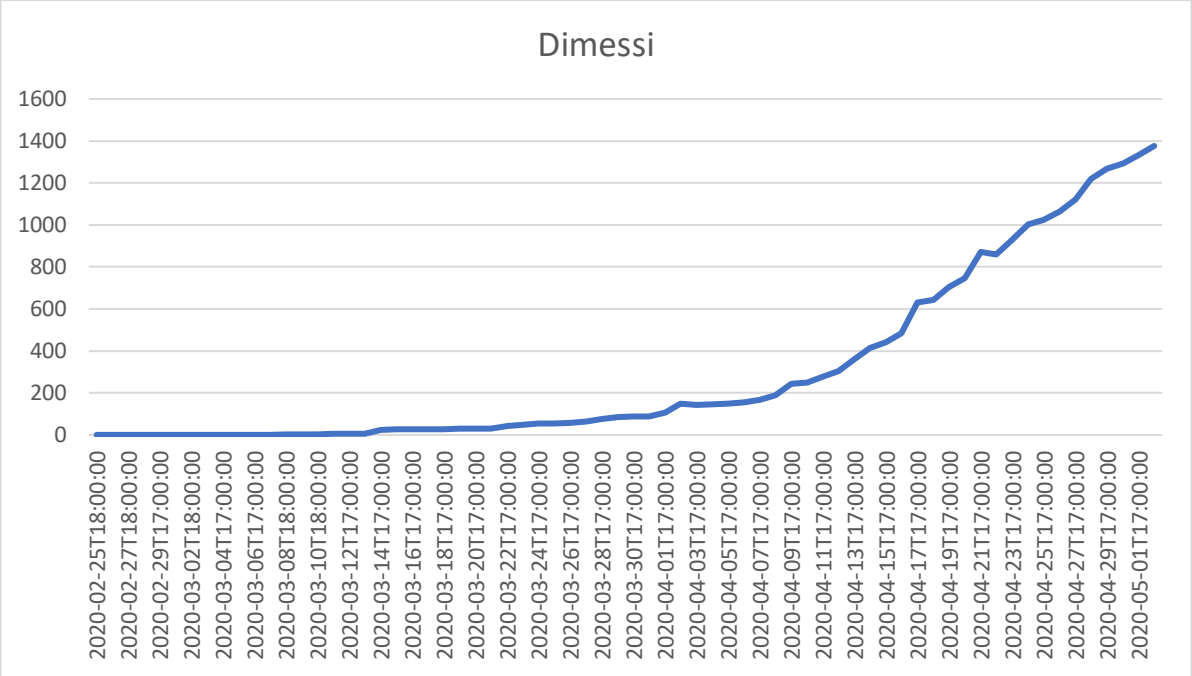
6)Numero ospedalizzati in Campania

```
select c.data,c.totale_ospedalizzati
from contagiregioni2 c join regioni2 r on c.codice_regione=r.codice_regione
where r.denominazione_regione='Campania'
```


Ospedalizzati
800
700
600
500
400
300
200
100
0
2020-02-24T18:00:00
2020-02-26T18:00:00
2020-02-28T18:00:00
2020-03-01T17:00:00
2020-03-03T18:00:00
2020-03-05T17:00:00
2020-03-07T18:00:00
2020-03-09T18:00:00
2020-03-11T17:00:00
2020-03-13T17:00:00
2020-03-15T17:00:00
2020-03-17T17:00:00
2020-03-19T17:00:00
2020-03-21T17:00:00
2020-03-23T17:00:00
2020-03-25T17:00:00
2020-03-27T17:00:00
2020-03-29T17:00:00
2020-03-31T17:00:00
2020-04-02T17:00:00
2020-04-04T17:00:00
2020-04-06T17:00:00
2020-04-08T17:00:00
2020-04-10T17:00:00
2020-04-12T17:00:00
2020-04-14T17:00:00
2020-04-16T17:00:00
2020-04-18T17:00:00
2020-04-20T17:00:00
2020-04-22T17:00:00
2020-04-24T17:00:00
2020-04-26T17:00:00
2020-04-28T17:00:00
2020-04-30T17:00:00
2020-05-02T17:00:00

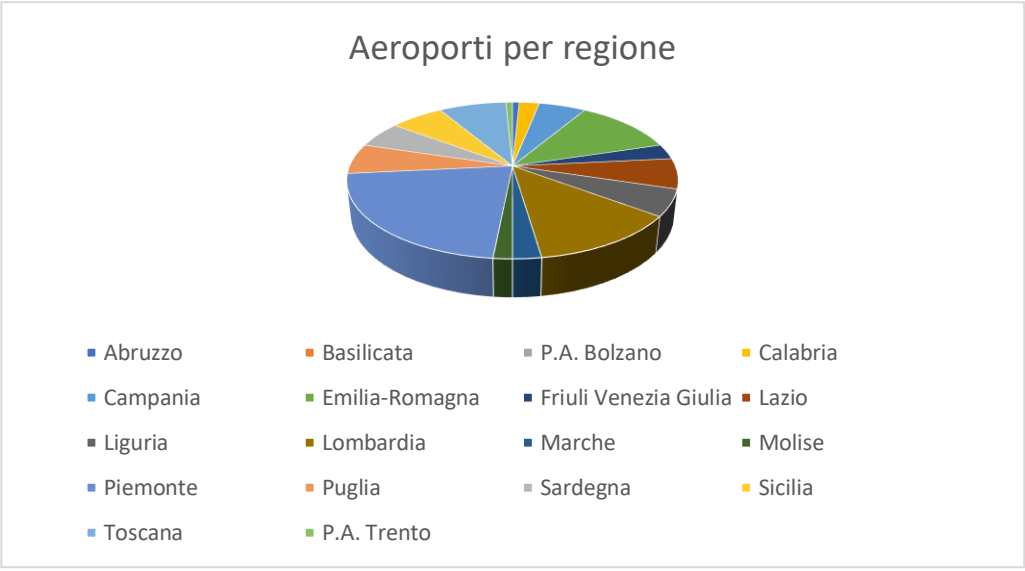
7)Selezionare il numero di guariti in Campania.

```
select c.data,c.dimessi_guariti

from contagiregioni2 c join regioni2 r on c.codice_regione=r.codice_regione

where r.denominazione_regione='Campania'
```



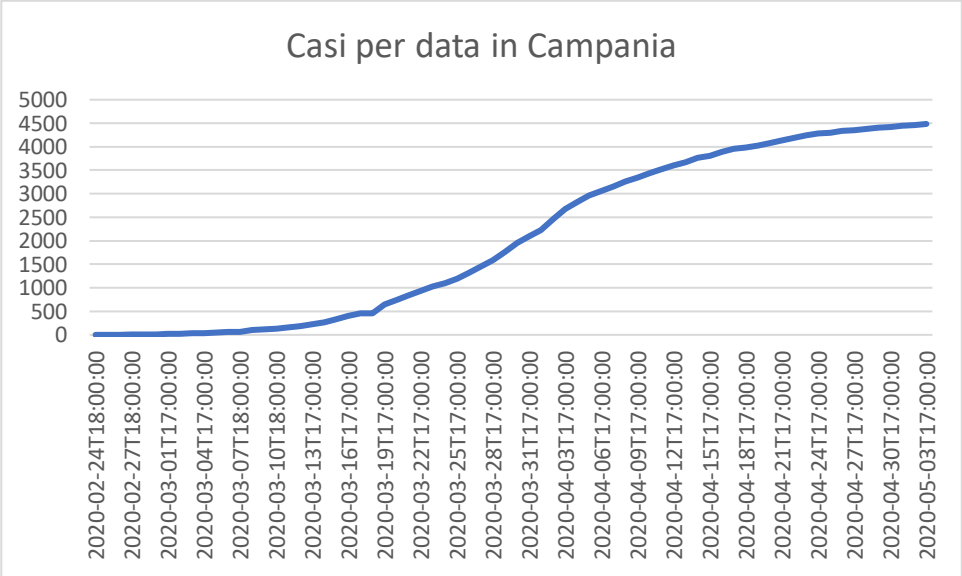
8)Numero aereoporti per regione

```
select r.denominazione_regione,r.aeroporti

from regioni2 r
```



9) Casi per data nella regione Campania

```
select c.data,c.totale_casi

from contagiregioni2 c join regioni2 r on c.codice_regione=r.codice_regione

where r.denominazione_regione='Campania'
```



PUNTO 5

5.1 Trigger

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 di dati e sono utilizzati per preservare l'integrità dei dati o implementare particolari regole di business.

--Trigger che a partire da un inserimento nella tabella master popola le tabelle normalizzate

```
create or replace trigger popola
after insert on MASTER
for each row
begin
insert into CONTAGI2(Data,Cod_provincia,Totale_casi,Note_it,Note_en)
VALUES(:new.Data,:new.Codice_provincia,:new.Totale_casi,:new.Note_it,:new.Note_en);
end;
```

--Trigger che non permette l'inserimento di tuple all'interno della tabella master se la data inserita è maggiore di quella attuale.

```
create or replace trigger orario
before insert on MASTER
for each row
declare
errore exception;
begin
if :new.data>sysdate
then raise errore;
end if;
exception
when errore then raise_application_error(-20001,'Data non valida');
end;
```

5.2 Stored Procedure

Le procedure in PL/SQL consistono in blocchi PL/SQL che è possibili attivare mediante una chiamata a procedura e possono essere dotate di parametri di input/output.

-Procedura che stampa a video il numero di casi in una data provincia e determinata data

```
CREATE OR REPLACE PROCEDURE contagi_proc(
   CODPROV CONTAGI2.cod_provincia%type,IN_data CONTAGI2.data%type)
   IS
 risultati_trovati CONTAGI2%rowtype;
   BEGIN
   SELECT *
   INTO risultati_trovati
```

```
FROM CONTAGI2
WHERE cod_provincia=CODPROV AND DATA=IN_data;
dbms_output.put_line('I casi nella provincia avente codice= '||risultati_trovati.cod_provincia||' sono: '||risultati_trovati.totale_casi||' in data '||risultati_trovati.data);
end;
```

-Procedura che stampa a video il numero di abitanti e il numero di comuni in una data provincia

```
CREATE OR REPLACE PROCEDURE contagi_pro(
CODPROV PROVINCE2.denominazione_provincia%type)
IS
risultati_trovati PROVINCE2%rowtype;
BEGIN
SELECT *
INTO risultati_trovati
FROM PROVINCE2
WHERE denominazione_provincia=CODPROV ;
dbms_output.put_line('La popolazione nella provincia di '||risultati_trovati.denominazione_provincia||' è pari a '||risultati_trovati.popolazione||' abitanti e vi sono ben '||risultati_trovati.ncomuni||' comuni');
end;

CALL contagi_pro('Milano');
```

-Procedura che permette di aumentare i casi in un determinato giorno in una data provincia e determinata data

```
CALL CONTAGI_PROC(10,'2020-04-23T17:00:00');
CREATE OR REPLACE PROCEDURE contagi_au(
CODPROV CONTAGI2.cod_provincia%type,AUMENTO NUMBER,IN_data CONTAGI2.data%type)
IS
BEGIN
```

```
UPDATE CONTAGI2

SET totale_casi=totale_casi+AUMENTO

WHERE cod_provincia=CODPROV AND DATA=IN_data;

END;


CALL contagi_au(10,100,'2020-04-23T17:00:00');
```

-Procedura che stampa a video la densità di tutte le province italiane

```
create or replace procedure pro2 is
cursor my_cursor is
select P.densità,P.denominazione_provincia
from PROVINCE2 P
WHERE P.denominazione_provincia LIKE '%o';
errore exception;
risultati_trovati my_cursor%rowtype;

begin
open my_cursor;
fetch my_cursor into risultati_trovati;

if my_cursor%notfound then
raise errore;
else if my_cursor%found then
loop
DBMS_OUTPUT.PUT_LINE('La densità di:' ||risultati_trovati.denominazione_provincia||'
è:'||risultati_trovati.densità||'ab./km2');
fetch my_cursor into risultati_trovati;
exit when my_cursor%notfound;


end loop;
end if;
end if;
exception
when errore then
DBMS_OUTPUT.PUT_LINE('ERRORE');
close my_cursor;
end;

call pro2();
```

-Procedura che stampa il numero di casi nella provincia di Caserta in data '2020-04-23T17:00:00'

```
CREATE OR REPLACE PROCEDURE procedurea_ IS
cursor my_cursor is
select C.totale_casi,P.denominazione_provincia,C.cod_provincia,C.data
FROM PROVINCE2 P join CONTAGI2 C ON P.codice_provincia=C.cod_provincia
WHERE P.denominazione_provincia='Caserta' AND C.DATA='2020-04-23T17:00:00';
casi_trovati my_cursor%rowtype;
begin
open my_cursor;

loop

fetch my_cursor into casi_trovati;
exit when my_cursor%notfound;
DBMS_OUTPUT.PUT_LINE('Provincia:'||casi_trovati.denominazione_provincia||' avente come
codice:'||casi_trovati.cod_provincia);
DBMS_OUTPUT.PUT_LINE('Casi:'||rpad(casi_trovati.totale_casi,10)||'Data:'||casi_trovati.data);



end loop;

close my_cursor;
end;

call procedura_1();
```

PUNTO 6

6.1 INDICI

Creazione indice sulla tabella PROVINCE2 per i campi codice_provincia,denominazione_provincia,sigla_provincia

```
CREATE INDEX idx_province
ON PROVINCE2(codice_provincia,denominazione_provincia,sigla_provincia)
```

Creazione indice sulla tabella PROVINCE2 per i campi cod_regione,codice_provincia

```
CREATE INDEX idx_fk
ON PROVINCE2(cod_regione,cod_provincia)
```

Creazione indice sulla tabella PROVINCE2 per i campi lat,longe

```
CREATE INDEX pos
ON PROVINCE2(lat,longe)
```

Creazione indice sulla tabella PROVINCE2 per i campi popolazione,superficie,densità

```
CREATE INDEX prov
ON PROVINCE2(popolazione,superficie,densità)
```

Creazione indice sulla tabella PROVINCE2 per i campi N_ospedali,scuole

```
CREATE INDEX strut
ON PROVINCE2(N_ospedali,Scuole)
```

Creazione indice sulla tabella REGIONI2 per i campi Denominazione_regione,stato

```
CREATE INDEX reg
ON REGIONI2(Denominazione_regione,stato)
```

Creazione indice sulla tabella CONTAGI2 per i campi N_ospedali,scuole

```
CREATE INDEX casi
ON CONTAGI2(Cod_provincia,totale_casi)
```

Bibliografia

A. Chianese, V.Moscato, A.Picariello, Sistemi di basi di dati e applicazioni, Santarcangelo Romagna, Maggiolini Editore, 2015