Sistemi Informativi

Alessandro Lori

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

3 Aprile 2009
Lab 05

Query su PostgreSQL (ex.1)

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

Trovare le città per cui la somma degli stipendi degli impiegati che lavorano nell'ufficio 202 è superiore a 250 mila euro:

Query su PostgreSQL (ex.1)

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

Trovare le città per cui la somma degli stipendi degli impiegati che lavorano nell'ufficio 202 è superiore a 250 mila euro:

```
SELECT S.citta
FROM Impiegato I
     INNER JOIN Sedi SE ON (I.dipartimento = SE.dipartimento)
WHERE I.ufficio = 202
GROUP BY S.citta
HAVING sum(Stipendio) > 250;
```

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();
```

Ex. 2

Creare una tabella `exSedi` con i seguenti campi:

- *dipartimento*: text (chiave primaria);
- *indirizzo*: text;
- *citta*: text;

Ex. 3

Creare una funzione trigger che archivi le tuple cancellate da Sedi in exSedi.

Ex. 3

Creare una funzione trigger che archivi le tuple cancellate da Sedi in exSedi.

```
CREATE FUNCTION archivia_sede() RETURNS trigger AS $archivia_sede$
    BEGIN
       -- Aggiorna la tabella delle sedi non più in uso
       INSERT INTO ex_sedi SELECT OLD.dipartimento, OLD.indirizzo, OLD.citta;
       RETURN OLD;
    END
$archivia_sede$ LANGUAGE plpgsql;
```

Ex. 4

Creare un trigger `archivia_sede` che archivi le tuple cancellate da Sedi in exSedi.

Ex. 4

Creare un trigger `archivia_sede` che archivi le tuple cancellate da Sedi in exSedi.

```
CREATE TRIGGER archivia_sede
AFTER DELETE ON sedi
   FOR EACH ROW EXECUTE PROCEDURE archivia_sede();
```

Progettazione

Fase fondamentale del *ciclo di vita* di un sistema informativo.

- **Prog. concettuale**: costruzione, a partire dalle specifiche, di uno schema concettuale in grado di descrivere al meglio le specifiche sui dati; costruzione graduale con raffinamento in passaggi successivi.
- **Prog. logica**: costruzione, dato il *carico applicativo*, di uno schema logico che descrive in maniera corretta ed efficiente tutte le informazione contenute nello schema concettuale.
  Prevede due fasi:
  - *ristrutturazione* (semplicità, efficienza);
  - *traduzione* (verifica di qualità).

Progettazione (II)

- Prog. concettuale $\Rightarrow$ schema concettuale:
  - diagramma E-R;
  - business rules;
  - dizionario dei dati.
- Prog. logica $\Rightarrow$ schema logico:
  - schema e vincoli SQL;
  - asserzioni SQL;
  - trigger;
  - logica dell'applicazione.

Business rules

- descrizione di un concetto;
- vincolo di integrità:
  < concetto > `deve/non deve` *< espr. su concetti >*;
  es. il direttore di un dipartimento deve afferire a tale dipartimento.
- derivazione:
  < concetto > `si ottiene` *< op. su concetti >*;
  es. il num. di impiegati di un dipartimento si ottiene contando gli impiegati che vi afferiscono.

Strumenti CASE

CASE (Computer Aided Software Engineering):

- interfaccia grafica;
- dizionario dei dati;
- generazione di SQL;
- integrazione con DB.

Nota: ogni tool usa una sua notazione (non esiste standard).

TDM: ERD

Relazione:

- identificante;
- non identificante;
- M:N;
- informativa.

Ex. 5

Creare TDM-ERD e script SQL per il seguente diagramma E-R: