Esercizi di SQL

È riportato di seguito un insieme di esercizi risolti in SQL. Per ogni esercizio una o più soluzioni equivalenti sono presentate.

Esercizio 1.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

PERSONE(CodFisc, Nome, Cognome, Indirizzo, Città)
MULTE(IdMulta, CodFisc, DataMulta, Somma)

Interrogazione
Selezionare il nome e il cognome delle persone per cui il numero di multe ricevute nel 2005 è superiore al numero di multe ricevute nel 2004 dalla stessa persona.

Soluzione

```
SELECT Nome, Cognome
 FROM PERSONE P, MULTE M
WHERE P.CodFisc=M.CodFisc
  AND DataMulta>='1/1/2005'
  AND DataMulta<='31/12/2005'
GROUP BY P.CodFisc, Nome, Cognome
HAVING COUNT(*)>(SELECT COUNT(*) FROM MULTE M2
                 WHERE M2.CodFisc=P.CodFisc
                   AND DataMulta>='1/1/2004'
                   AND DataMulta<='31/12/2004');
```

Esercizio 2.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

SALA_RIUNIONE(CodiceSala, CodiceSede, NomeSala)
SEDE(CodiceSede, Nome, Indirizzo, Città, Stato)

Operazione.
Eliminare dalla base di dati tutte le sedi tedesche e le relative sale riunione.

Soluzione

```
DELETE FROM SALA_RIUNIONE
 WHERE CodiceSede IN (SELECT CodiceSede
                      FROM SEDE
                      WHERE Stato='Germania');

DELETE FROM SEDE
 WHERE Stato='Germania';
```


Esercizio 3.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

RIVISTA (CodR, NomeR, Editore)
ARTICOLO (CodA, Titolo, Argomento, CodR)

Interrogazione
Selezionare il codice e il nome delle riviste che hanno pubblicato almeno 1 articolo.

Soluzione

```
SELECT DISTINCT RIVISTA.CodR, NomeR
FROM RIVISTA, ARTICOLO
WHERE RIVISTA.CodR=ARTICOLO.CodR;
```

Esercizio 4.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

DIPENDENTE(Matr, NomeD)
LINGUE-CONOSCIUTE(Matr, Lingua)
PROGETTO(CodP, NomeP)
LAVORA-IN(Matr, CodP, DataInizio, DataFine, Mansione)

Interrogazione
Per i dipendenti che hanno lavorato complessivamente per più di 6 mesi nello stesso progetto, selezionare matricola, nome e numero totale di progetti diversi in cui hanno lavorato.

Soluzione

```
SELECT D.Matr, NomeD, COUNT(DISTINCT CodP)
FROM DIPENDENTE D, LAVORA-IN LI
WHERE LI.Matr=D.Matr
  AND D.Matr IN
      (SELECT Matr FROM LAVORA-IN
       GROUP BY Matr, CodP
       HAVING SUM(MONTHS(DataFine)-MONTHS(DataInizio))>6
GROUP BY D.Matr, NomeD;
```

MONTHS rappresenta una semplificazione dell'applicazione delle funzioni che permettono di estrarre il mese di una data.

Esercizio 5.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

EDITORE(CodE, NomeEditore, Indirizzo, Città)
PUBBLICAZIONE(CodP, Titolo, NomeAutore, CodE)
LIBRERIA(CodL, NomeLibreria, Indirizzo, Città)
VENDITA(CodP, CodL, Data,CopieVendute)

Interrogazione
Selezionare il nome degli editori per cui almeno 10 pubblicazioni sono state vendute nel 2002 nelle librerie di Roma in più di 2.000 copie.

Soluzione

```
SELECT NomeEditore FROM EDITORE E, PUBBLICAZIONE P1
WHERE P1.CodE=E.CodE
  AND CodP IN
      (SELECT CodP FROM VENDITA V, LIBRERIA L
       WHERE V.CodL=L.CodL
         AND Data>='1/1/2002' AND Data<='31/12/2002'
         AND L.Citta='Roma'
       GROUP BY CodP
       HAVING SUM(CopieVendute)>2000)
GROUP BY E.CodE, NomeEditore
HAVING COUNT(*)>=10;
```

Esercizio 6.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

RIVISTA (CodR, NomeR, Editore)
ARTICOLO (CodA, Titolo, Argomento, CodR)

Interrogazione
Selezionare il codice delle riviste che hanno pubblicato almeno 1 articolo di motociclismo (Argomento='motociclismo').

Soluzione

```
SELECT DISTINCT CodR FROM ARTICOLO
WHERE Argomento='motociclismo';
```

Esercizio 7.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

RIVISTA (CodR, NomeR, Editore)
ARTICOLO (CodA, Titolo, Argomento, CodR)

Interrogazione
Selezionare il numero di articoli pubblicati sulla rivista D100 (CodR='D100').

Soluzione

```
SELECT COUNT(*) FROM ARTICOLO
WHERE CodR='D100';
```


Esercizio 8.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

DIPENDENTE(Matricola, Nome, Cognome, Filiale)
LINGUE_CONOSCIUTE(Matricola, Lingua)

Interrogazione.
Selezionare matricola, nome e cognome dei dipendenti che conoscosco sia il francese sia l'inglese.

Soluzione

```
SELECT Matricola, Nome, Cognome
FROM DIPENDENTE D, LINGUE_CONOSCIUTE L
WHERE D.Matricola=L.Matricola
  AND Lingua='Francese'
  AND Matricola IN (SELECT Matricola
                    FROM LINGUE_CONOSCIUTE
                    WHERE Lingua='Inglese');
```

Esercizio 9.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

FORNITORE(PartitaIVA, RagioneSociale, Stato)
CLIENTE(CodiceFiscale, Nome, Cognome, Stato)

Interrogazione.
Selezionare gli stati associati ad almeno un cliente oppure ad almeno un fornitore.

Soluzione

```
SELECT Stato
FROM FORNITORE
UNION
SELECT Stato
FROM CLIENTE;
```

Esercizio 10.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

GARA(CodG, Luogo, Data, Disciplina)
ATLETA(CodA, Nome, Nazione, DataNascita)
PARTECIPAZIONE(CodG, CodA,PosizioneArrivo, Tempo)

Interrogazione
Selezionare il codice e il nome degli atleti italiani che sono giunti primi in almeno una gara di lancio del peso.

Soluzione

```
SELECT DISTINCT A.CodA, Nome FROM ATLETA A, PARTECIPAZIONE P, GARA G
WHERE P.CodA=A.CodA
  AND P.CodG=G.CodG
  AND Disciplina='lancio del peso'
  AND Nazione='Italia'
  AND PosizioneArrivo=1;
```


Esercizio 11.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

DOCENTE(CodDocente, Nome, Cognome, DataNascita)
CORSI(CodCorso, NomeCorso, Sede, CodDocente)
LEZIONI_SVOLTE(CodCorso, Data, OraInizio, Aula, Durata)

Operazione di gestione da eseguire.
Scrivere, nell'ordine appropriato, i comandi SQL necessari per eliminare le tabelle CORSI e LEZIONI_SVOLTE.

Soluzione

```
DROP TABLE LEZIONI_SVOLTE;

DROP TABLE CORSI;
```

Esercizio 12.

È dato lo schema relazionale costituito dalla seguente tabella (le chiavi primarie sono sottolineate)

DOCENTE(CodDocente, Nome, Cognome, AnnoNascita)

Operazione di gestione da eseguire.
Scrivere il comando SQL necessario per aggiungere un vincolo sulla colonna AnnoNascita della tabella DOCENTE tale da imporre che AnnoNascita assuma valori successivi al 1899.

Soluzione

```
ALTER TABLE DOCENTE
 ADD CONSTRAINT VincoloAnnoDiNascita CHECK(AnnoNascita>1899);
```

Esercizio 13.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

PROFESSORI(Matricola, Nome, Cognome, Dipartimento)
PRESIDI(Matricola, Facoltà, DataNomina)
VICE_PRESIDI(Matricola, Facoltà, DataNomina)

Interrogazione.
Selezionare la matricola dei professori che non sono presidi.

Soluzione

```
SELECT Matricola
FROM PROFESSORI
WHERE Matricola NOT IN (SELECT Matricola
                        FROM PRESIDI);
```

Oppure

```
SELECT Matricola
FROM PROFESSORI
EXCEPT
SELECT Matricola
FROM PRESIDI;
```

Esercizio 14.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

SALA_RIUNIONE(CodiceSala, CodiceSede, NomeSala, Dimensione, NumeroPosti)
SEDE(CodiceSede, Nome, Indirizzo, Città, Stato)

Operazione.
Ridurre a 50 il numero massimo di posti per le sale più piccole di 100mq che hanno attualmente un numero di posti superiore a 50.

Soluzione

```
UPDATE SALA_RIUNIONE SET NumeroPosti=50
 WHERE Dimensione<100
   AND NumeroPosti>50;
```

Esercizio 15.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

RIVISTA (CodR, NomeR, Editore)
ARTICOLO (CodA, Titolo, Argomento, CodR)

Interrogazione
Selezionare il codice, il nome e il numero di articoli pubblicati su ogni rivista.

Soluzione

```
SELECT RIVISTA.CodR, NomeR, COUNT(*)
FROM RIVISTA, ARTICOLO
WHERE RIVISTA.CodR=ARTICOLO.CodR
GROUP BY RIVISTA.CodR, NomeR;
```

Esercizio 16.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

RIVISTA (CodR, NomeR, Editore)
ARTICOLO (CodA, Titolo, Argomento, CodR)

Interrogazione
Selezionare il codice delle riviste che hanno pubblicato almeno due articoli di motociclismo.

Soluzione

```
SELECT CodR FROM ARTICOLO
 WHERE Argomento='motociclismo'
GROUP BY CodR
HAVING COUNT(*)>1;
```


Esercizio 17.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

CORSO (CodCorso, NomeC, Anno, Semestre)
ORARIO_LEZIONI (CodCorso, GiornoSettimana, OraInizio, OraFine, Aula)

Interrogazione
Selezionare codice corso, nome corso e numero totale di ore di lezione settimanali per i corsi del terzo anno per cui il numero complessivo di ore di lezione settimanali è superiore a 10 e le lezioni sono in più di tre giorni diversi della settimana.

Soluzione

```
SELECT C.CodCorso, C.NomeC, SUM(OraFine-OraInizio)
FROM CORSO C, ORARIO_LEZIONI OL
WHERE C.CodCorso=OL.CodCorso
  AND C.Anno = 3
GROUP BY C.Corso, C.NomeC
HAVING SUM(OraFine-OraInizio)>10
   AND COUNT(DISTINCT GiornoSettimana)>3;
```

Esercizio 18.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

ALLOGGIO (CodA, Indirizzo, Città, Superficie, CostoAffittoMensile)
CONTRATTO-AFFITTO (CodC, DataInizio, DataFine, NomePersona, CodA)
N.B. Superficie espressa in metri quadri. Per i contratti in corso, DataFine è NULL.

Interrogazione
Selezionare, per le città in cui sono stati stipulati almeno 100 contratti, la città, il costo mensile massimo degli affitti, il costo mensile medio degli affitti, la durata massima dei contratti, la durata media dei contratti e il numero totale di contratti stipulati.

Soluzione

```
SELECT Citta, MAX(CostoAffittoMensile), AVG(CostoAffittoMensile),
       MAX(DataFine-DataInizio), AVG(DataFine-DataInizio), COUNT(*)
FROM ALLOGGIO A, CONTRATTO-AFFITTO C
WHERE A.CodA=C.CodA
GROUP BY Citta
HAVING COUNT(*)>=100;
```


Esercizio 19.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

SALA_RIUNIONE(CodiceSala, CodiceSede, NomeSala, NumeroPosti)
SEDE(CodiceSede, Nome, Indirizzo, Città, Stato)

Operazione.
Impostare a 100 il numero di posti della sala numero 1 (CodiceSala=1) collacata presso la sede 15 (CodiceSede=15).

Soluzione

```
UPDATE SALA_RIUNIONE SET NumeroPosti=100
 WHERE CodiceSala=1 AND CodiceSede=15;
```

Esercizio 20.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

RIVISTA (CodR, NomeR, Editore)
ARTICOLO (CodA, Titolo, Argomento, CodR)

Interrogazione
Selezionare gli editori che non hanno mai pubblicato articoli di motociclismo.

Soluzione

```
SELECT Editore FROM RIVISTA
WHERE Editore NOT IN
      (SELECT Editore FROM ARTICOLO A, RIVISTA R
       WHERE A.CodR=R.CodR
         AND Argomento='motociclismo');
```

Esercizio 21.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

GRANPREMIO(NumGP, Anno, Stato)
PILOTA(CodP, Cognome, Nome, Nazione)
PARTECIPAZIONE(NumGP, Anno, CodP)

Operazione di gestione da eseguire.
Scrivere il comando SQL necessario per aggiungere un vincolo tale da imporre che l'attributo NumGP della tabella GRANPREMIO possa assumere esclusivamente valori maggiori o uguali a 1.

Soluzione

```
ALTER TABLE GRANPREMIO
 ADD CONSTRAINT VincoloNumGP CHECK(NumGP>=1);
```


Esercizio 22.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

```
ORCHESTRA(CodO, NomeO, NomrDirettore, numElementi)
CONCERTI(CodC, Data, CodO, CodS, PrezzoBiglietto)
SALE(CodS, NomeS, Città, Capienza)
```

Interrogazione
Selezionare il codice e il nome delle orchestre con più di 30 elementi che hanno tenuto concerti sia a Torino, sia a Milano e non hanno mai tenuto concerti a Bologna.

Soluzione

```
SELECT CodO, NomeO FROM ORCHESTRA
WHERE NumElementi>30
  AND CodO IN
      (SELECT C1.CodO FROM CONCERTI C1, SALE S1
       WHERE C1.CodS=S1.CodS
         AND S1.Citta='Torino')
  AND CodO IN
      (SELECT C2.CodO FROM CONCERTI C2, SALE S2
       WHERE C2.CodS=S2.CodS
         AND S2.Citta='Milano')
  AND CodO NOT IN
      (SELECT C3.CodO FROM CONCERTI C3, SALE S3
       WHERE C3.CodS=S3.CodS
         AND S3.Citta='Bologna');
```

Esercizio 23.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

```
RIVISTA (CodR, NomeR, Editore)
ARTICOLO (CodA, Titolo, Argomento, CodR)
```

Interrogazione
Selezionare il codice e il nome delle riviste che hanno pubblicato almeno 10 articoli di automobilismo e almeno 25 articoli di motociclimo.

Soluzione

```
SELECT CodR, NomeR FROM RIVISTA
 WHERE CodR IN
       (SELECT CodR ARTICOLO
         WHERE Argomento='automobilismo'
        GROUP BY CodR
        HAVING COUNT(*)>=10)
   AND CodR IN
       (SELECT CodR ARTICOLO
         WHERE Argomento='motociclismo'
        GROUP BY CodR
        HAVING COUNT(*)>=25);
```

Esercizio 24.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

```
RIUNIONE(CodR, Descrizione, DataRiunione)
DIPENDENTE(CodD, Nome, Cognome, DataNascita, Città)
PARTECIPA_RIUNIONE(CodD, CodR)
```

Interrogazione
Visualizzare il codice dei dipendenti che hanno partecipato solamente alle riunioni alle quali ha partecipato il dipendente D100 (CodD='D100').

Soluzione

```
SELECT CodD
FROM PARTECIPA_RIUNIONE
WHERE CodR IN
      (SELECT CodR FROM PARTECIPA_RIUNIONE
      WHERE CodD='D100')
GROUP BY CodD
HAVING COUNT(*)= (SELECT COUNT(*) FROM PARTECIPA_RIUNIONE
                  WHERE CodD='D100');
```

Oppure

```
SELECT CodD
FROM DIPENDENTE D
WHERE CodD NOT IN
      (SELECT CodD FROM PARTECIPA_RIUNIONE
       WHERE CodR NOT IN
             (SELECT CodR FROM PARTECIPA_RIUNIONE
              WHERE CodD='D100'));
```

Oppure

```
SELECT CodD
FROM DIPENDENTE D
WHERE NOT EXISTS
      (SELECT * FROM PARTECIPA_RIUNIONE PR
       WHERE PR.CodD=D.CodD
         AND PR.CodR NOT IN
             (SELECT CodR FROM PARTECIPA_RIUNIONE
              WHERE CodD='D100'));
```

Esercizio 25.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

SALA_RIUNIONE(CodiceSala, CodiceSede, NomeSala, NumeroPosti)
SEDE(CodiceSede, Nome, Indirizzo, Città, Stato)
PRENOTAZIONE(CodiceSala, CodiceSede, Data, OraInizio, Durata)

Operazione.
Eliminare dalla base di dati le sale riunioni con meno di 10 posti e tutte le prenotazioni relative a tali sale.

Soluzione

```
DELETE FROM PRENOTAZIONE
 WHERE (CodiceSala, CodiceSede) IN
       (SELECT CodiceSala, CodiceSede
        FROM SALA_RIUNIONE
        WHERE NumeroPosti<10);

DELETE FROM SALA_RIUNIONE
 WHERE NumeroPosti<10);
```

Esercizio 26.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

QUIZ(CodQuiz, Argomento, Punteggio)
STUDENTE(Matricola, Nome, Indirizzo, Città)
RISULTATO_TEST(Matricola, CodQuiz,RispostaCorretta)

Interrogazione
Selezionare il nome degli studenti di Torino che hanno conseguito il punteggio massimo possibile nei quiz di matematica.

Soluzione

```
SELECT Nome FROM STUDENTE S, RISULTATO_TEST R, QUIZ Q
WHERE Citta='Torino'
  AND R.Matricola=S.Matricola
  AND R.CodQuiz=Q.CodQuiz
  AND RispostaCorretta='si'
  AND Argomento='matematica'
GROUP BY S.Matricola, Nome
HAVING SUM(Punteggio)=(SELECT SUM(Punteggio) FROM QUIZ
                       WHERE Argomento='matematica');
```

Esercizio 27.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

RIUNIONE(CodR, Descrizione, DataRiunione)
DIPENDENTE(CodD, Nome, Cognome, DataNascita, Città)
PARTECIPA_RIUNIONE(CodD, CodR)

Interrogazione
Visualizzare il codice dei dipendenti che hanno partecipato a tutte le riunioni che si sono svolte a gennaio 2006.

Soluzione

```
SELECT CodD
FROM PARTECIPA_RIUNIONE PR, RIUNIONE R
WHERE PR.CodR=R.CodR AND DataRiunione>=‘‘01/01/2006’’
AND DataRiunione<=‘‘31/01/2006’’
GROUP BY CodD
HAVING COUNT(*)=(SELECT COUNT(*)
                 FROM RIUNIONE
                 WHERE DataRiunione>=‘‘01/01/2006’’
                 AND DataRiunione<=‘‘31/01/2006’’);
```

Esercizio 28.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

GRANPREMIO(NumGP, Anno, Stato)
PILOTA(CodP, Cognome, Nome, Nazione)
PARTECIPAZIONE(NumGP, Anno, CodP)

Operazione di gestione da eseguire.
Scrivere i comandi SQL necessari per creare tutte le tabelle riportate nello schema relazionale. Selezionare per ogni attributo il tipo di dato ritenuto più opportuno e indicare i vincoli di integrità ritenuti necessari.

Soluzione

```
CREATE TABLE GRANPREMIO
( NumGP  SMALLINT,
  Anno   SMALLINT,
  STATO  VARCHAR(15) NOT NULL,
  PRIMARY KEY(NumGP, Anno)
);

CREATE TABLE PILOTA
( CodP     SMALLINT,
  Cognome  VARCHAR(20) NOT NULL,
  Nome     VARCHAR(20) NOT NULL,
  Nazione  VARCHAR(15) NOT NULL,
  PRIMARY KEY(CodP)
);

CREATE TABLE PARTECIPAZIONE
```

```
( NumGP  SMALLINT,
  Anno   SMALLINT,
  CodP   SMALLINT,
  PRIMARY KEY(NumGP, Anno, CodP),
  FOREIGN KEY (NumGP, Anno) REFERENCES GRANPREMIO(NumGP, Anno)
              ON DELETE CASCADE
              ON UPDATE CASCADE,
  FOREIGN KEY CodP REFERENCES PILOTA(CodP)
              ON DELETE CASCADE
              ON UPDATE CASCADE
);
```

Esercizio 29.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

CONTRIBUENTE(CodFiscale, Nome, Via, Città)
DICHIARAZIONE(CodDichiarazione, Tipo, Reddito)
PRESENTA(CodFiscale, CodDichiarazione, Data)

Interrogazione
Visualizzare codice, nome e media dei redditi dichiarati dal 1990 in poi per i contribuenti tali che il massimo reddito da loro dichiarato dal 1990 in poi sia superiore alla media dei redditi calcolata su tutte le dichiarazioni nella base di dati.

Soluzione

```
SELECT C.CodFiscale, C.Nome, AVG(Reddito) FROM CONTRIBUENTE C,
DICHIARAZIONE D, PRESENTA P
WHERE C.CodFiscale=P.CodFiscale
  AND D.CodDichiarazione=P.CodDichiarazione
  AND P.Data>'1/1/1990'
GROUP BY C.CodFiscale, C.Nome
HAVING MAX(D.Reddito)>(SELECT AVG(REddito) FROM DICHIARAZIONE);
```

Esercizio 30.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

RIVISTA (CodR, NomeR, Editore)
ARTICOLO (CodA, Titolo, Argomento, CodR)

Interrogazione
Selezionare il codice e il nome di tutte le riviste presenti nella base di dati.

Soluzione

```
SELECT CodR, NomeR FROM RIVISTA;
```

Esercizio 31.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

SALA_RIUNIONE(CodiceSala, CodiceSede, NomeSala)
SEDE(CodiceSede, Nome, Indirizzo)

Operazione.
Inserire nella base di dati la sala 5 con nome: 'Sala conferenze', CodiceSede: 1.

Soluzione

```
INSERT INTO SALA_RIUNIONI(CodiceSala, NomeSala, CodiceSede)
        VALUES(5, 'Sala conferenze', 1);
```

Esercizio 32.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

GRANPREMIO(NumGP,Anno,Stato,Città)
PILOTA(CodP,Nome,Nazionalità)
PARTECIPA(NumGP,Anno,CodP)

Interrogazione
Selezionare gli anni in cui si sono tenuti gran premi in almeno 15 stati diversi e meno di 2 gran premi in Italia.

Soluzione

```
SELECT Anno FROM GRANPREMIO
WHERE Anno NOT IN
      (SELECT Anno FROM GRANPREMIO
       WHERE Stato='Italia'
       GROUP BY Anno
       HAVING COUNT(*)>=2)
GROUP BY Anno
HAVING COUNT(DISTINCT Stato)>=15;
```

Esercizio 33.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

PERSONA(CodF,Cognome,DataNascita)
CAMPO(CodCampo,Coperto)
PRENOTAZIONI(CodCampo,Data,OraInizio,OraFine,CodFisc)

L'attributo Coperto della relazione CAMPO può assumere valore 'si' e 'no'.

Interrogazione
Selezionare il codice del campo e il numero di prenotazioni fatte nell'anno 1981 per tutti i campi che sono stati prenotati da almeno 50 persone diverse nel corso dell'anno 1980.

Soluzione

```
SELECT CodCampo, COUNT(*) FROM PRENOTAZIONI
WHERE Data>='1/1/1981'
  AND Data<='31/12/1981'
  AND CodCampo IN
      (SELECT CodCampo FROM PRENOTAZIONI
       WHERE Data>='1/1/1980'
         AND Data<='31/12/1980'
       GROUP BY CodCampo
       HAVING COUNT(DISTINCT CodFisc)>=50)
GROUP BY CodCampo;
```

Esercizio 34.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

DIPENDENTE(Matricola, Nome, Cognome, Filiale)
LINGUE_CONOSCIUTE(Matricola, Lingua)

Interrogazione.
Selezionare matricola, nome e cognome dei dipendenti che non conoscosco l'inglese.

Soluzione

```
SELECT Matricola, Nome, Cognome
FROM DIPENDENTE
WHERE Matricola NOT IN (SELECT Matricola
                        FROM LINGUE_CONOSCIUTE
                        WHERE Lingua='Inglese');
```

Esercizio 35.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

ALBERGO(CodA, Nome, Categoria)
SERVIZIO(CodS, NomeServizio)
SERVIZI_OFFERTI(CodA, CodS)

Operazione di gestione da eseguire.
Scrivere i comandi SQL necessari per creare tutte le tabelle riportate nello schema relazionale. Selezionare per ogni attributo il tipo di dato ritenuto più opportuno e indicare i vincoli di integrità ritenuti necessari.

Soluzione

```
CREATE TABLE ALBERGO
( CodA      SMALLINT,
  Nome      VARCHAR(20) NOT NULL,
  Categoria SMALLINT,
  PRIMARY KEY(CodA)
);

CREATE TABLE SERVIZIO
( CodS         SMALLINT,
  NomeServizio VARCHAR(20) NOT NULL,
  PRIMARY KEY(CodS)
);

CREATE TABLE SERVIZI_OFFERTI
( CodA   SMALLINT,
  CodS   SMALLINT,
  PRIMARY KEY(CodA, CodS),
  FOREIGN KEY CodA REFERENCES ALBERGO(CodA)
              ON DELETE CASCADE
              ON UPDATE CASCADE,
  FOREIGN KEY CodS REFERENCES SERVIZIO(CodS)
              ON DELETE CASCADE
              ON UPDATE CASCADE
);
```


Esercizio 36.

È dato lo schema relazionale costituito dalle seguenti tabelle (le chiavi primarie sono sottolineate)

MEDICO(Matr, Nome)
MEDICINALE(CodM, NomeM)
PAZIENTE(CodP, NomeP, DataNascita)
PRESCRIZIONE(Matr, CodM, CodP, Data)

Interrogazione
Selezionare codice e nome dei pazienti a cui non sono mai stati prescritti medicinali oppure sono stati prescritti medicinali solo dopo il compimento di 40 anni.

Soluzione

```
SELECT CodP, NomeP FROM PAZIENTE
WHERE CodP NOT IN
      (SELECT P.CodP FROM PAZIENTE P, PRESCRIZIONE PR
       WHERE PR.CodP=P.CodP
         AND YEARS(PR.Data)-YEARS(P.DataNascita)<40);
```

YEARS rappresenta una semplificazione dell'applicazione delle funzioni che permettono di estrarre l'anno di una data.

Esercizi SQL di Preparazione all'Esame

difficoltà delle query:
* = facile
** = media
* ** = difficile

1. Si consideri la base di dati definita dal seguente schema relazionale:

   Programmi(Codice, NomeProgramma, Rete, Tipologia)
   Artisti(CProgramma, NomeArt)
   Sondaggi(CodProg, Auditel)

   Gli attributi CProgramma e CodProg sono chiavi esterne di Programmi.

   Esprimere le seguenti interrogazioni:

   a. (*) Restituire l'indice Auditel del programma con codice P17

   SELECT Auditel FROM Sondaggi WHERE CodProg = 'P17'

   b. (**) Restituire l'indice Auditel dei programmi di Fiorello

   ```
   SELECT Auditel
   FROM Sondaggi S, Artisti A
   WHERE    S.CodProg = A.CProgramma AND A.NomeArt = 'Fiorello'
   ```

   c. (***) Restuire i nomi dei programmi con almeno due artisti

   ```
   SELECT DISTINCT NomeProgramma
   FROM Programmi P, Artisti A1, Artisti A2
   WHERE    P.Codice = A1.CProgramma AND
            P.Codice = A2.CProgramma AND
            A1.NomeArt > A2.NomeArt
   ```

   d. (***) Restituire i nomi dei programmi con un solo artista

   È sufficiente togliere (*except*) i programmi con almeno 2 artisti (query 1.c) dai programmi con almeno un artista:

   ```
   (SELECT DISTINCT NomeProgramma
    FROM Programmi, Artisti
   WHERE Codice = CProgramma)
       EXCEPT
   (SELECT DISTINCT NomeProgramma
    FROM Programmi P, Artisti A1, Artisti A2
    WHERE   P.Codice = A1.CProgramma AND
            P.Codice = A2.CProgramma AND
            A1.NomeArt > A2.NomeArt)
   ```

2. Si consideri la base di dati definita dal seguente schema relazionale:

   Partiti(Codice,NomePartito,Coalizione)
   Candidati(CPartito,Citta, NomeCand)
   Sondaggi(CodPart,PercFavorevoli)

   Gli attributi CPartito e CodPart sono chiavi esterne di Partiti.

   Esprimere le seguenti interrogazioni:

   a. (*) Restituire la percentuale di favorevoli del partito con codice A13.

   SELECT PercFavorevoli FROM Sondaggi WHERE CodPart = 'A13'

   b. (**) Restituire la percentuale di favorevoli del partito del candidato con nome 'Rossi'.

   SELECT PercFavorevoli FROM Sondaggi S, Candidati C
   WHERE S.CodPart = C.CPartito AND C.NomeCand = 'Rossi'

   c. (***) Restuire i nomi dei partiti con esattamente due candidati

   È sufficiente togliere (*except*) i partiti con almeno 3 candidati dai programmi con almeno due candidati:

   (SELECT DISTINCT NomePartito
   FROM Partiti P, Candidati C1, Candidati C2
   WHERE C1.CPartito = P.Codice AND C2.CPartito = P.Codice AND
   C1.Citta > C2.Citta)
   EXCEPT
   (SELECT DISTINCT NomePartito
   FROM Partiti P, Candidati C1, Candidati C2, Candidati C3
   WHERE C1.CPartito = P.Codice AND C2.CPartito = P.Codice AND
   C3.CPartito = P.Codice AND
   C1.Citta > C2.Citta AND C2.Citta > C3.Citta)

   d. (**) Restituire i nomi dei dei partiti con percentuale di favorevoli del 30%.

   SELECT NomePartito FROM Partiti P, Sondaggi S
   WHERE P.Codice = S.CodPart AND S.PercFavorevoli = 30

   e. (*) Restituire i codici dei partiti con percentuale di favorevoli del 30%.

   SELECT CodPart FROM Sondaggi WHERE PercFavorevoli = 30

3. Si consideri la base di dati definita dal seguente schema relazionale:

   Impiegato(CodFisc,Nome,Cognome,Recapito,Dip)
   Lavora_Su(Imp,Prog)
   Progetto(CodP,NomeP,Budget)
   Dipartimento(CodD,NomeD,Indirizzo,Citta)

   L'attributo Dip è chiave esterna di Dipartimento. L'attributo Imp è chiave esterna di Impiegato. L'attributo Prog è chiave esterna di Progetto.

   Esprimere le seguenti interrogazioni:

   a. (*) Trovare il nome delle persone con cognome 'Rossi'.

   SELECT Nome FROM Impiegato WHERE Cognome = 'Rossi'

   b. (**) Trovare i nomi dei dipartimenti in cui lavorano gli impiegati che stanno sul progetto 4.

   SELECT D.NomeD
   FROM Dipartimento D, Impiegato I, Lavora_Su L
   WHERE I.Dip = D.CodD AND I.CodFisc = L.Imp AND L.Prog = 4

   c. (***) Per ogni progetto in cui lavorano almeno due impiegati, si recuperi il numero del progetto e il nome del progetto.

   SELECT DISTINCT P.CodP, P.NomeP
   FROM Progetto P, Lavora_Su L1, Lavora_Su L2
   WHERE L1.Prog = P.CodP AND L2.Prog = P.CodP AND L1.Imp > L2.Imp

   d. (*) Trovare il nome e cognome delle persone con recapito '011 6706733'

   SELECT Nome, Cognome FROM Impiegato WHERE Recapito = '011 6706733'

   e. (***) Per ogni progetto in cui lavorano al più due impiegati, si recuperi il numero del progetto e il nome del progetto.

   È sufficiente togliere (*except*) i progetti con almeno 3 impiegati dall'elenco di tutti I progetti:

   (SELECT DISTINCT P.CodP, P.NomeP FROM Progetto P)
   EXCEPT
   (SELECT DISTINCT P.CodP, P.NomeP
   FROM Progetto P, Lavora_Su L1, Lavora_Su L2, Lavora_Su L3
   WHERE L1.Prog = P.CodP AND L2.Prog = P.CodP AND L3.Prog = P.CodP AND
   L1.Imp > L2.Imp AND L2.Imp > L3.Imp)

CAPITOLO 4

ESERCIZI SU SQL

(le soluzioni sono riportate da pag. 5 in poi)
(Nota: gli esercizi non sono sempre in ordine di difficoltà)

Esercizio 1

Si prendano tutti gli schemi relazionali considerati negli esercizi sulla progettazione logica pubblicati sulla pagina web del corso. Si diano le definizioni SQL delle relazioni di ciascun schema relazionale, facendo particolare attenzione alle chiavi primarie, ai vincoli intrarelazionali e ai vincoli d'integrità referenziale.

Esercizio 2

Dare le definizioni SQL delle tre tabelle

```
FONDISTA(Nome, Nazione, Età)
GAREGGIA(NomeFondista, NomeGara, Piazzamento)
GARA(Nome, Luogo, Nazione, Lunghezza)
```

rappresentando in particolare i vincoli di foreign key della tabella `GAREGGIA`

Esercizio 3

Dare le definizioni SQL delle tabelle

```
AUTORE (Nome, Cognome, DataNascita, Nazionalità)
LIBRO (TitoloLibro, NomeAutore, CognomeAutore, Lingua)
```

Per il vincolo foreign key specificare una politica di cascade sulla cancellazione e di set null sulle modifiche.

Esercizio 4

Dato lo schema dell'esercizio precedente, spiegare cosa può capitare con l'esecuzione dei seguenti comandi di aggiornamento:

1. `delete from AUTORE where Cognome = 'Rossi'`
2. `update LIBRO set Autore= 'Umberto' where Autore = 'Eco'`
3. `insert into AUTORE(Nome,Cognome) values('Antonio','Bianchi')`
4. `update AUTORE set Nome = 'Italo' where Cognome = 'Calvino'`

Esercizio 5

Con riferimento ad una relazione `PROFESSORI(CF, Nome, Eta, Qualifica)`, scrivere le interrogazioni SQL che calcolano l'età media dei professori di ciascuna qualifica, nei due casi seguenti:

1. se l'età non è nota si usa per essa il valore nullo
2. se l'età non è nota si usa per essa il valore 0.

Esercizio 6

Dato il seguente schema:

```
AEROPORTO(Città, Nazione,NumPiste)
VOLO(IdVolo,GiornoSett,CittàPart,OraPart,CittàArr,OraArr,TipoAereo)
AEREO(TipoAereo,NumPasseggeri,QtaMerci)
```

scrivere le interrogazioni SQL che permettono di determinare:

1. Le città con un aeroporto di cui non è noto il numero di piste;
2. Le nazioni da cui parte e arriva il volo con codice AZ274;
3. I tipi di aereo usati nei voli che partono da Torino;
4. I tipi di aereo e il corrispondente numero di passeggeri per i tipi di aereo usati nei voli che partono da Torino. Se la descrizione dell'aereo non è disponibile, visualizzare solamente il tipo;
5. Le città da cui partono voli internazionali;
6. Le città da cui partono voli diretti a Bologna, ordinate alfabeticamente;
7. Il numero di voli internazionali che partono il giovedì da Napoli;

8. Il numero di voli internazionali che partono ogni settimana da città italiane (farlo in due modi, facendo comparire o meno nel risultato gli aeroporti senza voli internazionali);
9. Le città francesi da cui partono più di venti voli alla settimana diretti in Italia;
10. Gli aeroporti italiani che hanno solo voli interni. Rappresentare questa interrogazione in quattro modi:
    a) con operatori insiemistici;
    b) con un interrogazione nidificata con l'operatore not in;
    c) con un interrogazione nidificata con l'operatore not exists;
    d) con l'outer join e l'operatore di conteggio
11. Le città che sono servite dall'aereo caratterizzato dal massimo numero di passeggeri;

Esercizio 7

Dato il seguente schema:

```
DISCO(NroSerie, TitoloAlbum,Anno,Prezzo)
CONTIENE(NroSerieDisco,CodiceReg,NroProg)
ESECUZIONE(CodiceReg,TitoloCanz,Anno)
AUTORE(Nome,TitoloCanzone)
CANTANTE(NomeCantante,CodiceReg)
```

formulare le interrogazioni SQL che permettono di determinare:

1. I cantautori (persone che hanno cantato e scritto la stessa canzone) il cui nome inizia per 'D';
2. I titoli dei dischi che contengono canzoni di cui non si conosce l'anno di registrazione;
3. I pezzi del disco con numero di serie 78574, ordinati per numero progressivo, con indicazione degli interpreti per i pezzi che hanno associato un cantante;
4. Gli autori e i cantanti puri, ovvero autori che non hanno mai registrato una canzone e cantanti che non hanno mai scritto una canzone;
5. Gli autori solisti di “collezioni di successi” (dischi in cui tutte le canzoni sono di un solo cantante e in cui almeno tre registrazioni sono di anni precedenti la pubblicazione del disco);
6. I cantanti che non hanno mai registrato una canzone come solisti;
7. I cantanti che hanno sempre registrato canzoni come solisti.

Esercizio 8

Considerare la base di dati relazionale definita per mezzo delle seguenti istruzioni:

```
create table Studenti (
     matricola numeric not null primary key,
     cognome char(20) not null,
     nome char(20) not null,
     eta numeric not null
);
create table Esami (
     codiceCorso numeric not null,
     studente numeric not null references Studenti(matricola),
     data date not null,
     voto numeric not null,
     primary key (codiceCorso, studente, data)
).
```

Si supponga che vengano registrati anche gli esami non superati, con voti inferiori al 18.
Formulare in SQL:

1. l’interrogazione che trova gli studenti che non hanno superato esami;
2. l’interrogazione che trova gli studenti che hanno riportato in almeno un esame un voto più alto di Archimede Pitagorico;
3. l’interrogazione che trova i nomi degli studenti che hanno superato almeno due esami;
4. l’interrogazione che trova, per ogni studente, il numero di esami superati e la relativa media.

Esercizio 9

Dare una sequenza di comandi di aggiornamento che modifichi l'attributo Stipendio della tabella Impiegato, aumentando di 200€ gli stipendi sotto i 30000€ e diminuendo del 5% gli stipendi sopra i 30000€ (gli stipendi di 30.000 € rimangono invariati).

Esercizio 10

Considerare la base di dati relazionale con il seguente schema:

- `PRODOTTI (Codice, Nome, Categoria)`
- `VENDITE (CodiceProdotto, Data, Incasso)`
  con vincolo di integrita referenziale fra l'attributo `CodiceProdotto` e la relazione `PRODOTTI`.

e la sua seguente istanza:

PRODOTTI		
Codice	Nome	Categoria
101	A	Bevanda
102	B	Bevanda
103	C	Pasta
104	D	Biscotti

VENDITE		
CodiceProdotto	Data	Incasso
101	24/11/2008	2.000
101	25/11/2008	1.000
102	23/11/2008	2.500
102	24/11/2008	4.000
103	25/11/2008	1.320

mostrare il risultato delle tre seguenti interrogazioni:

1. ```
   select Codice
   from Prodotti
   where not exists
       (select *
       from Vendite
       where CodiceProd=Codice);
   ```
2. ```
   select Codice
   from Prodotti
   where not exists
       (select *
       from Vendite
       where Data = 2008-11-24 and CodiceProd=Codice);
   ```
3. ```
   select Codice
   from Prodotti
   where not exists
       (select *
       from Vendite
       where Data = 2008-11-24);
   ```

Esercizio 11

Con riferimento all'esercizio precedente mostrare le istruzioni SQL per creare e popolare l'istanza di basi di dati mostrata.
   ```

Esercizio 12

Considerare la base di dati relazionale definita per mezzo delle seguenti istruzioni:

```
create table Tariffa (
     tipoAuto numeric not null primary key,
     costoAlKm numeric not null
);
create table Automobile (
     targa numeric not null primary key,
     tipologia char(20) not null references Tariffa(TipoAuto),
     lunghezza char(20) not null
);
create table Transito (
     codice numeric not null primary key,
     auto numeric not null references Automobile(targa),
     orarioIngresso numeric not null,
     orarioUscita numeric not null,
     KmPercorsi numeric not null
)
```

Formulare in SQL:

1. l'interrogazione che restituisce, per ogni transito, i dati del veicolo, del transito e il costo del pedaggio (ottenuto moltiplicando il costo al Km per i Km percorsi);
2. l'interrogazione che restituisce tutti i dati delle automobili che sono transitate più di una volta sull'autostrada;
3. l'interrogazione che restituisce le auto che in ogni transito hanno percorso sempre lo stesso numero di Km;
4. l'interrogazione che restituisce i dati dei transiti (gli stessi richiesti nella prima domanda) per cui la velocità media è superiore ai 140Km/h (si assuma che una differenza tra i due orari produca il risultato espresso in ore).

Esercizio 13

Si consideri una base di dati che gestisce dati relativi ai voli in partenza da un dato aeroporto (ad esempio Roma), con le seguenti relazioni:

- `AEROPORTI(Codice,Citta,Nome)`
- `AEREI(Codice,Nome,NumeroPosti)`
- `VOLI(Compagnia,Numero,Destinazione,OraPart,OraArr,Aereo)`
  con vincoli di integrità referenziale fra `Destinazione` e la relazione `AEROPORTI` e fra `Aereo` e la relazione `AEREI`.

Formulare in SQL:

1. l'interrogazione che trova le città raggiungibili con un volo diretto che utilizzi un aereo con almeno 200 posti.
2. l'interrogazione che trova le città raggiungibili con voli diretti e, per ciascuna, mostra il numero di tali voli.

SOLUZIONI

Esercizio 2

```
Create Table FONDISTA
(
     Nome  character(20) primary key,
     Nazione  character(30),
     Età integer
)
Create table GARA
(
     Nome  character(20) primary key,
     Luogo  character(20),
     Nazione character(20),
     Lunghezza integer
)
Create table GAREGGIA
(
     NomeFondista character(20) references FONDISTA(Nome),
     NomeGara character(20) references GARA(Nome),
     Piazzamento integer,
     primary key (NomeFondista, NomeGara)
)
```

Esercizio 3

```
Create table AUTORE
(
     Nome character(20),
     Cognome character(20),
     DataNascita date,
     Nazionalità character(20),
     primary key(Nome, Cognome)
)
Create table LIBRO
(
     TitoloLibro character(30) primary key,
     NomeAutore character(20),
     CognomeAutore character(20),
     Lingua character(20),
     foreign key (NomeAutore, CognomeAutore)
          references AUTORE(Nome, Cognome)
          on delete cascade on update set NULL
)
```

Esercizio 4

1. Il comando cancella dalla tabella AUTORE tutte le tuple con Cognome = 'Rossi'. A causa della politica cascade anche le tuple di LIBRO con CognomeAutore = 'Rossi' verranno eliminate.
2. Il comando non è corretto: Nome e Cognome sono attributi della tabella AUTORE e non della tabella LIBRO
3. Il comando aggiunge una nuova tupla alla tabella AUTORE. Non ha alcun effetto sulla tabella LIBRO
4. Le tuple di AUTORE con Cognome = Calvino vengono aggiornate a Nome = Italo. A causa della politica set null gli attributi NomeAutore e CognomeAutore delle tuple di Libro con CognomeAutore = Calvino vengono posti a NULL.

Esercizio 5

1. Le funzioni aggregative escludono dalla valutazione le ennuple con valori nulli:
```
select Qualifica, avg(Eta) as EtaMedia
from Professori
group by Qualifica
```

2. E' necessario escludere esplicitamente dal calcolo della media le tuple con il valore che denota l'informazione incompleta:
```
select Qualifica, avg(Eta) as EtaMedia
from Professori
where Eta <> 0
group by Qualifica.
```

Esercizio 6

1.
```
select Città
from AEROPORTO
where NumPiste is NULL
```

2.
```
select A1.Nazione, A2.Nazione
from AEROPORTO as A1 join VOLO on A1.Città=CittàArr
     join AEROPORTO as A2 on CittàPart=A2.Città
where IdVolo= 'AZ274'
```

3.
```
select TipoAereo
from VOLO
where CittàPart='Torino'
```

4.
```
select VOLO.TipoAereo, NumPasseggeri
from VOLO left join AEREO on VOLO.TipoAereo=aereo.TipoAereo
where CittàPart= 'Torino'
```

5.
```
select CittàPart
from AEROPORTO as A1 join VOLO on CittàPart=A1.Città
     join AEROPORTO as A2 on CittàArr=A2.Città
where A1.Nazione <> A2.Nazione
```

6.
```
select CittàPart
from VOLO
where CittàArr= 'Bologna'
order by CittàPart
```

7.
```
select count(*)
from VOLO join AEROPORTO on CittàArr=Città
where CittàPart = 'Napoli' and Nazione <> 'Italia' and
GiornoSett= 'Giovedì'
```

8.

a)
```
select count(*), CittàPart
from AEROPORTO as A1 join VOLO  on A1.Città=CittàPart
     join AEROPORTO as A2 on CittàArr=A2.Città
where A1.Nazione='Italia' and A2.Nazione <> ' Italia'
group by CittàPart
```

b)
```
select count(CittàArr)
from AEROPORTO as A1 join VOLO  on A1.Città=CittàPart
     join AEROPORTO as A2 on CittàArr=A2.Città
where A1.Nazione='Italia' and A2.Nazione <> ' Italia'
group by CittàPart
```

9.
```
select CittàPart
from AEROPORTO as A1 join VOLO on A1.Città=CittàPart
     join AEROPORTO as A2 on CittàArr=A2.Città
where A1.Nazione='Francia' and A2.Nazione= 'Italia'
group by CittàPart
Having count(*) >20
```

10.

a)
```
select CittàPart
from VOLO join AEROPORTO on CittàPart=Città
where Nazione = 'Italia'
except
select CittàPart
from AEROPORTO as A1 join VOLO on A1.Città=CittàPart
     join AEROPORTO as A2 on CittàArr=A2.Città
where (A1.Nazione=' Italia ' and A2.Nazione<>' Italia ' )
```

b)
```
select CittàPart
from VOLO join AEROPORTO on CittàPart=Città
where Nazione= 'Italia' and CittàPart not in
   ( select CittàPart
     from AEROPORTO as A1 join VOLO on
     A1.Città=CittàPart join AEROPORTO as A2 on CittàArr=A2.Città
     where A1.Nazione='Italia' and A2.Nazione<>'Italia' )
```

c)
```
select CittàPart
from VOLO join AEROPORTO as A1 on CittàPart=Città
where Nazione= 'Italia' and
not exists ( select *
             from VOLO join AEROPORTO as A2 on A2.Città=CittàArr
             where A1.Città=CittàPart and A2.Nazione<>'Italia' )
```

d)
```
select CittàPart
from AEROPORTO as A1 join VOLO on A1.Città=CittàPart
     left join AEROPORTO as A2
          on (CittàArr=A2.Città and A2.Nazione='Italia')
where A1.Nazione='Italia'
group by CittàPart
having count (district A2.Nazione)= 1 )
```

11.
```
select CittàPart
from VOLO join AEREO on VOLO.TipoAereo=AEREO.TipoAereo
where NumPasseggeri=( select max(NumPasseggeri)
                      from AEREO )
union
select CittàArr
from VOLO join AEREO on VOLO.TipoAereo=AEREO.TipoAereo
where NumPasseggeri=( select max(NumPasseggeri)
                      from AEREO )
```

Esercizio 7

1.
```
select NomeCantante
from CANTANTE join ESECUZIONE on
     CANTANTE.CodiceReg=ESECUZIONE.CodiceReg
     join AUTORE on ESECUZIONE.TitoloCanz=AUTORE.TitoloCanzone
where Nome=NomeCantante and Nome like 'd%'
```

2.
```
select TitoloAlbum
from DISCO join CONTIENE on DISCO.NroSerie=CONTIENE.NroSerieDisco
     join ESECUZIONE on CONTIENE.CodiceReg=ESECUZIONE.CodiceReg
where ESECUZIONE.anno is NULL
```

3.
```
select NroProg, TitoloCanz, NomeCantante
from (CONTIENE left join CANTANTE on
     CONTIENE.NroSerieDisco=CANTANTE.CodiceReg)
     join ESECUZIONE on CONTIENE.codiceReg= ESECUZIONE.CodiceReg
where NroSerieDisco=78574
order by NroProg
```

4.
```
select Nome
from AUTORE
where Nome not in ( select NomeCantante
                    from CANTANTE )
union
select NomeCantante
from CANTANTE
where NomeCantante not in ( select Nome
                            from AUTORE )
```

5.
```
select NroSerie
from DISCO
where NroSerie not in
( select NroSerieDisco
  from CONTIENE join CANTANTE as S1 on
          CONTIENE.CodiceReg=S1.CodiceReg
       join CANTANTE as S2 on CONTIENE.CodiceReg=S2.CodiceReg
  where  S1.NomeCantante<>S2.NomeCantante
) and NroSerie in
( select NroSerieDisco
  from CONTIENE join ESECUZIONE on CodiceReg= CodiceReg
       join DISCO on DISCO.NroSerie=CONTIENE.NroSerieDisco
  where ESECUZIONE.Anno<DISCO.Anno
  group by NroSerieDisco
  having count(*) >=3
)
```

6.
```
select distinct NomeCantante
from CANTANTE
where NomeC not in
   ( select S1.NomeCantante
     from CANTANTE as S1
     where CodiceReg not in
        ( select CodiceReg
          from CANTANTE S2
          where S2.NomeCantante <> S1.NomeCantante ) )
```

7.
```
select NomeCantante
from CANTANTE
where NomeCantante not in
( select S1.NomeCantante
  from CANTANTE as S1 join ESECUZIONE on CodiceReg=S1.CodiceReg
       join CANTANTE as S2 on CodiceReg=S2.CodiceReg )
  where S1.NomeCantante<> S2.NomeCantante
)
```

Esercizio 8

1.
```
select *
from Studenti
where matricola not in ( select distinct studente
                         from Esami );
```

2.
```
select *
from Studenti join Esami on matricola=studente
where voto > any ( select voto
                   from Esami join Studenti on studente = matricola
                   where cognome = 'pitagorico'
                         and nome = 'archimede');
```

3.
```
select distinct s.nome, s.cognome
from Studenti s,
     Esami e1 join Esami e2 on (e1.studente = e2.studente)
where e1.codiceCorso > e2.codiceCorso and e1.voto >= 18 and
      e2.voto >= 18 and s.matricola = e1.studente;
```

4.
```
select s.nome, s.cognome, count(*), avg(voto)
from Studenti s, Esami e
where s.matricola = e.studente
group by s.nome, s.cognome
```

Esercizio 9

1.
```
update Impiegato set Stipendio= Stipendio+200
where Stipendio < 30000
```
2.
```
update Impiegato set Stipedio= Stipendio*0.95
where Stipendio > 30000
```

Esercizio 10

1.

CODICE
104

2.

CODICE
103
104

3.

CODICE
103

Esercizio 11

```
create table Prodotti (
     codice numeric not null primary key,
     nome char(20) not null,
     categoria char(20) not null
);
```

```
create table Vendite (
     codiceProd numeric not null references Prodotti(codice),
     data date not null,
     incasso numeric not null,
     primary key (codiceProd, data)
);

delete * from Prodotti;
insert into Prodotti values(101,A,Bevanda);
insert into Prodotti values(102,B,Bevanda);
insert into Prodotti values(103,C,Pasta);
insert into Prodotti values(104,D,Biscotti);

delete * from Vendite;
insert into Vendite values(101,2008-11-24,2000);
insert into Vendite values(101,2008-11-25,1000);
insert into Vendite values(102,2008-11-23,2500);
insert into Vendite values(102,2008-11-24,4000);
insert into Vendite values(103,2008-11-25,1320);
```

Esercizio 12

1. ```
   select A.targa, A.tipologia, A.lunghezza, T.kmPercorsi * TA.costoAlKm
   from Transito T join Automobile A on (T.auto = A.targa)
        join Tariffa TA on (TA.tipoAuto = A.tipologia);
   ```

2. ```
   select A.targa, A.tipologia, A.lunghezza
   from Automobile A join Transito T on (T.auto = A.targa)
        join transito T2 on (T1.auto = T2.auto)
   where T1.codice > T2.codice;
   ```

3. ```
   select A.targa
   from Automobile A join Transito T on (A.targa = T.auto)
        left join Transito T2 on (T1.auto = T2.auto)
   where T1.codice > T2.codice
   except
   select A.targa
   from Automobile A join Transito T on (A.targa = T.auto)
        join Transito T2 on (T1.auto = T2.auto)
   where T1.codice > T2.codice and (T1.km <> T2.Km);
   ```

4. ```
   select A.*, T.*, T.KmPercorsi * TA.costoKM
   from Transito T join Automobile A on (T.auto = A.targa)
        join Tariffa TA on (TA.tipoAuto = A.Tipologia)
   where (T.KmPercorsi / (T.uscita-T.ingresso)) > 140
   ```

Esercizio 13

1. ```
   select distinct Citta
   from Aeroporti join Voli on Aeroporti.Codice = Voli.Destinazione
        join Aerei on Aerei.Codice = Voli.Aereo
   where Aerei.NumeroPosti >= 200;
   ```

2. ```
   select Citta, count(*)
   from Aereoporti join Voli on Aeroporti.Codice = Voli.Destinazione
   group by Citta
   ```

Esercizi SQL di Preparazione all'Esame

difficoltà delle query:
* = facile
** = media
* ** = difficile

1. Si consideri la base di dati definita dal seguente schema relazionale:

   Programmi(<u>Codice</u>, NomeProgramma, Rete, Tipologia)
   Artisti(<u>CProgramma</u>, <u>NomeArt</u>)
   Sondaggi(<u>CodProg</u>, Auditel)

   Gli attributi CProgramma e CodProg sono chiavi esterne di Programmi.

   Esprimere le seguenti interrogazioni:

   a. (*) Restituire l'indice Auditel del programma con codice P17
   b. (**) Restituire l'indice Auditel dei programmi di Fiorello
   c. (***) Restuire i nomi dei programmi con almeno due artisti
   d. (***) Restituire i nomi dei programmi con un solo artista

2. Si consideri la base di dati definita dal seguente schema relazionale:

   Partiti(<u>Codice</u>,NomePartito,Coalizione)
   Candidati(<u>CPartito</u>,<u>Citta</u>, NomeCand)
   Sondaggi(<u>CodPart</u>,PercFavorevoli)

   Gli attributi CPartito e CodPart sono chiavi esterne di Partiti.

   Esprimere le seguenti interrogazioni:

   a. (*) Restituire la percentuale di favorevoli del partito con codice A13.
   b. (**) Restituire la percentuale di favorevoli del partito del candidato con nome 'Rossi'.
   c. (***) Restuire i nomi dei partiti con esattamente due candidati
   d. (** Restituire i nomi dei dei partiti con percentuale di favorevoli del 30%.
   e. (*) Restituire i codici dei partiti con percentuale di favorevoli del 30%.

3. Si consideri la base di dati definita dal seguente schema relazionale:

   Impiegato(CodFisc,Nome,Cognome,Recapito,Dip)
   Lavora_Su(Imp,Prog)
   Progetto(CodP,NomeP,Budget)
   Dipartimento(CodD,NomeD,Indirizzo,Citta)

   L'attributo Dip è chiave esterna di Dipartimento. L'attributo Imp è chiave esterna di Impiegato. L'attributo Prog è chiave esterna di Progetto.

   Esprimere le seguenti interrogazioni:

   a. (*) Trovare il nome delle persone con cognome 'Rossi'.
   b. (**) Trovare i nomi dei dipartimenti in cui lavorano gli impiegati che stanno sul progetto 4.
   c. (***) Per ogni progetto in cui lavorano almeno due impiegati, si recuperi il numero del progetto e il nome del progetto.
   d. (*) Trovare il nome e cognome delle persone con recapito '011 6706733'.
   e. (***) Per ogni progetto in cui lavorano al più due impiegati, si recuperi il numero del progetto e il nome del progetto.

Esercizi SQL di Preparazione all'Esame

difficoltà delle query:
* = facile
** = media
* ** = difficile

1. Si consideri la base di dati definita dal seguente schema relazionale:

   Programmi(Codice, NomeProgramma, Rete, Tipologia)
   Artisti(CProgramma, NomeArt)
   Sondaggi(CodProg, Auditel)

   Gli attributi CProgramma e CodProg sono chiavi esterne di Programmi.

   Esprimere le seguenti interrogazioni:

   a. (*) Restituire l'indice Auditel del programma con codice P17

   SELECT Auditel FROM Sondaggi WHERE CodProg = 'P17'

   b. (**) Restituire l'indice Auditel dei programmi di Fiorello

   ```
   SELECT Auditel
   FROM Sondaggi S, Artisti A
   WHERE    S.CodProg = A.CProgramma AND A.NomeArt = 'Fiorello'
   ```

   c. (***) Restuire i nomi dei programmi con almeno due artisti

   ```
   SELECT DISTINCT NomeProgramma
   FROM Programmi P, Artisti A1, Artisti A2
   WHERE    P.Codice = A1.CProgramma AND
            P.Codice = A2.CProgramma AND
            A1.NomeArt > A2.NomeArt
   ```

   d. (***) Restituire i nomi dei programmi con un solo artista

   È sufficiente togliere (*except*) i programmi con almeno 2 artisti (query 1.c) dai programmi con almeno un artista:

   ```
   (SELECT DISTINCT NomeProgramma
    FROM Programmi, Artisti
   WHERE Codice = CProgramma)
       EXCEPT
   (SELECT DISTINCT NomeProgramma
    FROM Programmi P, Artisti A1, Artisti A2
    WHERE   P.Codice = A1.CProgramma AND
            P.Codice = A2.CProgramma AND
            A1.NomeArt > A2.NomeArt)
   ```

2. Si consideri la base di dati definita dal seguente schema relazionale:

   Partiti(Codice,NomePartito,Coalizione)
   Candidati(CPartito,Citta, NomeCand)
   Sondaggi(CodPart,PercFavorevoli)

   Gli attributi CPartito e CodPart sono chiavi esterne di Partiti.

   Esprimere le seguenti interrogazioni:

   a. (*) Restituire la percentuale di favorevoli del partito con codice A13.

   SELECT PercFavorevoli FROM Sondaggi WHERE CodPart = 'A13'

   b. (**) Restituire la percentuale di favorevoli del partito del candidato con nome 'Rossi'.

   SELECT PercFavorevoli FROM Sondaggi S, Candidati C
   WHERE S.CodPart = C.CPartito AND C.NomeCand = 'Rossi'

   c. (***) Restuire i nomi dei partiti con esattamente due candidati

   È sufficiente togliere (*except*) i partiti con almeno 3 candidati dai programmi con almeno due candidati:

   (SELECT DISTINCT NomePartito
   FROM Partiti P, Candidati C1, Candidati C2
   WHERE C1.CPartito = P.Codice AND C2.CPartito = P.Codice AND
   C1.Citta > C2.Citta)
   EXCEPT
   (SELECT DISTINCT NomePartito
   FROM Partiti P, Candidati C1, Candidati C2, Candidati C3
   WHERE C1.CPartito = P.Codice AND C2.CPartito = P.Codice AND
   C3.CPartito = P.Codice AND
   C1.Citta > C2.Citta AND C2.Citta > C3.Citta)

   d. (**) Restituire i nomi dei dei partiti con percentuale di favorevoli del 30%.

   SELECT NomePartito FROM Partiti P, Sondaggi S
   WHERE P.Codice = S.CodPart AND S.PercFavorevoli = 30

   e. (*) Restituire i codici dei partiti con percentuale di favorevoli del 30%.

   SELECT CodPart FROM Sondaggi WHERE PercFavorevoli = 30

3. Si consideri la base di dati definita dal seguente schema relazionale:

   Impiegato(CodFisc,Nome,Cognome,Recapito,Dip)
   Lavora_Su(Imp,Prog)
   Progetto(CodP,NomeP,Budget)
   Dipartimento(CodD,NomeD,Indirizzo,Citta)

   L'attributo Dip è chiave esterna di Dipartimento. L'attributo Imp è chiave esterna di Impiegato. L'attributo Prog è chiave esterna di Progetto.

   Esprimere le seguenti interrogazioni:

   a. (*) Trovare il nome delle persone con cognome 'Rossi'.

   SELECT Nome FROM Impiegato WHERE Cognome = 'Rossi'

   b. (**) Trovare i nomi dei dipartimenti in cui lavorano gli impiegati che stanno sul progetto 4.

   SELECT D.NomeD
   FROM Dipartimento D, Impiegato I, Lavora_Su L
   WHERE I.Dip = D.CodD AND I.CodFisc = L.Imp AND L.Prog = 4

   c. (***) Per ogni progetto in cui lavorano almeno due impiegati, si recuperi il numero del progetto e il nome del progetto.

   SELECT DISTINCT P.CodP, P.NomeP
   FROM Progetto P, Lavora_Su L1, Lavora_Su L2
   WHERE L1.Prog = P.CodP AND L2.Prog = P.CodP AND L1.Imp > L2.Imp

   d. (*) Trovare il nome e cognome delle persone con recapito '011 6706733'

   SELECT Nome, Cognome FROM Impiegato WHERE Recapito = '011 6706733'

   e. (***) Per ogni progetto in cui lavorano al più due impiegati, si recuperi il numero del progetto e il nome del progetto.

   È sufficiente togliere (*except*) i progetti con almeno 3 impiegati dall'elenco di tutti I progetti:

   (SELECT DISTINCT P.CodP, P.NomeP FROM Progetto P)
   EXCEPT
   (SELECT DISTINCT P.CodP, P.NomeP
   FROM Progetto P, Lavora_Su L1, Lavora_Su L2, Lavora_Su L3
   WHERE L1.Prog = P.CodP AND L2.Prog = P.CodP AND L3.Prog = P.CodP AND
   L1.Imp > L2.Imp AND L2.Imp > L3.Imp)

Corso di Basi di Dati 11. Esercitazioni in SQL: Altri esercizi

Guido Pezzini

A.A. 2016–2017

Esercizi

Esempio: consideriamo le seguenti tabelle

ARTICOLI(Id, Nome, Colore)
FORNITORI(Id, Nome, Indirizzo)
CATALOGO(Articolo, Fornitore, Prezzo)

Catalogo.Articolo si riferisce ad Articoli.Id, Catalogo.Fornitore si riferisce a Fornitori.Id, ma può essere `NULL` se un articolo non ha un fornitore.

Esercizio: trovare i nomi dei pezzi (eliminando le ripetizioni) che hanno un fornitore.

Esercizi

Soluzione:

```
select distinct A.Nome
from Articoli A, Catalogo C
where A.Id = C.Articolo
and C.Fornitore is not NULL;
```

Esercizi

ARTICOLI(Id, Nome, Colore)
FORNITORI(Id, Nome, Indirizzo)
CATALOGO(Articolo, Fornitore, Prezzo)

Esercizio: trovare i nomi dei fornitori che hanno tutti gli articoli, sempre eliminando le ripetizioni.

Esercizi

Soluzione:

```
select distinct F.Nome
from Fornitori F
where not exists (select *
                  from Articoli A
                  where A.Id not in (select C.Articolo
                                     from Catalogo C
                                     where
                                     C.Fornitore = F.Id))
```

Esercizi

Esempio:

GENITORI(Figlio, Genitore)

Esercizio: estrarre le coppie di fratelli (cioè coppie di persone con almeno un genitore in comune), eliminando le ripetizioni. Il risultato dovrà avere due colonne chiamate "Nome1" e "Nome2".

Esercizi

Soluzione:

```
select distinct G1.Figlio AS Nome1,
G2.Figlio AS Nome2
from Genitori G1, Genitori G2
where G1.Genitore = G2.Genitore
and G1.Figlio <> G2.Figlio
```

Esercizi

Esempio:

LIBRI(Id, Titolo, IdAutore, Anno)
AUTORI(Id, Nome)

Esercizio: estrarre i nomi degli autori che hanno scritto più di dieci libri.

Esercizi

Soluzione 1:

```
select A.Nome
from Autori A
where 10 < (select count(*)
            from Libri L
            where A.Id = L.IdAutore)
```

Esercizi

Soluzione 1:

```
select A.Nome
from Autori A
where 10 < (select count(*)
            from Libri L
            where A.Id = L.IdAutore)
```

Soluzione 2:

```
select distinct A.Nome
from Autori A, Libri L
where A.Id = L.IdAutore
group by A.Id
having count(*) > 10
```

Altri esercizi

PRODOTTI(Id, Nome, Marca)
VENDITE(IdProdotto, Anno, Quantità)

In queste tabelle, VENDITE.IdProdotto si riferisce a PRODOTTI.Id.

Per ogni (nome di) prodotto che abbia almeno una vendita negli anni compresi fra il 2012 e il 2015 (estremi inclusi), calcolare la quantità totale venduta in tali anni.

Dare il risultato in una tabella con due colonne: "NomeProdotto" e "QuantitàTot".

Altri esercizi

Soluzione:

```
select P.Nome as NomeProdotto,
       sum(V.Quantità) as QuantitàTot
from Vendite V, Prodotti P
where V.IdProdotto = P.Id
      and V.Anno >= 2012
      and V.Anno <= 2015
group by V.Prodotto
```

Altri esercizi

Esercizio: Dare le definizioni SQL delle tabelle:

FARMACIE(Id, Indirizzo),
FARMACI(Id, Nome),
CATALOGO(IdFarmacia, IdFarmaco, Prezzo).

1. Le colonne tipo "Id..." e "Prezzo" devono essere numeri interi, le altre devono essere stringhe di lunghezza massima 30 caratteri.
2. "Prezzo" deve avere valore di default 0.
3. Specificare in FARMACIE e in FARMACI la colonna `Id` come chiave primaria,
4. Inserire un vincolo di `foreign key` su CATALOGHI.`IdFarmacia`, che si deve riferire a FARMACIE.`Id`, e un vincolo di `foreign key` su CATALOGHI.`IdFarmaco`, che si deve riferire a FARMACI.`Id`,
5. Specificare su entrambi i vincoli `no action` (cioè impedire l'operazione) in caso di modifiche o cancellazioni di dati.

Altri esercizi

Soluzione (prime due tabelle):

```
create table Farmacie (
Id int primary key,
Indirizzo varchar(30) );

create table Farmaci (
Id int primary key,
Nome varchar(30) );
```

Altri esercizi

Soluzione (terza tabella):

```
create table Catalogo (
IdFarmacia int,
IdFarmaco int,
Prezzo int default 0,
foreign key (IdFarmacia) references Farmacie(Id)
on update no action
on delete no action,
foreign key (IdFarmaco) references Farmaci(Id)
on update no action
on delete no action );
```

ESERCIZI

INTERROGAZIONI SQL

Interrogazioni base di dati “Azienda”

- Mostrare nome e cognome di tutti gli impiegati del dipartimento 5 che lavorano più di 10 ore alla settimana sul progetto ‘ProdottoX’
- Mostrare il nome e cognome di tutti gli impiegati che hanno una persona a carico con il loro stesso nome di battesimo
- Mostrare il nome e cognome di tutti gli impiegati che hanno come diretto supervisore ‘Franklin Wong’
- Mostrare il nome e cognome di tutti gli impiegati che guadagnano oltre $10.000 in più rispetto allo stipendio più basso

Base di Dati "Catalogo"

- Considerate lo schema seguente:

  Fornitori (*fid:* `integer`, *fnome:* `string`, *indirizzo:* `string`)
  Pezzi (*pid:* `integer`, *pnome:* `string`, *colore:* `string`)
  Catalogo (*fid:* `integer`, *pid:* `integer`, *costo:* `real`)

  I campi chiave sono sottolineati. La relazione Catalogo elenca i prezzi praticati da ciascun fornitore.

Interrogazione 1

- Trovare i *fid* dei fornitori che forniscono pezzi rossi o pezzi verdi

```
SELECT C.fid
FROM   Parti P, Catalogo C
WHERE  (P.colore='rosso' OR
        P.colore='verde') AND
        C.pid=P.pid
```

Formulazione alternativa I1

```
SELECT  C.fid
FROM    Parti P, Catalogo C
WHERE   P.colore='rosso' AND
        C.pid=P.pid
UNION
SELECT  C.fid
FROM    Parti P, Catalogo C
WHERE   P.colore='verde' AND
        C.pid=P.pid
```


Interrogazione 2

- Trovare i *fid* dei fornitori che forniscono pezzi rossi e pezzi verdi

```
SELECT  C.fid
FROM    Catalogo C, Pezzi P
WHERE   P.colore='rosso'AND
        C.pid=P.pid
INTERSECT
SELECT  C.fid
FROM    Catalogo C, Pezzi P
WHERE   P.colore='verde'AND
        C.pid=P.pid
```

Base di dati "Compagnia Aerea"

- Considerate lo schema seguente:

Volo (*vno:* `integer`, *da:* `string`, *a:* `string`, *distanza:* `integer`, *partenza:* `time`, *arrivo:* `time`)
Aereo (*aid:* `integer`, *anome:* `string`, *autonomia:* `integer`)
Certificato (*pid:* `integer`, *aid:* `integer`)
Personale (*pid:* `integer`, *pnome:* `string`, *salario:* `integer`)

- La relazione Personale descrive piloti così come gli altri tipi di impiegati
- Ogni pilota è certificato per certi aerei e solo i piloti sono certificati per volare.

Interrogazione 1

- Trovare i *pid* dei piloti certificati per qualche aereo Boeing.

```
SELECT  C.pid
FROM    Aereo A, Certificato C
WHERE   A.aid=C.aid AND
        A.nome LIKE 'Boeing%'
```

Interrogazione 2

- Trovare gli *aid* di tutti gli aerei che possono essere usati per voli diretti da Los Angeles a Chicago.

```
SELECT    A.aid
FROM      Aereo A, Volo V
WHERE     V.da='Los Angeles' AND V.a='Chicago'
          AND A.autonomia > V.distanza
```


SQL Avanzato

Formulazione alternativa I2

```
SELECT C.fid
FROM   Catalogo C, Pezzi P
WHERE  P.colore='rosso' AND C.pid=P.pid
       AND C.fid IN
           (SELECT C1.fid
            FROM Catalogo C1, Pezzi P1
            WHERE P1.colore='verde' AND
                  C1.pid=P1.pid)
```


Formulazione alternativa I2

```
SELECT  C.fid
FROM    Catalogo C, Pezzi P
WHERE   P.colore='rosso' AND C.pid=P.pid
       AND EXISTS(SELECT P2.pid
                  FROM Catalogo C2,
Pezzi P2
                  WHERE P2.colore='verde'
                       AND C2.fid=C.fid
                       AND P2.pid=C2.pid)
```

Interrogazione 3

- Trovare i *pid* dei pezzi forniti da almeno due fornitori diversi.

```
SELECT    C.pid
FROM      Catalogo C
WHERE     EXISTS
          (SELECT       C1.fid
           FROM         Catalogo C1
           WHERE        C1.pid=C.pid
                    AND C1.fid<>C.fid)
```


Formulazione alternativa I3

```
SELECT    C.pid
FROM      Catalogo C
GROUP BY  C.pid
HAVING    COUNT(*) > 1
```

Formulazione alternativa I2

- Trovare gli *aid* di tutti gli aerei che possono essere usati per voli diretti da Los Angeles a Chicago.

```
SELECT    A.aid
FROM      Aereo A
WHERE     A.autonomia >
          (SELECT   V.distanza
           FROM     Volo V
           WHERE    V.da='Los Angeles' AND
                    V.a='Chicago')
```


Interrogazione 3

- Trovare i pid dei piloti certificati **solo** per aerei Boeing

```
SELECT DISTINCT eid
FROM certified c
WHERE NOT EXISTS
              (SELECT *
               FROM aircraft a, certified c1
               WHERE a.aid = c1.aid AND
               c1.eid = c.eid AND
               a.aname NOT LIKE 'Boeing%')
```

Interrogazione 4

- Trovare i pid dei piloti certificati per **tutti** gli aerei Boeing

```
SELECT DISTINCT eid
FROM employees e
WHERE NOT EXISTS
           (SELECT *
            FROM aircraft a
            WHERE a.aname LIKE 'Boeing%' AND
                  NOT EXISTS
                     (SELECT *
                      FROM certified c
                      WHERE e.eid = c.eid
                            AND c.aid = a.aid))
```


Interrogazione 5

- Trovare i nomi dei piloti che possono operare su aerei con un autonomia superiore a 3000 miglia, ma che non sono certificati per alcun aereo Boeing

```
SELECT  DISTINCT P.pnome
FROM    Aereo A, Certificato C, Personale P
WHERE   P.pid=C.pid AND C.aid=A.aid
        AND A.autonomia>3000
        AND P.pid NOT IN
               (SELECT C1.pid
                FROM   Certificato C1, Aereo A1
                WHERE  C1.aid=A1.aid
                    AND A1.anome LIKE 'Boeing%')
```

Interrogazione 6

- Trovare i *pid* del personale con il salario più alto

```
SELECT   P.pid
FROM     Personale P
WHERE    P.stipendio=(SELECT
                      MAX(P1.stipendio)
                      FROM   Personale P1)
```


Interrogazione 7

- Trovare i *pid* del personale con il 2° stipendio più alto

```
SELECT   P.pid
FROM     Personale P
WHERE    P.stipendio =
         (SELECT MAX(P1.stipendio)
          FROM   Personale P1
          WHERE  P1.stipendio<>
             (SELECT MAX(P2.stipendio)
              FROM   Personale P2))
```

Interrogazione 8

- Visualizzare il nome e lo stipendio di ogni dipendente non-pilota il cui stipendio è superiore allo stipendio medio dei piloti.

```
SELECT  P.dnome, P.stipendio
FROM    Personale P
WHERE   P.pid NOT IN
              (SELECT  C.pid
               FROM    Certificato C)
        AND P.stipendio >
              (SELECT AVG(P1.stipendio)
               FROM   Personale P1,
                      Certificato C1
               WHERE  P1.pid = C1.pid)
```


Interrogazione 9

- Trovare i nomi degli aerei tali che tutti i piloti abilitati al volo su quegli aerei guadagnano almeno € 80000,00

```
SELECT  DISTINCT A.anome
FROM    Aereo A
WHERE   A.aid NOT IN
        (SELECT C.aid
         FROM Certificato C, Personale
P
         WHERE C.pid = P.pid  AND
               P.stipendio < 80000)
```

Interrogazione 10

- Trovare i *pid* del personale certificato esattamente per tre aerei

```
SELECT   C.pid
FROM     Certificato C
GROUP BY C.pid
HAVING   COUNT(*) = 3
```


Interrogazione 11

- Per ciascun pilota abilitato al volo su più di tre aerei, trovare il *pid* e la massima distanza percorribile dagli aerei su cui è abilitato.

```
SELECT   C.pid, MAX(A.autonomia)
FROM     Certificato C, Aereo A
WHERE    C.aid = A.aid
GROUP BY C.pid
HAVING   COUNT(*)>3
```

Interrogazione 12

- Per tutti gli aerei che percorrono una distanza superiore alle 1000 miglia, trovare il nome dell'aereo e lo stipendio medio dei piloti abilitati al volo su quell'aereo.

```
SELECT   A.anome, AVG(P.stipendio)
FROM     Certificato C, Aereo A,
         Personale P
WHERE    C.aid = A.aid AND C.pid = P.pid
         AND A.autonomia>1000
GROUP BY A.aid, A.anome
```


Interrogazione 13

- Visualizzare i nomi del personale abilitato al volo solo su aerei che possono coprire distanze superiori alle 1000 miglia.

```
SELECT P.pnome
FROM   Certificato C, Aereo A,
                      Personale P
WHERE  C.aid = A.aid AND C.pid = P.pid
GROUP BY p.pid, p.pnome
HAVING  EVERY(A.autonomia>1000)
```

Interrogazione 14

- Identificare le rotte che possono essere percorse da tutti i piloti con uno stipendio superiore a €100000,00

```
SELECT  DISTINCT V.da, V.a
FROM    Volo V
WHERE   NOT EXISTS
        (SELECT  *
         FROM    Personale P
         WHERE   P.stipendio > 100000
                 AND NOT EXISTS
                    (SELECT *
                     FROM   Aereo A, Certificato C
                     WHERE A.autonomia > V.distanza
                           AND P.pid = C.pid
                           AND A.aid = C.aid))
```


Nota su I9 e I14

- Nella interrogazione I9 siamo interessati agli aerei i cui piloti abilitati guadagnano almeno 80.000,00 Euro
  - Presa una coppia di aerei che risponde all'interrogazione, gli insiemi dei piloti abilitati per ciascun aereo possono essere diversi, ma in ciascun insieme tutti i piloti guadagnano almeno 80.000,00 Euro
- Nella interrogazione I14 siamo interessati alle rotte che possono essere percorse da tutti i piloti che guadagnano 100.000,00 Euro

Base di dati "Università"

- **Studente** (*snum:* `integer`, *snome:* `string`, *specializzazione:* `string`, *anno:* `string`, *età:* `integer`)
- **Corso** (*cnome:* `string`, *orario:* `time`, *aula:* `string`, *pid:* `integer`)
- **Iscritto** (*snum:* `integer`, *cnome:* `integer`)
- **Professore** (*pid:* `integer`, *pnome:* `string`, *dipartId:* `integer`)

Interrogazione 15

- Trovare i nomi dei professori per i quali il totale degli iscritti in un loro corso è minore di cinque.

```
SELECT   DISTINCT P.pnome
FROM     Professore P, Corso C,
         Iscritto I
WHERE    C.cnome=I.cnome
         AND C.pid=P.pid
GROUP BY (P.pid,P.pnome)
HAVING COUNT(*) < 5
```

Interrogazione 16

- Trovare i nomi degli studenti non iscritti ad alcun corso

```
SELECT   DISTINCT S.snome
FROM     Studente S
WHERE    S.snum NOT IN
         (SELECT    I.snum
          FROM      Iscritto I)
```


Interrogazione 17

- Trovare i nomi dei corsi che hanno almeno 5 iscritti

```
SELECT     C.cnome
FROM       Corso C, Iscritto I
WHERE      C.cnome = I.cnome
GROUP BY   C.cnome
HAVING     COUNT(*)>=5
```

Interrogazione 19

- Trovare i nomi degli studenti iscritti al maggior numero di corsi

```
SELECT   DISTINCT S.snome
FROM       Studente S, Iscritto I
WHERE      S.snum = I.snum
GROUP BY   (I.snum,S.snome)
HAVING   COUNT(*) >= ALL
         (SELECT COUNT(*)
         FROM Iscritto I1
         GROUP BY I1.snum)
```


Interrogazione 20

- Per ciascun valore di età che appare in Studente, trovare l'anno di corso nel quale è più frequente trovare studenti di quell'età.
  - Per esempio, se ci sono più studenti di 20 anni al secondo anno piuttosto che negli altri, si dovrebbe visualizzare (20,2).

Formulazione I20

```
SELECT S.età, S.anno
FROM  Studente s1
GROUP BY età, anno
HAVING COUNT(*) >= ALL
                  (SELECT  COUNT(*)
                   FROM    Studente s2
                   WHERE   s2.età = s1.età
                   GROUP BY s2.anno)
ORDER BY Stat.età
```