

- Schema relazionale:
Sailors(sid,sname,rating,age),
Boats(bid,bname,color),
Reserves(sid^{Sailors},bid^{Boats},day)

- La relazione Sailors contiene informazioni relative ai marinai (i.e., codice, nome, livello ed eta): sid è chiave primaria
- La relazione Boats contiene informazioni relative alle barche (i.e., codice, nome e colore): bid è chiave primaria
- La relazione Reserves contiene informazioni relative agli imbarchi (i.e., codice del marinaio, codice della barca in cui si è imbarcato e giorno di imbarco): sid e bid sono chiave primarie e chiavi esterne rispettivamente da Sailors e Boats

- Trovare i nomi dei marinai che hanno un imbarco su almeno una nave
- ```
SELECT DISTINCT sname
 FROM Sailors, Reserves
 WHERE Sailors.sid=Reserves.sid;
```

- Trovare i nomi dei marinai che sono imbarcati sulla nave 103

```
SELECT sname
FROM Sailors, Reserves
WHERE Sailors.sid=Reserves.sid AND
 Reserves.bid='103';
```

- Trovare i nomi dei marinai che sono imbarcati su una nave rossa

- ```
SELECT sname
FROM   Sailors, Reserves, Boats
WHERE  Sailors.sid=Reserves.sid AND Reserves.bid=Boats.bid
AND    Boats.color='red';
```

•Trovare i nomi dei marinai che sono imbarcati su una nave rossa o verde

```
• SELECT DISTINCT sname
  FROM Sailors, Reserves, Boats
 WHERE Sailors.sid=Reserves.sid AND
        Reserves.bid=Boats.bid
        AND (Boats.color='red' OR Boats.color='green');
```

- CORSI_ACQUA(NomeF,Lunghezza, Profondita_media)
- COMUNI(NomeC, Regione, Popolazione)
- BAGNA(NomeF^{CORSI_ACQUA},NomeC^{COMUNI})
- La relazione CORSI_ACQUA contiene informazioni relative ai fiumi (i.e., nome fiume, lunghezza, profondità media): nome fiume è chiave primaria della relazione
- La relazione COMUNI contiene informazioni relative ai comuni (i.e., Nome comune, regione e popolazione): nome comune è chiave primaria della relazione
- La relazione BAGNA contiene informazioni sui fiumi che attraversano le città: nomeF e nomeC sono chiavi primarie e chiavi esterne rispettivamente da CORSI_ACQUA e COMUNI

- CORSI_ACQUA(NomeF,Lunghezza, Profondita_media)
- COMUNI(NomeC, Regione, Popolazione)
- BAGNA(NomeF^{CORSI_ACQUA},NomeC^{COMUNI})

Determinare i nomi dei comuni lombardi, con popolazione maggiore di 500000 abitanti, percorsi da almeno un corso d'acqua.

```
SELECT DISTINCT NomeC
FROM BAGNA, COMUNI
WHERE COMUNI.NomeC=BAGNA.NomeC
      AND Regione='Lombardia'
      AND Popolazione> 500000;
```

- CORSI_ACQUA(NomeF,Lunghezza, Profondita_media)
- COMUNI(NomeC, Regione, Popolazione)
- BAGNA(NomeF^{CORSI_ACQUA},NomeC^{COMUNI})

Determinare i nomi dei fiumi con lunghezza compresa tra 20 e 30 Km che attraversano almeno un comune in Liguria

```
SELECT CORSI_ACQUA.NomeF
FROM CORSI_ACQUA, BAGNA, COMUNI
WHERE CORSI_ACQUA.NomeF=BAGNA.NomeF
      AND COMUNI.NomeC=BAGNA.NomeC
      AND Regione='Liguria'
      AND Lunghezza BETWEEN 20 AND 30;
```

- CORSI_ACQUA(NomeF,Lunghezza, Profondita_media)
 - COMUNI(NomeC, Regione, Popolazione)
 - BAGNA(NomeF^{CORSI_ACQUA},NomeC^{COMUNI})
- Determinare i nomi dei comuni che non sono percorsi da alcun corso d'acqua con lunghezza inferiore ai 3 Km

```
SELECT DISTINCT COMUNI.NomeC
FROM COMUNI, BAGNA
WHERE COMUNI.NomeC=BAGNA.NomeC
AND NomeF NOT IN (SELECT NomeF FROM CORSI_ACQUA WHERE Lunghezza<3);
```

oppure

```
SELECT DISTINCT COMUNI.NomeC
FROM COMUNI, BAGNA
WHERE COMUNI.NomeC=BAGNA.NomeC
AND NomeF IN (SELECT NomeF FROM CORSI_ACQUA WHERE Lunghezza>=3);
```

- Quadro(Nome,Anno,periodo,id-museo,autore)
- Pittore(CF,Nome,Cognome,Nazionalità,Anno-di-nascita)
- Museo(Id,Nome,Annodicostruzione,Città,Nazione,Cognomedirettore)

- 1) Determinare i nomi di tutti i quadri dipinti da Salvator Dalì
- 2) Determinare i nomi dei musei che espongono quadri di De Chirico

- Pittore(CF,Nome,Cognome,Nazionalità,Anno-di-nascita)
 - Museo(Id,Nome,Annodicostruzione,Città,Nazione,Cognomedirettore)
 - Quadro(Nome,Anno,periodo,id-museo^{Museo},autore^{Pittore})
- La relazione Pittore contiene informazioni relative ai pittori (i.e., codice fiscale, nome, cognome, nazionalità e anno di nascita): chiave primaria è codice fiscale
 - La relazione Museo contiene informazioni relative ai musei (i.e., codice, nome, anno di costruzione, città, nazione e cognome del direttore): Id, ovvero il codice del museo, è chiave primaria della relazione
 - La relazione Quadro contiene informazioni relative ai quadri (i.e., nome del quadro, anno di realizzazione, periodo artistico): chiave primaria la coppia nome del quadro e autore, dove autore è chiave esterna dalla relazione Pittore. Inoltre id-museo è chiave esterna da Museo

- Pittore(CF,Nome,Cognome,Nazionalità,Anno-di-nascita)
- Museo(Id,Nome,Annodicostruzione,Città,Nazione,Cognomedirettore)
- Quadro(Nome,Anno,periodo,id-museo^{Museo},autore^{Pittore})

1) Determinare i nomi di tutti i quadri dipinti da Salvator Dalì

```
SELECT Quadro.Nome
FROM Quadro, Pittore
WHERE Quadro.autore= Pittore.CF AND
      Pittore.Nome='Salvator' AND Cognome='Dali';
```

- Pittore(CF,Nome,Cognome,Nazionalità,Anno-di-nascita)
- Museo(Id,Nome,Annodicostruzione,Città,Nazione,Cognomedirettore)
- Quadro(Nome,Anno,periodo,id-museo^{Museo},autore^{Pittore})

2) Determinare i nomi dei musei che espongono quadri di De Chirico

```
SELECT Museo.Nome
FROM Quadro, Pittore, Museo
WHERE Quadro.autore= Pittore.CF
      AND Quadro.id-museo=Museo.id
      AND Pittore.Cognome='De Chirico';
```

STUDENTI (Matricola, Nome, Cognome, Sesso, Diploma, Età)
 PROFESSORI (CodiceProf, Nome, Dipartimento, OrarioRicevimento)
 CORSI (Nome, Periodo, Aula, CodiceProf^{PROFESSORI})
 PIANI_DI_STUDIO (Matricola^{STUDENTI}, Corso^{CORSI})

- La relazione STUDENTI contiene informazioni relative agli studenti di una università (i.e., matricola, nome, cognome, sesso, diploma ed età). Chiave primaria è la matricola
- La relazione PROFESSORI contiene informazioni relative ai professori di una università (i.e., codice, nome, cognome, dipartimento, orario ricevimento). Chiave primaria è il codice del professore
- La relazione CORSI contiene informazioni relative ai corsi erogati dall'università (i.e., nome del corso, periodo, aula e codice del professore). Chiave primaria è il codice del corso, mentre CodiceProf è chiave esterna da PROFESSORI
- La relazione PIANI_DI_STUDIO contiene informazioni relative ai piani di studio degli studenti (i.e., matricola dello studente e codice del corso selezionato). Chiave primaria è la coppia Matricola (matricola studente) e Corso (i.e., codice del corso), dove Matricola è chiave esterna da STUDENTI e Corso è chiave esterna da CORSI

STUDENTI (Matricola, Nome, Cognome, Sesso, Diploma, Età)
PROFESSORI (CodiceProf, Nome, Dipartimento, OrarioRicevimento)
CORSI (Nome, Periodo, Aula, CodiceProf^{PROFESSORI})
PIANI_DI_STUDIO (Matricola^{STUDENTI}, Corso^{CORSI})

Trovare il numero di studenti che hanno nel piano di studi il corso di Basi di dati

```
SELECT COUNT(*)  
FROM PIANI_DI_STUDIO  
WHERE Corso = 'Basi di dati';
```

STUDENTI (Matricola, Nome, Cognome, Sesso, Diploma, Età)
PROFESSORI (CodiceProf, Nome, Dipartimento, OrarioRicevimento)
CORSI (Nome, Periodo, Aula, CodiceProf^{PROFESSORI})
PIANI_DI_STUDIO (Matricola^{STUDENTI}, Corso^{CORSI})

Trovare il nome dello studente più giovane

```
SELECT Nome  
FROM STUDENTI  
WHERE Età = (SELECT MIN(Età)  
             FROM STUDENTI);
```

- **Dati gli schemi**

Customers(cid, cname, city)

Agents(aid, aname, city)

Products(pid, pname, city, quantity, price)

Orders(ordnum, cid, aid, pid, qty, cost)

- **Calcolare, per ogni tipo di prodotto, il numero di pezzi venduti**

```
SELECT  pid, SUM(qty) AS total
```

```
FROM    orders
```

```
GROUP BY pid;
```

- **La seguente query non è corretta. Perché?**

```
SELECT  pid, cid, SUM(qty) AS total
```

```
FROM    orders
```

```
GROUP BY pid;
```

- La funzione SUM calcola un singolo valore per ogni gruppo. E' necessario che ogni attributo nella SELECT abbia valore univoco. In questo caso, l'attributo cid non verifica questa condizione.
- Bisogna escludere per forza cid?

- No. Basta includere cid nel GROUP BY:

```
SELECT  pid, cid, SUM(qty) AS total
FROM    orders
GROUP BY pid, cid;
```

Calcola per ogni prodotto il numero di pezzi venduti per cliente

- La query seguente è sintatticamente corretta?

```
SELECT  pid, SUM(qty)
FROM    orders
WHERE   sum(qty)>1000
GROUP BY pid;
```

- No. La funzione di aggregazione SUM non può apparire nella clausola WHERE. Inoltre, la condizione espressa nel WHERE elimina tuple prima che il GROUP BY faccia il partizionamento...

- ...ossia, quando il query processor controlla la condizione nel WHERE, non ha ancora calcolato i valori della funzione SUM perché non ha ancora eseguito il GROUP BY
- Sintassi corretta:
SELECT pid, SUM(qty)
FROM orders
GROUP BY pid
HAVING SUM(qty)>1000;

- Schema relazionale:
Sailors(sid,sname,rating,age),
Boats(bid,bname,color),
Reserves(sid^{Sailors},bid^{Boats},day)
- La relazione Sailors contiene informazioni relative ai marinai (i.e., codice, nome, livello ed età): sid è chiave primaria
- La relazione Boats contiene informazioni relative alle barche (i.e., codice, nome e colore): bid è chiave primaria
- La relazione Reserves contiene informazioni relative agli imbarchi (i.e., codice del marinaio, codice della barca in cui si è imbarcato e giorno di imbarco): sid e bid sono chiavi primarie e chiavi esterne rispettivamente da Sailors e Boats

- Trovare l'età media dei marinai con rating uguale a 10

```
SELECT AVG (age)
FROM Sailors
WHERE rating=10;
```

- Trovare l'età del marinaio più giovane per ogni valore del rating

```
SELECT rating, MIN (age)
FROM Sailors
GROUP BY rating;
```

- Trovare tra tutti i marinai con età maggiore di 18, l'età del marinaio più giovane per ogni livello di rating. Si è interessati solo a livelli di rating a cui sono associati almeno due marinai di età maggiore di 18

```
SELECT rating, MIN (age) as MinAge
FROM Sailors
WHERE age >= 18
GROUP BY rating
HAVING COUNT (*) > 1;
```

- Trovare i marinai con rating piu' alto

```
SELECT sid
FROM Sailors
WHERE rating >= ALL (SELECT rating
                    FROM Sailors);
```

- Trovare i nomi dei marinai il cui rating e' piu' alto di quello di un qualsiasi marinaio di eta' maggiore di 24 anni

```
SELECT sid
FROM Sailors
WHERE rating > ANY (SELECT rating
                    FROM Sailors
                    WHERE age > '24');
```

SCHEMA RELAZIONALE:

- ATTORI (CodAttore, Nome, AnnoNascita, Nazionalità);
- RECITA (CodAttore^{ATTORI}, CodFilm^{FILM})
- FILM (CodFilm, Titolo, AnnoProduzione, Nazionalità, Regista, Genere)
- PROIEZIONI (CodProiezione, CodFilm^{FILM}, CodSala^{SALE}, Incasso, DataProiezione)
- SALE (CodSala, Posti, Nome, Città)

dove gli attributi sottolineati sono chiave primaria, e in RECITA CodAttore è chiave esterna da ATTORI, mentre CodFilm è chiave esterna da FILM. In PROIEZIONI, CodFilm è chiave esterna da FILM e CodSala è chiave esterna da SALE.

Scrivere le interrogazioni SQL che restituiscono le seguenti informazioni:

1. Per ogni regista, il numero di film diretti dopo il 1990
2. Per ogni regista, l'incasso totale di tutte le proiezioni dei suoi film
3. Per ogni film di S.Spielberg, il titolo del film, il numero totale di proiezioni a Pisa e l'incasso totale
4. Per ogni regista e per ogni attore, il numero di film del regista con l'attore
5. Il regista ed il titolo dei film in cui recitano meno di 6 attori
6. Per ogni film prodotto dopo il 2000, il codice, il titolo e l'incasso totale di tutte le sue proiezioni
7. Il numero di attori dei film in cui appaiono solo attori nati prima del 1970
8. Per ogni film di fantascienza, il titolo e l'incasso totale di tutte le sue proiezioni
9. Per ogni film di fantascienza il titolo e l'incasso totale di tutte le sue proiezioni successive al 1/1/01
10. Per ogni film di fantascienza che non è mai stato proiettato prima del 1/1/01 il titolo e l'incasso totale di tutte le sue proiezioni
11. Per ogni sala di Pisa, che nel mese di gennaio 2005 ha incassato più di 20000 €, il nome della sala e l'incasso totale (sempre del mese di gennaio 2005)
12. I titoli dei film che non sono mai stati proiettati a Pisa

Soluzioni

- 1)

```
SELECT Regista, count(*)
FROM Film
WHERE AnnoProduzione > 1990
GROUP BY Regista;
```
- 2)

```
SELECT Film.Regista, sum(p.Incasso) as IncassoTotale
FROM Film, Proiezioni
WHERE Film.CodFilm = Proiezioni.CodFilm
GROUP BY Film.Regista;
```
- 3)

```
SELECT Film.Titolo, count(*) as NumeroProiezioni, sum(Proiezioni.Incasso) as
IncassoTotale
FROM Film, Proiezioni, Sale
WHERE Film.CodFilm = Proiezioni.CodFilm and Proiezioni.CodSala=Sale.CodSala and
Film.Regista = 'S.Spielberg' and Sale.Città = 'Pisa'
GROUP BY Film.CodFilm, Film.Titolo;
```

Soluzioni

- ```
4) SELECT Film.Regista, Attori.Nome, count(*) as NumeroFilm
 FROM Attori, Recita, Film
 WHERE Attori.CodAttore=Recita.CodAttore and Recita.CodFilm = Film.CodFilm
 GROUP BY Film.Regista, Attori.CodAttore, Attori.Nome;

5) SELECT Film.Regista, Film.Titolo
 FROM Film, Recita
 WHERE Film.CodFilm = Recita.CodFilm
 GROUP BY Film.CodFilm, Film.Titolo, Film.Regista
 HAVING count(*) < 6;

6) SELECT Film.CodFilm, Film.Titolo, sum(Film.Incasso) as IncassoTotale
 FROM Film, Proiezioni
 WHERE Film.AnnoProduzione > 2000 and Film.CodFilm = Proiezioni.CodFilm
 GROUP BY Film.CodFilm, Film.Titolo;
```

## Soluzioni

- ```
7) SELECT Film.Titolo, count(*) as NumeroAttori
   FROM Attori, Recita, Film
   WHERE Attori.CodAttore=Recita.CodAttore and Recita.CodFilm = Film.CodFilm
   GROUP BY Film.CodFilm, Film.Titolo
   HAVING max(Attori.AnnoNascita) < 1970;

8) SELECT Film.Titolo, sum(Proiezioni.Incasso) as IncassoTotale
   FROM Film, Proiezioni
   WHERE Film.Genere="Fantascienza"and Film.CodFilm = Proiezioni.CodFilm
   GROUP BY Film.CodFilm, Film.Titolo;

9) SELECT Film.Titolo, sum(Proiezioni.Incasso) as IncassoTotale
   FROM Film, Proiezioni
   WHERE Film.Genere="Fantascienza"and Film.CodFilm = Proiezioni.CodFilm and
   Proiezioni.Data > 1/1/01
   GROUP BY Film.CodFilm, Film.Titolo;
```

Soluzioni

```
10) SELECT Film.Titolo, sum(Proiezioni.Incasso) as IncassoTotale
FROM Film, Proiezioni
WHERE Film.Genere="Fantascienza" and Film.CodF= Proiezioni.CodF
GROUP BY Film.CodFilm, Film.Titolo
HAVING min(Proiezioni.Data) >= 1/1/01;
```

```
11) SELECT Sala.Nome, sum(Proiezioni.Incasso)
FROM Sala, Proiezioni
WHERE Proiezioni.CodSala=Sala.CodSala and Sala.Citta = 'Pisa' and Proiezioni.DataProiezione between 1/1/05 and
31/1/05
GROUP BY Sala.CodSala, Sala.Nome
HAVING sum(Proiezioni.Incasso) > 20.000;
```

```
12) SELECT Film.Titolo
FROM Film
WHERE "Pisa" not in (
    SELECT Sala.Città
    FROM Proiezioni, Sala
    WHERE Film.CodFilm = Proiezioni.CodFilm and Proiezioni.CodSala = Sala.CodS) ;
```