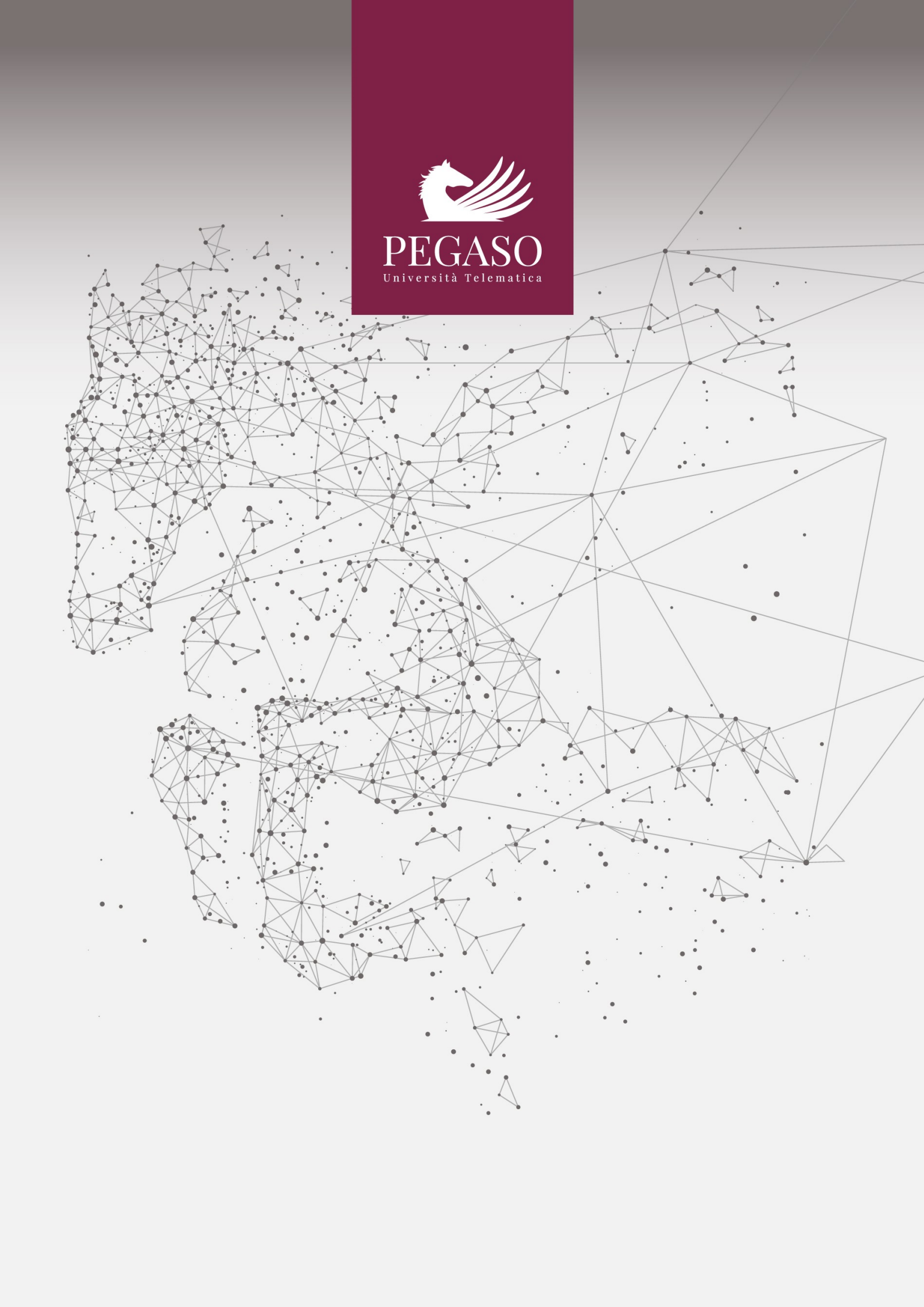
PEGASO
Università Telematica

Indice

1. Introduzione

Un'istruzione SELECT recupera zero o più righe da una o più tabelle di database del database. Nella maggior parte delle applicazioni, SELECT è il comando DQL (data query language) più comunemente utilizzato. Poiché SQL è un linguaggio di programmazione dichiarativo, le query SELECT specificano un set di risultati, ma non specificano come calcolarlo. Il database traduce la query in un "piano di query" che può variare tra esecuzioni, versioni di database e software di database. Questa funzionalità è denominata "Query Optimizer" in quanto è responsabile della ricerca del miglior piano di esecuzione possibile per la query, all'interno dei vincoli applicabili.

In questa unità didattica, si mostrano altre parti e operatori per costruire query più complesse.

2. Uso di variabili

Un alias è una funzionalità di SQL supportata dalla maggior parte, se non da tutti, dei sistemi di gestione dei database relazionali (RDBMS). Gli alias forniscono agli amministratori di database e ad altri utenti di database la possibilità di ridurre la quantità di codice richiesta per una query e semplificare la comprensione delle query. Inoltre, l'aliasing può essere utilizzato come tecnica di offuscamento per proteggere i nomi reali dei campi del database.

In SQL, si può dare un alias sia alle tabelle che alle colonne. Un alias di tabella è anche chiamato un nome di correlazione. Un programmatore può utilizzare un alias per assegnare temporaneamente un altro nome a una tabella o colonna per la durata di una query SELECT. Assegnare un alias in realtà non rinomina la colonna o la tabella. Questo è spesso utile quando le tabelle o le loro colonne hanno nomi molto lunghi o complessi. Un nome alias potrebbe essere qualsiasi cosa, ma in genere viene mantenuto breve. Ad esempio, potrebbe essere comune utilizzare un alias di tabella come "pi" per una tabella denominata "price_information".

Abbiamo precedentemente visto che nelle interrogazioni SQL è possibile associare un nome alternativo detto *alias* alle tabelle che compaiono come argomento della clausola FROM ed alle colonne che compaiono nella clausola SELECT. Gli alias SQL vengono quindi utilizzati per assegnare a una tabella o a una colonna in una tabella, un nome temporaneo. Per definire un alias basta la parola chiave **AS.**

Per un Alias nella Colonna, la sintassi è la seguente:

SELECT *column_name* AS *alias_name*

FROM *table_name;*

Mentre per un Alias nella tabella, la sintassi è:

SELECT *column_name(s)*

FROM *table_name* AS *alias_name;*

La motivazione del perché usare gli Alias, come detto precedentemente, è quella di avere un riferimento in modo più compatto, si evita di riscrivere il nome esteso della tabella.

Riprendiamo lo schema, formato dalle due tabelle Impiegato e Dipartimento di Figura 1 e Figura 2 e vediamo un esempio di utilizzo degli Alias.

Nome	Cognome	Dipart	Ufficio	Stipendio	Città
Mario	Rossi	Amministrazione	10	45	Milano
Carlo	Bianchi	Produzione	20	36	Torino
Giovanni	Verde	Amministrazione	20	40	Roma
Franco	Neri	Distribuzione	16	45	Napoli
Carlo	Rossi	Direzione	14	80	Milano
Lorenzo	Gialli	Direzione	7	73	Genova
Paola	Rosati	Amministrazione	75	40	Venezia
Marco	Franco	Produzione	20	46	Roma

Figura 1: tabella Impiegato.

Nome	Indirizzo	Città
Amministrazione	Via Livio 32	Milano
Produzione	Via Lavier 4	Torino
Distribuzione	Via Segre 9	Roma
Direzione	Via Livio 32	Milano
Ricerca	Via Venosa 6	Milano

Figura 2: tabella Dipartimento.

Interrogazione 17: estrarre tutti gli impiegati che hanno lo stesso cognome (ma diverso nome) di impiegati del dipartimento Produzione.

La query SQL che recupera i dati di interesse è la seguente:

- SELECT I1.Cognome, I1.Nome
- FROM Impiegato [AS] I1, Impiegato [AS] I2
- WHERE I1.Cognome=I2.Cognome AND
  - I1.Nome<>I2.Nome AND
  - I2.Dipart='Produzione'

Questa query provoca la duplicazione delle tabelle associate alle variabili I1 e I2. Infatti vengono create due variabili I1 e I2, ciascuna con tutte le righe della tabella Impiegato.

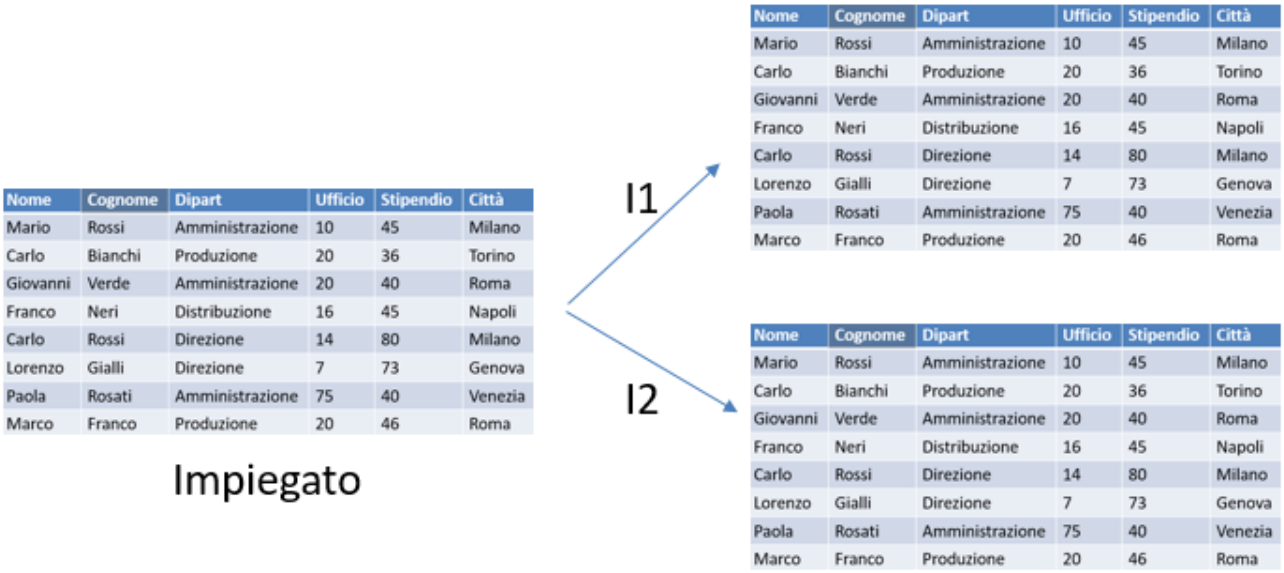

Figura 3: le due variabili utilizzate nella query.

Questa interrogazione confronta ciascuna riga di Impiegato con tutte le righe di Impiegato associate al dipartimento Produzione. In particolare, come si può notare dalla Figura 3, la suddetta query utilizza due variabili I1 e I2. È come se fossero create due tabelle uguali a quella di partenza. Vediamo adesso un altro esempio:

Interrogazione 18: estrarre il nome e lo stipendio dei capi degli impiegati che guadagnano più di 40:

- SELECT I1.Nome **AS** NomeC,I1.Stipendio **AS** StipC
- FROM Impiegati I1, Supervisione, Impiegati I2
- WHERE I1.Matricola=Supervisione.Capo AND
- I2.Matricola=Supervisione.Impiegato AND
- I2.Stipendio>40

Lasciamo al lettore l'individuazione del result-set.

3. Ordinamento

La clausola SELECT restituisce un result-set non ordinato, in linea con il fatto che una relazione è costituita da un insieme non ordinato di tuple. Sorge però spesso la necessità di costruire un ordine sulle righe delle tabelle:

- Es.: un utente vuole sapere quali sono gli stipendi più elevati che vengono erogati dall'azienda.

  È necessario quindi avere uno strumento SQL che torni i dati ordinati in base all'attributo Stipendio.

SQL permette di creare un ordinamento nel result-set attraverso la clausola: **ORDER BY.** La clausola SQL ORDER BY viene utilizzata quindi per ordinare i dati in ordine crescente o decrescente, in base a una o più colonne. Alcuni database ordinano i risultati della query in ordine crescente per impostazione predefinita. La clausola funziona nel modo seguente:

SELECT *column1*, *column2 ...*

FROM *table_name*

ORDER BY *column1, column2,* ASC|DESC;

Dove:

- ASC=ordine ascendente (default)
- DESC=ordine discendente

Dato lo schema di Figura 4, vogliamo costruire un'altra interrogazione.

Targa	Marca	Modello	NroPatente
KB 574 WW	Fiat	Punto	VR 2030020Y
GA 652 FF	Fiat	Panda	VR 2030020Y
BJ 747 XX	Lancia	Ypsilon	PZ 1012436B
ZB 421 JJ	Fiat	Uno	MI 2020030U

Figura 4: la tabella automobile.

Interrogazione 19: estrarre il contenuto della tabella Automobile ordinato in base alla marca (discendente) e al modello.

La query SQL sarà:

- SELECT *
- FROM Automobile
- **ORDER BY** Marca desc, Modello

  Da notare la presenza di due attributi nella clausola ORDER BY. Il risultato è illustrato in Figura 5.

Targa	Marca	Modello	NroPatente
BJ 747 XX	Lancia	Ypsilon	PZ 1012436B
GA 652 FF	Fiat	Panda	VR 2030020Y
KB 574 WW	Fiat	Punto	VR 2030020Y
ZB 421 JJ	Fiat	Uno	MI 2020030U

Figura 5: risultato della query.

4. Operatori aggregati

Nelle interrogazioni viste finora le condizioni di selezione (clausola Where) venivano valutate su ciascuna riga indipendentemente da tutte le altre. Si potrebbe ad esempio verificare quali dipartimenti hanno sede a Milano, dalla tabella Dipartimento.

Gli operatori aggregati costituiscono una delle più importanti estensioni di SQL rispetto all'algebra relazionale. In algebra relazionale tutte le condizioni vengono valutate su una tupla alla volta, poiché la condizione è infatti un predicato che viene valutato su ciascuna tupla indipendentemente da tutte le altre.

Nei contesti reali viene richiesto però di valutare delle proprietà che dipendono da insiemi di tuple e per tali motivi si introducono gli **operatori aggregati.** Vediamo un esempio:

Interrogazione 20*:* estrarre il numero di impiegati del dipartimento Produzione. La query è la seguente:

- SELECT **count(*)**
- FROM Impiegato
- WHERE Dipart='Produzione'

La soluzione è quella illustrata in Figura 6. Come si può notare, il risultato è un numero intero.

Nome	Cognome	Dipart	Ufficio	Stipendio	Città
Mario	Rossi	Amministrazione	10	45	Milano
Carlo	Bianchi	Produzione	20	36	Torino
Giovanni	Verde	Amministrazione	20	40	Roma
Franco	Neri	Distribuzione	16	45	Napoli
Carlo	Rossi	Direzione	14	80	Milano
Lorenzo	Gialli	Direzione	7	73	Genova
Paola	Rosati	Amministrazione	75	40	Venezia
Marco	Franco	Produzione	20	46	Roma

Figura 6: risultato della select con l'operatore count.

Lo standard SQL prevede 5 operatori aggregati:

- La funzione COUNT () che restituisce il numero di righe che corrisponde a un criterio specificato.
- La funzione AVG () che restituisce il valore medio di una colonna numerica.
- La funzione SUM () che restituisce la somma totale di una colonna numerica.
- La funzione MAX() che restituisce il massimo di una colonna numerica.
- La funzione MIN() che restituisce il minimo di una colonna numerica.

Bibliografia

- Atzeni P., Ceri S., Fraternali P., Paraboschi S., Torlone R. (2018). Basi di Dati. McGraw-Hill Education.
- Batini C., Lenzerini M. (1988). Basi di Dati. In Cioffi G. and Falzone V. (Eds). Calderini. Seconda Edizione.