Sistemi Informativi

Alessandro Lori

C.d.L. Ing. Gestionale
Università di Pisa

27 Marzo 2009
Lab 04

Confronto tra insiemi

Impiegato(matricola, nome, cognome, dipartimento, ufficio, stipendio)

Trovare nome e cognome degli impiegati il cui stipendio è maggiore dello stipendio di qualunque altro impiegato di nome “Antonio”:

```
SELECT I1.nome, I1.cognome
FROM “Impiegato” I1
WHERE I1.stipendio > ANY
        (SELECT I2.stipendio
         FROM “Impiegato” I2
         WHERE I2.Nome = ’Antonio’))
```

Confronto tra insiemi

Impiegato(matricola, nome, cognome, dipartimento, ufficio, stipendio)

Trovare nome e cognome degli impiegati il cui stipendio è maggiore dello stipendio di qualunque altro impiegato di nome “Antonio”:

```
SELECT I1.nome, I1.cognome
FROM “Impiegato” I1
WHERE I1.stipendio > ANY
        (SELECT I2.stipendio
         FROM “Impiegato” I2
         WHERE I2.Nome = ’Antonio’))
```

Nota per PostgreSQL

Attenzione all’uso delle virgolette!

LIKE

SQL fornisce il supporto al *pattern matching* attraverso l'operatore `LIKE` ed i caratteri *jolly*:

- % → zero o più caratteri qualunque;
- _ → esattamente un carattere qualunque;

es.
"_A%": qualunque stringa di almeno 2 caratteri con la seconda lettera uguale ad A;
"_A__B%": qualunque stringa di almeno 4 caratteri con la seconda lettera uguale ad A e la quinta a B;

LIKE (II)

Impiegato(matricola, nome, cognome, dipartimento, ufficio, stipendio)

Trovare nome e cognome degli impiegati il cui nome inizia per A e la cui terza lettera è una d.

```
SELECT I.nome, I.cognome
FROM Impiegato I
WHERE I.nome LIKE 'A_d%';
```

LIKE (ex.1)

Impiegato(matricola, nome, cognome, dipartimento, ufficio, stipendio)

Trovare nome e cognome degli impiegati il cui nome inizia per A e termina per y.

LIKE (ex.1)

Impiegato(matricola, nome, cognome, dipartimento, ufficio, stipendio)

Trovare nome e cognome degli impiegati il cui nome inizia per A e termina per y.

```
SELECT I.nome, I.cognome
FROM Impiegato I
WHERE I.nome LIKE 'A%y';
```

Operatori di aggregazione

Valutano proprietà su insiemi di tuple e vengono applicati alla tabella risultato dopo la valutazione del `from` e del `where`.

`COUNT` (< * | [`DISTINCT`|`ALL`] *ListaAttributi* >)

< `SUM` | `MAX` | `MIN` | `AVG` > ([`DISTINCT`|`ALL`] *AttrEspr*)

count

Impiegato(matricola, nome, cognome, eta, ufficio, dipartimento)

Trovare il numero di impiegati del dipartimento Produzione:

```
SELECT COUNT(*)
FROM "Impiegato" I
WHERE I.dipartimento = 'Produzione';
```

COUNT (ex.2)

Impiegato(matricola, nome, cognome, eta, ufficio, stipendio)

Trovare il numero di diversi valori dell'attributo *stipendio* tra tutte le tuple di Impiegato:

COUNT (ex.2)

Impiegato(matricola, nome, cognome, eta, ufficio, stipendio)

Trovare il numero di diversi valori dell'attributo *stipendio* tra tutte le tuple di Impiegato:

```
SELECT COUNT(DISTINCT stipendio)
FROM Impiegato I;
```

count (ex.3)

Impiegato(matricola, nome, cognome, eta, ufficio, dipartimento)

Trovare il numero degli uffici del dipartimento Logistica:

count (ex.3)

Impiegato(matricola, nome, cognome, eta, ufficio, dipartimento)

Trovare il numero degli uffici del dipartimento Logistica:

```
SELECT count(distinct ufficio)
FROM Impiegato I
WHERE I.dipartimento = 'Logistica';
```

max, min, avg, sum

Impiegato(matricola, nome, cognome, eta, ufficio, stipendio)

Trovare gli stipendi minimo, massimo e medio fra quello di tutti gli impiegati:

```
SELECT min(I.stipendio), MAX(I.stipendio), AVG(I.stipendio)
FROM Impiegato I;
```

max, min, avg, sum (ex.4)

Impiegato(matricola, nome, cognome, eta, ufficio, dipartimento)
Supervisione(capo, impiegato)

Trovare la somma degli stipendi dei capi che supervisionano impiegati del dipartimento Contabilità

max, min, avg, sum (ex.4)

Impiegato(matricola, nome, cognome, eta, ufficio, dipartimento)
Supervisione(capo, impiegato)

Trovare la somma degli stipendi dei capi che supervisionano impiegati del dipartimento Contabilità

```
SELECT SUM(C.stipendio)
FROM Impiegato C
     INNER JOIN Supervisione S ON (C.matricola = S.capo)
     INNER JOIN Impiegato I ON (S.impiegato = I.matricola)
WHERE I.dipartimento = 'Contabilità';
```

Clausola `GROUP BY`

Nasce dall'esigenza di applicare operazioni di aggregazione a ciascuno dei membri di un gruppo di tuple in una relazione.

`SELECT [DISTINCT]` **lista-select**
`FROM` **lista-from**
`WHERE` **qualificazione**
`GROUP BY` **lista-gruppo**
`HAVING` **qualificazione-gruppo**

- La **lista-select** contiene lista di colonne e lista di termini del tipo **aggop** (*nome-colonna*) [AS *nuovo-nome*];
- la **lista-select** specifica nomi di colonne delle tabelle nominate nella lista-from; ogni colonna in lista-select deve apparire anche in **lista-gruppo**.
- espressioni in **qualificazione-gruppo** devono avere un singolo valore per gruppo.

Clausola `GROUP BY` (II)

Valutazione di una query

Strategia di valutazione concettuale:

1. Prodotto cartesiano delle tabelle in **lista-from**;
2. cancellazione righe che non soddisfano le condizioni in **qualificazione**;
3. cancellazione colonne non in `SELECT, GROUP BY, HAVING`;
4. ordinamento tabella secondo la clausola `GROUP BY`;
5. applicazione della **qualificazione-gruppo**;
6. generazione di singola tupla per gruppo (aggregazione);
7. se specificato `DISTINCT`, cancellazione righe duplicate.

Clausola GROUP BY (III)

Impiegato(matricola, nome, cognome, eta, dipartimento, stipendio)
Sedi(dipartimento, indirizzo, citta)

Trovare l'età media per ogni valore di stipendio con almeno 2 impiegati con sede a Pisa:

```
SELECT I.stipendio, AVG(I.eta) AS media_eta
FROM Impiegato I
     INNER JOIN Sedi SE ON (I.dipartimento = SE.dipartimento)
WHERE SE.citta = 'Pisa'
GROUP BY I.stipendio
HAVING COUNT(*) > 1;
```

Clausola `GROUP BY` (ex.5)

Impiegato(matricola, nome, cognome, eta, dipartimento, stipendio)
Sedi(dipartimento, indirizzo, citta)
Supervisione(capo, impiegato)

Trovare per ogni valore di stipendio con al massimo 5 impiegati l'età media dei capi che supervisionano impiegati con sede a Pisa:

Clausola `GROUP BY` (ex.5)

Impiegato(matricola, nome, cognome, eta, dipartimento, stipendio)
Sedi(dipartimento, indirizzo, citta)
Supervisione(capo, impiegato)

Trovare per ogni valore di stipendio con al massimo 5 impiegati l'età media dei capi che supervisionano impiegati con sede a Pisa:

```
SELECT C.stipendio, AVG(C.eta) AS media_eta
FROM Impiegato C
     INNER JOIN Supervisione S ON (I.matricola = S.capo)
     INNER JOIN Impiegato I ON (S.impiegato = I.matricola)
     INNER JOIN Sedi SE ON (I.dipartimento = SE.dipartimento)
WHERE SE.citta = 'Pisa'
GROUP BY C.stipendio
HAVING COUNT(*) <= 5;
```

Ordinamento

Costruzione di un ordine tra le tuple di una relazione:

`ORDER BY` *AttrDiOrdinamento* [`asc` | `desc`]
{, *AttrDiOrdinamento* [`asc` | `desc`]}

Ordinamento (ex.6)

Impiegato(matricola, nome, cognome, eta, stipendio)

Trovare nome, cognome ed età degli impiegati, visualizzando i risultati in ordine decrescente di età:

Ordinamento (ex.6)

Impiegato(matricola, nome, cognome, eta, stipendio)

Trovare nome, cognome ed età degli impiegati, visualizzando i risultati in ordine decrescente di età:

```
SELECT I.nome, I.cognome
FROM Impiegato I
ORDER BY eta DISC;
```

Trigger

Trigger: procedura eseguita dal DBMS in risposta ad un evento in basi di dati *attive*.

Esempi d'utilizzo:

- campo contatore con autoincremento;
- invio automatico di ordini in caso un magazzino scenda sotto una certa giacenza;
- statistiche o log;
- tabelle di backup attivate in caso di aggiornamento o cancellazione di tuble da un database in maniera trasparente all'utente.

Trigger (II)

Struttura di un trigger:

- **evento**: cambiamento nella base di dati che *attiva* il trigger;
- **condizione**: test o query eseguita all'attivazione del trigger;
- **azione**: procedura eseguita all'attivazione del trigger se la condizione è soddisfatta.

Trigger (III)

- Può essere attivato da un inserimento, cancellazione o da un aggiornamento, indipendentemente dall'utente e dall'applicazione che ha causato l'evento;
- se la condizione è una query è *vera* se l'insieme di risposta è *non vuoto*;
- l'azione può esaminare la risposta all'interrogazione, accedere a vecchi e nuovi valori delle tuple modificate, modificare la base di dati o chiamare procedure nel linguaggio ospite;
- le procedure possono essere definite usando il linguaggio preferito (oltre a PL/pgSQL).

Trigger (IV)

L'azione del trigger può essere eseguita:

- **prima** (`BEFORE`) o **dopo** (`AFTER`) il comando attivante;
- a **livello di riga** (`FOR EACH ROW`) o a **livello di istruzione** (`FOR EACH STATEMENT`).

es. trigger

```
CREATE FUNCTION archivia_impiegati() RETURNS trigger AS $archivia_impiegati$
    BEGIN
       -- Aggiorna la tabella di backup
       INSERT INTO ex_impiegati SELECT OLD.nome, OLD.cognome, OLD.matricola;
       RETURN OLD;
    END
$archivia_impiegati$ LANGUAGE plpgsql;


CREATE TRIGGER archivia_impiegati
AFTER DELETE ON impiegati
   FOR EACH ROW EXECUTE PROCEDURE archivia_impiegati();
```

es. trigger (II)

Creare una tabella **statistiche** con i seguenti campi:

- *tipo_accesso*: char(1), NOT NULL;
- *nome_utente*: char(20) NOT NULL;
- *tempo*: timestamp with time zone;

es. trigger (III)

Creare la seguente funzione trigger.

```
CREATE OR REPLACE FUNCTION statistiche_impiegati() RETURNS TRIGGER AS
$statistiche_impiegati$
    BEGIN
       -- Crea una riga nella tabella che riflette le modifiche fatte
       -- nel database
       IF (TG_OP = 'DELETE') THEN
           INSERT INTO statistiche SELECT 'D', user, now();
           RETURN OLD;
       ELSIF (TG_OP = 'UPDATE') THEN
           INSERT INTO statistiche SELECT 'U', user, now();
           RETURN OLD;
       ELSIF (TG_OP = 'INSERT') THEN
           INSERT INTO statistiche SELECT 'I', user, now();
           RETURN OLD;
       END IF;
       RETURN NULL;
    END;
$statistiche_impiegati$ LANGUAGE plpgsql;
```

es. trigger (IV)

Impostare i seguenti trigger.

```
CREATE TRIGGER aggiorna_statistiche
AFTER INSERT OR UPDATE OR DELETE ON Impiegati
   FOR EACH ROW EXECUTE PROCEDURE statistiche_impiegati();

CREATE TRIGGER aggiorna_statistiche
AFTER INSERT OR UPDATE OR DELETE ON Sedi
   FOR EACH ROW EXECUTE PROCEDURE statistiche_impiegati();

CREATE TRIGGER aggiorna_statistiche
AFTER INSERT OR UPDATE OR DELETE ON Supervisione
   FOR EACH ROW EXECUTE PROCEDURE statistiche_impiegati();
```