Corso di Laurea in Ingegneria Gestionale
Sapienza - Università di Roma

Corso di Basi di Dati
A.A. 2019/2020

7 – SQL : Check, Asserzioni,Viste

Tiziana Catarci

Ultimo aggiornamento : 08/11/2019

Costrutti Avanzati di SQL

- **Vincoli di CHECK**
- **Asserzioni**
- **Viste**

Vincoli di CHECK

- SQL permette di specificare *vincoli di integrità generici* sugli attributi e le tabelle attraverso la clausola ***check*.**

CREATE TABLE *NomeTabella*(
NomeAttributo Dominio **[check (***Condizione***)]**
……..
[check (*Condizione***)]**
)

- Le condizioni utilizzabili sono le stesse che possono apparire come argomento della clausola **where** di un'interrogazione SQL.
- La condizione contenuta nel vincolo di *check* **deve essere sempre verificata** affinché la base di dati sia *corretta*.

Vincoli di CHECK - Esempio

TABELLE DI ESEMPIO :

Impiegato

Nome	Cognome	Dipart	StipAnn

Dipartimento

Nome	Città

ESEMPIO :

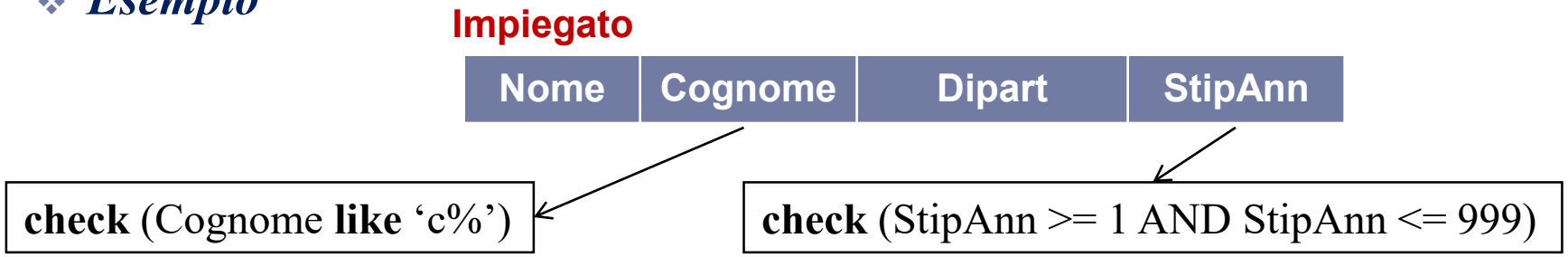
```
CREATE TABLE Impiegato(
Nome VARCHAR(20),
Cognome VARCHAR(20) check (Cognome LIKE 'c%'),
Dipart VARCHAR(20),
StipAnn INT check (StipAnn >= 1 AND StipAnn <= 999)
)
```

Nella creazione della tabella **Impiegato,** vengono accettate solo tuple in cui il valore dell'attributo ***Cognome*** inizia con la '**c**' e in cui lo ***Stipendio*** è compreso tra **1** e **999**.

Vincoli di CHECK - **ATTENZIONE**

- La condizione contenuta nel vincolo di *check* viene valutata **immediatamente dopo** l'inserimento\modifica di una tupla nella tabella in cui il vincolo è definito.

❖ ***Esempio***



INSERT INTO Impiegato(Nome,Cognome,Dipart,StipAnn)
VALUES ('Marco', 'Cesa', 'Produzione', 800)

L'inserimento va a **buon fine**, perché soddisfa tutti i vincoli (compresi quelli di *check*) specificati per la tabella.

Vincoli di CHECK - **ATTENZIONE**

- In caso di violazione del vincolo, l'operazione di modifica\inserimento che ha causato tale violazione viene "disfatta" dal sistema (**Rollback parziale**).
  - ***Esempio***

Impiegato

Nome	Cognome	Dipart	StipAnn

check (Cognome **like** 'c%')

check (StipAnn >= 1 AND StipAnn <= 999)

INSERT INTO Impiegato(Nome,Cognome,Dipart,StipAnn)
VALUES ('Andrea', 'Marrella', 'Amministrazione', 800)

Inizialmente l'inserimento della tupla viene effettuato senza verificare i vincoli di *check* specificati per la tabella.

A questo punto si verifica se l'inserimento ha violato qualche vincolo. Dato che il valore 'Marrella' non inizia con la 'c', un vincolo di *check* è stato violato. La tupla viene eliminata dalla tabella.

Rollback parziale

Nome	Cognome	Dipart	StipAnn
Andrea	Marrella	Amministrazione	800

Vincoli di CHECK

- Il vincolo di *check* è un costrutto molto potente :
  - la condizione specificata nel vincolo può far riferimento ad altri attributi della stessa tabella oppure ad **attributi in tabelle differenti**.
  - tutti i vincoli predefiniti (PRIMARY KEY, UNIQUE, FOREIGN KEY, …) possono essere espressi attraverso il vincolo di *check*.

Vincoli di CHECK - Esempio

ESEMPIO **:**

```
CREATE TABLE Impiegato(
Nome VARCHAR(20),
Cognome VARCHAR (20),
Dipart VARCHAR (20),
StipAnn INT,
constraint checkCognome
check (Cognome like 'c%' OR
        Dipart in (SELECT NomeDip
                   FROM Dipartimento
                   )
       )
)
```

Ad ogni vincolo di check si può associare un nome (attraverso la clausola **constraint**).

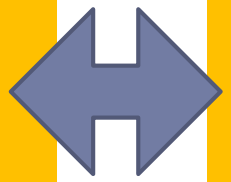
Un vincolo di check può essere espresso come un'interrogazione.

Nella creazione della tabella **Impiegato,** vengono accettate solo tuple in cui il valore dell'attributo ***Cognome*** inizia con la **'c'** oppure in cui il dipartimento a cui afferisce l'impiegato è contenuto nella tabella ***Dipartimento***.

Vincoli di CHECK - Esempio

ESEMPIO **:** Rappresentare il vincolo di **PRIMARY KEY** sugli attributi *Nome* e *Cognome* utilizzando un vincolo di check

```
CREATE TABLE Impiegato(
Nome VARCHAR (20),
Cognome VARCHAR (20),
Dipart VARCHAR (20),
StipAnn INT,
PRIMARY KEY (Nome,
Cognome)
)
```



```
CREATE TABLE Impiegato(
Nome VARCHAR(20),
Cognome VARCHAR(20),
Dipart VARCHAR(20),
StipAnn INT,
check (Nome is not null AND
        Cognome is not null AND
        1 >= (SELECT count(*)
              FROM Impiegato I
              WHERE Nome = I.Nome
                    and
                    Cognome =
                          I.Cognome
              )
        )
)
```

E' preferibile l'utilizzo dei vincoli predefiniti, che permettono una rappresentazione più compatta. Infatti, in questo caso, il vincolo di chiave primaria richiede una rappresentazione complicata con l'uso dell'operatore aggregato *count*.

Vincoli di CHECK - Esempio

ESEMPIO **:** Rappresentare il vincolo di **CHIAVE (unique)** sugli attributi *Nome* e *Cognome* utilizzando un vincolo di check

```
CREATE TABLE Impiegato(
Nome VARCHAR(20),
Cognome VARCHAR(20),
Dipart VARCHAR(20),
StipAnn INT,
UNIQUE(Nome, Cognome)
)
```

```
CREATE TABLE Impiegato(
Nome VARCHAR(20),
Cognome VARCHAR(20),
Dipart VARCHAR(20),
StipAnn INT,
check (1 >= (SELECT count(*)
             FROM Impiegato I
             WHERE Nome = I.Nome
                   and
                   Cognome =
                         I.Cognome
        )
    )
)
```

Vincoli di CHECK - **ATTENZIONE**

- Quando un vincolo di ***check*** coinvolge due o più tabelle, possono emergere comportamenti anomali non desiderati.

```
CREATE TABLE Impiegato(
Nome VARCHAR (20),
Cognome VARCHAR (20),
Dipart VARCHAR (20),
StipAnn INT,
check ( (SELECT count(distinct Dipart) FROM Impiegato)  +
          (SELECT count(Nome) FROM Dipartimento) <100
        )
)
```

- Se inseriamo nella tabella **Dipartimento** un numero di Dipartimenti superiore a 100, il vincolo rimane soddisfatto, dato che il controllo viene effettuato **solo** quando inseriamo tuple dentro la tabella **Impiegato**...ciò **non è soddisfacente** per le nostre richieste.

Asserzioni

- Per evitare eventuali spiacevoli anomalie comportate dai vincoli di ***check*** su più tabelle, si possono utilizzare le ***asserzioni***.
- Le asserzioni rappresentano vincoli non associati a nessun attributo o tabella in particolare, ma appartengono **direttamente** allo schema della Base di Dati.

CREATE ASSERTION *NomeAsserzione*
check (*Condizione***)**

- Le asserzioni permettono di esprimere vincoli particolari, come vincoli su più tabelle o vincoli che richiedono una cardinalità minima\massima per una tabella.

Asserzioni - Esempio

ESEMPIO **:**

```
CREATE ASSERTION NonPiùDi100(
check ( (SELECT count(distinct Dipart) FROM Impiegato)  +
          (SELECT count(Nome) FROM Dipartimento) < 100
        )
)
```

- Ad esempio, se il numero di valori contenuti nell'attributo ***Nome*** della tabella **Dipartimento** è superiore a 100 e la tabella **Impiegato** è vuota, il vincolo **non risulta soddisfatto**.

ESEMPIO **: Creare un vincolo che imponga che nella tabella *Impiegato* vi sia almeno una riga**

```
CREATE ASSERTION AlmenoUna(
check (1 <= (SELECT count(*)
                FROM Impiegato)
        )
)
```

Asserzioni – Esercizio – 1\3

ESERCIZIO **: Consideriamo la base di dati costituita dalle seguenti relazioni**

Impiegato

ID	Nome	Età	Salario

Dipartimento

ID	Budget	ManagerID

- 1) Costruire la tabella Impiegato, assicurando che ogni Impiegato guadagni almeno 10.000 Euro.

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

Asserzioni – Esercizio – 2\3

ESERCIZIO **: Consideriamo la base di dati costituita dalle seguenti relazioni**

Impiegato

ID	Nome	Età	Salario

Dipartimento

ID	Budget	ManagerID

- 2) Costruire un vincolo che assicuri che tutti i manager abbiano età > 30

```
CREATE TABLE Dipartimento(
ID INT PRIMARY KEY,
Budget REAL,
ManagerID INT,
FOREIGN KEY (ManagerID) REFERENCES Impiegato
check ( 30 < (SELECT I.Età
              FROM Impiegato I
              WHERE I.ID = ManagerID
             )
        )
)
```

NON VA BENE!
Il vincolo può essere facilmente aggirato. Infatti basta aggiornare l'età di un manager direttamente nella tabella **Impiegato**, senza violare alcun vincolo di **Dipartimento.**

Asserzioni – Esercizio – 3\3

ESERCIZIO **: Consideriamo la base di dati costituita dalle seguenti relazioni**

Impiegato

ID	Nome	Età	Salario

Dipartimento

ID	Budget	ManagerID

- 2) Costruire un vincolo che assicuri che tutti i manager abbiano età > 30.

```
CREATE TABLE Dipartimento(
ID INT PRIMARY KEY,
Budget REAL,
ManagerID INT,
FOREIGN KEY (ManagerID)
        REFERENCES Impiegato
)
```

E' preferibile l'utilizzo di un'asserzione, che effettua il controllo ogni qualvolta una delle due tabelle coinvolte viene aggiornata.

```
CREATE ASSERTION EtàManager (
check(30 < (SELECT I.Età
            FROM Impiegato I,
                 Dipartimento D
            WHERE I.ID =
                  D.ManagerID
           )
     )
)
```

Politiche di controllo nei vincoli di integrità

- 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*.
  - Quando un vincolo immediato non è soddisfatto, l'operazione di modifica che ha causato la violazione è stata appena eseguita e il sistema può disfarla. Questo modo di procedere è chiamato **Rollback Parziale** (guardare l'esempio a pag. 6).
  - Tutti i vincoli predefiniti (PRIMARY KEY, UNIQUE, FOREIGN KEY, NOT NULL) sono verificati in modo **immediato**.

Politiche di controllo nei vincoli di integrità

- I vincoli differiti sono verificati solo al termine dell'esecuzione di una serie di operazioni (che costituisce una ***transazione***).
- Il controllo ***differito*** viene tipicamente introdotto per gestire situazioni in cui non è possibile costruire una situazione consistenze con una singola modifica della base di dati.
  - Si pensi, ad esempio, ad una tabella **Dipartimenti** il cui attributo *Direttore* è associato ad un vincolo di integrità referenziale verso una tabella **Impiegato**, e la tabella **Impiegato** presenta a sua volta un attributo *Dipart* associato ad un vincolo di integrità referenziale verso la tabella **Dipartimenti**.
  - Se entrambi i vincoli fossero immediati, non sarebbe possibile modificare lo stato iniziale vuoto delle due tabelle, in quanto ogni singolo comando di inserimento di tuple non rispetterebbe il vincolo di integrità referenziale.
  - Il controllo differito permette di gestire agevolmente questa situazione.

Politiche di controllo nei vincoli di integrità

- 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**.
- Grazie ai meccanismi di controllo *immediato\differito*, l'esecuzione di un comando di modifica dell'istanza della base di dati che soddisfa tutti i vincoli, produrrà una nuova istanza **consistente** della base di dati (cioè, che soddisfa a sua volta tutti i vincoli).

Viste

- SQL permette di specificare *tabelle virtuali* in cui le righe non sono esplicitamente memorizzate nella base di dati, ma sono calcolate quando necessario → le ***viste.***

CREATE VIEW *NomeVista* **[** (*ListaAttributi*) **]**
AS SelectSQL
[with check option]

- Le viste 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".***

Viste – Esempio

TABELLE DI ESEMPIO **:**

Impiegato

Nome	Cognome	Dipart	StipAnn

Dipartimento

Nome	Città

ESEMPIO **: Definire una vista ImpiegatiAmmin che contiene tutti gli impiegati del dipartimento Amministrazione con uno stipendio superiore a 10 mila euro**

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

Viste – Esempio

TABELLE DI ESEMPIO **:**

Impiegato

Nome	Cognome	Dipart	StipAnn

Dipartimento

Nome	Città

ESEMPIO **: Definire una vista ImpiegatiAmminPoveri definita a partire dalla vista ImpiegatiAmmin, che contiene tutti gli impiegati amministrativi con uno stipendio compreso tra 10 mila e 50 mila euro**

```
CREATE VIEW ImpiegatiAmminPoveri  as
SELECT *
FROM ImpiegatiAmmin
WHERE StipAnn < 50
```

Viste aggiornabili

- SQL-92 permette di aggiornare **solo quelle viste definite su una sola tabella di base** e che per la loro definizione **non usano operatori di aggregazione** o il comando ***distinct.***
- Queste restrizioni relative alla politica aggiornamenti sono **FONDAMENTALI** per rendere l'aggiornamento alla tabella sottostante non ambiguo.
  - A ciascuna riga della tabella di base corrisponderà una sola riga della vista (le viste mantengono i duplicati).
  - Un'operazione di modifica\cancellazione nella vista deve soddisfare tutti i vincoli di integrità della tabella sottostante, altrimenti non viene eseguita.

ESEMPIO **:**

```
INSERT INTO ImpiegatiAmmin (Nome,Cognome,Dipart,StipAnn)
VALUES('Marco','Grigi','Amministrazione','40')
```

La tupla viene inserita correttamente nella vista **ImpiegatiAmmin** e nella relazione sottostante **Impiegato**.

Viste aggiornabili

ESEMPIO :

```
INSERT INTO ImpiegatiAmmin (Nome,Cognome,Dipart,StipAnn)
VALUES('Marco','Grigi','Amministrazione',7)
```

La tupla viene inserita correttamente nella relazione sottostante **Impiegato,** ma non nella vista **ImpiegatiAmmin,** che non accetta Stipendi < 10.

Gli inserimenti che non soddisfano le condizioni con cui è stata definita una vista possono essere disabilitati attraverso la clausola **with check option**. In questo caso saranno accettati solo inserimenti di righe che continuano ad apparire nella vista.

ESEMPIO :

```
CREATE VIEW ImpiegatiAmmin( Nome,Cognome,
                            Dipart,StipAnn ) as
SELECT Nome,Cognome,Dipart,StipAnn
FROM Impiegato
WHERE Dipart = 'Amministrazione' and StipAnn > 10
with check option
```

In questo caso l'inserimento della tupla relativa all'esempio precedente **verrebbe rifiutato**.

Viste – Esempio 1\7

TABELLA DI ESEMPIO **:**

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESEMPIO **: Definire una vista DipartimentiMilano che contiene tutti i nomi dei dipartimenti che si trovano a Milano, con una politica di inserimento controllato**

```
CREATE VIEW DipartimentiMilano ( Nome ) as
SELECT Nome
FROM Dipartimento
WHERE Città='Milano'
with check option
```

```
SELECT * FROM DipartimentiMilano
```

DipartimentiMilano

Nome
Amministrazione
Direzione
Ricerca

Viste – Esempio 2\7

Dipartimento

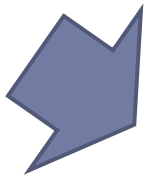
Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

DipartimentiMilano

Nome
Amministrazione
Direzione
Ricerca

ESEMPIO **: Inserire la tupla 'Sviluppo' nella vista DipartimentiMilano**

INSERT INTO DipartimentiMilano (Nome)
VALUES ('Sviluppo')



L'inserimento fallisce, perché la riga verrebbe correttamente inserita all'interno di **Dipartimento** (con il valore di ***Città*** posto a NULL), ma tale risultato non sarebbe visibile in **DipartimentiMilano** (data la presenza di ***with check option*** nella creazione della vista).

Viste – Esempio 3\7

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

DipartimentiMilano

Nome
Amministrazione
Direzione
Ricerca

ESEMPIO **: Inserire la tupla NULL nella vista DipartimentiMilano**

INSERT INTO DipartimentiMilano (Nome)
VALUES (NULL)

L'inserimento fallisce, perché si vìola il vincolo di PRIMARY KEY nella tabella sottostante **Dipartimento**.

Viste – Esempio 4\7

TABELLA DI ESEMPIO **:**

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESEMPIO **: Definire una vista DipartimentiD che contiene tutti i nomi dei dipartimenti che hanno nome che inizia con la 'D'**

```
CREATE VIEW DipartimentiD (Nome) as
SELECT Nome
FROM Dipartimento
WHERE Nome like 'D%'
```

```
SELECT * FROM DipartimentiD
```

DipartimentiD

Nome
Distribuzione
Direzione

Viste – Esempio 5\7

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

DipartimentiD

Nome
Distribuzione
Direzione

ESEMPIO **: Inserire la tupla 'DirezioneAvanzata' nella vista DipartimentiD**

INSERT INTO DipartimentiD (Nome)
VALUES ('DirezAvanz')

L'inserimento avviene correttamente sia nella vista **DipartimentiD** che nella tabella sottostante **Dipartimento.**

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano
DirezAvanz	NULL

DipartimentiD

Nome
Distribuzione
Direzione
DirezAvanz

Viste – Esempio 6\7

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano
DirezAvanz	NULL

DipartimentiD

Nome
Distribuzione
Direzione
DirezAvanz

ESEMPIO **: Inserire la tupla 'Sviluppo' nella vista DipartimentiD**

INSERT INTO DipartimentiD (Nome)
VALUES ('Sviluppo')

L'inserimento avviene correttamente sia nella tabella sottostante **Dipartimento**, ma **non** nella vista **DipartimentiD**

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano
DirezAvanz	NULL
Sviluppo	NULL

DipartimentiD

Nome
Distribuzione
Direzione
DirezAvanz

Viste – Esempio 7\7

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano
DirezAvanz	NULL
Sviluppo	NULL

DipartimentiD

Nome
Distribuzione
Direzione
DirezAvanz

ESEMPIO **: Inserire la tupla 'DIS' nella tabella Dipartimento**

INSERT INTO Dipartimento (Nome)
VALUES ('DIS')

L'inserimento avviene correttamente nella tabella sottostante **Dipartimento**. In questo caso anche la vista **DipartimentiD** beneficia dell'inserimento.

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano
DirezAvanz	NULL
Sviluppo	NULL
DIS	NULL

DipartimentiD

Nome
Distribuzione
Direzione
DirezAvanz
DIS

Le Viste e le interrogazioni

- Le viste in SQL possono anche servire per formulare interrogazioni che non sarebbero altrimenti esprimibili, aumentando il potere espressivo del linguaggio.
  - Ad esempio, permettono di definire in SQL interrogazioni che richiedono di utilizzare diversi operatori aggregati in cascata.

ESEMPIO **: Estrarre il numero medio di Dipartimenti per ogni città**

Dipartimento

Nome	Città

```
SELECT avg(count(Nome))
FROM Dipartimento
GROUP BY Città
```

L'interrogazione è **scorretta**, in quanto la sintassi SQL **non permette** di combinare in cascata la valutazione di diversi operatori aggregati.

Le Viste e le interrogazioni

ESEMPIO **: Estrarre il numero medio di Dipartimenti per ogni città**

Dipartimento

Nome	Città

Si può sfruttare la definizione di un'apposita vista.

```
CREATE VIEW DipartCittà( NomeCittà, NroDipart )  as
SELECT Città, count(Nome)
FROM Dipartimento
GROUP BY Città
```

```
SELECT avg(NroDipart)
FROM DipartCittà
```

Le Viste e le interrogazioni

ESEMPIO **: Estrarre il dipartimento caratterizzato dal massimo della somma per gli stipendi**

Impiegato

Nome	Cognome	Dipart	Stipendio

```
CREATE VIEW BudgetStipendi( Dip, TotaleStipendi)  as
SELECT Dipart, sum(Stipendio)
FROM Impiegato
GROUP BY Dipart
```

```
SELECT Dip
FROM BudgetStipendi
WHERE TotaleStipendi = (SELECT max(TotaleStipendi)
                        FROM BudgetStipendi)
```

Viste - Conclusioni

- 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.
- I comandi GRANT o 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 molto potente.
- Le viste possono essere interrogate proprio come relazioni ordinarie, ma sono consentite solo forme limitate di aggiornamento.