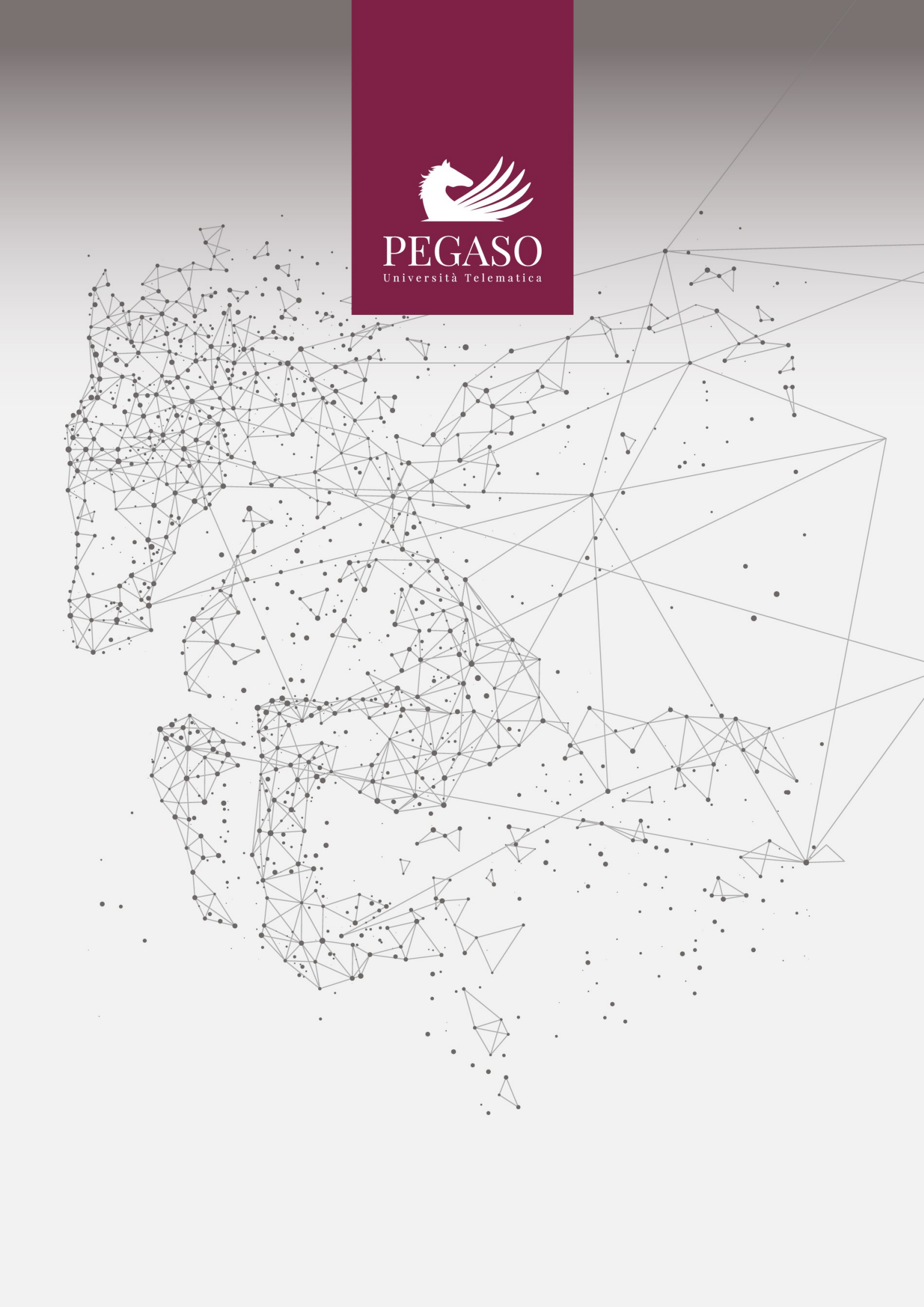
PEGASO
Università Telematica

Indice

1. Funzioni scalari

In questa sezione vengono introdotte le *Funzioni Scalari* ovvero un insieme di funzioni molto utili nel calcolo di alcuni valori scalari come ad esempio il calcolo della data, dell'ora, della lunghezza di una stringa, etc., etc. Per approfondimenti in merito, lo studente può far riferimento al sito https://www.w3schools.com/sql il quale rappresenta un ottimo strumento on-line per l'apprendimento di tutte le suddette funzioni ma anche di tutto il linguaggio SQL.

Cominciamo col dire che il linguaggio SQL mette a disposizione diverse funzioni che tornano valori elementari. Le modalità di funzionamento sono le seguenti:

- Possono essere usate all'interno delle espressioni del linguaggio che restituiscono valori di un dominio elementare in corrispondenza di ogni tupla su cui viene valutata la query;
- Restituiscono un valore semplice per ogni tupla;
- Sono suddivise in *famiglie* e dipendono dallo standard e dai dbms commerciali utilizzati.

Una suddivisione, non esaustiva, delle funzioni scalari è la seguente, come funzioni a livello di ennupla che restituiscono singoli valori:

- **Temporali**
  - current_date, extract(year from …)
- **Manipolazione stringhe**
  - char_length, lower, upper
- **Conversione**
  - cast
- **Condizionali**
  - …

1.1 Esempi di famiglie dello standard SQL-2

Vediamo alcuni esempi di utilizzo delle funzioni scalari, suddivise per gruppi o famiglie come segue:

- Funzioni temporali
  - current_date(): torna la data di sistema attuale
  - current_time(): torna l'ora attuale di sistema
  - current_timestamp(): torna data e ora nel formato timestamp

- Funzioni di manipolazioni di stringhe
  - lower(), upper(): Trasformano i caratteri di una stringa da maiuscolo a minuscolo e viceversa
  - char_lenght(): restituisce la lunghezza di una stringa
  - substring(): restituisce una parte di una stringa
- Funzioni di conversione di dominio
  - cast(): consente di convertire valori da un dominio ad un altro
    - Es: cast(data as char[10]) converte il valore di una data in una stringa di 10 caratteri

Esempi: (https://www.w3schools.com/sql/)

- SELECT CURRENT_DATE() AS Date;

Il risultato è espresso nella Tabella 1. Come si può notare, il result-set della query è una tabella contenente solo l'attributo Date (è stato usato AS) con la data come unico valore di dominio. Attenzione al formato della data (formato americano).

Date
2019-05-12

Tabella 1: result-set della query basata sdella funzione scalare Date().

Adesso vediamo l'utilizzo di un'altra funzione scalare:

SELECT LOWER("SQL Tutorial is FUN!") AS LowercaseText;

Il risultato è quello illustrato nella Tabella 2. Anche qui, come risultato si ha un attributo LowercaseText con un solo valore di dominio.

LowercaseText
sql tutorial is fun!

Tabella 2: result-set conseguenza dell'uso della funzione scalare LOWER().

2. Funzioni condizionali

Il linguaggio SQL prevede l'utilizzo delle cosiddette *Funzioni Condizionali* ovvero di alcune funzioni il cui obiettivo è quello di arricchire le modalità di costruzione delle query attraverso strumenti legati alla valutazione dei valori dei singoli attributi. Chi proviene dal mondo della programmazione o sta studiando tale disciplina, ritroverà all'interno di questa sezione concetti già noti.

Le funzioni condizionali sono funzioni che permettono di realizzare comandi SQL più compatti e facile da comprendere. Tra le caratteristiche ricordiamo:

- Non estendono il potere espressivo del linguaggio;
- Evitano si scrivere interrogazioni composte da unioni di tante interrogazioni più semplici.

Praticamente queste due caratterizzazioni ci dicono subito che queste funzioni ci facilitano la vita. Le famiglie di funzioni SQL-2 sono le seguenti:

- Coalescence
- Nullif
- Case

Vediamole adesso in dettaglio.

2.1 Funzione coalescence

La funzione coalescence ha le seguenti caratteristiche:

- Ammette come argomento una sequenza di espressioni;
- Restituisce il primo valore non nullo;

Vediamo un esempio.

***Interrogazione 58*:** estrarre i nomi i cognomi e i dipartimenti cui afferiscono gli impiegati, usando la stringa «ignoto» nel caso in cui non si conosca il dipartimento di appartenenza (campo NULL).

La tabella di partenza è la tabella Impiegato, rappresentata in Figura 1. Si noti come, volutamente, l'attributo Dipart contiene un valore NULL per l'impiegato Mario Rossi.

Nome	Cognome	Dipart	Ufficio	Stipendio	Città
Mario	Rossi	NULL	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.

Pertanto l'uso della funzione è il seguente:

SELECT nome, cognome, **coalescence**(Dipart,'Ignoto')

FROM Impiegato

Tale query produce come result-set la tabella di Figura 2.

Nome	Cognome	Dipart	Ufficio	Stipendio	Città
Mario	Rossi	Ignoto	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 2: result-set della query con la funzione coalescence().

Come si può notare, al posto del valore NULL è stato inserito il valore "Ignoto".

2.2 Funzione nullif

La funzione nullif() richiede come argomento una espressione e un valore costante; se l'espressione è pari al valore costante, la funzione restituisce il valore nullo altrimenti il valore dell'espressione. Vediamo adesso un esempio concreto del suo utilizzo.

Interrogazione 59: estrarre i nomi i cognomi e i dipartimenti cui afferiscono gli impiegati, restituendo il valore NULL per il dipartimento quando l'attributo Dipart contiene il valore 'Ignoto'.

La query è la seguente:

SELECT nome,cognome,**nullif**(Dipart,'Ignoto')

FROM Impiegato

Quando incontra il valore "Ignoto", nel result-set, per l'attributo selezionato, verrà impostato il valore a NULL, riproponendo in questo caso specifico la tabella di partenza. Praticamente questa funzione agisce in modo contrario alla funzione precedente coalescence().

2.3 Funzione case

La funzione case(): permette di specificare strutture condizionali all'intero di query più o meno complesse. Si supponga ad esempio di avere una tabella contenente informazioni sul tipo di veicolo definita attraverso il seguente schema:

Veicolo(Targa,Tipo,Anno,kWatt,Lunghezza,Nassi)

Interrogazione 60**:** Si vuole calcolare la tassa di circolazione dei veicoli immatricolati dopo il 1975.

Una possibile soluzione è quella illustrata in Figura 3. Come si può notare, la funzione case() è molto simile, se non uguale, alle stesse funzioni che si ritrovano in linguaggi di programmazione ad alto livello come ad esempio il linguaggio C. Il result-set di tale query sarà una tabella rappresentata da un solo attributo di nome "Tassa" dove, a seconda della tipologia di auto, si ha come valore il costo della tassa di circolazione. Come ultima osservazione, si noti che, nel caso in cui non dovessero essere soddisfatti i requisiti di "Auto" o "Moto", nel valore dell'attributo Tassa viene inserito il valore NULL.

```
select Targa,
        case Tipo
                when 'Auto' then 2.58 * KWatt
                when 'Moto' then (22.00 + 1.00 * KWatt)
                else null
        end as Tassa
    from Veicolo
    where Anno > 1975
```

Figura 3: la query dell'interrogazione 60.

3. Basi di dati attive

In questa sezione, viene illustrato il concetto di base di dati *Attiva,* ovvero di una base di dati dove, a seconda dell'operazione eseguita su di essa, la base di dati attiva una serie di operazioni.

Una base di dati attiva è una base di dati che contiene regole attive (chiamate *trigger*). Adesso, dopo la definizione, Andiamo a studiare in dettaglio tale proprietà.

3.1 I Trigger

I trigger seguono il paradigma: *Evento-Condizione-Azione,* ovvero:

- Quando un evento si verifica
- Se la condizione è vera
- Allora l'azione è eseguita

Questo modello consente computazioni reattive. Vediamo quindi questi tre concetti a che cosa corrispondono.

Ecco di seguito le definizioni:

- **Evento**
    - Normalmente una modifica dello stato del database: insert, delete, update
    - Quando accade l'evento, il trigger è *attivato*
- **Condizione**
    - Un predicato che identifica se l'azione del trigger deve essere eseguita
    - Quando la condizione viene valutata, il trigger è *considerato*
- **Azione**
    - Una sequenza di update SQL o una procedura
    - Quando l'azione è eseguita anche il trigger è *eseguito*

Lo standard SQL:1999 (SQL-3) sui trigger è stato fortemente influenzato da DB2 (IBM). Altri sistemi non seguono lo standard (esistono dagli anni 80'). Infine, vale la pena ricordare che la creazione dei trigger fa parte del DDL. Inoltre, I trigger presentano due livelli di granularità:

- Di tuple (row-level): Scatta sulla singola riga;
- Di primitive (statement-level): scatta per tutta la tabella.

Entrando più nel dettaglio di come si crea un trigger, diciamo che ogni trigger è caratterizzato da un insieme di attribute:

- nome
- target (tabella controllata)
- modalità (**before** o **after**)
- evento (**insert**, **delete** o **update**)
- granularità (statement-level o row-level)
- alias dei valori o tabelle di transizione
- azione
- timestamp di creazione

Vediamo un esempio in DB2 IBM:

```
create trigger ControllaStipendi
after update of Stipendio on Impiegato
for each row
when (new.Stipendio < old.Stipendio * 0.97)
begin
        update Impiegato
        set Stipendio = old.Stipendio * 0.97
        where Matr = new.Matr;
end;
```

Il suddetto trigger scatta ogni qualvolta dovesse essere modificato un valore dell'attributo Stipendio. Non entriamo più in dettaglio e lasciamo al lettore la possibilità di approfondire l'argomento sul proprio DBMS di riferimento.

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.

Sitografia

- https://www.w3schools.com/sql/sql_view.asp