Esercizi SQL

Diversi approcci, diversi stili

Se siete “tipi logici”, pensate prima alle espressioni in algebra relazionale, e quindi trasformatele in query SQL.

Se siete più “smanettoni”, imparare a creare query in SQL vi aiuterà negli esercizi di Algebra Relazionale.

Come risolvere gli esercizi? (1)

Ancor prima di leggere il testo della query dobbiamo capire:

1. Qual è il database che vogliamo utilizzare?
   - Comprendere la struttura della propria base di dati, per comprendere quali sono le “relazioni” e quali le “entità” in gioco.

2. Quali sono le *primary key* e le *foreign key*?
   - Le primary key si riconoscono dal fatto che sono indicate da attributi selezionati all’interno della relazione.

Esercizio 1 (a)

Sono date le seguenti relazioni:

Docente(Matricola,Cognome, Sede)
Studente(Matricola,Cognome)
Corso(Codice,Nome)
EdizioneCorso(Corso,Anno,Docente)
Esame(Studente,Corso,Anno)

Quali informazioni estraiamo da questo insieme di relazioni?

Esercizio 1 (b)

Docente(Matricola,Cognome, Sede)
Studente(Matricola,Cognome)
Corso(Codice,Nome)
EdizioneCorso(Corso,Anno,Docente)
Esame(Studente,Corso,Anno)

- Tutti gli attributi sottolineati sono Primary Key.
- Un corso è identificato da un codice, ma ogni corso può avere più di una edizione (*Corso* in *EdizioneCorso* è foreign key)
- Per quanto concerne *Esame*?

Come risolvere gli esercizi? (2)

Ora possiamo passare a leggere la query. Le domande che ci dobbiamo porre sono le seguenti:

1. Quali relazioni entrano in gioco?
   - componenti della clausola **`FROM`**

2. Qual è il risultato finale atteso?
   - componenti della clausola **`SELECT`**

3. Quali sono le condizioni che permettono di filtrare i dati?
   - "theta-join" tra tabelle (es. collegarle) o filtering semplice (**`WHERE`**)
   - filtering di valori aggregati (**`Group By/Having`**)
   - Left/Right Join (**`FROM`**)

Esercizio 1 (c)

Docente(Matricola,Cognome, Sede)
Studente(Matricola,Cognome)
Corso(Codice,Nome)
EdizioneCorso(Corso,Anno,Docente)
Esame(Studente,Corso,Anno)

Selezionare in SQL i cognomi distinti che sono sia di docenti sia di studenti.

Esercizio 1 (d)

Iniziamo individuando le relazioni coinvolte: Docente e Studente. Possiamo quindi utilizzare la keyword DISTINCT per evitare ridondanze (cognomi ripetuti) e, in alternativa:

1. Utilizzare l'operatore di intersezione (troppo facile)
2. Effettuare un *join* tra le due relazioni.

Esercizio 1: Risultati (a)

1) Tramite intersezione:

SELECT DISTINCT Cognome
FROM Studente

INTERSECT

SELECT DISTINCT Cognome
FROM Docente

Esercizio 1: Risultati (b)

2) Con Join:

```
SELECT DISTINCT Studente.Cognome
FROM Studente, Docente
WHERE Studente.Cognome =
        Docente.Cognome
```

Ottimizzazione: nested loop join

- Il costrutto "… JOIN … ON *condition*" esprime un **join esplicito**

- Quando il join fra due tabelle R e S viene definito in maniera esplicita, l'ottimizzatore di query lo traduce utilizzando l'algoritmo ottimizzato di NESTED LOOP JOIN

Algoritmo di nested loop join

Una delle due relazioni coinvolte è designata come esterna e l'altra come interna.

Supponendo di operare utilizzando due relazioni R e S, R esterna e S interna, e di essere in presenza di due predicati locali F_R su R e F_s su S, l'algoritmo procede ricercando, per ogni tupla T_R della relazione esterna che soddisfa F_R, tutte le tuple di S che soddisfano il predicato di join F_J, e successivamente che soddisfano F_S.

Esercizio 1: Risultati (c)

3) Con Join espresso in modo **esplicito**:

```
SELECT DISTINCT S.Cognome
FROM Studente S JOIN Docente D
        ON S.Cognome = D.Cognome
```

Nota Bene!

Non esiste un’unica query possibile e corretta. Anche qui come in programmazione vige la regola “ognuno programma seguendo il proprio stile” (l’importante è non lasciare spazio a “bug”).

Esempio di nested loop join (1)

•Selezionare in SQL i cognomi distinti dei docenti che hanno tenuto un corso prima del 2007 e che lavorano nella sede di Urbino.

```
SELECT D.Cognome, E.Anno
FROM EdizioneCorso E JOIN Docente D
ON E.Docente = D.Matricola
WHERE D.Sede='Urbino' AND E.Anno <= 2007
```

Esempio di nested loop join (2)

1 Scan tabella Docente per cercare tutti quelli con sede Urbino (applico F_R)

Matricola	Cognome	Sede
M1	Bianchi	Urbino
M2	Cairo	Bologna
M3	Benedetti	Urbino
M4	Cecchetti	Urbino
M5	Magnani	Bologna

2 Prendiamo le coppie con stessa Matricola (applico Join ON F_J)

Corso	Anno	Docente
Basi di dati	2006	M1
Sist inf	2006	M4
Program	2008	M5
Sist oper	2009	M4
Calcolo	2009	M4
Analisi	2005	M2

3 Scan il risultato per trovare Docenti che hanno tenuto Corsi prima del 2007 (applico F_S)

Cognome	Anno
Bianchi	2006
Cecchetti	2006
Cecchetti	2008
Cecchetti	2009

4 Otteniamo:

Cognome	Anno
Bianchi	2006
Cecchetti	2006

Esercizio 2

Sono date le seguenti relazioni:

Docente(Matricola,Cognome)
Studente(Matricola,Cognome)
Corso(Codice,Nome)
EdizioneCorso(Corso,Anno,Docente)
Esame(Studente,Corso,Anno)

Selezionare in SQL i cognomi dei docenti che hanno tenuto dei corsi in cui sono stati sostenuti esami da almeno 10 studenti

Esercizio 2 (a)

Mettiamo in **grassetto** le entità che entrano in gioco:

Docente(Matricola,Cognome)
Studente(Matricola,Cognome)
Corso?(Codice,Nome)
EdizioneCorso?(Corso,Anno,Docente)
Esame(Studente,Corso,Anno)

Selezionare in SQL i cognomi dei **docenti** che hanno tenuto dei **corsi** in cui sono stati sostenuti **esami** da almeno 10 studenti

Esercizio 2 (b)

Il testo presenta un'ambiguità in "che hanno tenuto dei corsi". Nella base di dati sono presenti due relazioni, sia quella **Corso**, sia quella **EdizioneCorso**: qualle delle due bisogna considerare?

Esame(Studente,Corso,Anno)

Es.: osserva la relazione esame: contiene informazioni su *Corso* ed *Anno*, quindi facciamo riferimento ad **EdizioneCorso**.

Esercizio 2 (c)

Docente(Matricola,Cognome)
EdizioneCorso(Corso,Anno,Docente)
Esame(Studente,Corso,Anno)

Selezionare in SQL i cognomi dei **docenti** che hanno tenuto **"edizioni di corsi"** in cui sono stati sostenuti **esami** da almeno 10 studenti.

1) Quali relazioni entrano in gioco?

FROM Docente D, EdizioneCorso C, Esame E

Esercizio 2 (d)

1. Qual è il risultato finale atteso?

```
SELECT DISTINCT D.Cognome
```

2. Colleghiamo le foreign key(s) con le primary key(s):

```
WHERE D.Matricola = C.Docente AND
      C.Corso = E.Corso AND
      C.Anno = E.Anno
```

Esercizio 2 (e)

Quali sono le condizioni che permettono di filtrare i dati?
Filtriamo gli esami aggregandoli per docente e filtrandolo per numero di esami sostenuti (risultato dell'aggregazione per corso).

Osserva:

a) un docente è univocamente identificato dalla matricola, ma dobbiamo aggregare anche per cognome perché quello è uno dei risultati che dev'essere restituito dalla SELECT.
b) vogliamo tenere distinti i corsi nell'aggregazione, altrimenti aggreghiamo per numero di esami complessivi per docente.

Esercizio 2 (e)

```
SELECT DISTINCT D.Cognome
FROM Docente D, EdizioneCorso C, Esame E
WHERE D.Matricola = C.Docente AND
      C.Corso = E.Corso AND
      C.Anno = E.Anno
GROUP BY D.Matricola, D.Cognome, C.Corso, C.Anno
HAVING COUNT(*)>10
```

Che cosa restituirebbe la query se tolgo invece l'aggregazione sul corso?

Risposta: il numero di studenti per docente

Esercizio 3

E' data la seguente relazione:

Treno(Codice, Partenza, Destinazione,KM)

Selezionare in SQL la **lunghezza massima e minima** delle tratte **da Milano a Bari** senza aver effettuato cambi

Esercizio 3 (a)

- Cosa significa "senza aver effettuato cambi"? Considero solamente un tragitto senza scambi, filtrando sull'andata ed il ritorno.
- Per quanto riguarda le lunghezze (massima e minima) abbiamo bisogno di utilizzare le funzioni che lo standard SQL ci mette a disposizione, ovvero **max()** e **min()**.

Esercizio 3 (b)

```
SELECT min(KM), max(KM)
FROM Treno
WHERE Partenza='Milano' AND
      Destinazione ='Bari'
```

Esercizio 4

Sono date le seguenti relazioni:

Regione(NomeRegione,Abitanti,Superficie)
Residenza(Matricola,Cognome,NomeRegione)

Selezionare in SQL le regioni con più abitanti di residenti.

Esercizio 4 (a)

Il numero dei residenti per regione si ottiene aggregando Residenza per NomeRegione.

```
SELECT NomeRegione, Count(*) as Residenti
FROM Residenza
GROUP BY NomeRegione
```

In questo modo si definisce una vista su di una tabella, che può essere utilizzata in una clausola **FROM**, allo scopo di confrontarla con Abitanti.

Esercizio 4 (b)

```
SELECT A.NomeRegione
FROM Regione A,
        (SELECT NomeRegione, Count(*) as Residenti
         FROM Residenza
         GROUP BY NomeRegione) R

WHERE A.NomeRegione = R.NomeRegione AND
          A.Abitanti > R.Residenti
```

Esercizio 5

Voto		
codFilm	codUtente	valutazione
11234	19023	7
21234	892	9
31234	892	8
41234	19291	6

Utente		
codUtente	alias	eta'
892	MarioRossi	20
19023	AleRossi	15
19291	AntonioBianchi	36

Film			
codFilm	titolo	anno	regista
11234	Blade Runner	1982	Scott
21234	Pulp Fiction	1994	Tarantino
31234	Django Unchained	2012	Tarantino
41234	Rush	2013	Howard

Scrivere in SQL una query che restituisce i nomi dei film con una media di valutazioni superiore a 7, per i film prodotti tra il 1990 e il 2000 (inclusi).

Esercizio 5 - Risultato

Scrivere in SQL una query che restituisce i nomi dei film con una media di valutazioni superiore a 7, per i film prodotti tra il 1990 e il 2000 (inclusi).

```
SELECT Film.titolo
FROM Voto NATURAL JOIN Film
WHERE Film.anno >= 1990 AND
      Film.anno <= 2000
GROUP BY Film.codFilm
HAVING AVG(Voto.valutazione) > 7
```

Esercizio 6

Voto		
codFilm	codUtente	valutazione
11234	19023	7
21234	892	9
31234	892	8
41234	19291	6

Utente		
codUtente	alias	eta'
892	MarioRossi	20
19023	AleRossi	15
19291	AntonioBianchi	36

Film			
codFilm	titolo	anno	regista
11234	Blade Runner	1982	Scott
21234	Pulp Fiction	1994	Tarantino
31234	Django Unchained	2012	Tarantino
41234	Rush	2013	Howard

Scrivere in SQL una query che restituisce l'eta' minima degli utenti che hanno dato a 'Blade Runner' un voto maggiore di 8.

Esercizio 6 - Risultato

Scrivere in SQL una query che restituisce l'eta' minima degli utenti che hanno dato a 'Blade Runner' un voto maggiore di 8.

```
SELECT MIN(Utente.eta)
FROM Voto NATURAL JOIN Utente
                NATURAL JOIN Film
WHERE Film.titolo = 'Blade Runner' AND
          Voto.valutazione > 8
```

Esercizio 7

Artista		
id_artista	nome	record_label
111	Daft Punk	Virgin
222	Pixies	4AD
333	Manic Street Preachers	Columbia
444	Cocteau Twins	4AD

Playlist		
nome	posizione	id_canzone
Summer	1	33333
Summer	2	22222
Party	1	11111
Party	2	44444

Album			
id_album	titolo	id_artista	anno
1111	Twinlights	444	1995
2222	Discovery	111	2001
3333	Human After All	111	2005
4444	Surfer Rosa	333	1988

Canzone				
id_canzone	titolo	id_album	riproduzioni	durata_sec
11111	Tsunami	5555	12	230
22222	Gigantic	4444	8	235
33333	Where is My Mind	4444	24	233

Scrivere in SQL una query che restituisce i nomi delle playlist contenenti almeno una canzone tratta da un album del 2001 o precedente di un artista della record label 4AD

Esercizio 7 - Risultato

Scrivere in SQL una query che restituisce i nomi delle playlist contenenti almeno una canzone tratta da un album del 2001 o precedente di un artista della record label 4AD

```
SELECT DISTINCT Playlist.nome
FROM Playlist
        JOIN Canzone ON
            Playlist.id_canzone = Canzone.id_canzone
        JOIN Album ON
            Canzone.id_album = Album.id_album
        JOIN Artista ON
            Album.id_artista = Artista.id_artista
WHERE Album.anno <= 2001  AND
            Artista.‘record_label‘ = ’4AD’
```