Basi di Dati

Esercizi Svolti di Basi di Dati

Anno Accademico: 2024/25

Giacomo Sturm

Dipartimento di Ingegneria Civile, Informatica e delle Tecnologie Aeronautiche
Università degli Studi "Roma Tre"

Sorgente del file LaTeX disponibile al seguente link:
https://github.com/00Darxk/Basi-di-Dati/

Indice

1 Esercitazione del 7/10/24

1.1 Esercizio 1

Si considerano le seguenti relazioni, senza valori nulli:

- $R_1(\underline{A}, B)$ con vincolo di integrità referenziale, tra B e la chiave D di R_2. Con cardinalità $M_1 = 500$;
- $R_2(\underline{D}, E, F, G)$, con vincolo di integrità referenziale, tra F e G e la chiave H, P di R_3. Con cardinalità $M_2 = 1000$;
- $R_3(\underline{H}, \underline{P}, Q)$, con cardinalità $M_3 = 200$.

Determinare la cardinalità (numero di ennuple) massima e minima delle seguenti relazioni derivate:

Domanda 1

L'equi-join tra R_3 e R_1 sulla condizione $Q = A$:

$$R_3 \bowtie_{Q=A} R_1$$

Non comprendendo alcun vincolo di integrità referenziale tra le dure relazioni, potrebbe non contenere alcuna ennupla, quindi avere una cardinalità minima pari a zero. Invece potrebbe contenere al massimo un numero di ennuple pari al numero di attributi Q non nulli, quindi pari alla cardinalità di R_3, M_3.

$$M_{D_1} \in [0, M_3] \tag{1.1.1}$$

Domanda 2

La proiezione sulla chiave H, P di R_3:

$$\pi_{H,P}(R_3)$$

Poiché coinvolge una chiave della relazione, questa relazione derivata conterra esattamente tutte le ennuple presenti nella relazione di partenza, quindi avrà una cardinalità sempre pari alla cardinalità di R_3, M_3:

$$M_{D_2} = M_3 \tag{1.1.2}$$

Domanda 3

La proiezione sugli attributi H e Q di R_3:

$$\pi_{H,P}(R_3)$$

Questi due attributi insieme non formano una superchiave, quindi potrebbero i valori potrebbero fra loro collassare, quindi produrre una singola ennupla di attributi distinti. Invece è possibile che questa relazione contenga tutte le ennuple distinte della relazione R_3, e quindi abbia una cardinalità massima pari a M_3:

$$M_{D_3} \in [1, M_3] \tag{1.1.3}$$

In generale quando una proiezione non coinvolge una superchiave dell'operando, è possibile, ma non garantito, che abbia lo stesso numero di ennuple dell'operando. L'unico discriminante in una proiezione è la presenza o meno della chiave, o superchiave, negli attributi considerati.

Domanda 4

L'equi-join tra R_1 ed R_2 sulla condizione $B = D$:

$$R_1 \bowtie_{B=D} R_2$$

Gli attributi B e la chiave D rappresentano un vincolo di integrità referenziale tra R_1 ed R_2. Ciascuna ennupla di R_1 può essere estesa con una singola ennupla distanza di R_2, poiché si effettua il join sulla sua chiave. La cardinalità di questa relazione è quindi M_1.

$$M_{D_4} = M_1 \tag{1.1.4}$$

Domanda 5

$$(R_1 \bowtie_{B=D} R_2) \bowtie_{(F=H)\wedge(G=P)} R_3$$

Analogamente alla domanda precedente, poiché si effettua un join tra R_3 e la relazione precedente, su una chiave di R_3 ed un vincolo di integrità referenziale, allora la cardinalità corrisponde alla cardinalità dell'altro operando, calcolata nella domanda precedente pari a M_1:

$$M_{D_5} = M_1 \tag{1.1.5}$$

Domanda 6

L'equi-join tra R_1 ed R_3 su $B = H$:

$$R_1 \bowtie_{B=H} R_3$$

Non sono presenti vincoli di integrità referenziale, o chiavi delle relazioni, quindi è possibile che non siano presenti ennuple, oppure che ogni ennupla della prima relazione, sia operata con ogni ennupla della seconda per un massimo di $M_1 \cdot M_3$. Bisogna considerare che l'attributo H non rappresenta una chiave primaria, quindi non si può effettuare lo stesso ragionamento della domanda 5.

$$M_{D_6} \in [0, M_1 \cdot M_2] \tag{1.1.6}$$

Domanda 7

$$(R_1 \bowtie_{B=D} R_2) \bowtie_{F=H} R_3$$

Il primo join interno ha cardinalità M_1, mentre nel secondo join, è presente solo una parte del vincolo di integrità referenziale. Quindi ogni ennupla del primo join può al minimo trovare una sua controparte in R_3, avendo una cardinalità minima di M_1. Mentre al massimo, non essendo completo il vincolo, è possibile che tutte le ennuple di R_1 siano combinate con tutte le ennuple di R_3:

$$M_{D_7} \in [M_1, M_1 \cdot M_3] \tag{1.1.7}$$

1.2 Esercizio 2

Si considera la seguente base di dati su RelaX.

Domanda 1

Trovare matricola, nome, età e stipendio degli impiegati che guadagnano più di 40. Si utilizza una semplice selezione sulla relazione impiegati, specificando la condizione di selezione, e non è necessario effettuare una proiezione, poiché si utilizzano tutti gli attributi:

```
σ Stipendio>40 (Impiegati)
```

Domanda 2

Trovare matricola, nome ed età degli impiegati che guadagnano più di 40. In questa interrogazione è necessario effettuare una proiezione sulla lista di attributi di interesse, dato il risultato alla precedente domanda:

```
π Matricola, Nome, Eta (σ Stipendio>40 (Impiegati))
```

Domanda 3

Trovare le matricole dei capi che guadagnano più di 40. Si può effettuare prima sia il join che la selezione, ma discussione in termini di efficienza non vengono trattate in questo corso. Si effettua una selezione sullo stipendio, come per le prime due domande. Si effettua un join tra questa relazione e supervisione sulla condizione matricola, di impiegati, ed impiegato, di supervisione. Per ottenere solamente l'attributo richiesto si effettua una proiezione su capo:

```
π Capo ( (Supervisione)
    ⋈ Matricola=Impiegato
    (σ Stipendio>40 (Impiegati))
)
```

Ovviamente quest'interrogazione produce un risultato solamente sui dati noti, in questa base di dati non sono presenti informazioni sui capi di alcuni impiegati, quindi non è possibile creare informazioni su di loro.

Domanda 4

Trovare nome e stipendio dei capi degli impiegati che guadagnano più di 40. Considerando i capi come impiegati, nella relazione impiegati sono presenti queste informazioni. Quindi dalla relazione precedente è possibile effettuare un join ulteriore con la relazione impiegati su `Capo = Impiegato`, ed in seguito una selezione per identificare le informazioni di interesse:

```
π Nome, Stipendio ( (Impiegati)
    ⋈ Impiegato=Capo
        (π Capo ( (Supervisione)
            ⋈ Matricola=Impiegato
            (σ Stipendio>40 (Impiegati)))))
```

Domanda 5

Trovare gli impiegati che guadagnano più del proprio capo, mostrando matricola, nome e stipendio dell'impiegato e del capo. Per trovare questo risultato bisognerebbe utilizzare due copie della stessa relazione impiegati per confrontare gli stipendi del capo, e dell'impiegato. Ma per poter confrontare questi attributi bisognerebbe rinominare gli attributi di una delle due, prima del join tra di loro, per correlare le informazioni necessarie. Il join si effettua prima tra le matricole dei capi e supervisione e poi nuovamente con la relazione impiegati, per correlare per ogni impiegato, le informazioni del suo capo. A questo punto si può effettuare una selezione per rimuovere le ennuple dove il capo guadagna di più del proprio impiegato, ed una proiezione finale sugli attributi di interesse:

```
π Matricola, Nome, Stipendio, MatricolaCapo, NomeCapo, StipendioCapo (
    σ Stipendio > StipendioCapo (
        (ρ MatricolaCapo ← Matricola, NomeCapo ← Nome,
           StipendioCapo ← Stipendio, EtaCapo ← Eta (Impiegati))
        ⋈ MatricolaCapo=Capo
        ((Supervisione) ⋈ Matricola=Impiegato (Impiegati))))
```

Questo processo è molto verboso, ed è possibile semplificarlo mediante l'utilizzo di viste. Utilizzano le viste si può rinominare la relazione impiegati a `Capi`, per poter utilizzare la notazione puntate per riferirsi ai suoi attributi:

```
Capi = ρ Capi (Impiegati)

π Impiegato.Matricola, Impiegato.Nome, Impiegato.Stipendio,
  Capi.Matricola, Capi.Nome, Capi.Stipendio (
    σ Stipendio > Capi.Stipendio ((Capi)
        ⋈ Capi.Matricola=Capo
        ((Supervisione) ⋈ Matricola=Impiegato (Impiegati))))
```

Domanda 6

Trovare matricola, nome e stipendio dei capi degli impiegati che guadagnano più di 40; per ciascuno, mostrare, matricola, nome e stipendio anche dell'impiegato. Questa rappresenta una versione

semplificata della domanda precedente, quindi invece di effettuare una selezione dopo il join tra capi ed impiegati, si effettua la selezione come per le domande precedenti:

```
π Impiegato.Matricola, Impiegato.Nome, Impiegato.Stipendio,
  Capi.Matricola, Capi.Nome, Capi.Stipendio (
    ((Capi) ⋈ Capi.Matricola=Capo
    ((Supervisione) ⋈ Matricola=Impiegato (σ Stipendio>40 (Impiegati)))))
```

Domanda 7

Trovare le matricole dei capi i cui impiegati guadagnano tutti più di 40. Quest'interrogazione è semplice, poiché è sufficiente individuare tutti i capi con almeno un impiegato che guadagna meno di 40, da togliere poi tramite una differenza tra insiemi alla vista capi. Rimangono sicuramente tutti i capi con impiegati che guadagnano più di 40:

```
π Capo (Supervisione) -
π Capo ((Supervisione) ⋈ Matricola=Impiegato
(σ Stipendio≤40 (Impiegati)))
```

2 Esercitazione del 18/10/24 e 25/10/24

Si considera la seguente base di dati:

```
CREATE TABLE Compositori (codice integer NOT NULL PRIMARY KEY,
                        cognome text ,
                        nome text);

CREATE TABLE Concerti (codice integer NOT NULL PRIMARY KEY,
                        titolo text ,
                        descrizione text);

CREATE TABLE Pezzi (codice  integer NOT NULL PRIMARY KEY,
                        titolo text,
                        autore integer NOT NULL REFERENCES Compositori,
                        durata integer);

CREATE TABLE Programmazione ( pezzo integer NOT NULL REFERENCES Pezzi,
                        concerto integer NOT NULL REFERENCES Concerti,
                        posizione integer,
                        PRIMARY KEY(pezzo, concerto));
```

Popolato dai seguenti valori:

```
insert into compositori values(1, 'Mozart', 'Wolfgang Amadeus');
insert into compositori values(2, 'Bach', 'Johann Sebastian');
insert into compositori values(3, 'Beethoven', 'Ludwig van');

insert into concerti values(1, 'Concerto di Febbraio', 'Selezione di musica Barocca');
insert into concerti values(2, 'Concerto di Marzo', 'Estratti di belle sinfonie');
insert into concerti values(3, 'Concerto di Giugno', 'Concerto a Villa Ada');

insert into pezzi values(1, 'Variazioni Goldberg', 2, 32);
insert into pezzi values(2, 'L''arte della fuga', 2, 38);
insert into pezzi values(3, 'Il clavicembalo ben temperato', 2, 85);
insert into pezzi values(4, 'Il flauto magico', 1, 95);
insert into pezzi values(5, 'Serenata in do minore k 388', 1, 45);
insert into pezzi values(6, 'Requiem', 1, 87);
insert into pezzi values(7, 'Sinfonia n. 6 in fa maggiore op. 68', 3, 91);
insert into pezzi values(8, 'Sinfonia n. 9 in re minore', 3, 91);
insert into pezzi values(9, 'Trio d''archi in mi bemolle maggiore op. 3', 3, 52);

insert into programmazione values(1, 1,1);
insert into programmazione values(2, 1,2);
```

```
insert into programmazione values(3, 2,2);
insert into programmazione values(4, 3,1);
insert into programmazione values(5, 2,3);
insert into programmazione values(5, 3,2);
insert into programmazione values(7, 2,1);
```

Vengono proposti una serie di esercizi su questo database.

Domanda 1

Determinare il titolo dei pezzi che hanno durata compresa tra 40 e 60 minuti. Per effettuare quest'operazione è sufficiente operare sulla singola tabella `Pezzi`:

```
SELECT titolo FROM Pezzi
WHERE durata>40 AND durata<60
```

Domanda 2

Determinare il titolo, nome e cognome dell'autore dei pezzi che hanno durata compresa tra 40 e 60 minuti. Oltre alla tabella `Pezzi`, è necessario ottenere i campi `nome` e `cognome` della tabella `Compositori`. Si effettua mediante un join, specificando come vengono associate le due relazioni, infatti il campo `autore` di `Pezzi` si riferisce alla chiave primaria della relazione `Compositori`:

```
SELECT titolo, nome, cognome
FROM Pezzi JOIN Compositori ON Pezzi.autore=Compositori.codice
WHERE durata>40 and durata<60
```

Domanda 3

Determinare il nome e cognome dei compositori dei pezzi presenti nel "Concerto di Giugno". Per ottenere questo si utilizza una vista. Nei database industriali la vista p contenuta nella cache. La vista può essere utilizzata come fosse un'altra tabella nel database, ed è possibile utilizzarla per effettuare altre interrogazioni nella stessa query. Si crea quindi una vista di tutti i pezzi presenti nel concerto di Giugno:

```
CREATE VIEW pezzi_giugno AS Select *
FROM Concerti JOIN Programmazione ON Concerti.codice=Programmazione.concerto
WHERE Concerti.titolo='Concerto di Giugno'
```

Per ottenere il nome ed il cognome dei compositori si effettua una join su questa vista, prima con i pezzi, e poi con la relazione compositori, selezionando solo gli attributi richiesti:

```
SELECT DISTINCT nome, cognome
FROM pezzi_giugno JOIN Pezzi ON pezzi_giugno.pezzo=Pezzi.codice
JOIN Compositori ON Compositori.codice=autore
```

Domanda 4

Determinare il nome e cognome dei compositori che non hanno pezzi nel "Concerto di Giugno". Utilizzando la stessa vista `pezzi_giugno`, è possibile utilizzare una semplice `EXCEPT` sulla relazione dei compositori per rimuovere quelli presenti nel concerto di Giugno:

```
SELECT DISTINCT nome, cognome
FROM Compositori EXCEPT
SELECT DISTINCT nome, cognome
FROM pezzi_giugno JOIN pezzi ON pezzo=pezzi.codice
JOIN Compositori ON autore=Compositori.codice
```

Domanda 5

Determinare il titolo e descrizione dei concerti in cui sono presenti pezzi di Mozart. Per ottenere questi risultati è sufficiente effettuare un'operazione di join tra tutte le relazioni del database, considerando solo le ennuple di valore specificato per l'attributo `cognome`:

```
SELECT DISTINCT concerti.titolo, descrizione
FROM Concerti JOIN Programmazione ON Concerti.codice=Programmazione.concerto
JOIN pezzi ON pezzi.codice=pezzo
JOIN Compositori ON autore=Compositori.codice
WHERE cognome='Mozart'
```

Domanda 6

Determinare il titolo e descrizione dei concerti in cui non sono presenti pezzi di Beethoven. Per effettuare quest'operazione è sufficiente trovare tutti i concerti dove sono presenti pezzi di Beethoven, come nella domanda precedente, ed in seguito rimuoverli dalla lista di tutti i concerti con un `EXCEPT` iniziale:

```
SELECT DISTINCT concerti.titolo, descrizione
FROM Concerti EXCEPT
-- concerti contenenti pezzi di Beethoven
SELECT DISTINCT concerti.titolo, descrizione
FROM Concerti join Programmazione ON Concerti.codice=Programmazione.concerto
JOIN pezzi ON pezzi.codice=pezzo
JOIN Compositori ON autore=Compositori.codice
WHERE cognome='Beethoven'
```

Domanda 7

Determinare il codice ed il titolo dei pezzi che non sono presenti in nessun concerto. Ordinare per codice. Si può effettuare in modo analogo alla precedente domanda, si ottengono i pezzi contenuti in tutti i concerti tramite un join tra le relazioni `Programmazione` e `Pezzi`. Si rimuove quindi questo

risultato dalla relazione `Pezzi`, tenendo conto di selezionare solamente gli attributi in comune, ed in seguito ordinando in base al codice con il comando `ORDER BY`, seguito dall'attributo su cui si vuole effettuare l'ordinamento:

```
SELECT DISTINCT pezzi.codice, titolo
FROM Pezzi EXCEPT
SELECT DISTINCT pezzi.codice, titolo
FROM Programmazione JOIN Pezzi ON Programmazione.pezzo=codice
ORDER BY pezzi.codice
```

Domanda 8

Determinare i pezzi che compaiono in ultima posizione in almeno un concerto. Mostrare codice, titolo, cognome dell'autore e durata del pezzo. Ordinare per codice. Si può utilizzare una vista `ultimi_pezzi`, dove vengono salvati i pezzi in ultima posizione nei concerti con il comando `MAX(posizione)`, raggruppandoli rispetto all'attributo `concerto`.

```
CREATE VIEW ultimi_pezzi AS
SELECT DISTINCT MAX(posizione), pezzo
FROM Programmazione
GROUP BY Programmazione.concerto
```

Da questa vista si possono ottenere le informazioni richieste con i singoli pezzi effettuando una join con le relazioni `Pezzi` e `Compositori`, ed ordinando con `ORDER BY` rispetto al codice dei pezzi:

```
SELECT DISTINCT pezzi.codice, pezzi.titolo, cognome, durata
FROM Pezzi JOIN Compositori ON Compositori.codice=autore
JOIN ultimi_pezzi ON pezzi.codice=pezzo
ORDER BY pezzi.codice
```

Domanda 9

Determinare i pezzi che compaiono in ultima posizione in tutti i concerti. Mostrare codice, titolo, cognome dell'autore e durata del pezzo. Ordinare per codice. Si può utilizzare la stessa vista realizzata alla domanda precedente, e si utilizza la stessa interrogazione alla domanda precedente, inserendo una condizione nel `WHERE` dove viene controllato che la vista `ultimi_pezzi` contenga solo un'ennupla, ovvero il pezzo che compare in ultima posizione in tutti i concerti:

```
SELECT DISTINCT pezzi.codice, pezzi.titolo, cognome, durata
FROM pezzi JOIN Compositori ON Compositori.codice=autore
JOIN ultimi_pezzi ON pezzi.codice=pezzo
WHERE (SELECT COUNT(pezzo) FROM ultimi_pezzi)=1
ORDER BY pezzi.codice
```

Domanda 10

Determinare coppie di pezzi con lo stesso titolo. Mostrare il titolo e i due codici (ordinando il risultato sul titolo). Nota bene: ogni coppia va mostrata una sola volta (ad esempio, se i pezzi 3 e 5 hanno stesso titolo, va mostrata solo la coppia 3,5 e non la coppia 5,3). Effettuando un'operazione di join sulla relazione `Pezzi` con sé stessa si ottiene una relazione dove sono presenti due volte le stesse coppie. Per ovviare a questo problema, si può utilizzare una `GROUP BY` rispetto al titolo dei pezzi, anche se questo tipo di operazioni vengono sconsigliate poiché mostrano un'ennupla casualmente tra le due, senza poter scegliere quale:

```
SELECT p1.titolo, p1.codice pezzo1, p2.codice pezzo2
FROM pezzi p1 JOIN pezzi p2 ON p1.titolo=p2.titolo
WHERE p1.codice!=p2.codice
GROUP BY p1.titolo, p1.codice
```

Per dare priorità ad una delle due coppie, ottenendo un risultato deterministico, si può utilizzare invece del costrutto `GROUP BY` una disuguaglianza nella `WHERE`, in questo modo si privilegia solo una delle due coppie:

```
SELECT p1.titolo, p1.codice pezzo1, p2.codice pezzo2
FROM pezzi p1 JOIN pezzi p2 ON p1.titolo=p2.titolo
WHERE p1.codice>p2.codice
```

Domanda 11

Determinare per ogni concerto, la durata totale (somma delle durate dei pezzi). Supporre che per tutti i concerti ci sia almeno un pezzo. Mostrare codice e titolo del concerto e durata totale. Ordinare per codice. Per ottenere le informazioni necessarie si usa una join tra le relazioni `Programmazione`, `Pezzi` e `Concerti`. Si utilizza il comando `SUM(durata)` per sommare tutte le durate dei pezzi, nello stesso concerto tramite `GROUP BY concerto`

```
SELECT Concerti.codice, Concerti.titolo, SUM(durata) AS durata_concerto
FROM Programmazione JOIN Pezzi ON Programmazione.pezzo=Pezzi.codice
JOIN Concerti ON Concerti.codice=concerto
GROUP BY concerto
ORDER BY Concerti.codice
```

Domanda 12

Determinare i concerti che hanno durata totale minore di 90 minuti; per ogni concerto, mostrare codice e durata totale. Ordinare per codice. Questa domanda è essenzialmente identica alla precedente, è sufficiente inserire la nuova condizione tramite il comando `HAVING`, che opera su tutte le ennuple della relazione:

```
SELECT Concerti.codice, Concerti.titolo, SUM(durata) AS durata_concerto
FROM Programmazione JOIN Pezzi ON Programmazione.pezzo=Pezzi.codice
```

```
JOIN Concerti ON Concerti.codice=concerto
GROUP BY concerto HAVING durata_concerto < 90
ORDER by Concerti.codice
```

Poiché è sostanzialmente uguale alla domanda precedente, si potrebbe realizzare una vista per rispondere alla domanda 11, utilizzata anche dalla 12:

```
CREATE VIEW concerti_con_durata AS
SELECT Concerti.codice, Concerti.titolo, SUM(durata) AS durata_concerto
FROM Programmazione JOIN Pezzi ON Programmazione.pezzo=Pezzi.codice
JOIN Concerti ON Concerti.codice=concerto
GROUP BY concerto
ORDER BY Concerti.codice

-- domanda 11:
SELECT * FROM concerti_con_durata

--domanda 12:
SELECT * FROM concerti_con_durata WHERE durata_concerto < 90
```

Domanda 13

Determinare codice, nome e cognome dei compositori che sono presenti in tutti i concerti. Ordinare per codice. Si può realizzare in modo poco elegante tramite una serie di `EXCEPT`. Si ottiene la relazione dei compositori per concerto effettuando una join tra `Programmazione`, `Pezzi` e `Compositori`, si ottiene il suo complementare effettuando una `EXCEPT` tra la join di `Programmazione` e `Compositori` per la relazione appena calcolata. Se un compositore è presente in tutti i concerti, non sarà presente in questa tabella. Quindi essenzialmente raggruppando per concerto, senza `GROUP BY`, si considerano solo le ennuple distinte senza l'attributo `concerto`, per cui si ottiene la tabella dei compositori non presenti in tutti i concerti. La sua relazione complementare conterrà quindi tutti i compositori presenti in tutti i concerti, ottenuta mediante un'altra `EXCEPT`:

```
-- compositori presenti in tutti i concerti:
SELECT DISTINCT codice, nome, cognome FROM Compositori EXCEPT

    -- compositori non presenti in tutti i concerti:
    SELECT DISTINCT codice, nome, cognome FROM(

        -- ogni compositore per ogni concerto:
        SELECT DISTINCT Compositori.codice, nome, cognome, concerto
        FROM Compositori JOIN Programmazione
        JOIN Concerti ON concerto=Concerti.codice EXCEPT

            -- compositori per ogni concerto:
            SELECT DISTINCT Compositori.codice, nome, cognome, concerto
```

```
            FROM Programmazione JOIN Pezzi ON Programmazione.pezzo=Pezzi.codice
            JOIN Compositori ON Pezzi.autore=Compositori.codice)
ORDER BY Compositori.codice
```

3 Esercitazione del 28/10/24

Sulla base del seguente schema di una base di dati:

```
CREATE TABLE IF NOT EXISTS volo_reale (
    id_volo_reale integer PRIMARY KEY NOT NULL,
    data_partenza_programmata date,
    numero_volo text REFERENCES volo(numero_volo),
    codice_tipo_aeromobile character(3) REFERENCES tipo_aeromobile(codice_tipo_aeromobile),
    data_partenza_reale date,
    data_arrivo_reale date,
    orario_arrivo_reale time ,
    orario_partenza_reale time ,
    UNIQUE (numero_volo, data_partenza_programmata))

CREATE TABLE IF NOT EXISTS volo (
    numero_volo text PRIMARY KEY NOT NULL,
    aeroporto_partenza character(3) REFERENCES aeroporto(codice_aeroporto),
    aeroporto_arrivo character(3) REFERENCES aeroporto(codice_aeroporto),
    orario_partenza_previsto time ,
    orario_arrivo_previsto time)
```

Domanda 1

Determinare per ogni aeroporto il volo di durata massima in partenza il giorno '2023-07-04', mostrare codice aeroporto di partenza, codice aeroporto di arrivo, numero volo, durata prevista ordinati per codice aeroporto di partenza e numero volo. Si crea una vista con tutti i voli con durata, per ogni aeroporto, in partenza il giorno specificato. In seguito su questa vista si considerano tutti i voli con durata uguale alla durata massima per ogni aeroporto. In seguito si ordina rispetto al numero del volo, e per ogni aeroporto:

```
CREATE VIEW voli_con_durata AS
SELECT aeroporto_partenza, aeroporto_arrivo, volo.numero_volo, orario_arrivo_previsto-orario_partenz
FROM volo_reale JOIN volo ON volo_reale.numero_volo=volo.numero_volo
WHERE data_partenza_programmata='2023-07-04';

SELECT * FROM voli_con_durata AS v
WHERE durata = (SELECT MAX(durata)
                FROM voli_con_durata
                WHERE aeroporto_partenza=v.aeroporto_partenza)
ORDER BY aeroporto_partenza, numero_volo
```

Domanda 2

Trovare il codice degli aeroporti con più di un volo con data partenza programmata il giorno "2023-07-06". Mostrare il codice dell'aeroporto e il numero di voli programmati in partenza il "2023-07-06", ordinati per codice dell'aeroporto. Sulla relazione ottenuta dalla join delle due relazioni, si considerano i voli nella data specificata tramite una `WHERE`, si raggruppa per il codice di aeroporto e si conta quanti voli sono presenti per ognuno tramite una `COUNT`. Per includere solamente aeroporti con più di un volo, si specifica con una `HAVING` solo gli aeroporti con più di un volo:

```
SELECT aeroporto_partenza, COUNT(*) AS numero_voli
FROM volo join volo_reale USING(numero_volo)
WHERE data_partenza_programmata='2023-07-06'
GROUP BY aeroporto_partenza HAVING numero_voli>1
ORDER BY aeroporto_partenza
```

Si utilizza la `USING` per evitare di scrivere eccessivamente.