Università di Milano Bicocca
Corso di Laurea in Statistica e Gestione delle Informazioni
Informatica (parte 3 CFU), solo studenti SGI.
mirko.cesarini@unimib.it

Simulazione d'esame.

La simulazione del tema d'esame si compone di 10 domande, da D1 a D10. Le domande da D1 a D8 sono domande a risposta chiusa, dove una e una sola domanda è corretta. Le domande D9 e D10 sono a risposta aperta.
Le soluzioni sono dopo la domanda D10.

Dopo le soluzioni delle domande da D1 a D10, trovate le domande aperte dei temi d'esame degli appelli scorsi e le soluzioni.

Se volete ulteriori delucidazioni sul compito contattate di persona i docenti. Domande fatte per email o per telefono non riceveranno risposta.

Negli appelli successivi la struttura del tema d'esame potrà subire delle leggere variazioni, es.:

- Al posto di 2 domande aperte potrebbe essere inserita un'unica domanda aperta
- Il numero di domande a risposta chiusa, potrebbe variare di qualche unità, in più o in meno.

Nessuna delle domande presenti in questo documento sarà presente negli appelli d'esame futuri.

D1
Le tabelle seguenti (tra parentesi graffe i nomi degli attributi, le chiavi primarie sono in MAIUSCOLO, le Foreign Key hanno la prima lettera maiuscola) ...

- Pietanza={CODICE, nome}
- Ingredienti={CODICE, descrizione, calorie_per_grammo}
- Composizione={Codice_pietanza, Codice_ingrediente, grammi}

... memorizzano i dati relativi a pietanze, gli ingredienti che compongono le pietanze, le calorie degli ingredienti. Gli attributi CODICE (entrambi), Codice_pietanza, Codice_ingrediente, calorie_per_grammo, grammi ospitano valori numerici interi.
L'attributo calorie_per_grammo contiene le calorie per 1 grammo di ingrediente, l'attributo grammi (nella tabella Composizione) indica quanti grammi di uno specifico ingrediente sono contenuti in una pietanza.
Quale query SQL consente di visualizzare il totale delle calorie fornite dalla pietanza "Pasta al pesto"? Assumete che nel database sia presente un'unica pietanza con tale nome, inoltre nella tabella risultato deve apparire una sola riga.

1. SELECT SUM(i.calorie_per_grammo*c.grammi) FROM Pietanza as p, Ingredienti as i, composizione as c WHERE p.codice=c.codice_pietanza AND c.codice_ingrediente = i.codice AND nome='Pasta al pesto';
2. SELECT i.calorie_per_grammo*c.grammi FROM Pietanza as p, Ingredienti as i, composizione as c WHERE p.codice=c.codice_pietanza AND c.codice_ingrediente = i.codice AND nome='Pasta al pesto';
3. SELECT SUM(i.calorie_per_grammo*c.grammi) FROM Pietanza as p, Ingredienti as i, composizione as c WHERE p.codice=c.codice_pietanza AND c.codice_ingrediente = i.codice GROUP BY i.codice;
4. SELECT i.calorie_per_grammo*c.grammi FROM Pietanza as p, Ingredienti as i, composizione as c WHERE nome='Pasta al pesto';
5. SELECT SUM(i.calorie_per_grammo*c.grammi) FROM Pietanza as p, Ingredienti as i, composizione as c GROUP BY p.codice HAVING nome='Pasta al pesto';

Le soluzioni sono dopo la domanda D10.

D2
Le tabelle seguenti (tra parentesi graffe i nomi degli attributi, le chiavi primarie sono in MAIUSCOLO, le Foreign Key hanno solo la prima lettera maiuscola) ...

- Taxi={TARGA nome_autovettura}
- Guidatore={COD, cognome, nome}
- Corsa={Targa, Cod_guidatore, data, ora, durata, importo}

... memorizzano i dati relativi alle corse effettuate da un gruppo di taxi (per corsa si intende un viaggio con un cliente pagante), le vetture in servizio e i guidatori dei taxi stessi.
La tabella corsa contiene un record per ognuna delle corse effettuate dai taxi. Ogni record memorizza le informazioni sul taxi usato per la corsa, sul guidatore, la data e l'ora d'inizio della corsa e la durata della corsa espressa in minuti.
Quale query SQL consente di visualizzare l'importo totale incassato da ogni tassista considerando tutte le corse presenti nel database? Dal risultato devono essere esclusi i tassisti che in totale hanno incassato meno di 50 euro. Nel risultato dovrà apparire una sola riga per ogni tassista.

1. SELECT g.cod, g.cognome, g.nome, SUM(importo) as tot FROM Corsa AS c, Guidatore AS g WHERE c.cod_guidatore=g.cod AND tot >=50 GROUP BY g.cod;
2. SELECT g.cod, g.cognome, g.nome, importo FROM Corsa AS c, Guidatore AS g WHERE c.cod_guidatore=g.cod AND importo >=50 GROUP BY g.cod;
3. SELECT g.cod, g.cognome, g.nome, SUM(importo) as tot FROM Corsa AS c, Guidatore AS g WHERE c.cod_guidatore=g.cod GROUP BY g.cod HAVING tot >=50;
4. SELECT g.cod, g.cognome, g.nome, SUM(importo) as tot FROM Corsa AS c, Guidatore AS g WHERE c.cod_guidatore=g.cod HAVING tot >=50;
5. SELECT g.cod, g.cognome, g.nome, SUM(importo) as tot FROM Corsa AS c, Guidatore AS g GROUP BY g.cod HAVING c.cod_guidatore=g.cod AND tot >=50;

Le soluzioni sono dopo la domanda D10.

D3
Un database
1. management system non permette di disciplinare l'accesso ai dati attraverso username e password
2. può contenere al massimo quattro tabelle
3. management system, può essere usato direttamente all'utente senza la mediazione di altre applicazioni client
4. management system non può gestire più database contemporaneamente
5. può essere interrogato con comandi SQL

D4
Una select SQL
1. riporta i nomi delle tabelle da cui prendere i dati dopo la clausola where
2. deve essere sempre conclusa da una virgola
3. indica i nomi degli attributi da visualizzare subito dopo la parola chiave select
4. permette di estrarre dati da non più di una tabella per volta
5. non permette di realizzare le operazioni descrivibili con gli operatori (dell'algebra relazionale) prodotto cartesiano, selezione e proiezione

D5
L'operatore AS usato in una select SQL
1. non può essere usato per nomi di tabelle lunghi
2. non può essere usato più di una volta all'interno di una stessa select
3. permette di rinominare temporaneamente il nome di una tabella
4. non permette di rinominare temporaneamente il nome di un attributo
5. può essere usato al posto della clausola having

D6
La clausola ORDER BY usata in una select SQL
1. può variare l'ordine di visualizzazione degli attributi da sinistra verso destra in una tabella
2. è incompatibile con la clausola group by
3. è incompatibile con la clausola where
4. non consente di specificare su quali attributi effettuare l'ordinamento
5. nessuna delle altre risposte

D7
La clausola HAVING usata in una select SQL
1. è incompatibile con la clausola group by
2. non può essere usata quando nella clausola from sono elencate più di una tabella
3. si può utilizzare solo congiuntamente alla clausola order by
4. permette di filtrare i risultati ottenuti da operazioni di aggregazione
5. può essere sempre sostituita da una clausola where equivalente

D8
In una base di dati,
1. possono esserci dei valori duplicati nell'attributo che funge da chiave primaria di una tabella
2. la chiave primaria permette di identificare univocamente due o più tuple all'interno di una stessa tabella
3. le chiavi non permettono di collegare informazioni tra tabelle diverse
4. la chiave esterna (chiamata anche chiave secondaria o foreign key) permette di accedere ad un database quando non ci si ricorda più la password

5. le tabelle possono essere dotate di chiavi primarie

D9 (domanda a risposta aperta) estratta dal tema d'esame del 21/11/2015
Testo domanda. Date le tabelle seguenti (tra parentesi i nomi degli attributi, le CHIAVI PRIMARIE sono scritte interamente in maiuscolo, le chiavi esterne hanno solo la prima lettera maiuscola)

- Citta: (ID_CITTA, nome)
- Treno: (ID_TRENO, nome)
- Biglietto: (ID_BIGLIETTO, Id_utente, Id_citta_partenza, Id_citta_arrivo, km, prezzo)
- Utente: (ID_UTENTE, nome, cognome, eta)

Gli attributi eta e km assumono valori interi. Scrivete una query che mostri per i soli utenti con età maggiore di 50 anni il rapporto tra il totale dei soldi spesi in biglietti e il totale dei km percorsi. Nel risultato dovranno apparire l'id_utente, il nome, il cognome e il rapporto di cui sopra, inoltre nel risultato non ci dovrà essere più di un record per utente.

D10 (domanda a risposta aperta) estratta dal tema d'esame del 21/11/2015
Testo domanda. Date le tabelle seguenti (tra parentesi i nomi degli attributi, le CHIAVI PRIMARIE sono scritte interamente in maiuscolo, le Chiavi Esterne hanno solo la prima lettera maiuscola)

- Giocatore: (ID_GIOCATORE, nome, cognome)
- Partita: (ID_PARTITA, data)
- Goal: (ID_GIOCATORE, ID_PARTITA, numerogoal)
- Compenso: (ID_GIOCATORE, MESE, compensomensile)

Le tabelle riportano i dati dei goal effettuati e dei compensi percepiti dai giocatori in una stagione. La tabella Goal riporta il numero di goal segnati da un giocatore per ogni partita, al suo interno la coppia di attributi ID_GIOCATORE, ID_PARTITA funge da chiave primaria, ognuno dei due attributi preso singolarmente funge da foreign key, rispettivamente per le tabelle Giocatore e Partita. Per ogni partita vengono riportati solamente i dati dei giocatori che hanno segnato goal (in altre parole, non sono presenti record con numerogoal pari a 0). Dato che per uno stesso giocatore il compenso può variare di mese in mese, la tabella Compenso riporta il compenso mensile percepito dai giocatori nei diversi mesi.
Con un'unica query, calcolate per ogni giocatore il rapporto tra stipendio totale percepito e totale dei goal effettuati. Fate attenzione che le tabelle Compenso e Goal hanno una diversa granularità.

Soluzioni domande a risposta chiusa

Domande a risposta chiusa

D1: 1
D2: 3
D3: 5
D4: 3
D5: 3
D6: 5
D7: 4
D8: 5

Soluzioni domande a risposta aperta

Nota metodologica: Per ogni domanda vengono proposte una o più soluzioni. Dato un quesito possono esistere più soluzioni corrette, non tutte le soluzioni possibili sono riportate nelle soluzioni.

D9 (si riportano solo alcuni esempi di soluzioni, ci possono essere altri esempi non riportati qua sotto di soluzioni sia corrette sia non corrette):

- Esempio di soluzione corretta
```
SELECT u.id_utente, u.nome, u.cognome, SUM(prezzo)/SUM(km)
FROM utente AS u, biglietto AS b
WHERE u.id_utente=b.id_utente AND u.eta > 50
GROUP BY u.id_utente;
```
- Altro esempio di soluzione corretta
```
SELECT u.id_utente, u.nome, u.cognome, SUM(prezzo)/SUM(km)
FROM utente AS u JOIN, biglietto AS b ON (u.id_utente=b.id_utente)
GROUP BY u.id_utente
HAVING u.eta > 50;
```
- Esempio di soluzione **NON corretta**:
```
SELECT u.id_utente, u.nome, u.cognome, SUM(prezzo/km)
FROM utente AS u, biglietto AS b
WHERE u.id_utente=b.id_utente AND u.eta > 50
GROUP BY u.id_utente;
```

D10 (si riportano solo alcuni esempi di soluzioni, ci possono essere altri esempi non riportati qua sotto di soluzioni sia corrette sia non corrette):

- Esempio di soluzione corretta
```
SELECT id_giocatore, nome, cognome, a.totcompenso/b.totgoal
FROM giocatore as g, (
    SELECT id_giocatore, SUM(numerogoal) AS totgoal
    FROM goal
    GROUP BY id_giocatore
) AS b,
(
    SELECT id_giocatore, SUM(compensomensile) AS totcompenso
    FROM compenso
    GROUP BY id_giocatore
) AS a
WHERE g.id_giocatore = a.id_giocatore AND a.id_giocatore=b.id_giocatore;
```

- Altro esempio di soluzione corretta

```
SELECT id_giocatore, a.totcompenso/b.totgoal
FROM (
    SELECT id_giocatore, SUM(numerogoal) AS totgoal
    FROM goal
    GROUP BY id_giocatore
) AS b,
(
    SELECT id_giocatore, SUM(compensomensile) AS totcompenso
    FROM compenso
    GROUP BY id_giocatore
) AS a
WHERE a.id_giocatore=b.id_giocatore
GROUP BY a.id_giocatore;      -- qua la cloausola GROUP BY non è sbagliata ma è inutile
```

- Esempio di soluzione **non corretta**

```
SELECT g.id_giocatore, g.cognome, SUM(c.compensomensile)/SUM(go.numerogoal)
FROM giocatore AS g, goal AS go, compenso AS c
WHERE g.id_giocatore=go.id_giocatore and g.id_giocatore=c.id_giocatore
GROUP BY id_giocatore;
```

L'ultima soluzione non è corretta perché, date due tabelle (relazioni) come le seguenti (saranno mostrati solo alcuni record di ogni tabella),

Goal

ID_GIOCATORE	ID_PARTITA	numerogoal
1	20	2
1	21	1

Compenso

ID_GIOCATORE	MESE	compensomensile
1	10	2000
1	11	1000

Giocatore

ID_GIOCATORE	Nome	Cognome
1	P.R.	C.

la condizione (nella clausola WHERE) g.id_giocatore=c.id_giocatore produrrebbe il seguente risultato intermedio

risultato intermedio, prima di applicare le funzioni di aggregazione

ID_GIOCATORE	Nome	Cognome	Mese	compensomensile	ID_Partita	numerogoal	…
1	P.R.	C.	10	2000	20	2	…
1	P.R.	C.	10	2000	21	1	…
1	P.R.	C.	11	1000	20	2	…
1	P.R.	C.	11	1000	21	1	…

Applicando le funzioni di aggregazione sul risultato intermedio di cui sopra, le somme di compensomensile e numerogoal produrrebbero dei risultati non corretti.

Altri esempi di domande a risposta aperta (e relative soluzioni)

Prova del 29/01/2015

I testi delle domande e delle risposte li potete trovare negli esercizi D9 e D10 della simulazione d'esame.

Prova del 29/01/2016

Testo domanda. Date le tabelle seguenti (tra parentesi i nomi degli attributi, le CHIAVI PRIMARIE sono scritte interamente in maiuscolo, le Foreign Key hanno solo la prima lettera maiuscola):

- Volo: (ID_VOLO, Id_citta partenza, Id_citta_arrivo, Id_pilota)
- Passeggero: (ID_PASSEGGERO, nome, cognome)
- Prenotazione: (ID_PRENOTAZIONE, Id_passeggero, Id_volo, data, prezzo_biglietto)
- Pilota: (ID_PILOTA, nome, cognome, stipendio_annuo)
- Citta: (ID_CITTA, nome)

Assumete che nella base di dati siano contenuti i dati raccolti in un anno di attività. Nello stesso anno, ogni volo è sempre stato condotto da uno e un solo pilota (sempre lo stesso). Con un'unica query, visualizzate, per ogni volo, l'Id_volo, la città di partenza, la città di arrivo, il rapporto tra la somma incassata con i biglietti e lo stipendio annuo speso per il pilota. Non mostrate i dati per quei voli che hanno un rapporto superiore a 100. Mostrate solamente una riga per ogni volo nella tabella risultato. Fate attenzione che i dati su stipendio annuo e su prezzo biglietto hanno una diversa granularità.

Soluzioni

```
select v.id_volo, v.id_citta_partenza, v.id_citta_arrivo, i.tot/p.stipendio_annuo as rap
from (
  select id_volo, sum(prezzo_biglietto) as tot
  from prenotazione as pr
  group by id_volo
) as i, volo as v, pilota as p
where i.id_volo = v.id_volo and v.id_pilota=p.id_pilota and i.tot/stipendio_annuo>100;
```

Altra possibile soluzione:

```
select v.id_volo, v.id_citta_partenza, v.id_citta_arrivo,
sum(prezzo_biglietto)/p.stipendio_annuo as rap
  from prenotazione as pr, volo as v, pilota as p
  where pr.id_volo=v.id_volo and v.id_pilota=p.id_pilota
  group by id_volo
having rap>100;
```

Prova del 19/02/2016

Testo domanda. Date le tabelle seguenti di un database (tra parentesi i nomi degli attributi, le CHIAVI PRIMARIE sono scritte interamente in maiuscolo, le Foreign Key hanno solo la prima lettera maiuscola):

- Opera: (ID_OPERA, titolo)
- Rappresentazione: (ID_RAPPRESENTAZIONE, giorno, Id_opera)
- Biglietto: (ID_BIGLIETTO, prezzo, Id_rappresentazione, Id_spettatore)
- Spettatore: (ID_SPETTATORE, nome, cognome)
- Attore: (ID_ATTORE, nome, cognome)
- Recitazionein: (ID_ATTORE, ID_OPERA)
- Il database contiene i dati di un anno delle opere messe in scena in un teatro, dei biglietti venduti e degli attori che hanno recitato.

Scrivete una query SQL che mostri per ogni opera, il rapporto tra il totale del prezzo dei biglietti incassati da un'opera e il numero di attori che recitano nell'opera stessa.
Nella tabella risultato ogni opera dovrà apparire una sola volta, inoltre dovete visualizzare solo i dati delle opere che hanno più di 10 attori.

Fate attenzione alla granularità delle diverse tabelle.

Soluzione

```
select o.id_opera, o.titolo, totale / num
from
(select b.id_opera, sum(prezzo) as totale
  from Biglietto as b, Rappresentazione as r
  where b.id_opera=r.id_opera
  group by id_opera
) as incasso,
(select id_opera, count(*) as num
  from recitazionein
  group by id_opera
) as rec, opera as o
where incasso.id_opera=rec.id_opera and rec.id_opera=o.id_opera and rec.num>10;
```

Prova del 20/04/2016

Testo domanda. Date le tabelle seguenti (tra parentesi i nomi degli attributi, le CHIAVI PRIMARIE sono scritte interamente in maiuscolo, le Foreign Key hanno solo la prima lettera maiuscola):

- Automobile: (TARGA, marca, modello, Id_proprietario
- Proprietario: (ID_PROPRIETARIO, nome, cognome,
- Id_citta_residenza)
- Citta: (ID_CITTA, nome, Id_regione)
- Regione: (ID_REGIONE, nome)
- Multe: (ID_MULTA, Targa_auto, importo, Id_regione) #Id_regione: regione in cui è stata presa la multa
- Il database contiene informazioni su automobili, proprietari, e multe prese dalle automobili.

Scrivete una query SQL che mostri per ogni regione: il nome della regione, il rapporto tra il totale delle multe date nella regione e il numero di abitanti della regione stessa. Nella tabella risultato, ogni regione deve apparire una sola volta.
Fate attenzione alla granularità delle diverse tabelle.

Soluzione

```
select r.nome, totmu.tot_importo / abit.nabitanti as rapporto
from Regione as r,
  (
    select re.id_regione as id_regione, count(*) as nabitanti
    from proprietario as p, citta as c, regione as re
    where p.id_citta_residenza=c.id_citta and c.id_regione=re.id_regione
    group by re.id_regione
  ) as abit,
  (
    select mu.id_regione as id_regione, sum(importo) as tot_importo
    from multe as  mu
    group by mu.id_regione
  ) as totmu
  where r.id_regione = abit.id_regione and abit.id_regione = totmu.id_regione;
```

Prova del 27/06/2016

Testo domanda. Date le tabelle seguenti (tra parentesi i nomi degli attributi, le CHIAVI PRIMARIE sono scritte interamente in maiuscolo, le Foreign Key hanno solo la prima lettera maiuscola):

- Automobile: (TARGA, marca, modello, Id_proprietario
- Proprietario: (ID_PROPRIETARIO, nome, cognome,
- Id_citta_residenza)
- Citta: (ID_CITTA, nome, Id_regione)
- Regione: (ID_REGIONE, nome)
- Multe: (ID_MULTA, Targa_auto, importo, Id_regione) #Id_regione: regione in cui è stata presa la multa
- Il database contiene informazioni su automobili, proprietari, e multe prese dalle automobili.

Il database contiene informazioni su automobili, proprietari, e multe prese dalle automobili.

Scrivete una query SQL che mostri per ogni proprietario la somma delle multe pagate dal proprietario stesso.
Nella tabella risultato, ogni proprietario deve apparire una sola volta, inoltre non devono essere visualizzati i proprietari che hanno pagato complessivamente meno di 1000 euro.

Soluzione

```
SELECT p.id_proprietario, p.nome, p.cognome, SUM(importo) AS tot
proprietario as p, automobile as a, multe as m
WHERE p.id_proprietario=a.id_proprietario AND a.targa=m.targa_auto
GROUP BY p.id_proprietario
HAVING tot >=1000;
```

Prova del 13/07/2016

Testo domanda. Date le tabelle seguenti di un database (tra parentesi i nomi degli attributi, le CHIAVI PRIMARIE sono scritte interamente in maiuscolo, le Foreign Key hanno solo la prima lettera maiuscola):

- AnagraficaProdotto: (ID_PRODOTTO, descrizione)
- AnagraficaCliente: (ID_CLIENTE, nome, cognome)
- Fattorino: (ID_FATTORINO, nome, cognome)
- Vendita: (ID_VENDITA, quantita, prezzo_unitario, Id_prodotto, Id_cliente, Id_consegna)
- Consegna: (ID_CONSEGNA, indirizzo, giorno, Id_fattorino)

Il database contiene i dati delle vendite a domicilio effettuate da un sito di e-commerce. Ogni cliente è registrato nell'AnagraficaCliente.
Scrivete una query SQL che mostri per ogni fattorino, il rapporto tra il totale del prezzo incassato con le vendite consegnate dal fattorino e il numero di consegne effettuate dal fattorino stesso.
Nella tabella risultato dovranno apparire anche il nome e il cognome del fattorino, inoltre ogni fattorino dovrà apparire una e una sola volta nel risultato. Prestate attenzione al fatto che una consegna può raggruppare più prodotti e anche più ordini (es., un utente puo' acquistare diverse volte e ricevere tutto con un'unica consegna, oppure acquisti di utenti diversi che abitano nello stesso condominio possono venire consegnati attraverso un'unica consegna). In altre parole, la granularità della tabella Consegna e della tabella Vendita è diversa.

Fate attenzione alla granularità delle diverse tabelle.

Soluzione

```
SELECT f.id_fattorino, f.nome, f.cognome, incassi.tot/cons.n AS rapporto
FROM fattorino AS F,
(
  SELECT c.id_fattorino, SUM(v.quantita*v.prezzo_unitario) as tot
  FROM vendita AS v, consegna AS c
  WHERE v.id_consegna=c.id_consegna
  GROUP BY c.id_fattorino) AS incassi,
(
  SELECT id_fattorino, COUNT(*) AS n
  FROM consegna
  GROUP BY id_fattorino
) AS cons
WHERE f.id_fattorino=incassi.id_fattorino and f.id_fattorino=cons.id_fattorino;
```

Prova del 14/09/2016

Testo domanda. Date le tabelle seguenti di un database (tra parentesi i nomi degli attributi, le CHIAVI PRIMARIE sono scritte interamente in maiuscolo, le Foreign Key hanno solo la prima lettera maiuscola):

- CasaFarmaceutica: (ID_CASA, nome)
- Farmaco: (ID_FARMACO, nome, Id_casa)
- Farmacia: (ID_NEGOZIO, nome, indirizzzo)
- Vendita: (Id_farmaco, Id_farmacia, n_confezioni, prezzo_totale, giorno)

Il database contiene i dati delle vendite di medicinali effettuate da un gruppo di farmacie. Ogni record della tabella vendita riporta l'acquisto di un farmaco effettuato da un cliente. L'attributo n_confezioni memorizza il numero di confenzioni acquistate dal cliente, l'attributo prezzo_totale rappresenta il prezzo totale pagato dal cliente per acquistare le confezioni. Scrivete una query SQL che mostri per ogni farmaco, il rapporto tra soldi incassati e numero di confezioni vendute. Per ogni farmaco dovete visualizzare il nome del farmaco stesso e della casa farmaceutica che lo vende. Nel risultato devono essere visualizzati solo quei farmaci il cui rapporto è maggiore o uguale a 50.

Soluzione

```
SELECT f.id_farmaco, f.nome, cf.nome, SUM(v.prezzo_totale)/SUM(n_confezioni) AS rapporto
FROM vendita AS v, farmaco AS f, casafarmaceutica AS cf
WHERE v.id_farmaco=f.id_farmaco AND f.id_casa=cf.id
GROUP BY f.id_farmaco
HAVING rapporto >=50;
```

Prova del 24/11/2016 – Domanda aperta 1

Testo domanda. Date le tabelle seguenti di un database (tra parentesi i nomi degli attributi, le CHIAVI PRIMARIE sono scritte interamente in maiuscolo, le Foreign Key hanno la prima lettera maiuscola, gli attributi id_... presenti nelle chiavi primarie, presi singolarmente fungono da foreigh key):

- Albergo: (ID_ALBERGO, nome, indirizzo)
- Cliente: (ID_CLIENTE, nome, cognome)
- Camera: (ID_ALBERGO, ID_CAMERA, n_letti)
- Prenotazione: (Id_cliente, Id_albergo, Id_camera, da_data, a_data, n_persone)
- Pulizia: (ID_ALBERGO, ID_CAMERA, DATA, ore)

Il database contiene i dati delle prenotazioni di camere effettuate dai clienti di una catena di alberghi e delle ore impiegate per pulire le camere. La tabella camere ha una chiave formata da due attributi perché in ogni albergo id_camera ha sempre valori compresi tra 1 e il numero di camere dello specifico albergo. Dato che diverse camere (in diversi alberghi) possono avere id_camera uguale a 1, allora per identificare univocamente una camera occorre indicare anche l'id_albergo. Gli attributi da_data, a_data della tabella prenotazioni contengono le date rispettivamente del primo e dell'ultimo giorno di prenotazione, l'attributo n_persone della stessa tabella contiene il numero di persone per cui viene effettuata la prenotazione (es., una camera a due letti potrebbe essere prenotata da 1 sola persona). La tabella pulizia indica, per ogni camera e per ogni giorno, il numero di ore impiegate per pulirla (le ore sono valori interi, possibili valori sono 1, 2, ...). Gli id nelle diverse tabelle sono sempre valori interi.

Scrivete una query SQL che mostri per ogni albergo il rapporto tra il numero di ore impiegate per pulire le camere e il numero di prenotazioni effettuate nell'albergo. Nel risultato i dati di un albergo devono essere riportati una sola volta, per ogni albergo devono essere mostrati anche id_albergo, nome e indirizzo. Dal risultato devono essere esclusi quegli alberghi che hanno un rapporto inferiore a 2. Fate attenzione alle granularità delle diverse tabelle.

Soluzione

```
select a.id_albergo, a.nome, a.indirizzo, pul_ag.so / pren_ag as n_pren as rap
from (
  select id_albergo, sum(ore) as so
  from pulizia
  group by id_albergo
) as pul_ag,
(
  select id_albergo, count(*) as n_pren
  from prenotazione
  group by id_albergo
) as pren_ag,
albergo as a
where a.id_albergo=pul_ag.id_albergo and a.id_albergo=pren_ag.id_albergo
having rap>=2;
```

Prova del 24/11/2016 – Domanda aperta 2

Testo domanda. Date le tabelle seguenti di un database (tra parentesi i nomi degli attributi, le CHIAVI PRIMARIE sono scritte interamente in maiuscolo, le Foreign Key hanno la prima lettera maiuscola, gli attributi id_... presenti nelle chiavi primarie, presi singolarmente fungono da foreigh key):

- Giornale: (ID_GIORNALE, nome)
- Giornalista: (ID_GIORNALISTA, nome, cognome)
- Articolo: (ID_ARTICOLO, Id_giornalista, titolo, testo)
- Pubblicazione: (Id_giornale, Id_articolo, compenso)

Il database contiene i dati degli articoli scritti da alcuni giornalisti, delle pubblicazioni degli articoli nei diversi giornali, e del compenso percepito da una giornalista ogni volta che un suo articolo viene pubblicato. Un articolo di un giornalista può essere pubblicato su giornali diversi con diversi compensi.

Scrivete una query SQL che mostri per ogni coppia di giornale e giornalista il rapporto tra i compensi che il giornale ha elargito al giornalista e il totale dei compensi spesi dal giornale per tutti gli articoli pubblicati (includendo tutti i giornalisti, incluso il giornalista del numeratore). Non c'è bisogno di gestire le coppie giornale e giornalista in cui il giornalista non ha mai avuto pubblicazioni sullo specifico giornale.

Nel risultato i dati di una coppia (giornale, giornalista) devono essere riportati una sola volta, per ogni coppia devono essere mostrati solo l'id_giornalista e l'id_giornale, non occorre mostrare il nome del giornale e il nome e cognome del giornalista. Dal risultato devono essere incluse solo quelle coppie (giornale, giornalista) che hanno un rapporto minore o uguale a 0.1 . Fate attenzione alle granularità delle diverse tabelle.

Esempi di possibili soluzioni

```
select g1.id_giornale, g1.id_giornalista, g1.t_comp1 / g2.t_comp2
from (
  select p.id_giornale, a.id_giornalista, sum(compenso) as t_comp1
  from pubblicazione as p, articolo as a
  where p.id_articolo=a.id_articolo
  group by p.id_giornale, a.id_giornalista
) as g1,
(
  select id_giornale, sum(compenso) as t_comp2
  from pubblicazione
  group by id_giornale
) as g2
where g1.id_giornale=g2.id_giornale and g1.t_comp1 / g2.t_comp2 <=0.1;
```

```
select p.id_giornale, a.id_giornalista, sum(compenso) / g2.t_comp2 as rap
from
(   select id_giornale, sum(compenso) as t_comp2
  from pubblicazione
  group by id_giornale
) as g2, pubblicazione as p, articolo as a,
where p.id_articolo=a.id_articolo and p.id_giornale=g2.id_giornale
group by p.id_giornale, a.id_giornalista
having rap <=0.1;
```

Prova del 24/11/2016 – Domanda aperta 3

Testo domanda. Date le tabelle seguenti di un database (tra parentesi i nomi degli attributi, le CHIAVI PRIMARIE sono scritte interamente in maiuscolo, le Foreign Key hanno la prima lettera maiuscola, gli attributi id_... presenti nelle chiavi primarie, presi singolarmente fungono da foreigh key):

- Artista: (ID_ARTISTA, nome, cognome)
- Brano: (ID_BRANO, Id_artista, nome_brano)
- Utente: (ID_UTENTE, cognome, nome)
- Vendite: (ID_UTENTE, ID_BRANO, prezzo, data)
- Pubblicita: (ID_BRANO, data, ora, importo)

Il database contiene i dati relativi alle vendite di brani musicali (attraverso piattaforme digitali) e alla pubblicità spesa per promuovere tali brani. La tabella artista contiene i dati relativi agli artisti autori dei brani riportati nella tabella brano. Le tabelle utente e vendite riportano i dati degli utenti e dei brani a loro venduti. La tabella pubblicità riporta le spese in pubblicità effettuate per promuovere i brani. Possono essere effettuate più inserzioni pubblicitarie relative ad uno stesso brano, in date e ore diverse. Ogni record della tabella pubblicità rappresenta il costo di una singola inserzione pubblicitaria. L'importo speso per un'inserzione pubblicitaria può variare in base alla data e all'ora in cui la pubblicità è avvenuta (cioè non si tratta di un valore costante).

Scrivete una query SQL che mostri per ogni artista il rapporto tra soldi incassati dalla vendita dei brani e soldi spesi per la promozione pubblicitaria (dei brani dell'artista). Nella query è sufficiente mostrare l'id dell'artista, non serve visualizzare il nome.

Nel risultato i dati di un artista devono essere riportati una sola volta. Nel risultato devono essere inclusi solo gli artisti che hanno un rapporto tra incassi e spese di pubblicità superiore a 10. Fate attenzione alle granularità delle diverse tabelle.

Soluzione

```
select inc.id_artista, inc.t_incassi / sp.t_spese
from (
  select b.id_artista, sum(prezzo) as t_incassi
  from vendite as v, brano as b
  where v.id_brano=b.id_brano
  group by b.id_artista
) as inc,
(
  select b.id_artista, sum(importo) as t_spese
  from pubblicta as p, brano as b
  where p.id_brano=b.id_brano
) as sp
where inc.id_artista=sp.id_artista and inc.t_incassi / sp.t_spese >10
```

Prova del 02/02/2017

Testo domanda. Date le tabelle seguenti (tra parentesi i nomi degli attributi, le CHIAVI PRIMARIE sono scritte interamente in maiuscolo, le Foreign Key hanno la prima o le prime due lettere maiuscole):

- Barca(ID_BARCA, nome_barca, Id_proprietario, peso)
- Proprietario(ID_PROPRIETARIO, nome_proprietario)
- Pescato(data, ID_barca, quantita)
- Marinaio(ID_MARINAIO, nome, cognome, eta)
- Orelavorate(data, ID_marinaio, ID_barca, ore)

Il database contiene informazioni sulle barche presenti in un porto, i proprietari, i marinai che lavorano sulle barche, la quantita di pesce pescato e il numero di ore lavorate dai marinai.

Scrivete una query SQL che mostri per ogni barca il rapporto tra la quantità di pesce pescato e il numero di ore lavorate. Nella tabella risultato, ogni barca deve apparire una sola volta, inoltre per ogni barca deve apparire sia il nome della barca si il nome del proprietario.
Fate attenzione alla granularità delle diverse tabelle.

Soluzione

```
select b.id_barca. b.nome, prop.nome_proprietario, pesc.totq/lav.toth
from barca as b, proprietario as prop, (
  select id_barca, sum(quantita) as totqt from  pescato group by id_barca
) as pesc, (
  select id_barca, sum(ore) as toth from orelavorate group by id_barca)
) as lav
where b.id_proprietario = prop.id_proprietario and b.id_barca=pesc.id_barca and
b.id_barca=lav.id_barca;
```

Prova del 23/02/2017

Testo domanda.

Date le tabelle seguenti (tra parentesi i nomi degli attributi, le CHIAVI PRIMARIE sono scritte interamente in maiuscolo, le Foreign Key hanno la prima o le prime due lettere maiuscole):

Azienda (ID_AZIENDA, denominazione)
Manager(ID_MANAGER, nome, stipendio_manager, ID_azienda)
Stagista (ID_STAGISTA, nome, cognome, eta)
Stage (ID_stagista, ID_azienda, ore_lavorate, stipendio)

La tabella azienda contiene informazioni su alcune aziende, la tabella Manager riporta informazioni sui manager che lavorano per le aziende. Un manager può lavorare per una sola azienda, una azienda può avere più manager alle sue dipendenze.
La tabella Stage riporta informazioni sugli stage effettuati dagli stagisti nelle aziende.

Scrivete un'unica query SQL che mostri per ogni Azienda il rapporto tra lo stipendio erogato ai manger e lo stipendio erogato agli stagisti, considerate solo gli stagisti con età >= 25 anni. Per ogni azienda, oltre al rapporto descritto sopra, devono essere visualizzati l'ID_azienda e la denominazione.
Fate attenzione alla granularità delle diverse tabelle

Soluzione

```
select a.id_azienda, a.denominazione, sm.tot_stip_man/st.tot_stip_stag
from azienda as a,
(
  select a.id_azienda, sum(stipendio_manager) as tot_stip_man
  from manager as m, azienda as a
  where a.id_azienda=m.id_azienda
  group by a.id_azienda
) as sm,
(
  select s.id_azienda, sum(stipendio) as tot_stip_stag
  from stage as s, stagista
  where stagista.eta>=25 and stagista.id_stagista=s.id_stagista
  group by s.id_azienda
) as st
where a.id_azienda=sm.id_azienda and sm.id_azienda=st.id_azienda;
```

Prova del 08/05/2017

Testo domanda. Date le tabelle seguenti (tra parentesi i nomi degli attributi, le CHIAVI PRIMARIE sono scritte interamente in maiuscolo, le Foreign Key hanno la prima o le prime due lettere maiuscole):

- Consulente (ID_CONSULENTE, nome, cognome, costo_orario)
- Progetto(ID_PROGETTO, nomeprogetto, ricavo_totale_progetto)
- Attivita (ID_progetto, ID_consulente, mese, ore_lavorate)

La tabella Consulente riporta informazioni su alcune persone che lavorano come consulenti per una azienda.
La tabella Progetto riporta informazioni sui progetti svolti dalla societa' di consulenza.
La tabella Attivita riporta le ore mensili che i consulenti hanno impiegato per lavorare nei diversi progetti.

Scrivete un'unica query SQL che mostri per ogni progetto: l'id_progetto, il nome del progetto, il ricavo totale e i costi per personale che hanno lavorato sul progetto. Nella query dovranno essere visualizzati solo quei progetti in cui il rapporto tra ricavo totale e costi del personale e' maggiore di 3. Ogni progetto deve apparire una sola volta.

Soluzione

```
select p.id_progetto, p.nome_progetto, p.ricavo_totale_progetto as RT,
sum(c.costo_orario*a.ore_lavorate) as CT
from attivita as a, progetto as p, consulente as c
where a.id_progetto=p.id_progetto and a.id_consulente=c.id_consulente
group by p.id_progetto
having RT/CT>3;
```

Prova del 28/06/2017

Testo domanda. Date le tabelle seguenti (tra parentesi i nomi degli attributi, le CHIAVI PRIMARIE sono scritte interamente in maiuscolo, le Foreign Key hanno la prima o le prime due lettere maiuscole):

- Cliente (ID_CLIENTE, nome, cognome)
- Opera (ID_OPERA, nome)
- Prenotazione (ID_cliente, ID_opera, data, posto, prezzo)
- AtrezzaturaScenica (ID_ATREZZATURA, nome)
- NoleggioAtrezzatura (ID_atrezzatura, ID_opera, costo)

La tabella Cliente riporta informazioni sui clienti di un teatro.
La tabella Opera contiene informazioni sulle opere messe in scena nel teatro.
La tabella Prenotazione memorizza le prenotazioni effettuate dai clienti e il prezzo pagato.
La tabella AtrezzaturaScenica riporta informazioni sulle atrezzature sceniche noleggiate dal teatro.
La tabella NoleggioAtrezzatura contiene informazioni sui costi delle atrezzature noleggiate per le singole opere. Ogni atrezzatura viene utilizzata per una sola opera.

Scrivete un'unica query SQL che mostri per ogni opera, il nome dell'opera, il totale incassato con le prenotazioni, il totale speso per il noleggio delle atrezzature di scena e il rapporto tra il totale incassato e il totale speso per le atrezzature di scena. Per ogni opera deve apparire una sola riga nella tabella risultato. Fate attenzione alla granularità delle tabelle.

Soluzione

```
select o.nome,
from opera as o, tot.incasso, nol.cos, tot.incasso / nol.cos as rapporto
  ( select id_opera, sum(costo) as cos from NoleggioAtrezzatura group by id_opera ) as nol,
  ( select id_opera, sum(prezzo) as incasso from Prenotazione group by id_opera ) as tot
  where o.id_opera=nol.id_opera and o.id_opera=tot.id_opera;
```

Prova del 02/02/2017

Testo domanda. Date le tabelle seguenti (tra parentesi i nomi degli attributi, le CHIAVI PRIMARIE sono scritte interamente in maiuscolo, le Foreign Key hanno la prima o le prime due lettere maiuscole):

- Barca (ID_BARCA, nome)
- Porto (ID_PORTO, nomeporto)
- Tariffe (ID_barca, ID_porto, costo_giornaliero, numero_giorni, giorno_inizio, mese_inizio, anno_inizio)
- Manutenzioni (ID_barca, importo, giorno, mese, anno)

Le tabelle Barca e Porto riportano informazioni rispettivamente su delle barche e dei porti.
La tabella Tariffe riporta le informazioni sulle tariffe applicate alle barche per la sosta nei porti. Costo_giornaliero riporta quanto una barca deve pagare al giorno, numero_giorni indica il numero di giorni che la barca e' stata in porto, infine giorno_inizio, mese_inizio e anno_inizio indicano il primo giorno del periodo in cui la barca si e' fermata in porto.
La tabella Manutenzioni indica gli importi spesi per le manutenzioni delle barche. Ci potrebbero essere più record per una stessa barca dentro la tabella.

Scrivete un'unica query SQL che mostri per ogni barca, il nome della barca, il totale dei soldi spesi per la permanenza nei porti e il totale dei soldi spesi per le manutenzioni. Per ogni barca deve apparire una sola riga nella tabella risultato. Fate attenzione alla granularità delle tabelle.

Soluzione

```
select b.id_barca, b.nome, sosta.tot_sosta, man.tot_manutenzione
from Barca as b,
  (select id_barca, sum(costo_giornaliero*numero_giorni) as tot_sosta from Tariffe
   group by id_barca) as sosta,
  (select id_barca, sum(importo) as tot_manutenzione from manutenzioni
   group by id_barca) as man
where b.id_barca = sosta.id_barca and b.id_barca=man.id_barca;
```

Prova del 29/09/2017

Testo domanda. Date le tabelle seguenti (tra parentesi i nomi degli attributi, le CHIAVI PRIMARIE sono scritte interamente in maiuscolo, le Foreign Key hanno la prima o le prime due lettere maiuscole):

- Azienda (ID_AZIENDA, denominazione)
- Automezzo (TARGA, descrizione, ID_azienda)
- Autista (ID_AUTISTA, nome, cognome, eta, stipendio, ID_azienda)
- Viaggi (ID_autista, Targa, ore_lavorate, km_percorsi)

La tabella azienda contiene informazioni su alcune aziende che gestiscono automezzi per il trasporto di merci, la tabella Automezzo riparta informazioni sugli automezzi utilizzati. Ogni automezzo appartiene ad una ed una sola azienda e l'attributo ID_azienda è una chiave esterna che permette di collegare ogni Automezzo all'azienda proprietaria. La tabella Autista memorizza informazioni sugli autisti degli automezzi, la tabella Viaggi contiene informazioni sui viaggi fatti dagli autisti con gli automezzi. Ogni autista lavora per una ed una sola azienda, individuata dalla chiave esterna ID_azienda.

Scrivete un'unica query SQL che mostri per ogni Azienda il rapporto tra i km percorsi e le ore lavorate dagli autisti.
Nella query devono essere considerati solo i viaggi svolti dagli autisti che hanno più di 50 anni (>50).
Devono essere visualizzati solo i dati delle aziende che hanno un rapporto (km percorsi su ore lavorate) maggiore di 70 (>70) e i dati devo essere visualizzati in ordine decrescente rispetto al rapporto.
Per ogni azienda, oltre al rapporto descritto sopra, devono essere visualizzati l'ID_azienda e la denominazione.

Soluzione

```
select az.id_azienda, az.denominazione, sum(v.km_percorsi)/sum(v.ore_lavorate) as rap
from Viaggi as v, Autista as aut, Azienda as az
where v.id_autista=aut.id_autista and aut.id_azienda=az.id_azienda and aut.eta>50
group by az.id_azienda
having rap>70
order by rap DESC;
```