Basi di Dati:
Corso di laboratorio

Lezione 3

Raffaella Gentilini

Sommario

Data Manipulation Language (DML)

Istruzioni del DML

Le istruzioni del DML di SQL sono:

- Modifica
  - `INSERT`: Inserisce nuove tuple nel DB
  - `UPDATE` : Cancella tuple dal DB
  - `DELETE` : Modifica tuple nel DB
- Interrogazione
  - `SELECT`: Esegue interrogazioni (query) sul DB

Data Manipulation Language (DML)

Istruzioni del DML

Le istruzioni del DML di SQL sono:

- Modifica
  - `INSERT`: Inserisce nuove tuple nel DB
  - `UPDATE` : Cancella tuple dal DB
  - `DELETE` : Modifica tuple nel DB
- Interrogazione
  - `SELECT`: Esegue interrogazioni (query) sul DB

L'Istruzione SELECT

Sintassi generale del comando SELECT

SELECT [ALL | DISTINCT] { * | <espressione> [AS <nome>] } [, . . .]
FROM <nome_tab> [[AS] <alias>]
[WHERE <predicato>]
[GROUP BY <espressione> [, . . .]]
[HAVING <predicato>]
[{ UNION | INTERSECT | EXCEPT [ALL] } <richiesta>]
[ORDER BY <espressione> [ASC | DESC] [, . . .]]

L'istruzione SELECT e' composta da 7 clausole, di cui 5 opzionali.

L'Istruzione SELECT

Sintassi generale del comando SELECT

SELECT [ALL | DISTINCT] { * | <espressione> [AS <nome>] } [, . . .]
FROM <nome_tab> [[AS]<alias>]
[WHERE <predicato>]
[GROUP BY <espressione> [, . . .]]
[HAVING <predicato>]
[{ UNION | INTERSECT | EXCEPT [ALL]}<richiesta>]
[ORDER BY <espressione> [ASC | DESC] [, . . .]]

Clausola SELECT

Permette di specificare gli attributi che devono apparire nel risultato della query.

L'Istruzione SELECT

Sintassi generale del comando SELECT

SELECT [ALL | DISTINCT] { * | <espressione> [AS <nome>] } [, . . .]
FROM <nome_tab> [[AS]<alias>]
[WHERE <predicato>]
[GROUP BY <espressione> [, . . .]]
[HAVING <predicato>]
[{ UNION | INTERSECT | EXCEPT [ALL]}<richiesta>]
[ORDER BY <espressione> [ASC | DESC] [, . . .]]

Clausola FROM

Specifica tabelle (/e) su cui viene posta la query .

L'Istruzione SELECT

Sintassi generale del comando SELECT

SELECT [ALL | DISTINCT] { * | <espressione> [AS <nome>] } [, . . .]
FROM <nome_tab> [[AS]<alias>]
[WHERE <predicato>]
[GROUP BY <espressione> [, . . .]]
[HAVING <predicato>]
[{ UNION | INTERSECT | EXCEPT [ALL]}<richiesta>]
[ORDER BY <espressione> [ASC | DESC] [, . . .]]

Clausola WHERE

Permette di filtrare le tuple in base alla specifica di condizioni da soddisfare.

L'Istruzione SELECT

Sintassi generale del comando SELECT

SELECT [ALL | DISTINCT] { * | <espressione> [AS <nome>] } [, . . .]
FROM <nome_tab> [[AS]<alias>]
[WHERE <predicato>]
[GROUP BY <espressione> [, . . .]]
[HAVING <predicato>]
[{ UNION | INTERSECT | EXCEPT [ALL]}<richiesta>]
[ORDER BY <espressione> [ASC | DESC] [, . . .]]

Clausola GROUP BY

Permette di definire dei gruppi.

L'Istruzione SELECT

Sintassi generale del comando SELECT

SELECT [ALL | DISTINCT] { * | <espressione> [AS <nome>] } [, . . .]
  FROM <nome_tab> [[AS]<alias>]
  [WHERE <predicato>]
  [GROUP BY <espressione> [, . . .]]
  [HAVING <predicato>]
  [{ UNION | INTERSECT | EXCEPT [ALL]}<richiesta>]
  [ORDER BY <espressione> [ASC | DESC] [, . . .]]

Clausola UNION, INTERSECT, EXCEPT

Permette di effettuare operazioni insiemistiche sui risultati della query.

L'Istruzione SELECT

Sintassi generale del comando SELECT

SELECT [ALL | DISTINCT] { * | <espressione> [AS <nome>] } [, . . .]
FROM <nome_tab> [[AS]<alias>]
[WHERE <predicato>]
[GROUP BY <espressione> [, . . .]]
[HAVING <predicato>]
[{ UNION | INTERSECT | EXCEPT [ALL]}<richiesta>]
[ORDER BY <espressione> [ASC | DESC] [, . . .]]

Clausola ORDER BY

Permette di ordinare le tuple del risultato della query.

La Clausola SELECT

SELECT [ALL | DISTINCT] { * | <espressione> [AS <nome>] } [, . . .]

- definisce le colonne nel risultato della query;

La Clausola SELECT

SELECT [ALL | DISTINCT] { * | <espressione> [AS <nome>] } [, . . .]

- definisce le colonne nel risultato della query;
- permette di operare un'operazione di proiezione sulla tabella intermedia generata dal resto della query;

La Clausola SELECT

SELECT [ALL | DISTINCT] { * | <espressione> [AS <nome>] } [, . . .]

- definisce le colonne nel risultato della query;
- permette di operare un'operazione di proiezione sulla tabella intermedia generata dal resto della query;
- la specifica esplicita degli attributi avviene elencando i suddetti in una lista, separati da virgola;

La Clausola SELECT

SELECT [ALL | DISTINCT] { * | <espressione> [AS <nome>] } [, . . .]

- definisce le colonne nel risultato della query;
- permette di operare un'operazione di proiezione sulla tabella intermedia generata dal resto della query;
- la specifica esplicita degli attributi avviene elencando i suddetti in una lista, separati da virgola;
- permette anche di rinominare colonne e di generarne di nuove.

SELECT senza proiezione

L'operatore *

SELECT [ALL | DISTINCT] { * | <espressione> [AS <nome>] } [, . . .]

- L'operatore * permette di recuperare tutti gli attributi della tabella specificata nella clausola FROM

L'operatore *: Esempio

Si consideri lo schema relazionale visto nelle lezioni precedente:

- *persona*(*id_persona*, *codice_fiscale*, *nome*, *cognome*, *data_nascita*)
- corso(*id_corso*, *id_insegnante*, *sigla*, *crediti*, *descrizione*)
- frequenza(*id_studente*,*id_corso*,*voto*): dove *id_studente* ed *id_corso* sono chiavi esterne su persona e corso

Example

```
SELECT * FROM  persona;
```

permette recupero dell'intero contenuto di *persona*; equivale a:

```
SELECT id_persona, codice_fiscale, nome, cognome, data_nascita
FROM  persona;
```

Tabelle vs Relazioni

Gli operatori DISTINCT e ALL

SELECT [ALL | DISTINCT] { * | <espressione> [AS <nome>] } [, . . .]

- Il risultato di una query SQL puo' contenere righe duplicate

```
SELECT nome
FROM persona
WHERE cognome = 'Rossi'
```

nome
Francesca
Paolo
Francesca

Tabelle vs Relazioni

Gli operatori DISTINCT e ALL

SELECT [ALL | DISTINCT] { * | <espressione> [AS <nome>] } [, . . .]

- Per eliminare le duplicazioni si usa l'opzione DISTINCT nella clausola SELECT

```
SELECT DISTINCT nome
FROM persona
WHERE cognome = 'Rossi'
```

nome
Francesca
Paolo

- ALL e' opzione di default nella clausola SELECT

Espressioni nella clausola SELECT

SELECT [ALL | DISTINCT] { * | <espressione> [AS <nome>] } [, . . .]

- La SELECT list puo' contenere non solo attributi, ma anche espressioni:

```
SELECT id_studente, voto + 2
FROM frequenza
WHERE id_corso = 5
```

id_studente	
1	30
2	18
3	19
. . .	. . .

- Si noti che in questo caso la seconda colonna e' priva di nome.

Espressioni nella clausola SELECT

SELECT [ALL | DISTINCT] { * | <espressione> [AS <nome>] } [, . . .]

- L'utilizzo dell'operatore || all'interno di un'espressione nella clausola SELECT permette di concatenare campi di tipo carattere

Example

```
SELECT  cognome || ' ' || nome ' ' || 'e\'nato il:' || data_nascita
FROM persona
```

Ridenominazione delle Colonne

SELECT [ALL | DISTINCT] { * | <espressione> [AS <nome>] } [, . . .]

L'operatore AS

- Ad ogni elemento della SELECT list e' possibile associare un nome a piacere.
  SELECT *id_studente*, voto+2 AS voto_calmierato
  FROM *frequenza*;
- La parola chiave AS puo' anche essere omessa.
  SELECT *id_studente*, voto+2 voto_calmierato
  FROM *frequenza*;

La Clausola FROM

FROM <nome_tab> [[AS]<alias>]

- Specifica le tabelle su cui si pone la query;
- Costruisce la tabella intermedia a partire da una o piu' tabelle, su cui operano le altre clausole (SELECT, WHERE ...)
- La tabella intermedia e' il risultato del prodotto caresiano delle tabelle elencate;
- La sintassi data e' semplificata (non include clausola esplicita di JOIN che vedremo nella prossima lezione).

Esempi

Example

1. Selezionare tutti i corsi con un credito:

   SELECT $*$
   FROM *corsi*
   WHERE *crediti* $= 1$;

2. Selezionare id, nome e cognome degli studenti:

   SELECT DISTINCT *id_studente*, *nome*, *cognome*
   FROM *persona*, *frequenza*
   WHERE *persona.id_persona* $=$ *frequanza.id_studente*;

   - $\sigma_E(R_1 \times R_2) \equiv R_1 \bowtie_E R_2$: La query sopra realizza un equi-join!

Pseudonimi e Alias

FROM <nome_tab> [[AS]<alias>]

- Per chiarezza, ogni nome di colonna puo'/deve essere scritto prefissandolo con il nome della tabella,
- si puo' usare uno pseudonimo (alias) in luogo del nome di tabella:

Esempi

Example

```
SELECT f.id_studente
FROM frequenza AS f
WHERE f.id_corso = 7;
```

Example

```
SELECT corso.id_corso
FROM corso, frequenza
WHERE corso.id_corso = frequenza.id_corso AND id_studente = 7;
```

Esempi

Example

```
SELECT f.id_studente
FROM frequenza AS f                    wrong!
WHERE frequenza.id_corso = 7;
```

Example

```
SELECT corso.id_corso
FROM corso, frequenza
WHERE corso.id_corso = frequenza.id_corso AND id_studente = 7;
```

Esempi

Example (Self-Join e Alias)

genitore	*figlio*
Giorgio	Fabio
Maria	Anna
Milva	Paolo

genitore	*figlio*
Fabio	Luca
Anna	Luca
Milva	Genoveffa

- L'uso di alias e' forzato quando si deve eseguire una self-join:
- Chi sono i nonni di Luca?

```
SELECT g1.genitore AS nonno
FROM genitore AS g_1, genitore AS g_2
WHERE g1.figlio = g2.genitore AND g2.figlio = 'Luca' ;
```

Ordinamento del risultato

[ORDER BY <espressione> [ASC | DESC] [, . . .]]

Per ordinare il risultato di una query secondo i valori di una o piu' colonne:

Ordinamento del risultato

[ORDER BY <espressione> [ASC | DESC] [, . . .]]

Per ordinare il risultato di una query secondo i valori di una o piu' colonne:

- si introduce la clausola ORDER BY,

Ordinamento del risultato

[ORDER BY <espressione> [ASC | DESC] [, . . .]]

Per ordinare il risultato di una query secondo i valori di una o piu' colonne:

- si introduce la clausola ORDER BY,
- per ogni colonna si specifica se l'ordinamento e' per valori ascendenti (operatore ASC) o discendenti (operatore DESC)

Ordinamento del risultato

[ORDER BY <espressione> [ASC | DESC] [, . . .]]

Per ordinare il risultato di una query secondo i valori di una o piu' colonne:

- si introduce la clausola ORDER BY,
- per ogni colonna si specifica se l'ordinamento e' per valori ascendenti (operatore ASC) o discendenti (operatore DESC)
- come nella clausola SELECT

Ordinamento del risultato: Esempio

Example

```
SELECT id_persona, cognome, nome
FROM persona
ORDER BY cognome ASC, nome ASC;
```

Ordinamento del risultato: Esempio

Example

```
SELECT id_persona, cognome, nome
FROM persona
ORDER BY cognome ASC, nome ASC;
```

id_persona	cognome	nome
19	Bianchi	Francesco
2	Rossi	Andrea
17	Rossi	Emilio
...	...	

La Clausola WHERE

[WHERE <predicato>]

- Permette di filtrare le tuple della tabella intermedia in base alla soddisfazione di un predicato
- Vine usata logica a tre valori: solo le tuple per cui il predicato e' vero appaiono nel risultato.
- I predicati possono essere combinati usando gli operatori logici AND, OR, NOT.

La Clausola WHERE

Example

1. Persone di nome Giovanni nate prima del 1970:

```
SELECT *
FROM persona
WHERE nome = 'Giovanni'  AND data_nascita <= '01/01/1970';
```

Predicati: Funzioni e Operatori

[WHERE <predicato>]

- Un predicato e' costituito da una formula logica su delle espressioni;
- Vedremo ora gli operatori e le funzioni piu' comuni che possono essere utilizzati nella definizione di espressioni e predicati;
- In PostgreSQL i comandi \df e \do permettono di elencare la lista degli operatori e delle funzioni disponibili.

Operatori Logici

Operatore	Descrizione
AND	congiunzione logica
OR	disgiunzione logica
NOT	negazione logica
...	...

Operatori di Comparazione

Operatore	Descrizione
<x> = <y>	<x> e' uguale a <y>
<x> != <y> o <x> <> <y>	<x> e' diverso da <y>
<x> < (<=,>,>=) <y>	<x> e' minore (minore o uguale, maggiore, maggiore uguale) a <y>
<a> BETWEEN <x> AND <y>	equivale ad <a> >= <x> AND a<=y
<a> NOT BETWEEN <x> AND <y>	equivale ad <a> < <x> OR a>y
<x> IS NULL	vero se <x> ha valore NULL
<x> IS NOT NULL	vero se il valore di <x> non e' NULL
<x> IS DISTINCT FROM <y>	<x> e' diverso da <y>, supporta NULL
<x> IS NOT DISTINCT FROM <y>	<x> e' uguale a <y>, supporta NULL

Operatori di Comparazione

Example

1. Corsi il cui numero di crediti e' strettamente superiore a 2:

```
SELECT *
FROM corso
WHERE crediti > 2;
```

2. Corsi il cui numero di crediti e' definito:

```
SELECT *
FROM corso
WHERE crediti  IS NOT NULL;
```

3. Corsi il cui numero di crediti e' compreso tra 2 e 3:

```
SELECT *
FROM corso
WHERE crediti  BETWEEN 2 AND 3;
```

Operatori Matematici

Operatore	Descrizione
<x> + <y>	addizione
<x> - <y>	sottrazione
<x> * <y>	moltiplicazione
<x> / <y>	divisione
<x> % <y>	resto
@(<x>)	valore assoluto
...	...

Funzioni ed Operatori su Stringhe

Operatore	Descrizione
<exp> || <exp>	concatenazione di stringhe
CHAR_LENGTH(<st>)	Numero caratteri della stringa
LOWER(<st>)	converte caratteri stringa in minuscole
UPPER(<st>)	converte caratteri stringa in maiuscole
...	...

Funzioni ed Operatori su Stringhe

Example

1. `SELECT '2*3=' || 2 * 3;`

   2*3=6

2. `SELECT CHAR_LENGTH('Anna');`

   4

3. `SELECT LOWER('Anna');`

   ---

   anna

Funzioni ed Operatori di Corrispondenza di Pattern

Operatore LIKE

- l'operatore LIKE mediante le 'wildcard' _ (un carattere arbitrario) e % (una stringa arbitraria), permette di esprimere dei pattern su stringa:

Example

Cognomi delle persone che finiscono con una 'i' ed hanno una 'i' in seconda posizione:

```
SELECT cognome
FROM persona
WHERE cognome  LIKE '_i%i';
```

Operatore IN

- L'operatore IN permette di esprimere condizioni di appartenenza ad un insieme

Example

```
SELECT cognome
FROM persona
WHERE id_persona  IN (5,18,30);
```

Funzioni su Data/Ora

Funzione	Descrizione
CURRENT_TIME	Ora corrente
CURRENT_DATE	Data corrente
CURRENT_TIMESTAMP	Data e ora corrente
...	...

Sommario

A titolo di ricapitolazione, vediamo come quanto visto a lezione ci permette di definire in SQL gli operatori dell'algebra relazionale:

- operatore di proiezione
- operatore di selezione
- operatore di prodotto cartesiano
- theta-join

Sommario

Traduzione dell'operatore di proiezione

L'operatore di proiezione $\Pi_{(A_1,\ldots,A_n)}(\textit{relazione})$ si traduce in SQL come:

SELECT DISTINCT $A_1, \ldots, A_n$ FROM *relazione*;

- DISTINCT permette di eliminare tuple identiche nel risultato.

Example

Ottenere un elenco di cognomi delle persone:

SELECT DISTINCT *cognome* FROM *persona*;

Sommario

Traduzione dell'operatore di selezione

L'operatore di selezione $\sigma_{(predicato)}(relazione)$ si traduce in SQL come:

SELECT * FROM *relazione* WHERE *predicato*;

Example

Selezionare i corsi con 2 crediti:

SELECT * FROM *corso* WHERE *crediti* = 2 ;

Sommario

Traduzione dell'operatore di prodotto cartesiano

L'operatore di prodotto cartesiano $relazione_1 \times relazione_2$ si puo' tradurre in SQL mediante la query:

$$\texttt{SELECT} \quad * \ \texttt{FROM} \quad relazione_1, relazione_2;$$

Sommario

Traduzione dell'operatore di theta-join

L'operatore di join:

$$relazione_1 \bowtie_{exp} relazione_2 \equiv \sigma_{exp}(relazione_1 \times relazione_2)$$

si puo' tradurre in SQL mediante la query:

$$\texttt{SELECT} * \texttt{FROM} \quad relazione_1, relazione_2$$
$$\texttt{WHERE} \quad exp;$$

Example

Produrre un elenco dei corsi con i relativi insegnanti, mediante un equi-join sulle tabelle *persona* e *corso*

$$\texttt{SELECT} * \texttt{FROM} \quad persona, corso$$
$$\texttt{WHERE} \quad persona.id\_persona = corso.id\_insegnante;$$

Bibliografia

Bibliografia ed Approfondimenti

- R.A.Elmasri, S.B. Navathe. Sistemi di Basi di Dati – Fondamenti: Capitolo 8 (8.4).
- Capitolo 6 (Data Manipulation) del manuale di PostgreSQL (http://www.postgresql.org/docs/manuals/)