

Derivate e viste SQL / 10-29

Colonne derivate (non inserisce "du zero")

Contatti calcolati automaticamente dal sistema

```
ALTER TABLE NomeTab ADD colonna_du integer  
GENERATED ALWAYS AS ((dataRest - dataNel) DAY); ] down  
[STORED ] ] compatibile
```

↳ versioni PostgreSQL >= 12

Attributi e relazioni su cui aggiungo colonne

Se si parte dalla stessa istanza il risultato è lo stesso
(non posso usare CURRENT_DATE)

Relazioni derivate

Ha una relazione con stesso schema, più una colonna

```
CREATE TABLE ClienteV  
(LIKE Cliente, bonus NUMERIC(4,2));
```

Non copio i dati! Ma solo lo schema

Non vi sono relazioni fra la relazione copiata e quella nuova
= Copie indipendenti

Esistono in noleggi conclusi e in corso (relazioni separate)

IL LKE crea relazioni vuote

CREATE TABLE Noleggio_A AS

SELECT colloc, dataNel, codcli

FROM Noleggio

WHERE dataRest IS NULL

WITH (NO) DATA → se mi interessa o meno l'istoria

(relazione vuota o meno)

↓

che viene nuovamente

motori di database memorizzata

È se delle triple in Noleggio vengano modificate e la query cambia valore? Le nuove tabella è indipendente, quindi non ha propria gestione delle modifiche dopo la creazione delle nuove tabella.

J

posso ottenere tale proprietà con il meccanismo delle viste

Viste

Ad esempio si vogliono gestire i noleggi di clienti ever os

Ci focalizziamo sul livello esterno, ovvero un accesso a oggetti che non sono "nuovi" (nuove tabelle) ma sono tabelle virtuali create su query eseguite sul livello logico.

In generale la vista non ha un file che corrisponde al contenuto della vista (può essere derivata).

Dunque gli accessi e le modifiche si propagano sulle relazioni di base

Tale meccanismo

- semplifica l'accesso ai dati
- fornisce indipendenza logica
- garantisce la privacità dei dati

CREATE VIEW Over 65 AS

```
SELECT *  
FROM Noleggio NATURAL JOIN Cliente  
WHERE (CurrentDate - DataN) YEAR >= 65
```

SELECT * FROM Over 65

possiamo usare come una relazione
ed effettuare le query su di essa

opz (x renominare)

CREATE VIEW nuova vista > [(lista di attr.)]

AS [integrazione]

[WITH [} LOCAL | CASCADING] CHECK OPTION]

CREATE VIEW V(M) AS

SELECT MAX(A) AS M

FROM R

alternative

} se il nome dell'attributo non
si deduce, lo applica
(ne basta uno per non far
dedurre tutto)

La vista permette di effettuare una sola volta certe operazioni (es: il join), comprendendo i dati senza ogni volta reperirli con query identiche.

Permette di ottenere dati e presentare solo dati aggregati garantendo riservatezza.

Possò fare join fra viste e relazioni. Possò anche definire viste di viste. Possò proiettare e fare selezioni su esse

Se vogliavo creare una vista su una vista mi mettiamo alcuni attributi posso mettere in join la vista su altre relazioni.

In presenza di creazione di viste su altre viste questo ultime vengono espresse nella loro definizione, la traduzione complessiva combina tutte le definizioni.

In semplificazione c'è ad "alto livello", ovvero coinvolge le modalità di accesso ai dati (ottenere risultati pronti per costituire nuova query) MA ogni volta che si esegue una query su viste il DBMS esegue operazioni aggiuntive (rispetto a query leggendo) → espansione = costi aggiuntivi + query attente

Le viste sono ottime soluzioni per migliorare la descrizione di ciò che vogliamo rappresentare ma male a livello di prestazione (e ogni vista ha espansione + esecuzione query)

Spesso puo' capitare che durante le espressioni capitino delle colonne di funzioni aggregate, non accettate in SQL
(PostgreSQL)

Possiamo inserire ed aggiornare dati nella vista. Non concesso
sempre per via di così eretici di espressione.

Se ha successo la modifica viene propagata sulla tabella base
Ad esempio inserimenti in V equivalgono a inserimenti in R

Non posso aggiornare un campo desirato poiché la modifica
è ambigua (può essere espresso su una o entrambe le
tabelle che lo compongono), si seguono delle regole generali:

- La cancellazione è permessa se univoca rispetto alle relazioni di base (avendo la vista definita su singole relazioni) e niente DISTINCT, GROUP BY, se sottintendono correlati
- La modifica è permessa se valgono le regole delle doppie e aggiorno campi non definiti da espressioni o funzioni
- L'inserimento è permesso se quanto sopra e gli attributi necessari vengono inseriti.

I DBMS permettono regole più flessibili, quanto sopra è sufficiente ma non necessarie.

La clausola ^{WITH} CHECK OPTION in fase di creazione delle viste permette di verificare se le modifiche / inserimenti di tuple successivi rispettano le condizioni delle query che ha creato la vista

Cio' mi assicura di non perdere informazioni, il sistema mi propone l'eventuale modifica/insozialimento. Se va a buon fine allora mi ritrovo tale tuple dentro le viste.