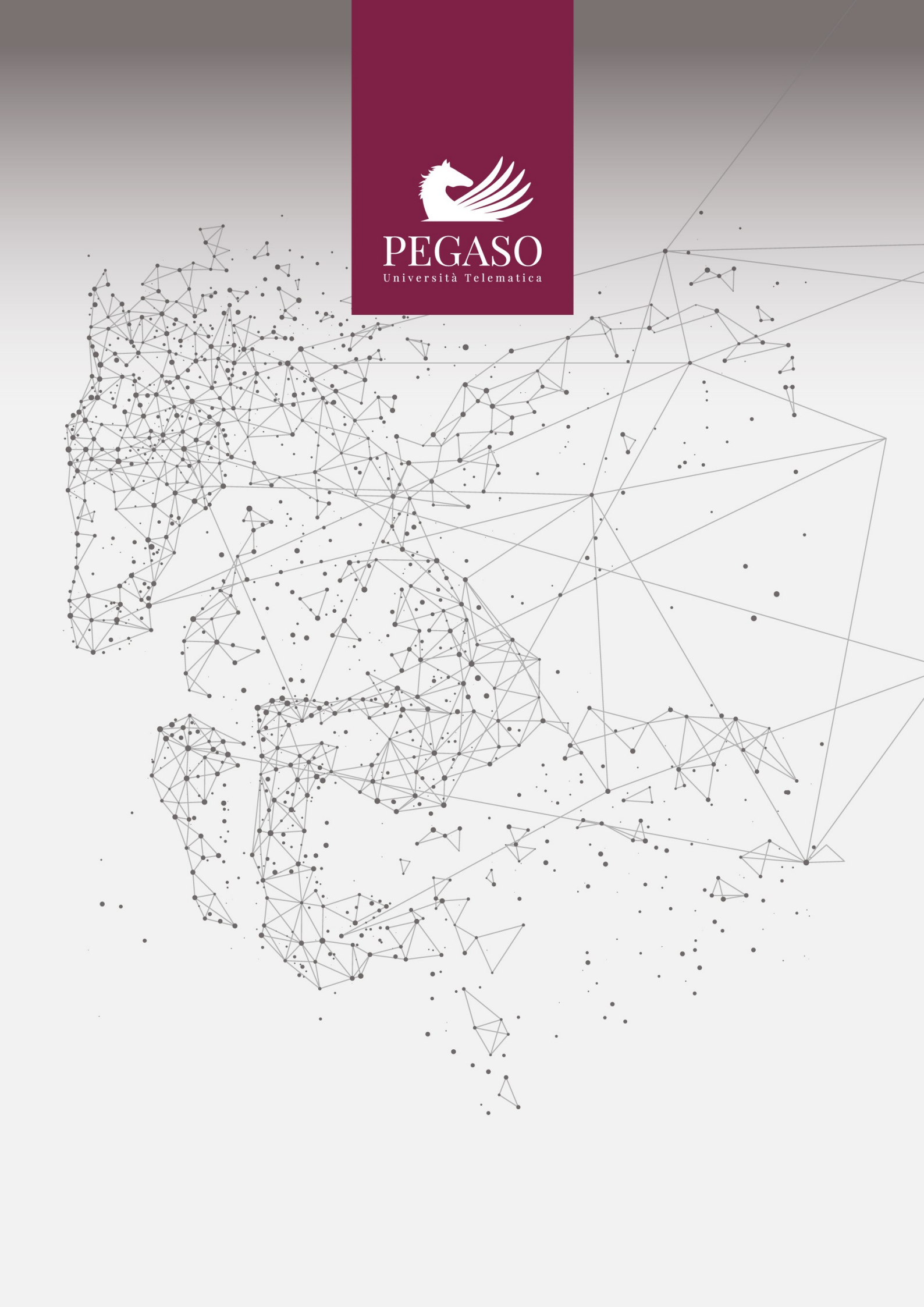
PEGASO
Università Telematica

Indice

1. Introduzione

Data Definition Language (DDL) è uno standard per i comandi che definiscono le diverse strutture o schemi in un database. Le istruzioni DDL creano, modificano e rimuovono oggetti di database come tabelle, indici e utenti. Le istruzioni DDL comuni sono CREATE, ALTER e DROP, peraltro già studiate in UDA precedenti.

I diagrammi degli schemi hanno una funzione importante perché costringono gli sviluppatori di database a trasporre le idee sulla carta. Ciò fornisce una panoramica dell'intero database, facilitando il lavoro futuro dell'amministratore del database.

In questa UDA studiamo proprio alcuni comandi SQL che consentono di imporre vincoli e modificare schemi di database.

2. Clausola CHECK

Il vincolo CHECK viene utilizzato per limitare l'intervallo di valori che può essere inserito in una colonna. Se si definisce un vincolo CHECK su una singola colonna, esso consente solo determinati valori per questa colonna. Se si definisce un vincolo CHECK su una tabella, è possibile limitare i valori a determinate colonne in base ai valori di altre colonne sulla riga[1]. Tale clausola specifica quindi vincoli di ennupla (e anche vincoli più complessi, non sempre supportati) in SQL-2. La sintassi è:

CHECK(Condizione)

Ad esempio, il seguente codice SQL crea un vincolo CHECK sulla colonna "Età" quando viene creata la tabella "Persone". Il vincolo CHECK garantisce che non è possibile avere una persona di età inferiore ai 18 anni. L'istruzione è illustrata in Figura 1.

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CHECK (Age>=18)
);
```

Figura 1: Un esempio di clausola CHECK.

Le condizioni ammissibili sono le stesse che possono apparire come argomento della clausola WHERE. La condizione deve essere sempre verificata affinché la base di dati sia corretta. Vediamo un altro esempio esplicativo.

[1] https://www.w3schools.com/sql/sql_check.asp

Esempio: creazione della tabella Impiegato con vincoli sugli attributi Sesso e Netto:

```
CREATE table Impiegato
(
        Matricola character(6),
        Cognome character(20),
        Nome character(20),
        Sesso character not null CHECK(sesso in ('M','F'))
        Stipendio integer,
        Ritenute integer,
        Netto integer,
        Superiore character(6),
        CHECK (Netto = Stipendio - Ritenute)
)
```

Figura 2: creazione della tabella Impiegato.

Il controllo in questo caso viene svolto al momento della modifica dati (cfr. in seguito).

Ancora un altro esempio: esprimere un vincolo che richiede che un impiegato abbia un manager del proprio dipartimento, a meno che la matricola non inizi con la cifra '1'.

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 3: tabella Impiegato.

Partendo quindi dalla tabella di Figura 2, vediamo come esprimere il vincolo richiesto. La soluzione è la seguente:

```
……
Superiore character (6)
CHECK (Matricola like '1%' or
Dipart=(SELECT Dipart
        FROM Impiegato  I
        WHERE I.Matricola=Superiore))
```

3. Asserzioni

Una asserzione specifica vincoli a livello di schema (SQL-2). L'istruzione è la seguente:

create assertion NomeAss check (Condizione)

Un esempio è questa istruzione:

create **assertion** AlmenoUnImpiegato

check (1 <= (select count(*) from Impiegato))

Le asserzioni permettono di esprimere vincoli particolari, come vincoli su più tabelle o vincoli che richiedono una cardinalità minima\massima per una tabella. Il seguente esempio è analogo al precedente:

Esempio: creare un vincolo che imponga che nella tabella Impiegato vi sia almeno una riga

CREATE ASSERTION AlmenoUna

(check (1 <= (SELECT count(*) FROM Impiegato)))

Esempio: Costruire la tabella Impiegato, assicurando che ogni Impiegato guadagni almeno 10.000 Euro. La sintassi dell'istruzione è illustrata in Figura 4.

```
CREATE TABLE Impiegato
(
        ID INT PRIMARY KEY,
        Nome VARCHAR(20),
        Età INT,
        Salario REAL,
        CHECK(Salario >= 10000)
)
```

Figura 4: creazione della tabella Impiegato con il vincolo CHECHK.

Valgono i seguenti accorgimenti:

- Ogni vincolo di integrità, definito tramite check o tramite asserzione, è associato ad una **politica di controllo** che specifica se il vincolo è immediato o differito.
- I vincoli immediati (valore di default) – sono verificati **dopo ogni operazione** che coinvolge le tabelle presenti nel vincolo di check o nell'asserzione.

- I vincoli differiti sono verificati **solo al termine dell'esecuzione di una serie di operazioni** (che costituisce una transazione).
- Quando si verifica una violazione di un vincolo differito al termine della transazione, non c'è modo di individuare l'operazione che ha causato la violazione.
- Diventa perciò necessario disfare l'intera sequenza di operazioni che costituiscono la transazione. in questo caso si esegue un **rollback**.

4. Le Viste

Le viste o view sono tabelle virtuali il cui contenuto dipende dalle altre tabelle di una base di dati.

Le righe non sono esplicitamente memorizzate nella base di dati, ma sono calcolate quando necessario. Si definisce una vista utilizzando il comando:

CREATE VIEW NomeVista [(ListaAttributi)] **AS** SelectSQL

[with [local | cascaded] **check** option]

Le viste sono caratterizzate dalle seguenti proprietà:

- Vengono definite associando un nome ed una lista di attributi al risultato dell'esecuzione di un'interrogazione;
- L'interrogazione interna (che può contenere anche altre viste) deve restituire un insieme di attributi pari a quelli contenuti nello schema della vista, nello stesso ordine;
- Una vista è una "relazione di cui viene memorizzata solo la definizione, piuttosto che l'insieme delle tuple";

Esempio: definire una vista *ImpiegatiAmmin* che contenga tutti gli impiegati amministrativi con uno stipendio superiore a 10. La vista è la seguente:

```
CREATE VIEW ImpiegatiAmmin(Matricola,Nome,Cognome,Stipendio)
AS SELECT Matricola,Nome,Cognome,Stipendio
FROM Impiegato
WHERE Dipart=Amministrazione' AND Stipendio>10
```

Mentre il risultato è quello mostrato in Figura 5.

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 5: la vista ImpiegatiAmmin.

4.1 Interrogazioni sulle viste

Sulle viste si possono fare normali interrogazioni come se fossero relazioni di base. Ad esempio l'istruzione:

select * from ImpiegatiAmmin

equivale a (e viene eseguita come)

select Matricola, Nome, Cognome, Stipendio
from Impiegato
where Dipart = 'Amministrazione' and
Stipendio > 10

Si ottiene la vista di Figura 6.

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: il result-set della select sulla vista.

4.2 Problematiche con le viste

La gestione delle viste richiede un minimo di accortezza in alcune operazioni. Infatti, lo standard SQL prevede che una vista sia aggiornabile solo quando una sola riga di ciascuna tabella di base corrisponde a una riga della vista. I sistemi commerciali considerano una vista aggiornabile solo se definita su una sola tabella. A questo scopo, la clausola CHECK OPTION specifica che sono ammessi aggiornamenti solo sulle righe della vista. Dopo ogni modifica tutte le righe devono continuare ad appartenere alla vista.

Vediamo un esempio: definire una vista *ImpiegatiAmminPoveri* che contiene tutti gli impiegati amministrativi con uno stipendio compreso tra 10 e 50. Il comando SQL è il seguente:

```
CREATE VIEW ImpiegatiAmminPoveri AS
SELECT *
FROM ImpiegatiAmmin
WHERE Stipendio<50
WITH CHECK OPTION
```

Il risultato è quello di Figura 7.

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
Paola	Rosati	Amministrazione	75	40	Venezia
Marco	Franco	Produzione	20	46	Roma

Figura 7: risultato delle select sulla vista.

Come ultima cosa, è importante osservare che ogni comando di aggiornamento fatto sulla vista, per poter essere propagato, non deve eliminare righe dalla vista. Quindi, in SQL, una vista è una tabella virtuale basata sul set di risultati di un'istruzione SQL. Una vista contiene righe e colonne, proprio come una tabella reale. I campi in una vista sono campi di una o più tabelle reali nel database. È possibile aggiungere le funzioni SQL, WHERE e JOIN a una vista e presentare i dati come se i dati provenissero da una singola tabella.

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