


*SQL: DDL, VI,
Aggiornamenti e Viste*

SQL è più di un semplice linguaggio di interrogazione

- ❖ Linguaggio di definizione dati (*Data-definition language, DDL*):
 - Crea/distrugge/modifica *relazioni* e viste
 - Definisce *vincoli di integrità* (VI)
- ❖ Linguaggio di aggiornamento
 - Inserisce/cancella/modifica (aggiorna) tuple
 - Interagisce strettamente con i VI
- ❖ Controllo di accesso:
- ❖ Può garantire/revocare il diritto di accesso e di manipolazione delle tabelle (relazioni/viste)



Creazione di relazioni

```
CREATE TABLE Barche
```


```
(bid:integer, bnome:char(10), colore:char(10))
```

- ❖ Crea la relazione Barche che già conosciamo e amiamo. Sono mostrati tre campi, con nomi e tipi

```
CREATE TABLE Prenota
```

```
(vnome:char(10)), bid:integer, giorno:date)
```

- ❖ *Un piccolo cambiamento: Prenota usa vnome invece di vid*
- ❖ *Nessun VI è stato specificato (ne parleremo più avanti)*



Distruzione e modifica di relazioni

DROP TABLE Barche

- Distrugge la relazione Barche. Le informazioni sullo schema *e* le tuple vengono cancellate

ALTER TABLE Barche

ADD COLUMN tipobarca:char(10)

- Lo schema di Barche viene modificato con l'aggiunta di un nuovo campo; ogni tupla nell'istanza corrente viene estesa con un valore *null* nel nuovo campo.



Creazione di indici

`CREATE INDEX indNomeColore ON Barche(bnome, colore)`

- ❖ Crea un indice ad albero B+ su Barche, con (bnome, colore) come chiave di ricerca.

- Domanda: qual è l'ordinamento sulla base dell'albero?

- ❖ Questo comando **NON** è incluso nello standard SQL/92!


- La sintassi in genere differisce leggermente tra i sistemi

- Ad esempio, `CREATE INDEX indNomeColore ON Barche`

- `WITH STRUCTURE = BTREE, KEY = (bnome, colore)`

- ❖ Per eliminare un indice (Sybase)

- `DROP INDEX Barche.indNomeColore`



Vincoli di integrità (revisione)

- ❖ Un VI descrive condizioni che ogni *istanza legale* di una relazione deve soddisfare
 - Inserimenti/cancellazioni/aggiornamenti che violano i VI non sono permessi
 - Possono essere usati per garantire la semantica dell'applicazione (ad esempio, *vid* è una chiave), o per prevenire inconsistenze (ad esempio, *vnome* deve essere una stringa, *età* deve essere <200)
- ❖ Tipi di VI: vincoli di dominio, vincoli di chiave primaria, vincoli di chiave esterna, vincoli generali
 - Vincoli di dominio: i valori dei campi devono essere del tipo corretto. Sempre verificati.

Chiavi primarie e candidate (riviste)

- ❖ *Chiave* di una relazione: insieme minimo di campi tale che in ogni istanza legale due tuple distinte siano diverse nei valori dei campi chiave.
 - Eventualmente molte *chiavi candidate* (specificate usando UNIQUE), una delle quali viene scelta come *chiave primaria*
 - I campi della chiave primaria non possono contenere valori *null*

```
CREATE TABLE Prenota
(vnome CHAR(10)
bid INTEGER,
giorno DATE,
PRIMARY KEY (vnome, bid, giorno))
```


```
CREATE TABLE Prenota
(vnome CHAR(10) NOT NULL,
bid INTEGER,
giorno DATE,
PRIMARY KEY (bid, giorno)
UNIQUE (vnome))
```

Chiavi esterne (riviste)

- ❖ *Chiave esterna*: insieme di campi in una relazione R che viene usato per “far riferimento” a tuple in un’altra relazione S.
 - I campi dovrebbero essere una chiave (idealmente primaria) di S
 - Nelle tuple di R, i valori dei campi devono essere uguali ai valori di qualche tupla di S, oppure essere *null*

```
CREATE TABLE Barche
(bid INTEGER,
bnome CHAR(10),
colore CHAR(10),
PRIMARY KEY (bid))
```

```
CREATE TABLE Prenota
(vnome CHAR (10) NOT NULL,
bid INTEGER,
giorno DATE,
PRIMARY KEY (bid, giorno)
UNIQUE (vnome)
FOREIGN KEY (bid)
REFERENCES Barche)
```



Vincoli generali

- ❖ Utili quando sono coinvolti VI più generali delle chiavi
- ❖ Possono usare interrogazioni per esprimere il vincolo
- ❖ I vincoli possono avere un nome

```
CREATE TABLE Velisti
(vid INTEGER,
vnome CHAR(10),
esperienza INTEGER,
età REAL,
PRIMARY KEY (vid),
CHECK (esperienza >= 1
        AND esperienza <= 10))
```

```
CREATE TABLE Prenota
(vnome CHAR(10),
bid INTEGER,
giorno DATE,
PRIMARY KEY (bid, giorno),
CONSTRAINT noPrenInterlago
CHECK ('Interlago' <>
      (SELECT B.bnome
       FROM Barche B
       WHERE B.bid = bid)))
```

Vincoli su relazioni multiple

- ❖ Contorto e sbagliato!
- ❖ Se Velisti è vuota, il numero delle tuple di Barche può essere qualunque cosa!
- ❖ Una ASSERZIONE è la soluzione corretta; non è associata con alcuna tabella

```
CREATE TABLE Velisti
(vid INTEGER,
vnome CHAR(10),
esperienza INTEGER,
età REAL,
PRIMARY KEY (vid),
CHECK
((SELECT COUNT (V.bid) FROM Velisti S)
+ (SELECT COUNT (B.bid) FROM Barche B)
< 100))
```

*Numero di barche
più numero di velisti < 100*

```
CREATE ASSERTION piccoloClub
CHECK
((SELECT COUNT (V.bid) FROM Velisti B)
+ (SELECT COUNT (B.bid) FROM Barche B) < 100)
```

Inserimento di nuovi record

- Inserimento di un singolo record:

```
INSERT INTO Velisti (vid, vnome, esperienza, età)  
VALUES (12, 'Emanuele', 5, 21.0)
```

- Inserimento di più record:

```
INSERT INTO Velisti (vid, vnome, esperienza, età)  
SELECT S.vid, S.nome, null, S.età  
FROM Studenti S  
WHERE S.età >= 18
```

➡ *Un comando INSERT che causa una violazione di un VI viene rifiutato*



Cancellazione di record

- ❖ Si possono cancellare tutte le tuple che soddisfano una condizione in una clausola WHERE:

```
DELETE  
FROM Velisti V  
WHERE V.esperienza IS NULL
```

- ❖ L'esempio cancella tutti i velisti inesperti; in generale, la clausola WHERE può contenere interrogazioni annidate, etc.
- ❖ Che si dovrebbe fare quando una cancellazione viola un vincolo di chiave esterna?

Modifica di record

- ❖ Per modificare i campi di tuple esistenti si usa il comando UPDATE
- ❖ La clausola WHERE viene applicata per prima e determina i campi da modificare. La clausola SET determina i nuovi valori
- ❖ Se il campo da modificare è usato anche per determinare il nuovo valore, il valore usato nella WHERE è il *vecchio*.

```
UPDATE Velisti V
SET V.esperienza = V.esperienza - 1
WHERE V.esperienza < 15
```

<u>sid</u>	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
62	rusty	8	25.0
58	rusty	10	35.0



```
UPDATE Velisti V
SET V.esperienza = V.esperienza - 1
WHERE V.esperienza >= 8
```

<u>sid</u>	sname	rating	age
22	dustin	7	45.0
31	lubber	7	55.5
62	rusty	7	25.0
58	rusty	9	35.0



Garantire l'integrità referenziale

- ❖ Consideriamo Barche e Prenota; *bid* in Prenota è una chiave esterna che fa riferimento a Barche
- ❖ Che si dovrebbe fare se si inserisce una tupla di Prenota con l'id di una barca che non esiste? (*Rifiutarla!*)
- ❖ Che si dovrebbe fare se si cancella una tupla di Barche?
 - Cancellare anche tutte le tuple di Prenota che vi fanno riferimento
 - Non permettere la cancellazione di una tupla di Barche per la quale esistono riferimenti
 - Impostare a un valore predefinito il bid delle tuple di Prenota che vi fanno riferimento
 - Impostare a *null* le tuple di Prenota che vi fanno riferimento
- ❖ Stesse scelte se la chiave primaria delle tuple di Barche viene aggiornata.

Integrità referenziale in SQL/92

- ❖ SQL/92 supporta tutte le 4 opzioni sulle cancellazioni e sugli aggiornamenti
 - Il valore predefinito è NO ACTION (la cancellazione/la modifica vengono rifiutate)
 - CASCADE (cancella anche tutte le tuple che fanno riferimento alla tupla cancellata)
 - SET NULL / SET DEFAULT (imposta al valore nullo o ad un valore di default il valore della chiave esterna delle tuple che fanno riferimento alla tupla cancellata)

```
CREATE TABLE Prenota
(vnome CHAR(10) NOT NULL,
bid INTEGER DEFAULT 1000,
giorno DATE,
PRIMARY KEY (bid, giorno)
UNIQUE (vnome)
FOREIGN KEY (bid)
REFERENCES Barche
ON DELETE CASCADE
ON UPDATE SET DEFAULT)
```

Viste

- Una vista non è che una relazione, ma ne viene memorizzata la definizione, piuttosto che l'insieme di tuple

```
CREATE VIEW VelistiAttivi (nome, età, giorno)
AS SELECT V.vnome, V.età, P.giorno
FROM Velisti V, Prenota P
WHERE V.nome = P.vnome AND V.età > 6
```

- ❖ Le viste possono essere cancellate usando il comando DROP VIEW
 - Come gestire DROP VIEW se c'è una vista su una tabella?
 - Il comando DROP TABLE ha opzioni per permettere all'utente di specificarlo .



Interrogazioni su viste

- ❖ *Valutate usando una tecnica nota come modifica dell'interrogazione*

- *Il riferimento a una vista è sostituito dalla definizione di quest'ultima.*

```
SELECT A.nome, MAX(A.giorno)
FROM VelistiAttivi A
GROUP BY A.nome
```

- ❖ *Notate come *vnome* sia stato rinominato in *nome* per corrispondere alla definizione della vista*

```
SELECT nome, MAX(A.giorno)
FROM
(SELECT V.vnome AS nome, V.età, P.giorno
FROM Velisti V, Prenota P
WHERE V.vnome = P.vnome
      AND V.esperienza > 6) AS A
GROUP BY A.nome
```

Aggiornamenti su viste

- ❖ Le viste sono come relazioni di base sulle interrogazioni
- ❖ Non è vero per gli aggiornamenti!
 - Aggiornamento di vista □ nell'aggiornamento delle relazioni sottostanti
 - Qualcosa di ambiguo o addirittura impossibile!
 - Ad esempio: cancellare (solo) le tuple evidenziate dall'istanza A della vista *VelistiAttivi*

<u>sname</u>	<u>bid</u>	<u>day</u>
dustin	101	10/10/96
rusty	104	12/15/96
rusty	103	11/12/96

P

<u>sid</u>	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
62	rusty	8	25.0
58	rusty	10	35.0

V

<u>name</u>	<u>age</u>	<u>day</u>
dustin	45.0	10/10/96
rusty	25.0	12/15/96
rusty	25.0	11/12/96
rusty	35.0	12/15/96
rusty	35.0	11/12/96

A

Viste aggiornabili

- SQL/92 consente di aggiornare solo le viste su tabelle singole senza aggregazioni

```
CREATE VIEW VelistiGiovani (vid, età, esperienza)
AS SELECT V.vid, V.età, V.esperienza
FROM Velisti V
WHERE V.età < 18
```

- ❖ Ogni tupla della vista è generate da esattamente una tupla della relazione sottostante; quindi qualunque comando di aggiornamento/cancellazione sulla vista può facilmente essere *trasportato* sulla relazione.
- ❖ Dovrebbe essere consentito l'inserimento di (94, 22.0, 7)?
 - L'aggiunta di WITH CHECK OPTION alla definizione della vista non lo consentirebbe (in caso contrario è permesso)

Viste e sicurezza

- ❖ Le viste possono essere usate per presentare le informazioni necessarie (o un loro riassunto), nascondendo al contempo i dettagli della/e relazione/i sottostante/i
 - Data VelistiAttivi, ma non Velisti o Prenota, possiamo trovare i velisti che hanno una prenotazione, ma non i bid delle barche che sono state prenotate.
- ❖ I comandi GRANT/REVOKE possono essere usati per controllare l'accesso alle relazioni e alle viste
- ❖ Insieme all'abilità di definire le viste, questo fornisce un meccanismo di controllo di accesso molto potente



Riassunto del DDL di SQL

- ❖ Il DDL supporta la creazione di relazioni, viste e indici. Le tabelle possono anche essere modificate (aggiungendo o cancellando campi e VI)
- ❖ Le viste possono essere interrogate proprio come relazioni ordinarie, ma sono consentite solo forme limitate di aggiornamento



Riassunto (segue)

- ❖ In SQL/92 sono supportati molti tipi di vincoli di integrità
 - Vincoli di dominio, specifica di chiavi primarie e candidate, chiavi esterne, e vincoli generali su una o più relazioni
 - Vincoli di chiave esterna, in particolare, interagiscono da vicino con comandi di inserimento/cancellazione/modifica, e gli utenti hanno diverse scelte rispetto a questa interazione