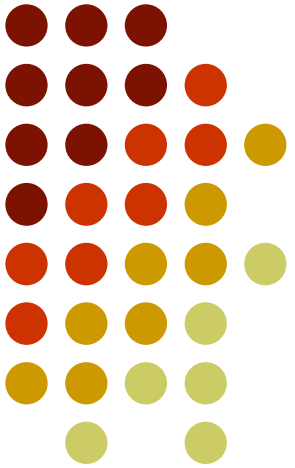
SQL: DML – vincoli di integrità & Viste

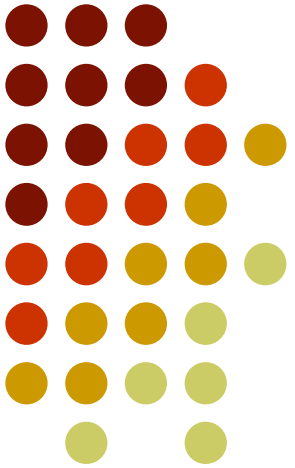
Elena Ferrari

Basi di Dati

A.A. 2020/2021



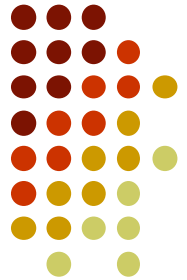
SQL: Vincoli di integrità



Vincoli di integrità

- In SQL, abbiamo già discusso la specifica di:
  - vincoli di obbligatorietà di colonne (NOT NULL)
  - vincoli di chiave (UNIQUE e PRIMARY KEY)
  - vincoli di integrità referenziale (chiavi esterne, FOREIGN KEY)

Vincoli di integrità



- SQL mette a disposizione anche altri costrutti per la specifica di generici vincoli di integrità
  - Nel comando CREATE TABLE è possibile definire:
    - **Vincoli CHECK su colonna**
    - **Vincoli CHECK su relazione**
  - Possibilità di definire asserzioni
- Vincoli check ed asserzioni sfruttano il linguaggio di query per definire le condizioni che le tuple devono soddisfare per verificare il vincolo

Vincoli CHECK su colonna

- Alla specifica della colonna viene affiancata la parola chiave CHECK seguita da una condizione, cioè un predicato o una combinazione booleana di predicati
- Tale condizione può anche contenere sotto-interrogazioni che fanno riferimento ad altre relazioni

Vincoli CHECK su colonna

- Esempi:

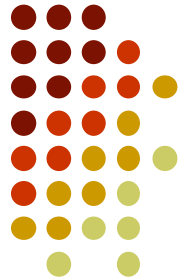
```
CREATE TABLE Film ( ...
valutaz DECIMAL(3,2) CHECK (valutaz BETWEEN 0.00 AND 5.00),
    ...);

CREATE TABLE Video ( ...
 tipo CHAR NOT NULL  CHECK (tipo IN ('d','v')),
    ...);
```

Vincoli CHECK su relazione

- Alla definizione di una relazione viene aggiunta la parola chiave CHECK seguita da un predicato o una combinazione booleana di predicati
  - la condizione può contenere sotto-interrogazioni che fanno riferimento ad altre tabelle
- Esempio:

```
CREATE TABLE Noleggio
        ( ...
         CHECK (dataRest >= dataNol));
```

Assegnare un nome ai vincoli



- È possibile assegnare un nome ai vincoli associati alle definizioni di relazioni facendo seguire la specifica del vincolo dalla parola chiave CONSTRAINT e dal nome
  - questa possibilità è prevista anche per i vincoli predefiniti
- Specificare un nome per i vincoli è utile per potervisi poi riferire (ad esempio per eliminarli, mediante ALTER TABLE DROP CONSTRAINT)

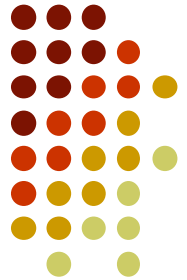
Esempio

```
CREATE TABLE  Video
  (colloc     DECIMAL(4) CONSTRAINT PKey PRIMARY KEY,
   titolo     VARCHAR(30) CONSTRAINT Tnn NOT NULL,
   regista    VARCHAR(20) CONSTRAINT Rnn NOT NULL,
   tipo       CHAR CONSTRAINT Snn NOT NULL DEFAULT 'd'
              CONSTRAINT Tok CHECK (tipo IN ('d','v')),
  CONSTRAINT FK FOREIGN KEY (titolo,regista));
```

Esempio

```
ALTER TABLE Video DROP CONSTRAINT Tok;

ALTER TABLE Video ADD CONSTRAINT Tok
        CHECK (tipo IN ('d','v','x'));

ALTER TABLE Video DROP CONSTRAINT Rnn;
```

Esempio



```
ALTER TABLE Video DROP CONSTRAINT Tok;

ALTER TABLE Video ADD CONSTRAINT Tok
        CHECK (tipo IN ('d','v','x'));

ALTER TABLE Video DROP CONSTRAINT Rnn;
```

Nota: l'aggiunta di un vincolo ad una relazione mediante ALTER TABLE ADD CONSTRAINT è possibile solo se tutte le tuple correntemente presenti nella relazione soddisfano il vincolo

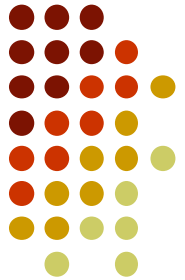
Vincoli CHECK

- Mediante l'uso di sotto-interrogazioni è possibile esprimere vincoli CHECK che verificano condizioni arbitrarie
- E' però consigliabile esprimere tramite vincoli CHECK solo condizioni che fanno riferimento a singole tuple della relazione cui associamo il vincolo:
  - migliore comprensibilità dello schema
  - maggiore efficienza nella verifica dei vincoli
- Condizioni che richiedano di esaminare più tuple della relazione o tuple di relazioni diverse andrebbero invece espresse tramite asserzioni (se il DBMS le supporta)

Asserzioni

- Sono elementi dello schema, manipolate da appositi comandi del DDL
- Servono per esprimere vincoli di integrità che coinvolgono più tuple o più relazioni
- Sintassi:

CREATE ASSERTION <nome asserzione>
CHECK (<condizione>);



Noleggio

colloc	dataNol	codCli	dataRest
1111	01-Mar-2006	6635	02-Mar-2006
1115	01-Mar-2006	6635	02-Mar-2006
1117	02-Mar-2006	6635	06-Mar-2006
1118	02-Mar-2006	6635	06-Mar-2006
1111	04-Mar-2006	6642	05-Mar-2006
1119	08-Mar-2006	6635	10-Mar-2006
1120	08-Mar-2006	6635	10-Mar-2006
1116	08-Mar-2006	6642	09-Mar-2006
1118	10-Mar-2006	6642	11-Mar-2006
1121	15-Mar-2006	6635	18-Mar-2006
1122	15-Mar-2006	6635	18-Mar-2006
1113	15-Mar-2006	6635	18-Mar-2006
1129	15-Mar-2006	6635	20-Mar-2006
1119	15-Mar-2006	6642	16-Mar-2006
1126	15-Mar-2006	6610	16-Mar-2006
1112	16-Mar-2006	6610	18-Mar-2006
1114	16-Mar-2006	6610	17-Mar-2006
1128	18-Mar-2006	6642	20-Mar-2006
1124	20-Mar-2006	6610	21-Mar-2006
1115	20-Mar-2006	6610	21-Mar-2006
1124	21-Mar-2006	6642	22-Mar-2006
1116	21-Mar-2006	6610	?
1117	21-Mar-2006	6610	?
1127	22-Mar-2006	6635	?
1125	22-Mar-2006	6635	?
1122	22-Mar-2006	6642	?
1113	22-Mar-2006	6642	?

Esempio

- Uno stesso video non può essere noleggiato contemporaneamente da due clienti:

```
CREATE ASSERTION SoloUno
CHECK (NOT EXISTS
        (SELECT colloc FROM Noleggio
            WHERE dataRest IS NULL
            GROUP BY colloc
            HAVING COUNT(*) > 1));
```

Controllo dei vincoli CHECK

- Un vincolo CHECK è violato se la valutazione della condizione di controllo restituisce FALSE come valore booleano di verità
- Cosa succede in presenza di valori nulli?
  - esempio: il vincolo CHECK(dataRest >= dataNol) per noleggi in corso?
- Questo problema connesso alla presenza di valori nulli si ha anche per decidere quali sono le tuple risultato di una interrogazione

Valori nulli

- SQL usa una logica a tre valori per valutare il valore di verità di una condizione di ricerca:
  - TRUE (T), FALSE (F), UNKNOWN (?)
- UNKNOWN (?) indica che il valore di verità di una condizione di ricerca applicata ad una data tupla non è determinabile
- Un predicato semplice valutato su un attributo a valore nullo dà come risultato della valutazione ?
- Il valore di verità di un predicato complesso viene calcolato in base alle tabelle di verità nel lucido successivo

Valori nulli

AND

	T	F	?
T	T	F	?
F	F	F	F
?	?	F	?

OR

	T	F	?
T	T	T	T
F	T	F	?
?	T	?	?

NOT

T	F
F	T
?	?

Esempio

- R

A	B	C	
a	?	c_1	t_1
a_1	b	c_2	t_2
a_2	?	?	t_3

- SELECT * FROM R WHERE A=a OR B=b;

  Il valore di verità della condizione per ogni tupla è il seguente:

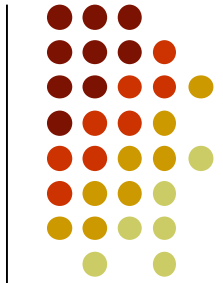
  t_1 T OR ? $\rightarrow$ T

  t_2 F OR T $\rightarrow$ T

  t_3 F OR ? $\rightarrow$?

  le tuple t_1 e t_2 verificano l'interrogazione

Valori nulli



- SELECT * FROM R WHERE A=a AND B=b;

  il valore di verità della condizione per ogni tupla è il seguente:

  t_1 T AND ? $\rightarrow$?

  t_2 F AND T $\rightarrow$ F

  t_3 F AND ? $\rightarrow$ F

  nessuna tupla verifica l'interrogazione

- SELECT * FROM R WHERE NOT C=c_1;

  il valore di verità della condizione per ogni tupla è il seguente:

  t_1 NOT T $\rightarrow$ F

  t_2 NOT F $\rightarrow$ T

  t_3 NOT ? $\rightarrow$?

  la tupla t_2 verifica l'interrogazione

Valori nulli

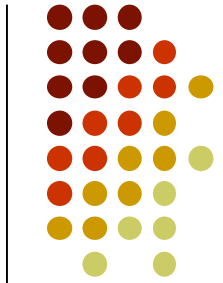
- Nelle espressioni (ad es. aritmetiche) se un argomento è NULL allora il valore dell'intera espressione è NULL
- Esempio: le tuple relative a noleggi correnti hanno durata (dataRest – dataNol) DAY indeterminata
- Nel calcolo delle funzioni di gruppo vengono escluse le tuple che hanno valore nullo per la colonna su cui la funzione è calcolata

  SUM(e1 + e2) può dare risultato diverso da SUM(e1) + SUM(e2)

Valori nulli

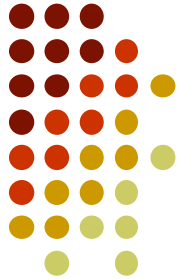
- Una funzione di gruppo può comunque restituire NULL se applicata ad un insieme vuoto di valori o contenente il solo valore NULL
- Se e1 ed e2 sono NULL, e1=e2 non è vero (è ?)

Valori nulli



- Una tupla per cui il valore di verità è ? non viene restituita dall'interrogazione
- Viceversa, in un vincolo di integrità se la valutazione della condizione di controllo restituisce ? il vincolo non è violato
  - il vincolo CHECK(dataRest >= dataNol) non è violato dai noleggi in corso

IS NULL



- Il predicato IS NULL applicato ad un attributo restituisce TRUE se la tupla ha valore nullo per l'attributo

Esempio

- R

A	B	C	
a	?	c_1	t_1
a_1	b	c_2	t_2
a_2	?	?	t_3

- SELECT * FROM R WHERE B IS NULL;

  restituisce le tuple t_1 e t_3
- SELECT * FROM R WHERE B IS NULL AND C IS NULL;

  restituisce la tupla t_3
- SELECT * FROM R WHERE B IS NULL OR C IS NULL;

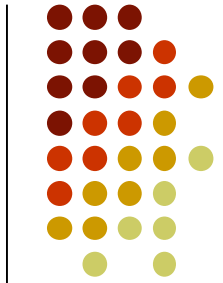
  restituisce le tuple t_1 e t_3

IS NOT NULL

- Il predicato IS NOT NULL applicato ad un dato attributo di una tupla restituisce TRUE se la tupla ha valore non nullo per l'attributo
- Esempio:
  - SELECT * FROM R WHERE B IS NOT NULL;

    restituisce la tupla t_2

Valori nulli



- SELECT colloc FROM Noleggio

  WHERE dataRest > CURRENT_DATE;

  non restituisce i noleggi in corso

- SELECT colloc FROM Noleggio

  WHERE dataNol > DATE ‘16-Nov-2020' OR dataRest > DATE ‘16-Nov-2020';

  restituisce anche noleggi in corso, purché siano iniziati dopo il 16 di Novembre 2020

- SELECT colloc FROM Noleggio

  WHERE NOT dataRest < CURRENT_DATE;

  non restituisce i noleggi in corso

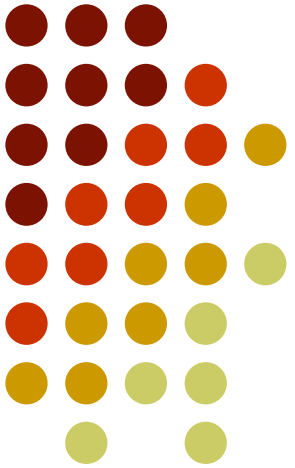
Valori nulli

- Le interrogazioni:
  - SELECT colloc FROM Noleggio
    WHERE dataRest = CURRENT_DATE OR
    NOT dataRest = CURRENT_DATE;
  - SELECT colloc FROM Noleggio
    WHERE dataRest = dataRest;

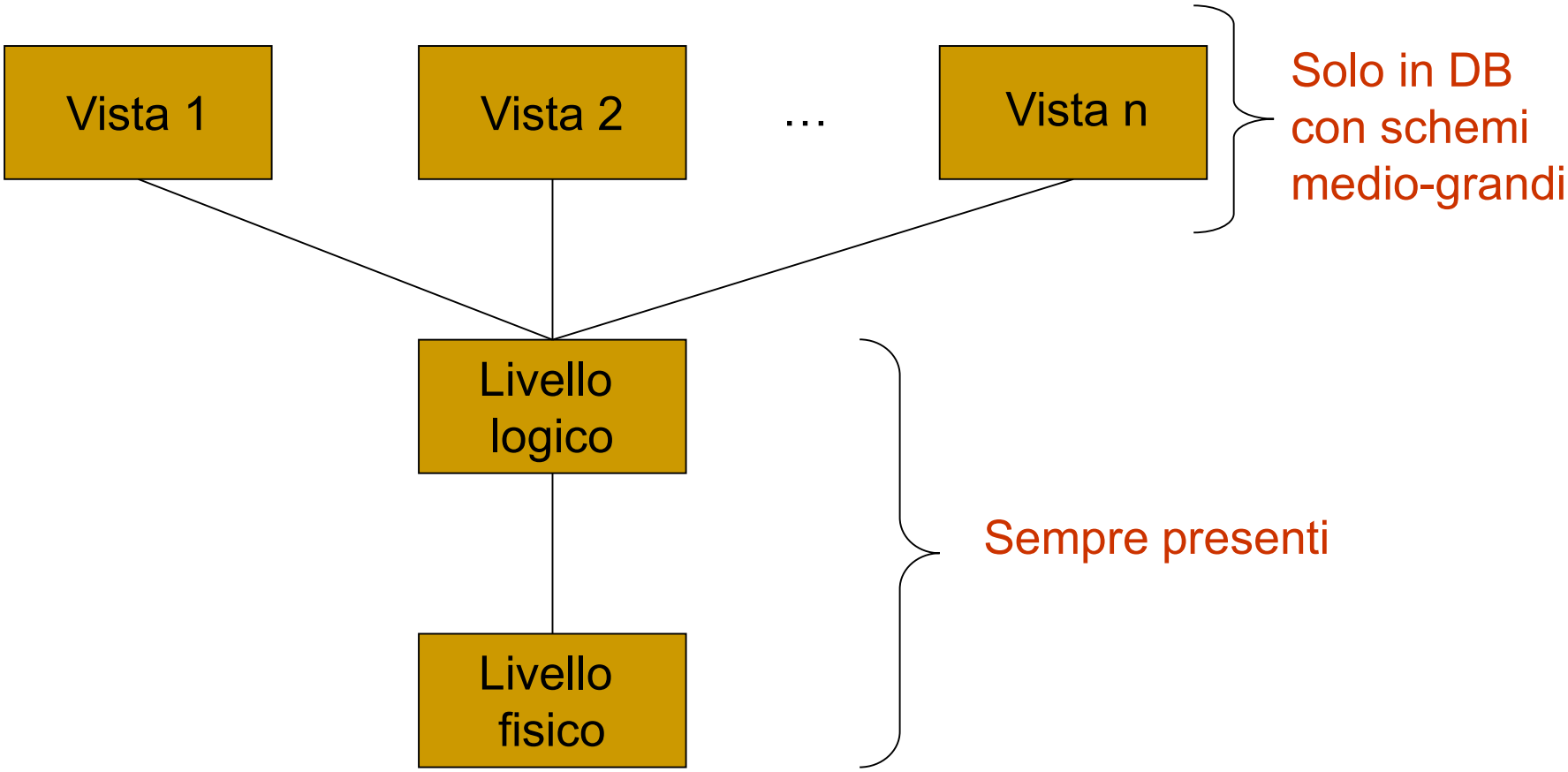
non restituiscono tutti i noleggi, ma solo i noleggi terminati, cioè:

SELECT colloc FROM Noleggio
WHERE dataRest IS NOT NULL;

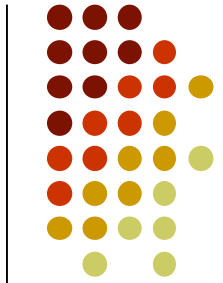
SQL: Viste



Livelli nella rappresentazione dei dati

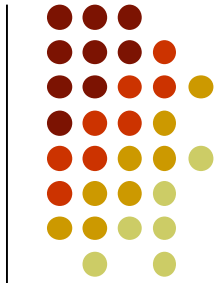


Viste



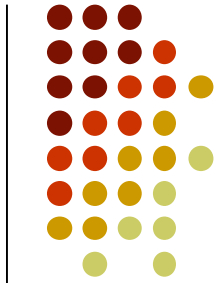
- Una vista è una relazione virtuale
  - il contenuto (tuple) è definito mediante un'interrogazione SQL sulla base di dati
    - il contenuto della vista dipende dal contenuto delle altre relazioni (di base) presenti nella base di dati
- il contenuto non è memorizzato fisicamente nella basi di dati
  - è ricalcolato tutte le volte che si usa la vista eseguendo l'interrogazione che la definisce

Viste



- Una vista è un oggetto della base di dati
  - può essere usata a quasi tutti gli effetti come una relazione di base
- Il meccanismo delle viste è utile per:
  - semplificare l'accesso ai dati
  - fornire indipendenza logica
  - garantire la protezione dei dati

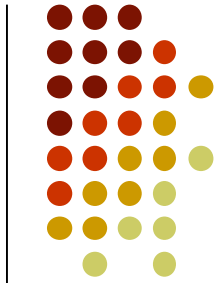
Viste



Comando creazione di viste:

```
CREATE VIEW <nome vista> [(<lista nomi colonne>)]
 AS <interrogazione>
 [WITH [{LOCAL| CASCADED}] CHECK OPTION];
```

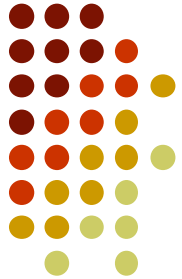
Viste



- <nome vista> è il nome della vista
- <interrogazione> è l'interrogazione di definizione della vista

  le colonne della vista corrispondono in numero e dominio alle colonne specificate nella clausola di proiezione di tale interrogazione
- <lista nomi colonne> è una lista di nomi da assegnare alle colonne della vista:
  - la specifica non è obbligatoria, tranne nel caso in cui l'interrogazione contenga nella clausola di proiezione colonne virtuali cui non è assegnato un nome

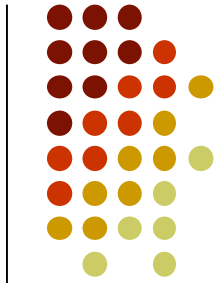
Viste



Cancellazione di una vista:

DROP VIEW <nome vista>;

Esempio

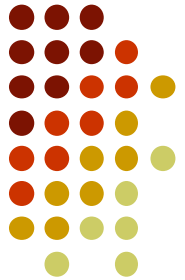


Vista contenente il codice cliente, la data di inizio noleggio e la collocazione dei video in noleggio da più di tre giorni:

```
CREATE VIEW Nol3gg AS
SELECT codCli, dataNol, colloc
FROM  Noleggio
WHERE dataRest IS NULL AND
    (CURRENT_DATE - dataNol) DAY > INTERVAL '3' DAY;
```

i nomi delle colonne della vista sono codCli, dataNol e colloc

Viste



- Nella definizione di viste è possibile usare tutte le funzionalità del linguaggio di interrogazione
- L'interrogazione di definizione di una vista può ad esempio contenere operazioni di join e fare uso di funzioni di gruppo ed espressioni:
  - **join**: può essere facile per alcuni utenti lavorare con una sola relazione piuttosto che eseguire join tra relazioni diverse (semplificazione dell'accesso ai dati)
  - **funzioni di gruppo**: può essere opportuno per alcuni utenti lavorare con dati aggregati piuttosto che con dati di dettaglio (garanzia di riservatezza)

Esempio

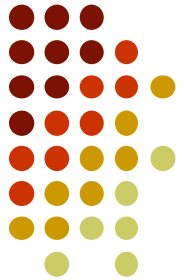
Vista che, per ogni cliente, contiene il codice, il numero di noleggi effettuati e la durata massima in giorni di tali noleggi:

```
CREATE VIEW InfoCli (codCli, numNol, durataM) AS
  SELECT codCli,COUNT(*), MAX((dataRest - dataNol) DAY)
  FROM Noleggio
  GROUP BY codCli;
```

Interrogazioni e aggiornamenti su viste

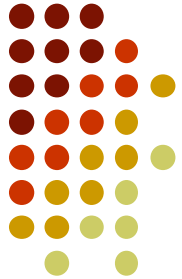
- Una volta definita, una vista è parte dello schema della base di dati e dovrebbe poter essere manipolata dall'utente "come" una relazione di base
- Su una vista si possono eseguire:
  - interrogazioni
  - aggiornamenti, sotto opportune condizioni

Interrogazioni su viste



- Su una vista ad esempio è possibile:
  - effettuare proiezioni
  - specificare condizioni di ricerca
  - effettuare dei join con altre relazioni o viste
  - effettuare raggruppamenti e calcolare funzioni di gruppo
  - definire altre viste

Esempio



Vogliamo determinare dalla vista InfoCli le informazioni relative al cliente 1119:

```
SELECT *
FROM InfoCli
WHERE codCli= 1119;
```

Esempio

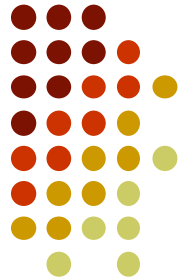
Vista creata su altra vista:

```
CREATE VIEW InfoCli2 AS
  SELECT codCli
   FROM InfoCli;
```

Aggiornamenti su viste

- L'esecuzione di un'operazione di aggiornamento su una vista deve poter essere propagata alle relazioni su cui la vista è definita
- Problemi:
  - in alcuni casi non è possibile realizzare l'operazione richiesta attraverso operazioni sulle relazioni di base oppure tale realizzazione non è univoca

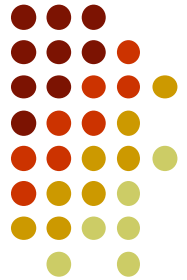
Aggiornamenti su viste



Problema (esempio modifica):

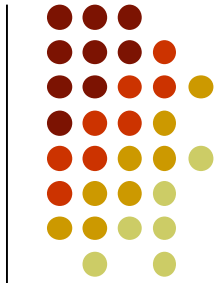
- Una modifica su una colonna di una vista viene realizzata tramite una modifica sulla colonna corrispondente nella relazione di base
- Se la colonna della vista è virtuale (cioè definita tramite un'espressione) non è possibile stabilire quale valore assegnare alla colonna (o alle colonne) corrispondente nella relazione di base per ottenere il valore specificato nella modifica sulla vista

Esempio



```
CREATE VIEW InfoNoleggio AS
 SELECT colloc, (dataRest - dataNol) DAY AS durata
 FROM Noleggio;
```

Aggiornamenti su viste



Problema (esempio inserimento):

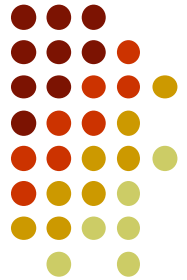
- Un inserimento su una vista viene realizzato tramite un inserimento sulla relazione di base
- Se la vista non contiene una colonna della relazione di base su cui è specificato un vincolo NOT NULL e per il quale non è specificato nello schema un valore di default:
  - Il comando di inserimento sulla vista non specifica un valore per tale colonna
  - La colonna è obbligatoria e senza valore di default

Aggiornamenti su viste

Problema 2 (esempio cancellazione):

- Una cancellazione su una vista viene realizzata tramite una cancellazione sulla relazione di base
- La cancellazione di una tupla da una vista definita come join di più relazioni di base può essere ottenuta:
  - mediante cancellazione da una delle relazioni di base
  - mediante cancellazione da tutte le relazioni di base
  - ponendo a NULL il valore dell'attributo di join in una o più di tali relazioni

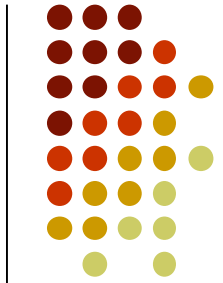
Aggiornamenti su viste



Sono permesse solo quelle operazioni di aggiornamento che sono mappabili in modo univoco in operazioni sulle relazioni di base su cui la vista è definita

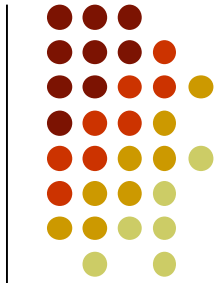
- molti DBMS consentono operazioni di aggiornamento solo su viste definite su singola relazione e ponendo delle restrizioni sulla query di formulazione (ad esempio non deve contenere GROUP BY…)

Check option



- Una vista essendo definita da un'interrogazione può contenere una condizione sul contenuto delle tuple delle relazioni su cui la vista è definita
- Solo le tuple che verificano tale condizione appartengono alla vista
- Un problema riguarda gli inserimenti nella vista di tuple che non verificano la condizione specificata dall'interrogazione di definizione della vista

Check option



- Esempio: supponiamo di voler inserire nella vista Nol3gg la tupla:

  (1128, CURRENT_DATE, 6635)

  - la data di noleggio specificata è oggi
  - la condizione nell'interrogazione (noleggio iniziato almeno tre giorni fa) non è verificata dalla nuova tupla
  - la tupla viene inserita in Noleggio ma non è poi ritrovata da interrogazioni sulla vista Nol3gg
- Per assicurare che le tuple inserite tramite una vista (o modificate tramite una vista) siano accettate solo se verificano la condizione nell'interrogazione di definizione della vista, si usa la clausola CHECK OPTION del comando CREATE VIEW

Esempio

- Se Nol3gg è definita come:

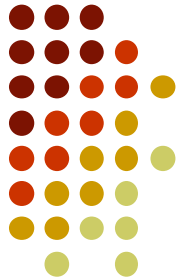
```
CREATE VIEW Nol3gg AS
 SELECT codCli, dataNol, colloc
 FROM  Noleggio
 WHERE dataRest IS NULL AND
    (CURRENT_DATE - dataNol) DAY > INTERVAL '3' DAY;
 WITH CHECK OPTION;
```

L'inserimento di tuple che non soddisfano l'interrogazione di definizione della vista, come (1128, CURRENT_DATE, 6635), non è permesso

Check option

- La situazione si complica nel caso di viste definite in termini di altre viste, perché ognuna di tali viste potrebbe essere definita con CHECK OPTION
- Per tale motivo, la CHECK OPTION può essere specificata con due possibili alternative: LOCAL e CASCADED (default):
  - LOCAL : verifica solo la query di definizione della vista in oggetto
  - CASCADED: verifica ricorsivamente tutte le query di definizione delle viste coinvolte

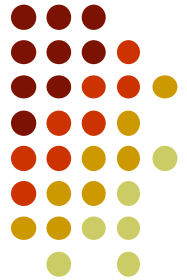
Esempio



P

CodP	NomeP	Colore	Taglia	Magazzino
P1	Maglia	Rosso	40	Torino
P2	Jeans	Verde	48	Milano
P3	Camicia	Blu	48	Roma
P4	Camicia	Blu	44	Torino
P5	Gonna	Blu	40	Milano
P6	Bermuda	Rosso	42	Torino

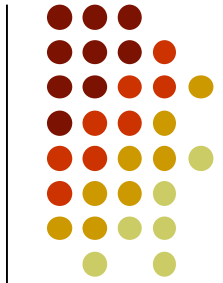
Esempio



```
CREATE VIEW PRODOTTI_TAGLIA_MEDIA_O_GRANDE(CodP,
NomeP, Taglia)   AS
SELECT CodP, NomeP, Taglia
FROM P
WHERE Taglia>=42 WITH CHECK OPTION;
```

- La vista è aggiornabile:
  - non si possono aggiornare le tuple presenti nella vista con valori di taglia minori di 42
  - non si possono inserire nella vista tuple con valori di taglia minori di 42

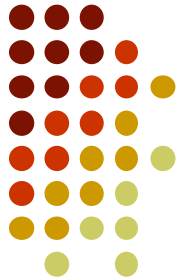
Esempio



```
CREATE VIEW PRODOTTI_TAGLIA_MEDIA (CodP, NomeP, Taglia)
AS
SELECT CodP, NomeP, Taglia
FROM PRODOTTI_TAGLIA_MEDIA_O_GRANDE
 WHERE Taglia<= 46 WITH CASCADED CHECK OPTION;
```

- La vista è aggiornabile:
  - si può aggiornare il contenuto della vista PRODOTTI_TAGLIA_MEDIA solo usando taglie comprese tra 42 e 46

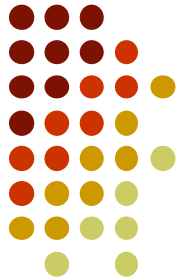
Esempio



P

CodP	NomeP	Colore	Taglia	Magazzino
P1	Maglia	Rosso	40	Torino
P2	Jeans	Verde	48	Milano
P3	Camicia	Blu	48	Roma
P4	Camicia	Blu	44	Torino
P5	Gonna	Blu	40	Milano
P6	Bermuda	Rosso	42	Torino

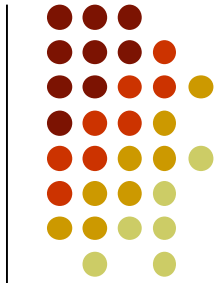
Esempio



UPDATE PRODOTTI_TAGLIA_MEDIA
SET Taglia=Taglia-2;

- Con CASCADED CHECK OPTION:
  - aggiornamento vietato a causa di PRODOTTI_TAGLIA_MEDIA_O_GRANDE

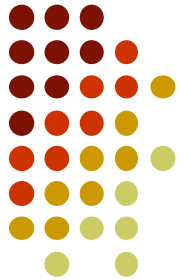
Esempio



```
CREATE VIEW PRODOTTI_TAGLIA_MEDIA (CodP, NomeP, Taglia)
AS
SELECT CodP, NomeP, Taglia
FROM PRODOTTI_TAGLIA_MEDIA_O_GRANDE
 WHERE Taglia<= 46 WITH LOCAL CHECK OPTION;
```

- La vista è aggiornabile:
  - il controllo è effettuato solo sulla vista PRODOTTI_TAGLIA_MEDIA
  - si può aggiornare con taglie <= 46

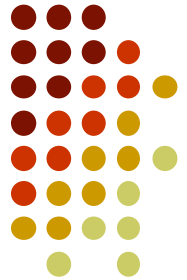
Esempio



```
UPDATE PRODOTTI_TAGLIA_MEDIA
SET Taglia=Taglia-2;
```

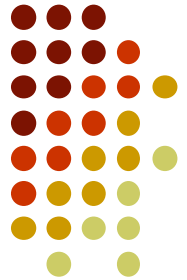
- Con LOCAL CHECK OPTION:
  - aggiornamento consentito

Caratteristiche di SQL non viste



- Alcune funzionalità non discusse:
  - SEQUENCE, per generare codici progressivi
  - colonne/relazioni derivate
  - nel linguaggio di interrogazione:
    - sotto-interrogazioni nella clausola FROM (danno luogo ad interrogazioni difficili da capire e da verificare)

Caratteristiche di SQL non viste



- Alcuni aspetti del modello dei dati di SQL che non sono stati trattati (vedi corso di modelli innovativi per la gestione dati):
  - caratteristiche object-relational:
    - tipi user-defined, tipi riga, tipi riferimento, tipi collezione
    - ereditarietà
  - trigger