SQL: operazioni sui dati SELECT

Fallucchi Francesca
fallucchi@info.uniroma2.it
http://art.uniroma2.it/fallucchi/
a.a. 2010/2011

SQL: introduzione

- Non è un linguaggio di programmazione
- Segue un paradigma dichiarativo: si specifica la descrizione dell'obiettivo
- l'interrogazione in SQL viene tradotta dai DBMS in linguaggio procedurale, quindi viene ottimizzata
- sulla traduzione si fanno ottimizzazioni algebriche …
- … e non (queste ultime dipendono dalle strutture sottostanti al DBMS in questione)

Istruzione SELECT

- Sintassi

```
SELECT ListaAttributi
FROM ListaTabelle
[ WHERE Condizione ]
```

- Le tre parti della select sono chiamate:
  - clausola Select o "target list"
  - clausola FROM
  - clausola WHERE

SELECT

```
SELECT * FROM <tabella>
```

- riporta in forma tabellare tutte le tuple di tabella
- il carattere "*" è un carattere jolly che ci permette di listare tutti gli attributi
- in alternativa è possibile listare il nome di ciascun attributo, o di un sottoinsieme di essi, definito in "tabella"

```
SELECT <attr1>,...,<attrN> FROM <tabella>
```

Selezioni

- Per richiedere l'applicazione di una selezione su una tabella è necessario esprimere, nella select, la formula proposizionale che definisce la condizione di selezione
- Si usa la clausola WHERE seguita da un'espressione che deve essere soddisfatta

Selezioni

- Per ottenere tutte le sole tuple della tabella che soddisfano la condizione data

```
SELECT *
FROM <tabella>
WHERE <condizione>;
```

Selezioni

- condizione è un'espressione booleana
- Alcuni esempi di espressioni booleane:

  Uso di operatori di confronto <termine> <operatore> <termine>:
  - <termine>: attributo, costante, <espressione>
  - <operatore>: >, ≥, <, ≤, <>, =
    - Codicefiscale = 'abcdef12g34h567i'
    - NascitaData > '01-jan-1994'

Selezioni

Uso di operatori logici [<espressione>] <operatore> <espressione>:
- <espressione>: espressione booleana
- <operatore>: "and", "or", "not"
  - NOT (Codicefiscale = 'abcdef12g34h567i')
  - NascitaData > '01-jan-1995' AND Residente='R'

- Condizione può diventare un'espressione anche molto complessa potendo comporla con qualunque funzione definita sui domini

Proiezioni

- Per la proiezione su una tabella è sufficiente listare i soli attributi desiderati

  SELECT [ALL] <attrX>,...,<attrY>
  FROM <tabella>
- La select ritorna comunque tante tuple quante sono quelle presenti nella tabella definite sui soli attributi citati

  SELECT CodiceFiscale, Nome, Cognome
  FROM CITTADINO

Esempio

Persone

Nome	Età	Reddito
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

Selezione e proiezione

- Nome e reddito delle persone con meno di trenta anni

$$PROJ_{Nome,\ Reddito}(SEL_{Eta<30}(Persone))$$

SELECT nome, reddito
FROM persone
WHERE eta < 30

Esempio

Persone

Nome	Età	Reddito
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

SELECT nome, reddito
FROM persone
WHERE eta < 30

Persone

Nome	Reddito
Andrea	21
Aldo	15
Filippo	30

Ridenominazione e Alias

- La clausola AS permette di definire un alias per un attributo
- Esempi:

```
SELECT Cognome AS NomeDiFamiglia
FROM Cittadino

SELECT Nome || ' ' || Cognome AS
            NomeComposto
FROM Cittadino
```

Ridenominazione e Alias

- Si può definire anche un alias per la tabella con il seguente formalismo:

  <tabella> <alias>

- Esempio:

```
SELECT Nome,Cognome
  FROM Cittadino C;
```

SELECT, abbreviazioni

```
SELECT nome, reddito
FROM persone
HERE eta < 30
```

```
SELECT p.nome as name
  p.reddito as reddito
  FROM persone p
  WHERE p.eta < 30
```

Selezione, senza proiezione

- Nome, età e reddito delle persone con meno di trenta anni

$SEL_{Eta<30}(Persone)$

```
SELECT *
FROM persone
WHERE eta < 30
```

Esempio

Persone

Nome	Età	Reddito
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

```
SELECT *
FROM persone
WHERE eta < 30
```

Persone

Nome	Età	Reddito
Andrea	27	21
Aldo	25	15
Filippo	26	30

SELECT, abbreviazioni

```
SELECT nome, eta, reddito
FROM persone
HERE eta < 30
```

```
SELECT *
FROM persone
WHERE eta < 30
```

Proiezione, senza selezione

- Nome e reddito di tutte le persone

$PROJ_{Nome, Reddito}(Persone)$

```
SELECT nome, reddito
FROM persone
```

Espressioni nella target list

```
SELECT Reddito
FROM Persone
WHERE Nome = 'Luigi'
```

Condizione complessa

```
SELECT *
FROM persone
WHERE reddito > 25
AND(eta < 30 or eta > 60)
```

Colonne calcolate

- In una SELECT, oltre agli attributi, è possibile specificare espressioni complesse che generano risultati calcolati.
  - COUNT (lista_att): conta gli attributi non nulli
  - SUM (espressione): somma
  - MAX (espressione): calcola il massimo
  - MIN (espressione): calcola in minimo
  - AVG (espressione): calcola la media

Condizione "LIKE"

- Le persone che hanno un nome che inizia per 'A' e ha una 'd' come terza lettera

```
SELECT *
FROM persone
WHERE nome like 'A_d%'
```

Gestione dei valori nulli

Impiegati

Matricola	Cognome	Filiale	Età
5998	Neri	Milano	45
9553	Bruni	Milano	NULL

- Gli impiegati la cui età è o potrebbe essere maggiore di 40

$SEL_{(Età > 40) OR (Età IS NULL)}$ (Impiegati)

Gestione dei valori nulli

- Gli impiegati la cui età è o potrebbe essere maggiore di 40

SEL $_{\text{Età > 40 OR Età IS NULL}}$ (Impiegati)

```
SELECT *
FROM impiegati
WHERE eta > 40 or eta is null
```

Proiezione, attenzione

- Cognome e filiale di tutti gli impiegati

Cognome	Filiale
Neri	Napoli
Neri	Milano
Rossi	Roma

PROJ $_{\text{Cognome, Filiale}}$ (Impiegati)

Distinct

- Per eliminare eventuali duplicazioni

`Sintassi`

```
SELECT DISTINCT <attrX>,...,<attrY>
        FROM <tabella>;
```

`Esempio`

```
SELECT DISTINCT Residente FROM
        CITTADINO;
```

Distinct

```
SELECT
    cognome, filiale
FROM impiegati
```

Cognome	Filiale
Neri	Napoli
Neri	Milano
Rossi	Roma
Rossi	Roma

```
SELECT DISTINT
    cognome, filiale
FROM impiegati
```

Cognome	Filiale
Neri	Napoli
Neri	Milano
Rossi	Roma

Ordinamento dei risultati

- La SELECT ritorna le istanze selezionate nell'ordine con cui sono state inserite nel database.

```
ORDER BY <column> [ASC[ENDING] |
        DESC[ENDING]] [, .... ]
```

- Ordinare il risultato in base ai valori delle colonne

```
SELECT * FROM CITTADINO ORDER
  BY Cognome, Nome, NascitaData DESC;
```

Ordinamento del risultato

- Nome e reddito delle persone con meno di trenta anni in ordine alfabetico

```
select nome, reddito
from persone
where eta < 30
order by nome
```

Ordinamento del risultato

```
SELECT nome, reddito
FROM persone
WHERE eta < 30
```

Persone

Nome	Reddito
Andrea	21
Aldo	15
Filippo	30

```
SELECT nome, reddito
FROM persone
WHERE eta < 30
ORDER BY nome
```

Persone

Nome	Reddito
Aldo	15
Andrea	21
Filippo	30

Raggruppamento dei risultati

- E' possibile raggruppare i risultati in base al contenuto di una o più colonne con la clausola

  GROUP BY ‹column› [, ...]

- Il risultato viene partizionato in gruppi tali che le istanze in uno stesso gruppo hanno lo stesso valore nelle colonne indicate

```
SELECT Residente, SUM(1)
FROM CITTADINO
GROUP BY Residente;
```

Raggruppamento dei risultati

- HAVING `associato a` GROUP BY

  restrizione delle istanze appartenenti ad un gruppo definendo una condizione di selezione ristretta alle sole funzioni aggregate

  esempi:

```
SELECT Dip, SUM(Stip) AS SommaStip
FROM Impiegato
GROU BY Dip
HAVING SUM(Stip) >100
```

Interrogazioni di tipo insiemistico

selectSQL

‹union| intersect| except› [all]

selectSQL

- La select da sola non permette di otterene union
- Gli operatori intersect e except possono essere espressi con altri costrutti di SQL come le interrogazioni nidificate
- I duplicati vengono eliminati di default