Cose

Scritte da Qualcuno

L'altro giorno

Contents

1 Viste Aggiornabili

cose scritte col create view per creare una vista che modifica lo schema esterno della base dati, dal punto di vista delle query questa vista è a tutti gli effetti una tabella della base, risulta in qualche modo la domanda, posso usare questa vista per modificare l'istanza della base dati.

posso fare set, oltre a get, di elementi della tabella? dal punto di vista formale, una vista è una funzione che estrae roba da una tabella

$vista = F(tabella)$

se posso determinare in maniera univoca qual'è l'aggiornamento della base dati che implica quell'aggiornamento della vista, allora posso aggiornare la tabella, allora la vista può essere aggiornabile, perchè invertibile, perchè posso arrivare univocamente alla riga che mi ha dato quella cosa nella tabella

condizioni, vincoli, che valendo questi vincoli la vista è aggiornabile, non si pretende di individuare tutti i casi in cui è invertibile, ma si definiscono delle condizioni sufficienti per dire "ok la vista è x, quindi la vista è aggiornabile"

questi sono i limiti posti da dbms per considerare una vista come aggiornabile, quando le considerano aggiornabili

- assenza di group by
- assenza di funzioni aggregate
- assenza di distinct
- assenza di join, impliciti o espliciti

quello a cui questi join si riferiscono è della parte esterna della vista, quindi

- solo una tabella
- niente distinct
- niente aggregati
- niente raggruppamento

1.1 Esempio

supponiamo di avere una base dati con tabelle

- imp : codice, nome, sede, ruolo, stipendio

- sedi : sede, responsabile, città

la città dove vanno a lavorare gli impiegati dipende da dove è localizzata la sede c'è un vincolo di chiave

definiamo una vista per tutti gli impiegati che lavorano a bologna

```
create view impBo(...)
select I.*
from impiegato I, join sedi S on (I.sede = S.sede)
where S.città = 'Bologna'
```

questa vista non è aggiornabile, nella clausola `from` hai due tabelle, quindi c'è un join implicito, questa vista non è aggiornabile per sti criteri.

ma formuliamola con una query annidata,

```
where impiegato.sede in (select le sedi a bologna)
```

oppure

```
create view ImpBo(...)
select I.*
from impiegato I where I.sede in (select S.sede from Sedi where S.città = 'Bologna')
```

in questo caso possiamo vedere che la query annidante rispetta tutti i vincoli di aggiornaiblità, quindi sta query che da le stesse informazioni ora è aggiornabile

nel caso tu debba usare più tabelle, se la vuoi aggiornabile, devi vedere se puoi farla con select annidate

1.2 Aggiornare query

```
insert into ImpBo(...) values(<dati di impiegato>)
```

ok, ma questa è una vista virtuale, non esiste come tabella, quindi la tupla specificata dove mi va a finire? la vista non ha un istanza materializzata, la vista è un getter che si pensa figo, per avere che quel getter dia quella tupla in più la tupla deve andare a finire nella tabella `imp`, e abbiamo potuto determinare dove doveva andare la tupla apputno perchè la vista era definita da una query reversibile.

se io dopo questo inserimento in `imp` facessi `select * from ImpBo`, allora vedrei che l'impiegato è stato aggiunto.

facciamo un'altra query del tipo

```
insert into ImpBo(...) values(<dati di impiegato>)
```

ma mettiamo che stavolta il codice della sede dato sia un codice di sede che già esiste, ma non si trova a bologna, aggiungo agli impiegati di bologna un impiegato che lavora nella sede di milano. Questa cosa possiamo vedere che non funziona benissimo l'inserimento non rispetta la query che definisce la vista

il controsenso viene dal fatto che la tupla non rispetta la select di definizione della vista, quando recupero i contenuti della vista quei dati non ci sono, non rispettano il comportamento della vista.

quindi nella definizione di una vista aggiornabile si può mettere un check per inserimento e aggiornamento, `with check option`, devo vedere se quell'aggiornamento è compatibile con la definizione della vista, così quando cerco di inserire o modificare una vista.

Se il dato inserito nella vista sarà visibile attraverso la definizione (`select`) della vista allora va bene, se il dato inserito non sarà visibile attraverso la definizione (`select`) della vista allora errore, qui il tizio di milano non sarà visibile dalla tabella di "vedi quelli di bologna", quindi errore.

visto che una vista puoi definirla in vista di altre viste, puoi avere due tipi di `with check option`

- `with local check option`, in vista di questa vista
- `with cascaded check option`, in vista di questa vista e delle altre viste da cui è definita.

ci sono qundi dei criteri in base ai quali la vista è aggiornabile.

Passo primo da ricordare, l'eventuale riga che viene aggiunta viene in realtà memorizzata nella tabella di base sulla quale la vista è definita, non nella vista, che è una vista virtuale. Nel caso delle viste aggiornabili è poi importante vedere se l'inserimento è compatibile o meno con la select di definzione della vista, se voglio che ci sia consistenza, e che quindi i dati inseriti nella vista siano visibili dalla vista, devo definire la vista con `with check option`, che obbliga il dbms a controllare che l'aggiornamento funzoini.

se non metto la with check option allora o

- la vista è read only
- la vista farà un po' i cazzi suoi

2 Ora fa prove compitini

puoi partecipare al compitino anche se non hai superato algoritmi, ma il compitino vale solo fino a gennaio febbraio, quindi se non passi algoritmi per

sta sessione il compitino bellissimo, non ci fai niente.

se non hai dato algoritmi non puoi partecipare al preappello di dicembre, ad esempio

2.1 Prova compitino VII

2.1.1 Schema

- Policeman : code, name, surname, address, areaCode
- Fine : code, policeman, car, date, infraction, streetName, areaCode, cost
- Infraction : code, description
- Car : plateNumber, type, owner
- Owner : code, name, surname, address, areaCode
- AreaCode : code, cityName, provinceName

(nelle query usa CAP e areaCode come sinonimi)

2.1.2 Prima query

per ogni cap, il codice ed il numero di multe medie fatte al giorno (ammissione dal Pala, la definizione $\pm$ torna $\pm$ non torna tantissimo)

da calcolare un numero, e la media di quel numero, mi sono un po' perso, so che va fatta

1. Pala dixit

   crea una vista in cui per ogni cap, e per ogni giorno, si contano le multe fatte

   ```
   create view multeAreaData(areaCode, date, numero) as
   select areaCode, date, count(code)
   from fine
   group by areaCode, date

   select areaCode, avg(numero)
   from multeAreaData
   group by areaCode
   ```

qui visto che code è chiave primaria non cambia fare

- `count`
- `count *`
- `count distinct`

se vuoi calcolare anche le aree per cui non è mai stato fatta una multa fai left join di qualcosa, poi count viene 0 nella vista, e avg verrà 0 nella select. Mi sono perso i dettagli, non stavo seguendo molto, pardon.

2.1.3 Seconda query

Il cap dove sono state fatte il maggior numero di multe, insieme a quel numero di multe

qui farei una query per legare cap e multe

```
create view capMulte(cap, multe) as
select areaCode, count(code)
from fine
where areaCode is not null -- se vuoi evitare problemi quando non specificano areaCode
group by areaCode

select cap from capMulte where multe = (select max multe from capMulte)
```

1. Pala dixit una possibile soluzoine è calcolare il numero di multe fatte per cap, e poi vedere il cap per cui è massimale

   ```
   create view multeArea(areaCode, numero) as
   select areaCode, count(code)
   -- count(code) permetterà di dare 0 per gli areaCode che non hanno mai avuto multe
   from fine
   group by areaCode

   select areCode, numero
   from multeArea
   where numero = (select max(numero) from multeArea)
   ```

   altrimenti

   ```
   having count(multe) >= all(select count(multe)
   ```

```
...
(select max(pippo) -- rinominato in un momento di euforia (a detta del Pala in per
from (select count(code) as pippo
from fine
group by areaCode))
```

2.1.4 Terza query

per ogni tipo di infrazione, il codice ed il numero medio di multe fatte in un giorno, `fine.infraction` indica il tipo di multa.

(come prima, il numero medio di multe fatte in un giorno, assumendo che sia stata fatta una multa per ogni tipo di infrazione)

come al solito si divide in vista che definisce il coso di cui fare la media

```
create view contatore(inf, dat, numero) as
select infraction, date count(fine.code)
from infraction
group by code, date
```

e fare la media dalla vista poi si fa la media

```
select inf, avg(numero)
from contatore
group by inf
```

1. Pala dixit

   vanno contatae usando una vista, il numero per ogni giorno e per ogni infrazione, poi si fa la media al variare di cosa si raggruppa per infraction, al variare di date, quindi nella seconda query togli il group by date

   ```
   create view multeTipoData(infraction, data, numero) as
   select infraction, date, count(code)
   from fine
   group by infraction, date

   select infraction, avg(numero)
   from multeTipData
   group by infraction
   ```

2.1.5 Quarta query

codice e descrizione del tipo di infrazione per cui sono state emesse il maggior numero di multe, inseme a tale numero

creiamo una vista del codice massimo e numero

```
create view infMaxNum(inf, num) as
select infraction, count(code)
from fine
where count(code) >= all(select count(code) from fine group by infraction)
group by infraction
```

poi abbiamo il codice e bisogna fare il join con la tabella infraction per avere anche la descrizione, quindi la select vera e propia.

```
select inf,descrizione,num
from infMaxNum, infraction
where infMaxNum.inf = infraction.code
```

1. Pala dixit

```
-- non so perchè ma gli andava di usare la tabella definita prima
-- la cosa non è necessaria, te fai come vuoi

-- questa basta definirla come "per ogni infrazione il numero di multe fatte"
create view multeTipo(infraction, numero) as
select infraction, sum(numero)
from multeTipoData
group by infraction


select infraction, numero
from multeTipo
where numero = (select max(numero) from multeTipo)
-- oppure
where numero >= all(select numero from multeTipo)
-- molti gradi di libertà

create view multeTipo (infraction, description, numero) as
select i.code, i.description, count(f.code)
from  infraction left join fine f on i.code = f.infraction
group by i.code, i.description
```

ai fini di questa select il join normale andava bene se dovevo contare quando certi tipi di multe non hanno multe fatte, allora avrei fatto un join esterno per considerare anche quei tipi

di solito posso usare indifferentemente i cosi della condizione, quelle righe avranno i campi di infraction a valori non nulli mentre i corrispondenti campi di fine saranno nulli, quindi qeulla condizione non sarà verificata per le righe obbligate e non ho capito che diceva

poi la query principale sarà

```
select infraction, description, numero
from multeTipo
where numero = (select max(numero) from multeTipo)
```

2.1.6 Quinta query

codice, nome, e cognome di ogni vigile, e costo totale delle multe fatte, ordinando i valori descrescenti del costo

io avrei fatto questa

```
select p.code, p.name, p.surname, sum(f.cost) costo
from policeman p, fine f
where f.policeman = p.code
group by p.code, p.name, p.surname
order by sum(f.cost) desc
```

1. Pala dixit serve la tabella policeman per i dati, e fine per il costo

   ```
   select p.code, p.name, p.surname, sum(f.cost) totale
   from policeman p, fine f
   group by p.code, p.name, p.surname
   group by p.code, p.name, p.surname
   order by totale desc -- desc necessario perchè di default il criterio è asc, ascer
   ```

   errata da parte di Sergio Cibecchini del join esterno

   ```
   ... -- stesso di sopra
   from policeman p left join fine f on f.policeman = p.code
   ... --stesso di spora
   ```

2.2 Prova compitino IX

era un compito a due file, quindi le query sono identiche a due a due

2.2.1 Schema

(la descrizione oltre alle tabelle, di che significa ogni tabella, è un saggio, vatti a vedere te le slide, tanto questo è il pdf delle prove compiti che ha messo su moodle)

- **Paziente** : cf, nome, cognome, indirizzo, città, telefono
- **Medico** : cf, nome, cognome, indirizzo, città, telefono
- **PrenotazoineVisita** : data, ora, paziente, medico
- **Visita** : codice, data, paziente, medico, costo
- **DettaglioVisita** : visita, controllo, inNorma, descrizione
- **Prescrizione** : visita, farmaco, posologia
- **Controllo** : codice, nome
- **Farmaco** : codice, nome

2.2.2 Prima query

il numero delle visite al termine delle quali non è stato prescritto il farmaco F018

facciamo una vista di tutte le visite in cui...

```
create view visiteInCuiNon(visita)
select visita.codice
except
(select prescrizione.visita
where farmaco = F018)

select count(visita)
from visitaInCuiNon
```

se vuoi fare lo scemo fai

```
select (select count (*) from visita) -
(select count(*) from prescrizione where farmaco = F018)
as questaCosaFunziona -- ci ho provato nel database impiegati e a postgres va bene
```

poi la solita epopea del tipo di count non l'ho considerata, ma visto che visita, codice è una superchiave non serve il `count distinct`, per ogni visita ogni codice compare al più una volta.

Questa minchiata è stata approvata dal Pala.

1. Pala dixit qui non è decisivo che sia prescritta qualcosa che non sia F018, visto che posso prescrivere qualcosa `<> F018` insieme a qualcosa che è `= F018`

   ```
   select count(codice)
   from visita
   where codice not in(
   select visita
   from prescrizione
   where farmaco = 'F018')
   ```

2.2.3 Seconda query

Il numero di visite in cui non è stato fatto il controllo C014

è identica alla query di sopra

1. Pala dixit stessa cosa

2.2.4 Terza query

il medico che ha visitato il maggior numero di diversi pazienti

a sto punto questo schema lo facciamo un design pattern quante volte lo rifila.

```
create view medicoPazienti(medico, numero) as
select medico, count(distinct paziente)
from visita
group by medico

select medico
from medicoPazienti
where numero = (select max(numero) form medicoPazienti)
```

1. Pala dixit si fa una vista in cui si contano per i medici il numero di pazienti (diversi) visitati e si fa il max di questa vista

   ```
   create view numPazienti as
   select medico, count(distinct paziente) num
   from visita
   group by medico

   select medico
   from numPazienti
   where num = (select max(num) form numPazienti)
   ```

   se vuoi contare tutti i medici, magari anche quelli che non hanno fatto nessuna visita, allora fai un join esterno.

   nell'economia della query in questione non ha molto senso vedere anche i medici che non hanno mai fatto nessuna visita, quindi il join esterno possiamo evitarlo.

2.2.5 Quarta query

il paziente che è stato visitato dal maggior numero diverso di medici

stessa cosa di sopra ma invertito

```
create view numPazienti as
select paziente, count(distinct medico) num -- abbiamo cambiato sta riga
from visita
group by paziente -- sta riga

select paziente -- e sta riga, solo quelle
from numPazienti
where num = (select max(num) form numPazienti)
```

2.2.6 Quinta query

numero medio di farmaci prescritti al termine delle visite

va fatto un join esterno per contare le visite in cui non prescrivi niente

come al solito creiamo una vista e poi `select avg(numero)`, siamo al secondo design patter del Pala.

```
create view numFarmaci(visita, numero) as
select v.codice, count(farmaco)
```

```
from visita v left join prescrizione f on v.codice = f.visita
group by v.codice

select avg(numero) as HaiCapitoVaBeneComeAlSolito
from numFarmaci
```

Il Pala ha fatto la stessa cosa

2.2.7 Sesta query

Uguale alla quinta. Il numero medio di controlli fatti in una visita

2.3 Prova compitino X

- Wine : code, name, color, year
- Color : code, name
- $\text{Grape}_{\text{variety}}$: code, name
- $\text{Wine}_{\text{grapevariety}}$: wine, $\text{grape}_{\text{variety}}$
- Food : code, name, recipe, food, category
- $\text{Food}_{\text{category}}$: code, name
- $\text{Wine}_{\text{foodpairing}}$: code, wine, food, rating
- Rating : code, description

io posso avere un rating a una stessa coppia più volte, posso dire più volte che le lasagne con l'Ambrusco ci stanno da 10.

2.3.1 Prima

Codici dei fini e delle pietanze che abbinati hanno avuto il massimo numero di valutazioni 'Perfect'

```
create view perfects(wine, food, numero)
select wine, food, count(rating)
from wine_food_pairing
where rating = 'Perfect'
-- qui ho fatto una piccolissima minchiata
-- il codice non dice un cazzo su che rating ha l'abbinamento
```

```
-- quello devi vederlo dalla tabella rating

select wine, food, numero
from perfects
where numero = (select max(numero) from perfects)
```

1. Pala dixit

   ```
   create view pairings as
   select wine, food, count(r.code) number -- funziona anche count(*), non count dist
   from wine_food_pairing wfp, rating r
   where rating = r.code and description = 'Perfect'
   group by wine,food

   select wine,food
   from pairings
   where number = (select max(number) from pairing)
   ```

2.3.2 Seconda

il numero di vini prodotti usando un solo vitigno (`grape_variety`)

```
select wine from wine_grape_variety having count(grape_variety) = 1
```

si poteva usando con `grape_variety` due volte

1. Pala dixit hey indovina un po' mi sono scordato il group by

   ```
   select wine from wine_grape_variety having count(grape_variety) = 1 group by wine
   ```

   altrimenti

   ```
   create view vitigni_per_vino as
   select wine, count(*) as numero
   from wine_grape_variety
   group by wine

   select count(wine)
   from vitigni_per_vino
   where numero = 1
   ```

per fare la conta devi fare una vista, o un (se sei scemo)

```
select count(*)
from (select wine from wine_grape_variety having count(grape_variety) = 1 group by
```

2.3.3 Terza

per ogni vino, il codice, il nome, ed il numero di valutazioni di abbinamenti che coinvolgono quel vino, mostrando 0 per i vini che non compaiono negli abbinamenti

```
select w.code, w.name, count(wfp.rating)
from wine w left join wine_food_pairing wfp on (wfp.wine = w.codice)
group by w.code, w.name
```

1. Pala dixit viene esplicitato per gli 0

   va usato wine perchè chiede codice e nome, e visto che chiede tutti i vini va obbligata ogni riga della tabella wine con un join esterno

   e contiamo `wine_food_pairing.code` per contare gli 0 , non può essere sostituito con `count(*)` visto che va considerato il count 0 per i vini non cosati

   ```
   select w.code, w.name, count(wfp.code)
   from wine w left join wine_food_pairing wfp on wine = w.code
   group by w.code, w.name
   ```