Domini degli attributi

Domanda 1

Date le seguenti dichiarazioni

```
stato BOOLEAN,
stato1 CHAR(1)
```

e gli assegnamenti

```
stato = 'f'
stato1 = 'f'
```

Indicare che cosa contengono i due attributi a seguito degli assegnamenti.

Risposta

L'attributo stato rappresenta il valore falso, l'attributo stato1 rappresenta il carattere 'f'. Solo il primo può essere usato per rappresentare il valore falso in una clausola WHERE.

Domanda 2

Indicare qual è la dichiarazione SQL esatta per l'attributo importo che deve rappresentare un valore in Euro.

Risposta

```
importo DECIMAL(8,2);
```

Domanda 3

Che cosa rappresenta la dichiarazione:

```
importo NUMERIC(10)
```

Risposta

Un attributo che può assumere solo valori interi per un massimo di 10 cifre.

Domanda 4

In PostgreSQL, data la dichiarazione

```
importo NUMERIC(5,2)
```

cosa succede se si assegna il valore 100.101 a importo?

Risposta

importo assume valore 100.10.

Domanda 5

Data la dichiarazione

```
nome CHAR(5)
```

Cosa succede se in un UPDATE si scrive

```
SET nome = '100.101'?
```

Risposta

Si verifica un errore perché il numero totale di caratteri di '100.101' è 7 mentre nome può rappresentare max 5 caratteri.

Domanda 6

Data la dichiarazione

```
nome CHAR(5)
```

Cosa succede se in un UPDATE si scrive

```
SET nome = '100'?
```

Risposta

nome assume valore '100' e viene rappresentato internamente come '100__' dove _ rappresenta uno spazio.

Domanda 7

Data le dichiarazioni

```
nome CHAR(80),
nome1 VARCHAR(80)
```

e gli assegnamenti

```
nome = 'Università          ' (Ci sono 10 caratteri spazio dopo 'à')
nome1 = 'Università     ' (Ci sono 5 caratteri spazio dopo 'à')
```

Qual è lo spazio di memorizzazione usato per rappresentare i valori e cosa ritorna l'istruzione

```
SELECT nome = nome1;?
```

Risposta

nome richiede l'equivalente dei 80 caratteri in byte + 1 byte, nome1 l'equivalente dei 15 caratteri in byte + 1 byte. SELECT ritorna vero.

Gestione Valori Nulli

Domanda 1

Si assuma che durante una selezione gli attributi a and b siano entrambi NULL e che l'attributo c abbia valore 100.

Come è valutata l'espressione `(c = 100 and a = b)`?

Risposta

L'espressione ritorna null, perché l'espressione null = null ritorna null, e c = 100 and null ritorna null.

Domanda 2

Si assuma che ci siano le tabelle A e B così costituite:

tabella A

a
1
null

tabella B

b
null
1

Si consideri quindi la seguente query

```
select *
from A JOIN B on a=b;
```

Qual è il risultato della query?

Risposta

a	b
1	1

Domanda 3

Si assuma che ci siano le tabelle A e B così costituite:

tabella A

a
1
null

tabella B

b
null
1

Si consideri quindi la seguente query

```
SELECT a
FROM A
WHERE a > ANY (
      SELECT * FROM B
);
```

Risposta

La query ritorna una tabella vuota (nessuna riga).

Definizione tabelle

Domanda 1

Si vuole vincolare l'attributo città presente in diverse tabelle ad assumere un valore tra un insieme di possibili valori. Qual è la dichiarazione più adatta per rappresentare tale vincolo e che non appesantisca le interrogazioni sull'attributo.

Risposta

Dichiarare un dominio

```
CREATE DOMAIN dom_città AS VARCHAR
      CHECK(value IN ('Arezzo', 'Verona',...))
```

e poi dichiarare l'attributo come

```
città dom_città;
```

In questo modo il valore dell'attributo è immediatamente disponibile in ogni interrogazione che coinvolga l'attributo città.

Alternative:

a) Non usare un dominio, ma indicare direttamente il vincolo CHECK nella definizione dell'attributo città: poiché l'attributo è usato in diverse tabelle, devo esplicitare il vincolo check ogni volta.
b) Non usare un dominio, ma usare una tabella a parte ed un vincolo di integrità referenziale: in questo modo dovrei fare un join ogni volta per recuperare il valore, poco efficiente.

Domanda 2

Si consideri la seguente dichiarazione in PostgreSQL:

```
CREATE TABLE A (
  id INTEGER UNIQUE NOT NULL,
  nome VARCHAR NOT NULL,
  prezzoUnitario DECIMAL(6,2),
  quantita NUMERIC(5,2)
)
```

Scrivere un vincolo per garantire che i due valori numerici siano non negativi.

Risposta

```
alter table A add constraint nonneg_check
      check (prezzoUnitario>=0 and quantita>=0);
```

Domanda 3

Si consideri la seguente dichiarazione:

```
CREATE TABLE A (
  id INTEGER PRIMARY KEY,
  nome VARCHAR,
  prezzoUnitario DECIMAL(6,2),
  quantita NUMERIC(5,2)
)
```

Si vuole garantire che tutti i valori di ogni tupla siano significativi e che i valori numerici (id escluso) siano non negativi. Quali vincoli vanno aggiunti?

Risposta

```
alter table A add check
      (nome is not null and
       prezzoUnitario is not null and
       quantita is not null and
       prezzoUnitario>=0 and
       quantita>=0);
```

Definizione vincoli attributo

Domanda 1

Indicare la dichiarazione di un attributo che permetta di rappresentare stringhe di lunghezza massima pari a 256 caratteri su cui non sono ammessi valori nulli e con valore di default 'non noto'.

Risposta

```
nomeAttributo VARCHAR(256) NOT NULL DEFAULT 'non noto'
```

Domanda 2

Indicare la dichiarazione di un attributo che permetta di rappresentare razionali positivi con 3 cifre decimali esatte e con valore di default 10%.

Risposta

```
nomeAttributo DECIMAL(10,3) DEFAULT '0.1'::decimal
      CHECK nomeAttributo >= 0)
```

Domanda 3

Indicare la dichiarazione che permetta di rappresentare un solo vincolo che rappresenti la proprietà:

quantità non negativa e prezzo * quantità < 100 quando quantità non è NULL.

Risposta

```
quantità DECIMAL,
prezzo DECIMAL,
CHECK(quantità >= 0 AND quantità*prezzo < 100)
```

Domanda 4

Indicare la dichiarazione corretta che permetta di rappresentare valori per la coppia (quantità, prezzo) che siano sempre significativi e distinti.

Risposta

```
quantità DECIMAL NOT NULL,
prezzo DECIMAL NOT NULL,
UNIQUE(quantità, prezzo)
```

Domanda 5

Indicare la dichiarazione che permetta di rappresentare la chiave primaria sulla coppia di attributi `(id, stato)` e una super chiave sulla terna `(nome,marca,stato)`.

Risposta

```
id INTEGER,
stato CHAR(3),
nome VARCHAR(64) NOT NULL,
marca CHAR(10) NOT NULL,
PRIMARY KEY(id, stato),
UNIQUE(stato,nome,marca)
```

Domanda 6

Indicare la dichiarazione che permetta di rappresentare una chiave non primaria (nome, cognome).

Risposta

```
nome VARCHAR NOT NULL,
cognome VARCHAR NOT NULL,
UNIQUE(nome, cognome)
```

Definizione vincoli interrelazionali

Domanda 1

Indicare la dichiarazione che consenta di rappresentare il vincolo di integrità referenziale: A.nome ⟶ B.marca.

Risposta

Nella tabella B: `marca VARCHAR UNIQUE`

Nella tabella A: `nome VARCHAR REFERENCES B(marca)`

Domanda 2

Indicare la dichiarazione che consenta di rappresentare il vincolo di integrità referenziale: (A.nome, A.id) ⟶ (B.marca, B.id).

Risposta

Nella tabella B:

```
marca CHAR(10) NOT NULL,
id SERIAL NOT NULL,
UNIQUE(marca,id)
```

Nella tabella A:

```
nome VARCHAR,
id INTEGER,
FOREIGN KEY (nome,id) REFERENCES B(marca,id)
```

Domanda 3

Si consideri la seguente sequenza di dichiarazioni. Perché PostgreSQL ritorna un errore?

```
CREATE TABLE A (
  id INTEGER UNIQUE NOT NULL,
  nome VARCHAR NOT NULL,
  magazzino INTEGER REFERENCES B,
  prezzo DECIMAL
)
CREATE TABLE B (
  id INTEGER PRIMARY KEY,
  nome VARCHAR NOT NULL,
  luogo VARCHAR
);
```

Risposta

Perché la definizione della tabella B deve essere posta prima della definizione della tabella A, essendoci un riferimento (tramite vincolo di chiave esportata) dalla tabella slave A verso la tabella master B.

Indici

Domanda 1

Data l'istruzione PostgreSQL

```
CREATE INDEX i1 ON spese USING HASH(id)
```

a) Spiegare in che tipo di espressioni può essere usato l'indice i1 se id è di tipo INTEGER.
b) Se id fosse di tipo VARCHAR(80) cosa cambierebbe?

Risposta

a) i1 è un indice che può essere usato solo quando ci sono selezioni del tipo id = espressione.
b) L'utilizzo di i1 non cambia, può essere usato solo in selezioni che testano uguaglianza stretta.

Domanda 2

Data l'istruzione PostgreSQL

```
CREATE INDEX i1 ON spese USING B-TREE (id)
```

Spiegare in che tipo di espressioni può essere usato l'indice i1 se id è di tipo INTEGER.

Risposta

i1 è un indice che può essere usato quando ci sono selezioni che coinvolgono id con gli operatori di confronto <, <=, =, >=, >; e può essere considerato anche in presenza degli operatori BETWEEN, IN, IS NULL, IS NOT NULL e LIKE.

Domanda 3

In una clausola WHERE è presente l'espressione

```
nome = 'Carlo' OR lower(cognome) < 'C'
```

dove nome e cognome sono due attributi con dominio VARCHAR(80) della tabella Persona. Che tipo di indici si potrebbero creare per aumentare le prestazioni della query?

Risposta

- `CREATE INDEX i1 ON Persona(nome varchar_pattern_opts)`
- `CREATE INDEX i2 ON Persona(lower(cognome) varchar_pattern_opts)`

Domanda 4

In una clausola WHERE è presente l'espressione

```
matricola = 'VR00001' AND voto = 30
```

dove matricola ha dominio VARCHAR(10) e voto ha dominio INTEGER. Che tipo di indici si potrebbero creare per aumentare le prestazioni della query?

Risposta

```
CREATE INDEX i1 ON esame(matricola, voto)
```

oppure due indici distinti, ma con prestazioni inferiori.

Domanda 5

Data l'istruzione

```
CREATE INDEX i1 ON spese(nome, cognome varchar_pattern_opts)
```

dove entrambi nome e cognome hanno dominio VARCHAR(80).

a) In the tipo di selezioni può essere usato l'indice?
b) Cosa cambia se non si indica l'opzione varchar_pattern_opts?

Risposta

a) i1 è un indice che può essere usato per risolvere selezioni del tipo nome = espressione1 AND cognome = espressione2.
b) In mancanza di varchar_pattern_opts, i confronti possono essere eseguiti solo tramite l'operatore di uguaglianza stretta = se il locale è it_IT.UTF-8, oppure si possono eseguire qualsiasi tipo di confronto con un locale diverso.

Query

Domanda 1

Data la tabella

```
INGREDIENTE_IN_RICETTA (
  ricetta_id INTEGER REFERENCES Ricetta(id),
  ingrediente_id INTEGER REFERENCES Ingrediente(id),
  quantita DECIMAL(4,2),
  PRIMARY KEY (ricetta_id, ingrediente_id)
)
```

Si scriva la query più efficiente che, per ogni ingrediente che è contenuto in almeno 3 ricette distinte visualizzi il suo identificatore e la quantità massima con cui tale ingrediente compare.

Risposta

```
SELECT ingrediente_id, max(quantita)
FROM INGREDIENTE_IN_RICETTA
GROUP BY ingrediente_id
HAVING count(ricetta_id)>=3;
```

Domanda 2

Data la tabella

```
PROVE_DI_ESAME (
  matricola INTEGER,
  inserogato INTEGER,
  voto DECIMAL(4,2),
  data DATE,
  PRIMARY KEY( matricola, inserogato, data )
)
```

Si scriva la query più efficiente che per ogni appello (rappresentato dalla coppia inserogato e data) mostri il numero di studenti che hanno ottenuto un voto maggiore o uguale di 18.

Risposta

```
SELECT inserogato, count(*)
FROM PROVE_DI_ESAME
WHERE voto >= 18
GROUP BY inserogato, data
```

Domanda 3

Data la tabella

```
PROVE_DI_ESAME (
  matricola INTEGER,
  inserogato INTEGER,
  voto DECIMAL(4,2),
  data DATE
)
```

Si scriva la query che, per ogni studente con almeno 3 prove svolte nel 2019, visualizzi la media dei voti ottenuti.

Risposta

```
SELECT matricola, AVG(voto)
FROM PROVE_DI_ESAME
WHERE EXTRACT(YEAR FROM data)=2019
GROUP BY matricola
HAVING count(*)>=3;
```

Esercizio 4

Data la tabella

```
ESAME(
  matricola INTEGER,
  inserogato INTEGER,
  voto DECIMAL(4,2),
  data DATE
)
```

Si scriva la query che, per ogni studente con almeno 3 esami di insegnamenti diversi, visualizzi la media dei voti ottenuti.

Risposta

```
SELECT matricola, AVG(voto)
FROM ESAMI
GROUP BY matricola
HAVING count(distinct inserogato)>=3;
```

Domanda 5

Data la tabella

```
PROVA (
  matricola INTEGER,
  id_competizione INTEGER,
  punteggio DECIMAL(4,2),
  data DATE
)
```

Si scriva la query che, per ogni atleta che ha partecipato ad almeno 3 competizioni diverse, visualizzi il massimo dei punteggi ottenuti.

Risposta

```
SELECT matricola, max(punteggio)
FROM prova
GROUP BY matricola
HAVING count(distinct id_competizione)>=3;
```

Domanda 6

Data la tabella:

```
InsErogato(id, annoaccademico, id_insegn, id_corsostudi, id_discriminante,
modulo, discriminantemodulo, nomemodulo, crediti, programma, id_facolta,
hamoduli, id_inserogato_padre, nomeunità, annierogazione)
```

Scrivere una query che determina gli insegnamenti erogati con modulo = 0 che hanno un numero di crediti minori della media dei crediti degli insegnamenti erogati con modulo = 0 del medesimo corso di studi.

Risposta

```
SELECT ie.id, ie.crediti
    FROM inserogato ie
        JOIN (
            SELECT avg(crediti),id_corsostudi
            FROM inserogato
            WHERE modulo=0
            GROUP BY id_corsostudi
        ) AS v1(crediti,cs) ON id_corsostudi=cs
WHERE ie.modulo=0 AND ie.crediti < v1.crediti;
```

Alternativa corretta, ma meno efficiente perché richiede il data- binding.

```
SELECT ie.id, ie.crediti
FROM inserogato ie
WHERE ie.modulo=0 AND ie.crediti < (SELECT avg(crediti)
   FROM inserogato
   WHERE id_corsostudi = ie.id_corsostudi AND modulo = 0);
```

Domanda 7

Data la tabella:

```
InsErogato(id, annoaccademico, id_insegn, id_corsostudi, id_discriminante,
modulo, discriminantemodulo, nomemodulo, crediti, programma, id_facolta,
hamoduli, id_inserogato_padre, nomeunità, annierogazione)
```

Scrivere una query che permette di determinare gli insegnamenti erogati da qualsiasi corso di studi con hanno numero di crediti pari al massimo dei crediti degli insegnamenti erogati del corso di studi con id = 4.

Risposta

```
SELECT ie.id, ie.crediti
FROM inserogato ie
WHERE ie.crediti >= ALL
  (SELECT crediti
   FROM inserogato
   WHERE id_corsostudi = 4)
```

Domanda 8

Date le tabelle:

```
Persona(id, cognome, nome, sesso, telefono, email)
Docenza(id, id_inserogato, id_persona, id_settore, creditilez, orelez)
InsErogato(id, annoaccademico, id_insegn, id_corsostudi, id_discriminante,
           modulo, discriminantemodulo, nomemodulo, crediti, programma,
           id_facolta, hamoduli, id_inserogato_padre, nomeunità, annierogazione)
```

Scrivere una query che determina i docenti che hanno insegnato nell'anno accademico 2010/2011 per un numero totale di crediti di lezione maggiore o uguale al totale dei crediti di lezione tenuti dallo stesso docente in tutti gli anni accademici diversi da 2010/2011.

Risposta

```
CREATE VIEW TOTALI AS (
  SELECT D.id_persona, IE.annoAccademico, SUM(d.creditilez) as totaleCrediti
  FROM DOCENZA AS D
  JOIN INSEROGATO AS IE ON IE.id=D.id_inserogato
```

```
  WHERE D.creditilez > 0
  GROUP BY D.id_persona, IE.annoAccademico)

SELECT docente, totaleCrediti
FROM TOTALI AS T
WHERE annoAccademico = '2010/2011' AND
  ROW(docente, totaleCrediti) IN (
    SELECT id_persona, max(totaleCrediti)
    FROM TOTALI
    WHERE annoAccademico <> '2010/2011'
    GROUP BY docente
);
```

Alternativa

```
SELECT docente, totaleCrediti
FROM( SELECT d.id_persona, SUM(d.creditilez)
      FROM DOCENZA AS D
      JOIN INSEROGATO AS IE ON IE.ID=D.ID_INSEROGATO
      WHERE IE.annoAccademico ='2010/2011' AND D.creditiLez > 0
      GROUP BY D.ID_persona ) AS TOTALI(docente, totaleCrediti) T
WHERE totaleCrediti >= ALL (
  SELECT SUM(d.creditilez)
  FROM DOCENZA AS D
  JOIN INSEROGATO AS IE ON IE.ID=D.ID_INSEROGATO
  WHERE IE.annoAccademico <>'2010/2011' AND D.creditiLez > 0
        AND D.id_persona = T.docente
  GROUP BY IE.annoAccademico
);
```

Domanda 9

Date le tabelle:

```
Docenza(id, id_inserogato, id_persona, id_settore, creditilez, orelez)
InsErogato(id, annoaccademico, id_insegn, id_corsostudi, id_discriminante,
           modulo, discriminantemodulo, nomemodulo, crediti, programma,
           id_facolta, hamoduli, id_inserogato_padre, nomeunità, annierogazione)
```

Scrivere una query che determina i docenti che hanno insegnato per un numero totale di crediti di lezione maggiore o uguale di quello dei colleghi limitando l'analisi all'anno accademico 2010/2011.

Risposta

```
SELECT docente, totaleCrediti
  FROM( SELECT d.id_persona, SUM(d.creditilez)
        FROM DOCENZA AS D
        JOIN INSEROGATO AS IE ON IE.ID=D.ID_INSEROGATO
```

```
        WHERE IE.ANNOACCADEMICO='2010/2011' AND D.CREDITILEZ > 0
        GROUP BY D.ID_persona ) AS TOTALI(docente, totaleCrediti)
  WHERE totaleCrediti >= ALL (
    SELECT SUM(d.creditilez)
    FROM DOCENZA AS D
    JOIN INSEROGATO AS IE ON IE.ID=D.ID_INSEROGATO
    WHERE IE.ANNOACCADEMICO='2010/2011' AND D.CREDITILEZ > 0
    GROUP BY D.ID_persona
  );
```

Domanda 10

Si consideri il seguente schema relazionale (chiavi primarie sottolineate):

```
CLIENTE(codiceFiscale, nome, cognome, telefono, dataIscrizione)
TESSERA(numero, idCliente, idSupermercato, dataInizio, punti)
SUPERMERCATO(id, nome, indirizzo, citta)
```

dove sono definite le seguenti chiavi esportate TESSERA.idCliente → CLIENTE e TESSERA.idSupermercato → SUPERMERCATO. L'attributo dataInizio è di tipo TIMESTAMP mentre l'attributo durata è di tipo INTERVAL.

Scrivere l'interrogazione in SQL per PostgreSQL che seleziona per ogni utente che ha accumulato un numero di punti totale (= su tutte le tessere in suo possesso) superiore alla media dei punti totali accumulati dagli utenti, il numero di punti raggruppati per nome del supermercato. Si noti che un cliente potrebbe avere più tessere per ciascun supermercato. Il risultato deve riportare il codice fiscale, il nome e cognome dell'utente, il numero della tessera, il nome del supermercato ed il numero di punti.

Risposta

Una vista che prepari i conteggi può essere di aiuto.

```
CREATE VIEW puntiPerCliente AS
SELECT idCliente, sum(punti) as puntiTotali
FROM tessera
GROUP BY idCliente;
```

Alla query si risponde quindi selezionando le righe di interesse nella vista.

```
SELECT c.codiceFiscale, c.nome, c.cognome,
       sum(t.punti) as punti, s.nome as supermercato
FROM puntiPerCliente p JOIN cliente c ON (p.idCliente = c.codiceFiscale)
JOIN tessera t ON (t.idCliente = p.idCliente)
JOIN supermercato s ON (t.idSupermercato = s.id)
WHERE (p.puntiTotali) >ANY (
        SELECT avg(puntiTotali)
        FROM puntiPerCliente
)
GROUP BY c.codiceFiscale, s.id;
```

Domanda 11

Si consideri il seguente schema relazionale (chiavi primarie sottolineate) contenente le formulazioni di prodotti di una pasticceria:

```
INGREDIENTE(nome, calorie, grassi, proteine, carboidrati, vegano, glutine)
COMPOSIZIONE(prodotto, ingrediente}, quantità)
PRODOTTO(nome, porzioni, tempoPreparazione)
```

dove vegano e glutine sono booleani e tempoPreparazione è di dominio INTERVAL.

Scrivere l'interrogazione in SQL per PostgreSQL che determina nell'insieme dei prodotti vegani e nell'insieme dei prodotti senza glutine, i prodotti con la massima quantità totale di proteine.

La tabella risultato deve riportare nome prodotto, totale di grassi, di proteine, di carboidrati, se è 'vegano' (colonna boolean) e se è 'senza glutine' (colonna boolean).

Suggerimento: due operatori di aggregazione per i booleani sono bool_and(espressione) e bool_or(espressione).

Si assuma che quantità*grassi dia il totale di grassi apportati dall'ingrediente al prodotto nella unità di misura richiesta(analogo per proteine e carboidrati).

Risposta

Creare una vista

```
CREATE VIEW valoriNutrizionali AS
SELECT r.nome, sum(i.grassi*c.quantità) as grassitotali,
       sum(i.proteine*c.quantità) as proteinetotali,
       sum(i.carboidrati*c.quantità) as carboidratitotali,
       bool_and(i.vegano) as vegano, bool_or(glutine) as conGlutine
FROM prodotto r JOIN composizione c ON r.nome=c.prodotto
JOIN ingrediente i ON c.ingrediente=i.nome
GROUP BY r.nome;
```

E poi calcolare:

```
SELECT nome, grassitotali, proteinetotali, carboidratitotali, vegano,
       not(conGlutine) as "senza Glutine"
FROM valoriNutrizionali as v
WHERE vegano is true AND proteinetotali IN (
      SELECT max(proteinetotali) FROM valoriNutrizionali WHERE vegano is true
      )
UNION
SELECT nome, grassitotali, proteinetotali, carboidratitotali, vegano,
       not(conGlutine) as "senza Glutine"
FROM valoriNutrizionali as v
WHERE conGlutine is false AND proteinetotali IN (
      SELECT max(proteinetotali) FROM valoriNutrizionali
      WHERE conGlutine is false
);
```

Domanda 12

Si consideri il seguente schema relazionale (chiavi primarie sottolineate, * per campo opzionale):

```
UTENTE(cf, nome, cognome, telefono, dataIscrizione, stato)
PRESTITO(idRisorsa, idBiblioteca, idUtente, dataInizio, durata)
RISORSA(id, biblioteca, titolo, tipo, stato)
```

dataInizio è TIMESTAMP e durata è INTERVAL.

Scrivere l'interrogazione in SQL per PostgreSQL che trova per ogni utente che abbia fatto un numero prestiti conclusi alla data odierna superiore alla media di tutti i prestiti conclusi, il numero di prestiti terminati e la loro durata totale. Il risultato deve riportare il codice fiscale dell'utente e i conteggi richiesti.

Risposta

Una vista che prepari i conteggi può essere di aiuto.

```
CREATE VIEW prestitiPerUtente AS
SELECT U.codiceFiscale AS idUtente, count(*) as nPrestiti, sum(durata) as durata
FROM Utente U JOIN PRESTITO P ON U.codicefiscale=P.idUtente
WHERE durata is not null AND (p.dataInizio + durata)::date < CURRENT_DATE
GROUP BY U.codiceFiscale;
```

Alla query si risponde quindi selezionando le righe di interesse nella vista.

```
SELECT idUtente, nPrestiti, durata
FROM prestitiPerUtente p
WHERE (nPrestiti) >ANY (
      SELECT max(nPrestiti)
      FROM prestitiPerUtente
);
```

Transazioni

Domanda 1

Si consideri il seguente schema relazionale (chiavi primarie sottolineate):

```
PRODOTTO(codice, nome, produttore, prezzo)
PRODUTTORE(codice, nome, nazione)
SOSTANZA(id, nome, categoria)
```

Si scrivano due transazioni in codice PostgreSQL dove:

- nella prima transazione si determina la somma dei prezzi di tutti i prodotti dei produttori con nazione diversa da 'Italia', e poi, si determina con un'istruzione separata, la somma dei prezzi di tutti i prodotti dei produttori con nazione diversa da 'Italia' raggruppati per nazioni.
- nell'altra si introducono dei nuovi prodotti (si assume che siano forniti da produttori che non stanno in Italia).

Notazione: si scriva VALUES (..),(..).. per rappresentare i valori che devono essere inseriti.

Che tipo di errore si può verificare se le due transazioni sono concorrenti? Che tipo di isolamento minimo si deve dichiarare in PostgreSQL per eliminare l'errore? In quale transazione?

Risposta

Possono verificarsi dei casi di inserimento fantasma. Se le transazioni iniziano insieme e, poi, l'insert e commit della seconda transazione viene effettuato dopo la prima select della prima ma prima della seconda select, si può verificare che i parziali non siano coerenti con il totale determinato nella prima select.

```
--Transazione 1:
BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SELECT sum(prezzo)
FROM prodotto p JOIN produttore pr ON p.produttore=pr.id
WHERE p.nazione <> 'Italia';

SELECT pr.nazione, sum(prezzo)
FROM prodotto p JOIN produttore pr ON p.produttore=pr.id
WHERE p.nazione <> 'Italia'
GROUP BY pr.nazione;
COMMIT;

--Transazione 2:
BEGIN;
INSERT INTO prodotto VALUES (...)
COMMIT;
```

Domanda 2

Si consideri il seguente schema relazionale (chiavi primarie sottolineate):

```
FARMACO(codice, nomeFarmaco , principioAttivo, produttore, prezzo)
PRODUTTORE(codice, nome, nazione)
SOSTANZA(id, nomeSostanza, categoria)
```

Si scrivano due transazioni in codice PostgreSQL cui:

- nella prima transazione si aumenta del 5% il prezzo a tutti i farmaci dei produttori con nazione diversa da 'Italia',
- nell'altra si abbassa il prezzo del 10% a tutti i farmaci che hanno prezzo inferiore a 1050 euro.

Che tipo di errore si può verificare se le due transazioni sono concorrenti? Che tipo di isolamento si deve dichiarare per eliminare l'errore? In quale transazione?

Risposta

Possono verificarsi dei casi di non-repeatable read. Se le transazioni iniziano insieme e, poi, l'update della seconda transazione viene effettuato dopo l'update della prima ma prima del commit della prima transazione, si può verificare il caso che un prezzo che era a 1000 diventi 1050 ma non è letto nella seconda transazione.

```
--Transazione 1:
BEGIN
UPDATE FARMACO SET prezzo = prezzo * 1.05
WHERE produttore IN
    (SELECT codice FROM Produttore p WHERE p.nazione <> 'Italia');
COMMIT;

--Transazione 2:
BEGIN TRANSACTION ISOLATION LEVEL REPEATABLE READ;
UPDATE FARMACO SET prezzo = prezzo * 0.9 WHERE prezzo = 1050;
COMMIT;
```

Connessione con Python e/o Java

Domanda 1

Si assuma che una base di dati in PostgreSQL contenga la tabella

```
Prodotti(id,nome, prezzo, dataUltimaModifica)
```

scrivere il codice di un programma Python che visualizza i prodotti (id, nome, prezzo) modificati nella settimana precedente alla data corrente e che poi li aggiorni aumentando il prezzo del 10%.

Risposta

```
with psycopg2.connect(database="migliorini", user='migliorini') as conn:
      conn.set_session(isolation_level='REPEATABLE READ')
      with conn.cursor() as cur:
            cur.execute('select id, nome, prezzo from prodotto where
dataUltimaModifica>=(CURRENT_DATE-7)')
            lista = cur.fetchall()
            if len(lista) > 0:
                  for row in lista:
                        print(row[0], row[1], row[2])
                  cur.execute('update prodotto set prezzo=prezzo*1.10 where
dataUltimaModifica>=(CURRENT_DATE-7)')
            else:
                  print('Non ci sono prodotti da aggiornare.')
conn.close()
```

Piano di esecuzione

Domanda 1

Si consideri il seguente piano di esecuzione

```
HashAggregate(cost=5398.33..5446.58 rows=4825 width=60)
  Group Key: i.nomeins, d.descrizione
  -> Hash Join  (cost=407.27..5374.20 rows=4825 width=60)
       Hash Cond: (ie.id_insegn = i.id)
       -> Hash Join  (cost=127.46..5028.06 rows=4825 width=25)
            Hash Cond: (ie.id_discriminante = d.id)
            -> Bitmap Heap Scan on inserogato ie (cost=122.40..4945.72 rows=7740 width=8)
                 Recheck Cond: ((annoaccademico)::text = '2009/2010'::text)
                 -> Bitmap Index Scan on ie_aa_it (cost=0.00..120.47 rows=7740 width=0)
                      Index Cond: ((annoaccademico)::text = '2009/2010'::text)
            -> Hash (cost=3.36..3.36 rows=136 width=25)
                 -> Seq Scan on discriminante d (cost=0.00..3.36 rows=136 width=25)
       -> Hash (cost=177.69..177.69 rows=8169 width=43)
            -> Seq Scan on insegn i (cost=0.00..177.69 rows=8169 width=43)
```

Supponendo che i dati non varino nel tempo e che le chiavi primarie hanno anche un indice hash, in base al piano di esecuzione conviene creare degli altri indici? Se sì, quali?

Risposta

I seq scan presenti servono per costruire delle hash table.
Visto però che i seq scan sono fatti su chiavi per le quali ci sono già gli indici, la risposta è no.

Domanda 2

Dato il seguente query plan:

```
 Aggregate  (cost=2.15..2.16 rows=1 width=16)
   ->  Nested Loop  (cost=0.00..2.15 rows=1 width=17)
         Join Filter: ((l.persona)::bpchar = (p.codfiscale)::bpchar)
         ->  Seq Scan on lezione l  (cost=0.00..1.07 rows=1 width=34)
               Filter: (durata > '01:00:00'::interval)
         ->  Seq Scan on persona p  (cost=0.00..1.04 rows=3 width=17)
               Filter: (datanascita > '1970-12-31'::date)
```

È possibile ottimizzare la query considerando solo le statistiche correnti del piano di esecuzione? Se sì, come? Se no, perché?

Risposta

No.

Entrambi i seq scan selezionano poche righe usando mediamente meno di 2 accessi al disco: è praticamente impossibile che un indice possa diminuire tale tempo di esecuzione.

Domanda 3

Si consideri il seguente piano di esecuzione (Piano1)

```
Hash Join  (cost=289.14..716.26 rows=150 width=47)
  Hash Cond: (ie.id_insegn = i.id)
  -> Hash Join (cost=9.34..434.39 rows=150 width=12)
       Hash Cond: (ie.id_discriminante = d.id)
       -> Bitmap Heap Scan on inserogato ie  (cost=4.28..426.93 rows=240 width=12)
             Recheck Cond: ((annoaccademico)::text = '2014/2015'::text)
             -> Bitmap Index Scan on ie_aa_it (cost=0.00..4.22 rows=240 width=0)
                  Index Cond: ((annoaccademico)::text = '2014/2015'::text)
       -> Hash (cost=3.36..3.36 rows=136 width=4)
             -> Seq Scan on discriminante d (cost=0.00..3.36 rows=136 width=4)
  -> Hash (cost=177.69..177.69 rows=8169 width=43)
       -> Seq Scan on insegn i  (cost=0.00..177.69 rows=8169 width=43)
```

e il seguente (Piano2)

```
Nested Loop  (cost=9.62..632.14 rows=150 width=47)
  -> Hash Join  (cost=9.34..434.39 rows=150 width=12)
       Hash Cond: (ie.id_discriminante = d.id)
       -> Bitmap Heap Scan on inserogato ie (cost=4.28..426.93 rows=240 width=12)
             Recheck Cond: ((annoaccademico)::text = '2014/2015'::text)
             -> Bitmap Index Scan on ie_aa_it  (cost=0.00..4.22 rows=240 width=0)
                   Index Cond: ((annoaccademico)::text = '2014/2015'::text)
       -> Hash (cost=3.36..3.36 rows=136 width=4)
             -> Seq Scan on discriminante d  (cost=0.00..3.36 rows=136 width=4)
  -> Index Scan using pippo on insegn i  (cost=0.28..1.31 rows=1 width=43)
       Index Cond: (id = ie.id_insegn)
```

Quali indici sono stati creati per permettere il passaggio da Piano 1 a Piano 2? Qual è il guadagno ottenuto?

Risposta

Nel Piano2 è stato usato anche l'indice creato con il comando

```
CREATE INDEX pippo ON insegn USING hash(id);.
```

Il guadagno ottenuto è di 82 accessi disco circa.

Domanda 4

Si consideri il seguente piano di esecuzione (Piano1)

```
Nested Loop  (cost=4249.15..5612.14 rows=684 width=92)
  -> Hash Join (cost=4249.15..5556.90 rows=684 width=12)
       Hash Cond: (d.id_inserogato = ie.id)
       -> Seq Scan on docenza d (cost=0.00..1266.84 rows=9088 width=8)
            Filter: (id_persona < 500)
       -> Hash (cost=4185.12..4185.12 rows=5122 width=8)
            -> Bitmap Heap Scan on inserogato ie  (cost=80.11..4185.12 rows=5122 width=8)
                 Recheck Cond: ((annoaccademico)::text = '2013/2014'::text)
                   -> Bitmap Index Scan on ie_aa_it  (cost=0.00..78.83 rows=5122 width=0)
                        Index Cond: ((annoaccademico)::text = '2013/2014'::text)
  -> Index Scan using insegn_id_idx on insegn i (cost=0.00..0.07 rows=1 width=43)
       Index Cond: (id = ie.id_insegn)
```

e il seguente (Piano2)

```
Nested Loop (cost=4371.87..5212.63 rows=684 width=92)
  -> Hash Join  (cost=4371.87..5157.39 rows=684 width=12)
       Hash Cond: (d.id_inserogato = ie.id)
       -> Bitmap Heap Scan on docenza d (cost=122.72..867.32 rows=9088 width=8)
            Recheck Cond: (id_persona < 500)
            -> Bitmap Index Scan on docenza_id_persona_idx1 (cost=0.00..120.45 rows=9088
width=0)
                 Index Cond: (id_persona < 500)
       -> Hash (cost=4185.12..4185.12 rows=5122 width=8)
             ->  Bitmap Heap Scan on inserogato ie  (cost=80.11..4185.12 rows=5122
width=8)
                   Recheck Cond: ((annoaccademico)::text = '2013/2014'::text)
                   -> Bitmap Index Scan on ie_aa_it (cost=0.00..78.83 rows=5122 width=0)
                        Index Cond: ((annoaccademico)::text = '2013/2014'::text)
  -> Index Scan using insegn_id_idx on insegn i (cost=0.00..0.07 rows=1 width=43)
       Index Cond: (id = ie.id_insegn)
```

Quali indici sono stati creati per permettere il passaggio da Piano 1 a Piano 2? Qual è il guadagno ottenuto?

Risposta

Nel Piano2 è stato usato anche l'indice creato con il comando

```
CREATE INDEX docenza_id_persona_idx1 ON docenza(id_persona);.
```

Il guadagno ottenuto è di 400 accessi disco circa.