SQL

SQL come Data Manipulation Language

-

Inserimento, cancellazione e aggiornamento di righe

Operazioni di aggiornamento

- Abbiamo visto come creare tabelle e come interrogarle. Vedremo ora come popolarle e modificarne il contenuto.
- Sono possibili tre tipi di aggiornamento dei dati contenuti nelle tabelle:
  - Inserimento (**INSERT**)
  - Eliminazione (**DELETE**)
  - Modifica (**UPDATE**)

Esempio

- Assumiamo di avere creato la seguente tabella, momentaneamente vuota:

ImpRaccomandati

Nome	*Cognome*	*Ufficio*	*Stipendio*

Inserimento di ennuple

```
insert into ImpAmministrazione
values('Mario','Rossi', 10, 45)
```

ImpRaccomandati

Nome	*Cognome*	*Ufficio*	*Stipendio*
Mario	Rossi	10	45

Inserimento di ennuple

```
insert into ImpRaccomandati(Cognome,Ufficio)
values('Verdi', 20)
```

ImpRaccomandati

Nome	*Cognome*	*Ufficio*	*Stipendio*
Mario	Rossi	10	45
	Verdi	20	

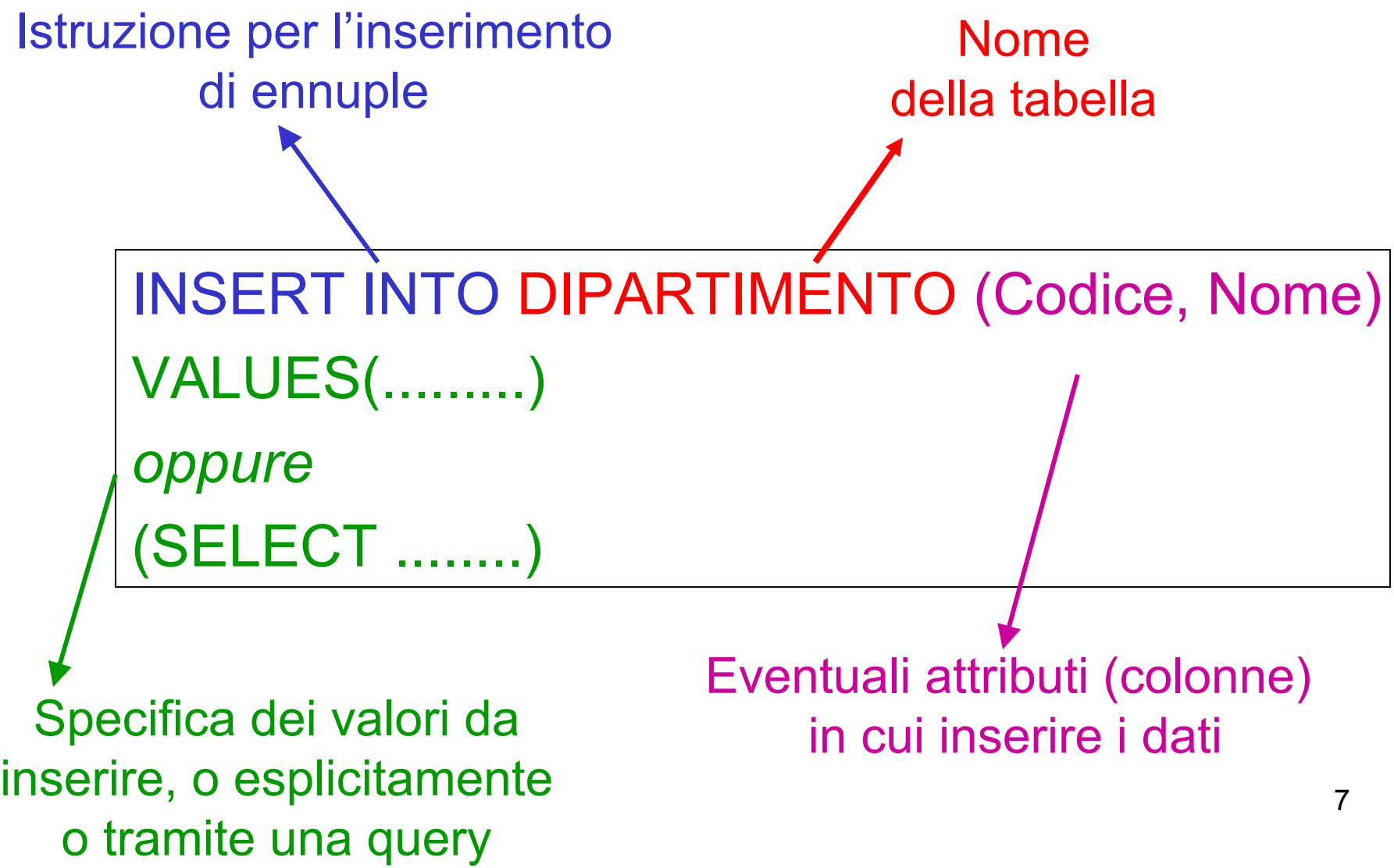
Inserimento di ennuple

```
insert into ImpRaccomandati
(select Nome, Cognome, Ufficio, Stipendio
from Impiegati
where Dipart='Direzione')
```

ImpRaccomandati

Nome	*Cognome*	*Ufficio*	*Stipendio*
Mario	Rossi	10	45
	Verdi	20	
Carlo	Rossi	14	80
Lorenzo	Lanzi	7	73

Inserimento di ennuple



Inserimento di ennuple

- L'ordinamento degli attributi (se presente) e dei valori è significativo.
- Le due liste debbono avere lo stesso numero di elementi.
- Se la lista di attributi è omessa, si fa riferimento a tutti gli attributi della relazione, secondo l'ordine con cui sono stati definiti.
- Se la lista di attributi non contiene tutti gli attributi della relazione, per gli altri viene inserito un valore nullo (che deve essere permesso) o un valore di default.

Eliminazione di ennuple

- Consideriamo la seguente tabella:

ImpRaccomandati

Nome	***Cognome***	***Ufficio***	***Stipendio***
Mario	Rossi	10	45
	Verdi	20	
Carlo	Rossi	14	80
Lorenzo	Lanzi	7	73

Eliminazione di ennuple

```
delete from ImpRaccomandati
where Cognome='Verdi'
```

ImpRaccomandati

Nome	*Cognome*	*Ufficio*	*Stipendio*
Mario	Rossi	10	45
~~Verdi~~	~~Verdi~~	~~20~~	
Carlo	Rossi	14	80
Lorenzo	Lanzi	7	73

Eliminazione di ennuple

```
delete from ImpRaccomandati
where Stipendio <> (select max(Stipendio)
                    from Impiegati)
```

ImpRaccomandati

Nome	***Cognome***	***Ufficio***	***Stipendio***
~~Mario~~	~~Rossi~~	~~10~~	~~45~~
Carlo	Rossi	14	80
~~Lorenzo~~	~~Lanzi~~	~~7~~	~~73~~

Eliminazione di ennuple

Eliminazione di ennuple

- Elimina le ennuple che soddisfano la condizione.
- Può causare eliminazioni da altre relazioni, se i vincoli di integrità referenziale sono definiti con politiche di reazione *cascade*.
- ATTENZIONE: se la where viene omessa, si intende where true.
- Per cancellare tutte le ennuple mantenendo lo schema: delete from Dipartimento
- Per cancellare anche lo schema:

  drop Dipartimento

Modifica di ennuple

- Consideriamo la seguente tabella:

ImpRaccomandati

Nome	***Cognome***	***Ufficio***	***Stipendio***
Mario	Rossi	10	45
Carlo	Rossi	14	80
Lorenzo	Lanzi	7	73

Modifica di ennuple

```
update ImpRaccomandati
set Stipendio = Stipendio * 1.5
```

ImpRaccomandati

Nome	*Cognome*	*Ufficio*	*Stipendio*
Mario	Rossi	10	**67.5**
Carlo	Rossi	14	**120**
Lorenzo	Lanzi	7	**109.5**

Modifica di ennuple

```
update ImpRaccomandati
set Stipendio = Stipendio - 30
where Ufficio not in (Select Ufficio
                      From Impiegati
                      Where Dipart = 'Direzione')
```

ImpRaccomandati

Nome	*Cognome*	*Ufficio*	*Stipendio*
Mario	Rossi	10	**37.5**
Carlo	Rossi	14	***120***
Lorenzo	Lanzi	7	***109.5***

Modifica di ennuple

```
update ImpRaccomandati
set Stipendio = null,
    Ufficio = default
```

ImpRaccomandati

Nome	***Cognome***	***Ufficio***	***Stipendio***
Mario	Rossi	0	*null*
Carlo	Rossi	0	*null*
Lorenzo	Lanzi	0	*null*

Modifica di ennuple

Istruzione per la modifica di ennuple

Nome della tabella

```
UPDATE NomeTabella
SET Attributo = < Espressione | SelectSQL | null | default >,
    Attributo = < Espressione | SelectSQL | null | default >
    ....
where condizione
```

Condizione di eliminazione (opzionale)

Specifica di uno o piu' attributi da modificare, con i valori (impliciti o espliciti) da sostituire.

Modifica di ennuple

- Attenzione a modifiche consecutive simili alle seguenti:

```
update ImpRaccomandati
set Stipendio = Stipendio * 1.1
where Stipendio <= 30

update ImpRaccomandati
set Stipendio = Stipendio * 1.15
where Stipendio > 30
```

Modifica di ennuple

- Ci aspetteremmo il seguente aggiornamento:

Stipendio
32
28

32x1.15 =
28x1.1 =

Stipendio
36.8
30.8

- Qualcuno vede l'errore?

Modifica di ennuple

Stipendio
32
28

Il primo blocco funziona:

```
update ImpRaccomandati
set Stipendio = Stipendio * 1.1
where Stipendio <= 30
```

Stipendio
32
30.8

```
update ImpRaccomandati
set Stipendio = Stipendio * 1.15
where Stipendio > 30
```

Modifica di ennuple

Stipendio
32
28

```
update ImpRaccomandati
set Stipendio = Stipendio * 1.1
where Stipendio <= 30
```

Stipendio
32
30.8

Il secondo fallisce.

```
update ImpRaccomandati
set Stipendio = Stipendio * 1.15
where Stipendio > 30
```

Stipendio
36.8
35.42

SQL

Vincoli Generici e Viste

Vincoli di integrità generici: check

- Abbiamo visto come SQL metta a disposizioni costrutti per definire i vincoli piu' frequenti di una base di dati relazionale.
- Talvolta sono richiesti vincoli piu' complessi e/o troppo specifici per dedicarvi un costrutto speciale.
- SQL permette la definizione di ulteriori vincoli tramite il costrutto check.

Vincoli di integrità generici: check

```
create table Impiegato (
Matricola character(6) primary key,
Cognome character(20),
Sesso char not null check (Sesso in ('M','F')),
Stipendio integer,
Superiore character(6),
check (Stipendio <= (select Stipendio
                     from Impiegato J
                     where Superiore = J.Matricola)
)
```

Vincoli di integrità generici: check

- Check viene seguito da una condizione, che puo' essere espressa come se si trattasse di una clausola where.
- Check e' piu' potente degli altri costrutti per specificare vincoli di SQL.
- ESERCIZIO: esprimere i vincoli not null, foreign key e unique utilizzando il costrutto check.

Vincoli di integrità generici: Asserzioni

- Alcuni vincoli possono non appartenere a un attributo o una tabella in particolare, ma direttamente allo schema.
- Check permette di definire questo tipo di vincoli:

create assertion *NomeAsserzione*
check (*Condizione*)

create assertion AlmenoUnImpiegato
check (1 <= (select count(*) from Impiegato))

Viste

•Le Viste sono tabelle virtuali, il cui contenuto dipende da quello di altre tabelle.

La lista Attributi e' opzionale

```
CREATE VIEW Nome ( ListaAttributi ) as
SelectSQL
modalita' di aggiornamento
```

Interrogazione che ne determina il contenuto.

Ne specifica il comportamento nel caso la si voglia aggiornare.

Viste

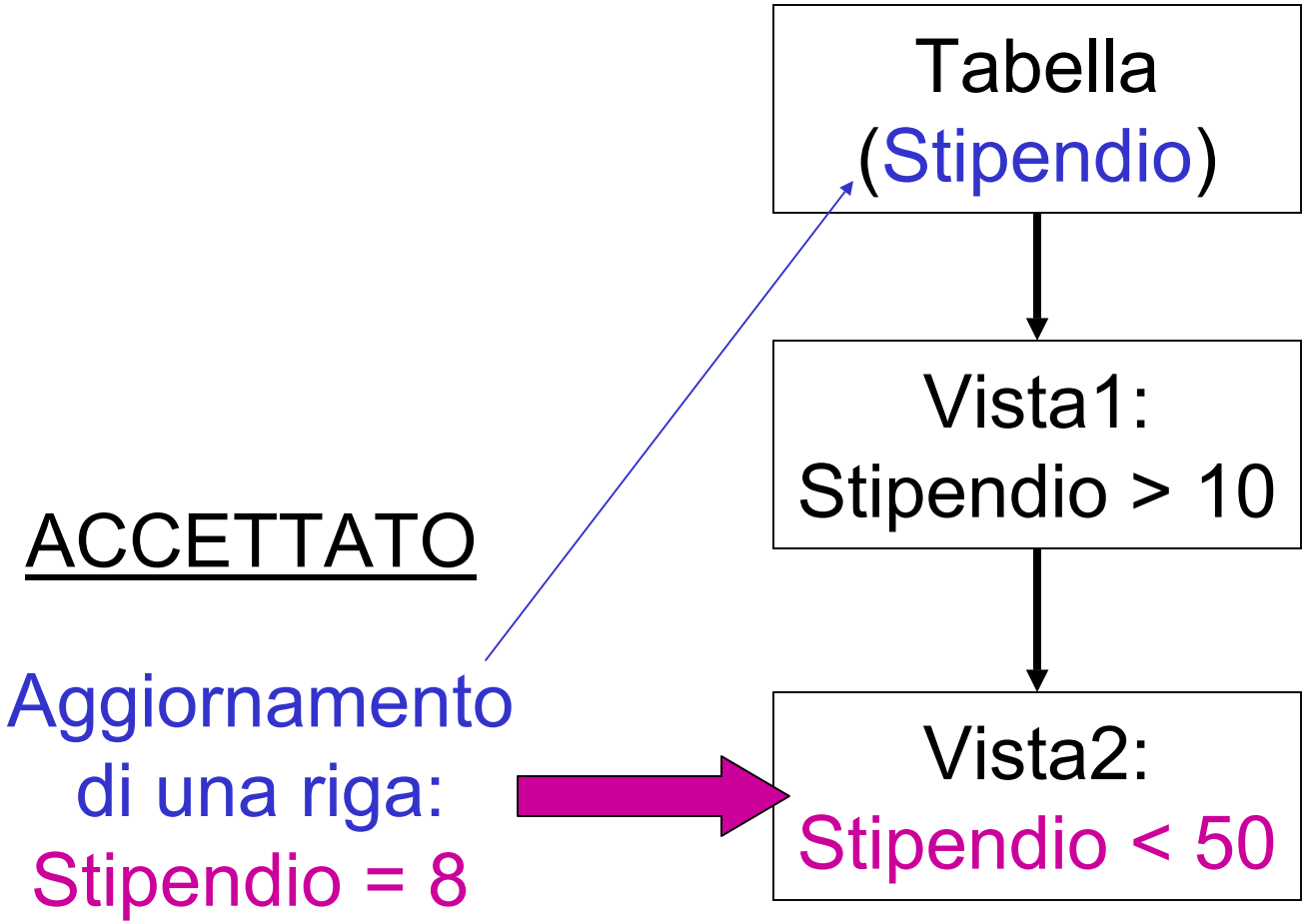
```
create view ImpiegatiAmmin
        (Matricola, Nome, Cognome, Stipendio) as
    select Matricola, Nome, Cognome, Stipendio
    from Impiegato
    where Dipart = 'Amministrazione' and
            Stipendio > 10

create view ImpiegatiAmminPoveri as
    select *
    from ImpiegatiAmmin
    where Stipendio < 50
    with check option
```

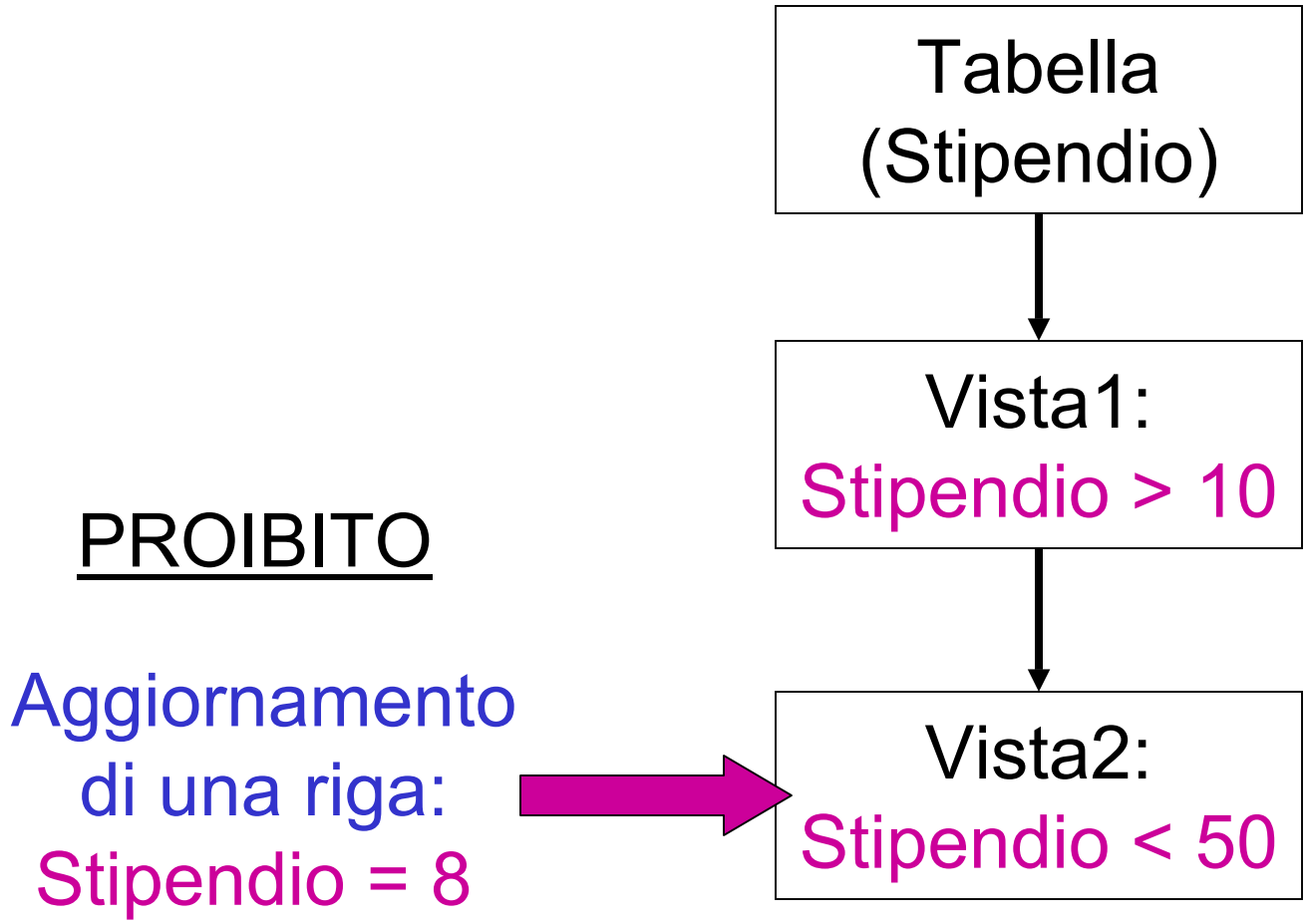
Opzioni su Viste

- `check option` e' permesso solo su viste aggiornabili. Controlla che dopo gli aggiornamenti le righe appartengano ancora alla vista.
- `local` (nel caso di viste su viste) specifica che il controllo deve essere verificato solo sulla vista corrente.
- `cascaded` (nel caso di viste su viste) specifica che l'aggiornamento deve soddisfare i vincoli di tutte le viste (default).

Opzioni su Viste: with *local* check option



Opzioni su Viste: with *cascade* check option



Interrogazioni tramite viste

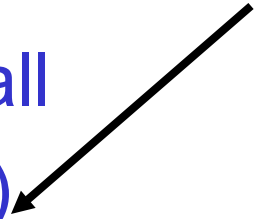
- Le viste permettono di scrivere interrogazioni che non sarebbero altrimenti possibili.
- Le viste aumentano dunque il potere espressivo di SQL.

Esempio 1

"Trova il dipartimento che spende il massimo in stipendi"

```
select Dipart
from Impiegato
group by Dipart
having sum(Stipendio) >= all
        (select sum(Stipendio)
        from Impiegato
        group by Dipart)
```

nidificazione della HAVING da evitare (non e' sempre supportata)



Soluzione con le viste

"Trova il dipartimento che spende il massimo in stipendi"

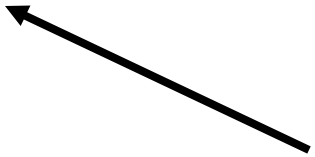
```
create view BudgetStipendi(Dip,TotaleStipendi) as
    select Dipart, sum(Stipendio)
    from Impiegato
    group by Dipart

select Dip
from BudgetStipendi
where TotaleStipendi =
                    (select max(TotaleStipendi)
                    from BudgetStipendi)
```

Esempio 2

"Trova il numero medio di uffici per ogni dipartimento"

```
select avg(count(distinct Ufficio))
from Impiegato
group by Dipart
```



nidificazione di funzioni aggregate non permessa

Soluzione con le viste

"Trova il numero medio di uffici per ogni dipartimento"

```
create view DipartUffici(NomeDip,NroUffici) as
   select Dipart, count(distinct Ufficio)
   from Impiegato
   group by Dipart

select avg(NroUffici)
from DipartUffici
```

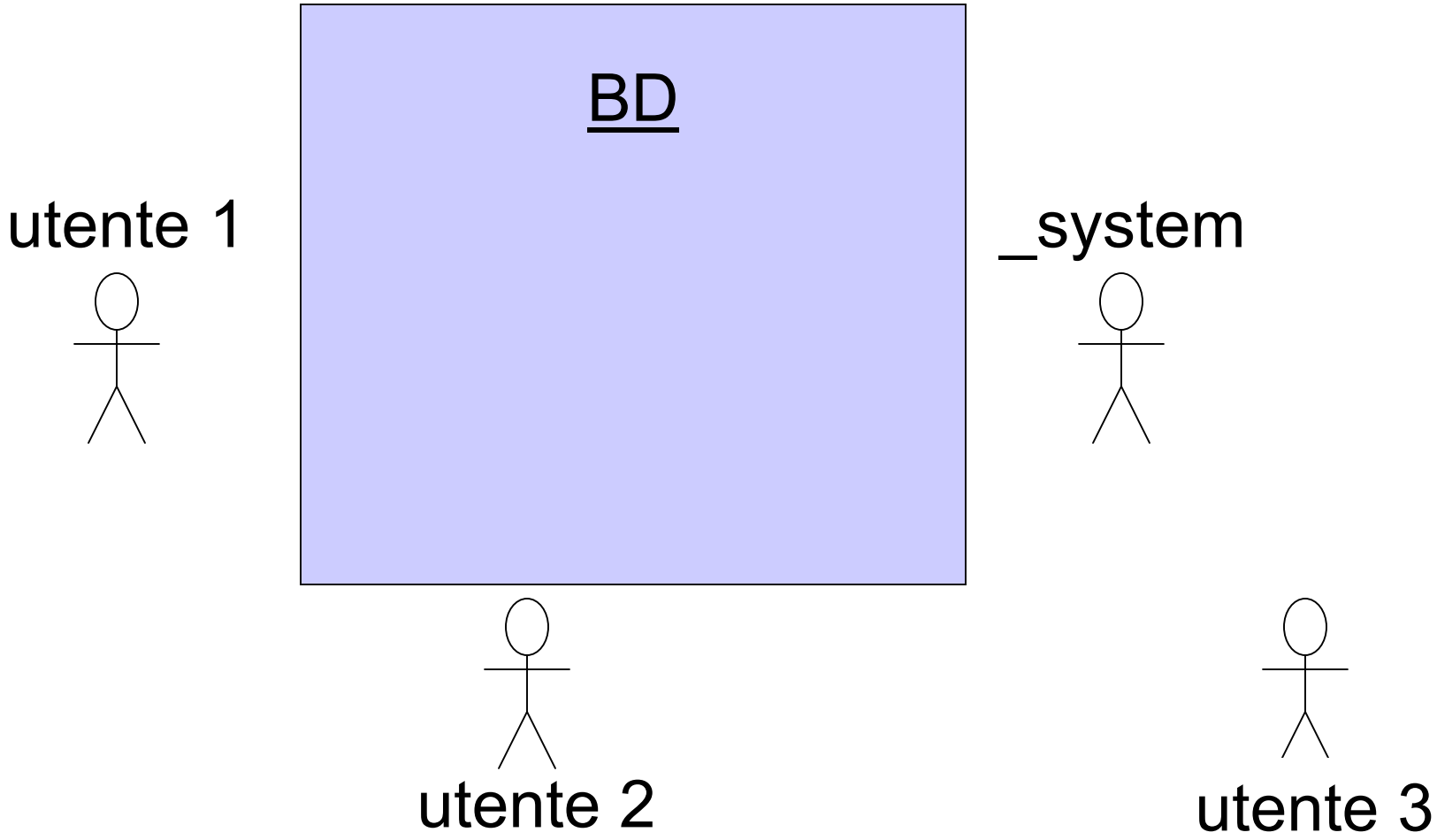
SQL

Controllo degli accessi

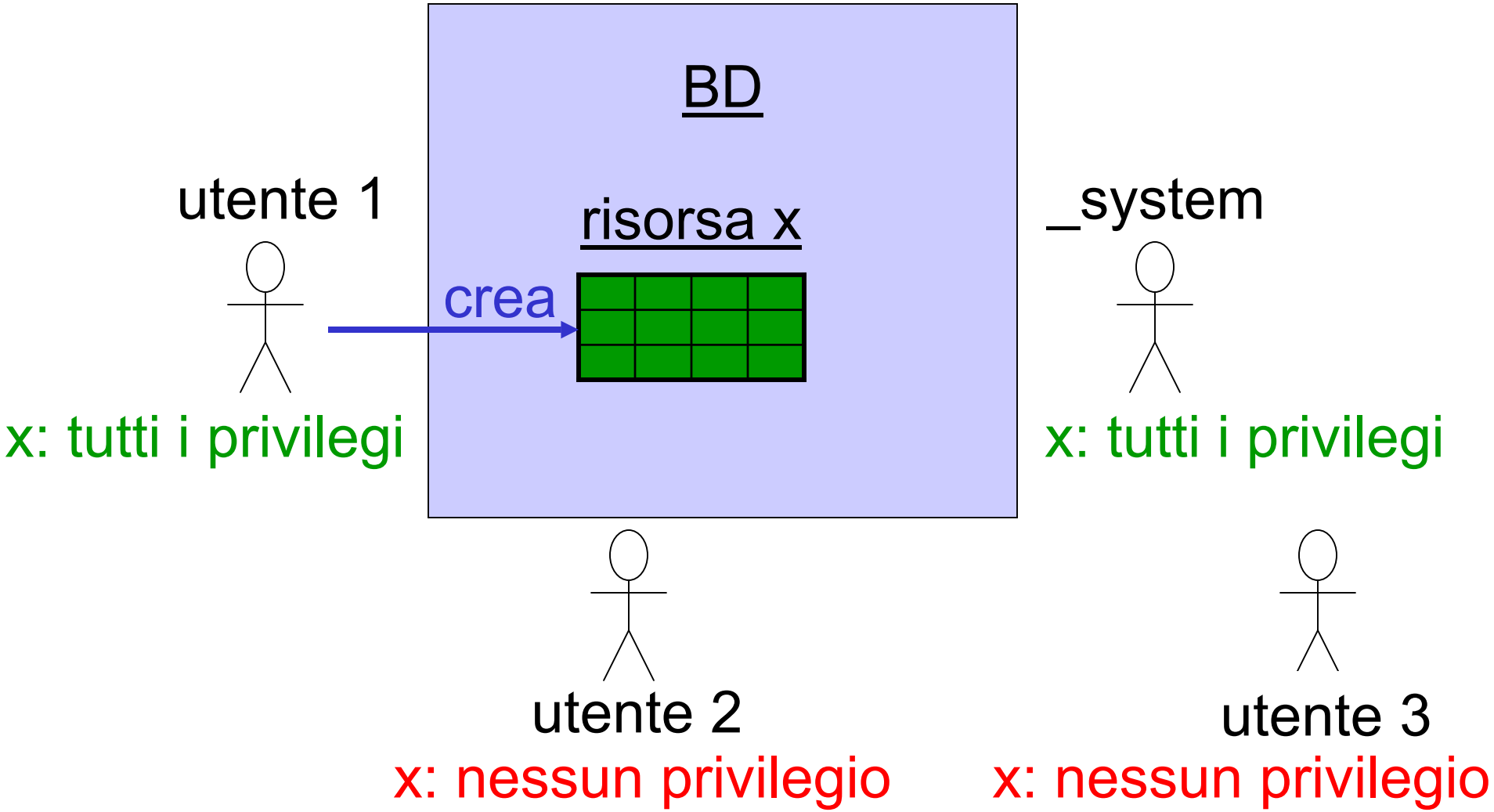
Controllo degli accessi

- Ogni componente dello schema (*risorsa*) può essere protetta (tabelle, attributi, viste, domini, ecc.).
- Il possessore della risorsa (colui che la crea) assegna dei *privilegi* agli altri utenti.
- Un utente predefinito (`_system`) rappresenta l'amministratore della base di dati ed ha completo accesso alle risorse.
- Ogni privilegio è caratterizzato da:
  - La risorsa a cui si riferisce.
  - L'utente che concede il privilegio.
  - L'utente che riceve il privilegio.
  - L'azione che viene permessa sulla risorsa.
  - Se il privilegio può esser trasmesso o meno ad altri utenti.

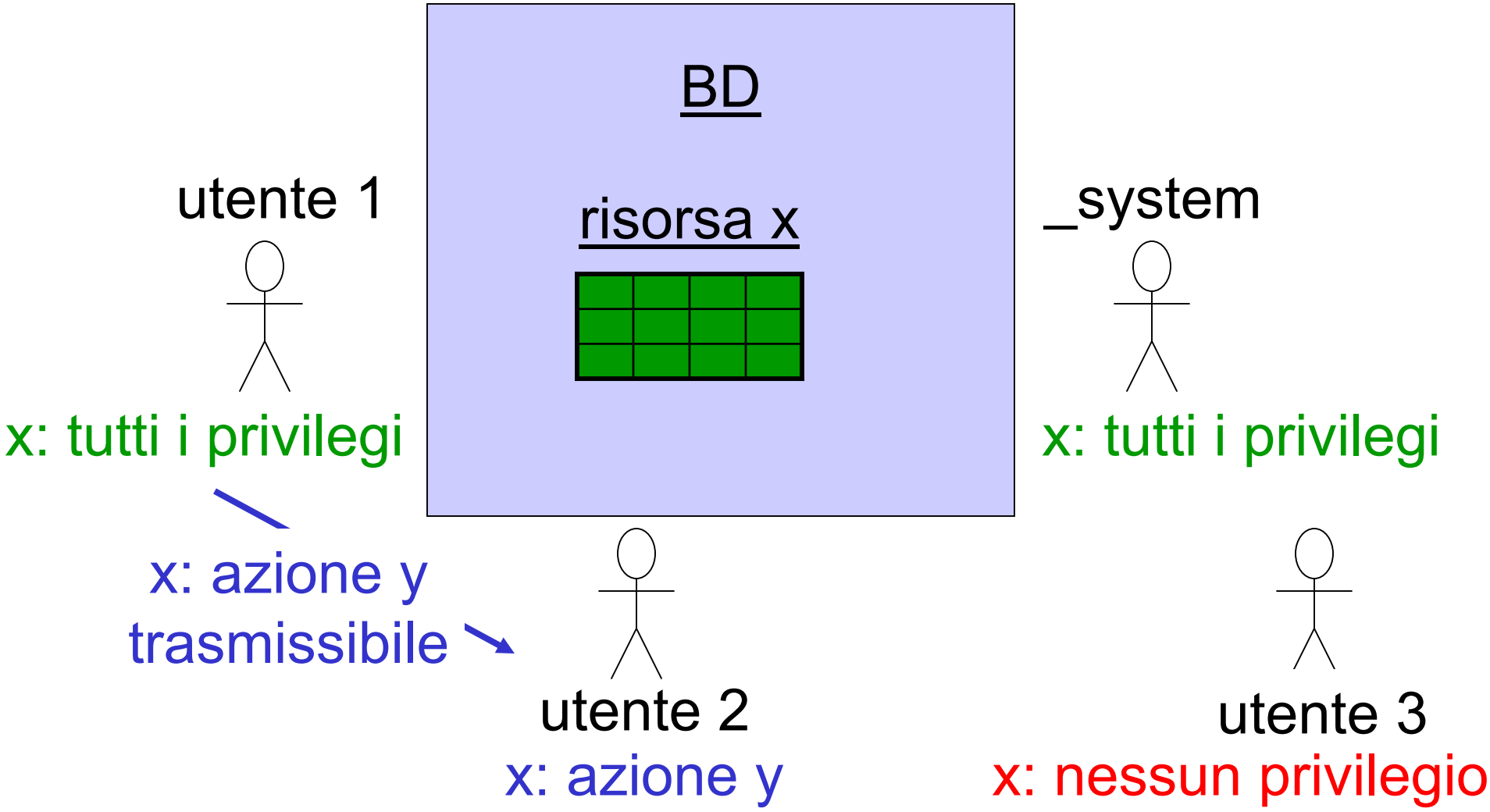
Controllo degli accessi



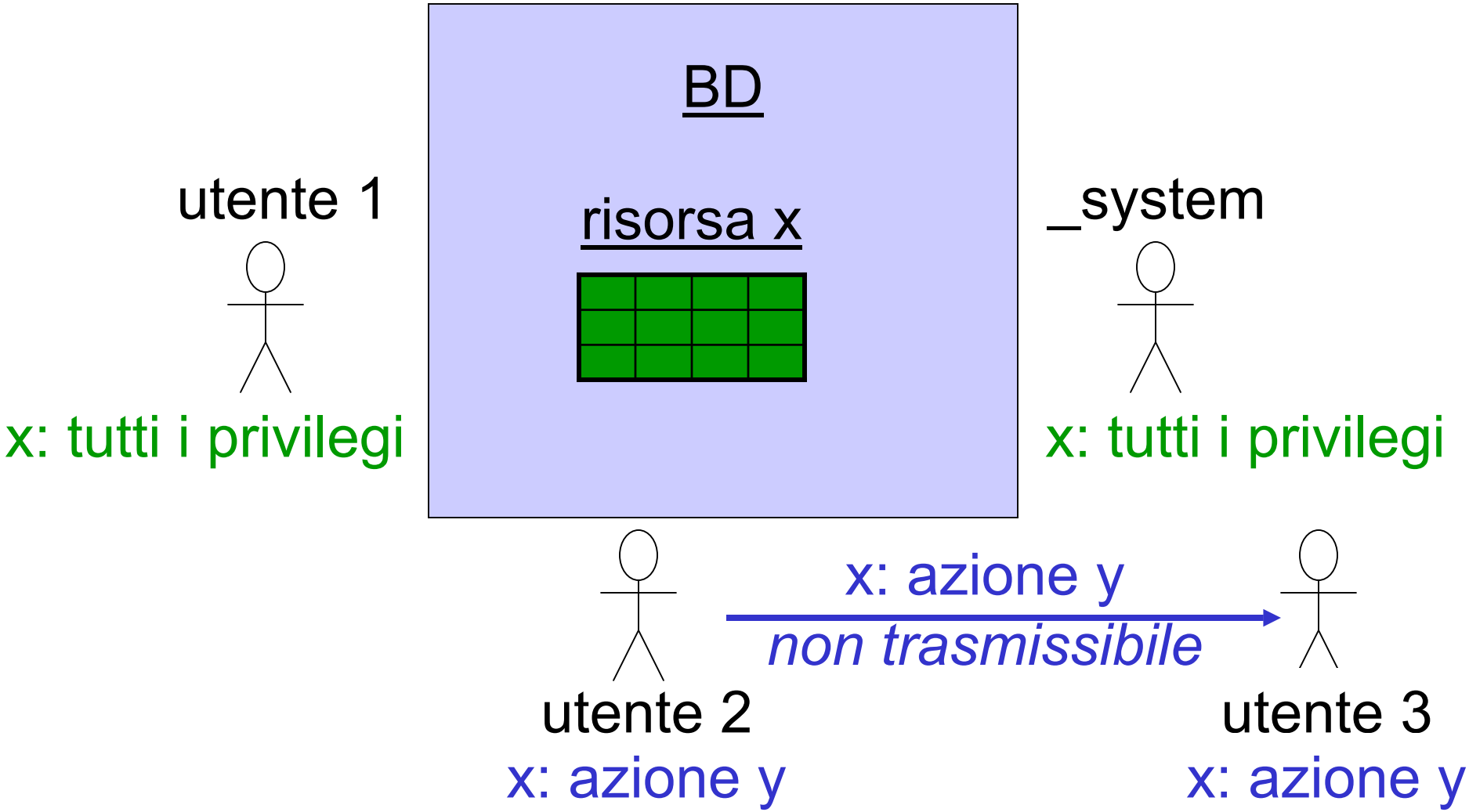
Controllo degli accessi



Controllo degli accessi



Controllo degli accessi



Tipi di privilegi

- SQL offre sei (+1) tipi di privilegi:
  - **insert** inserire un nuovo oggetto nella risorsa.
  - **update** modificare il contenuto di una risorsa.
  - **delete** rimuovere un oggetto dalla risorsa.
  - **select** accedere al contenuto della risorsa con una query.
  - **references** costruire un vincolo di integrità referenziale con la risorsa (può limitare la capacità di modificare la risorsa).
  - **usage** utilizzare la risorsa nella definizione di uno schema (per esempio, un dominio).
  - **all privileges** tutti i privilegi precedenti.

`grant` e `revoke`

- Per concedere un privilegio ad un utente:

  grant *Privilegio*
  on *Risorse*
  to *Utenti* *with grant option*

  opzionale

  – **grant option** specifica se gli utenti specificati possono propagare il privilegio ad altri utenti.

- Ad esempio:

grant select on Department to Stefano

grant all privileges on Impiegato to Paolo, Riccardo

grant select on Department to Marco with grant option

`grant` e `revoke`

- Per revocare privilegi precedentemente concessi:

  revoke *Privilegio*

  on *Risorse*

  to *Utenti restrict/cascade*

  restrict e' il valore di default

  – **grant option** in questo caso e' considerato un privilegio, e puo' essere revocato.

- Ad esempio:

revoke grant option on Department to Marco

Opzioni di `grant` e `revoke`

La revoca deve essere fatta dall'utente che aveva concesso i privilegi.

- **restrict** (di default) specifica che il comando non deve essere eseguito qualora la revoca dei privilegi all'utente comporti qualche altra revoca (dovuta ad un precedente grant option).
- **cascade** invece forza l'esecuzione del comando.
- Come sempre, attenti alle reazioni a catena.

SQL

Stored procedures

(Stored) Procedure

- SQL-2 consente di definire procedure (chiamate *stored procedures*).
- Le stored procedure sono parte dello schema:

```
procedure AssignCity(:Dep char(20),  :City char(20))
update Department
set City = :City
where Name = :Dep
```

- SQL-2 non consente di scrivere procedure complesse, ma solo singoli comandi SQL.
- Ogni sistema adotta le proprie regole al riguardo.
- Molti sistemi offrono estensioni di SQL che consentono di scrivere procedure complesse (per esempio, Oracle PL/SQL).

Procedure in Oracle PL/SQL

```
Procedure Addebita(CodConto char(5),Prelievo integer) is
    TroppoScoperto exeption;
    AmmontarePrec integer;
    NuovoAmmontare integer;
    Limite integer;
    begin
        select Ammontare, Scoperto into AmmontarePrec, Limite
            from ContoCorrente
            where CodiceConto = CodConto
            for update of Ammontare;
        NuovoAmmontare := AmmontarePrec - Prelievo;
        if NuovoAmmonotare > Limite
        then update ContoCorrente
            set Ammontare = NuovoAmmonotare
            where CodiceConto = CodConto;
       else insert into TransazioniOltreScoperto
             values(CodConto,Prelievo,sysdate);
        end if;
    end Addebita;
```