SQL

DEFINIZIONE DEI DATI

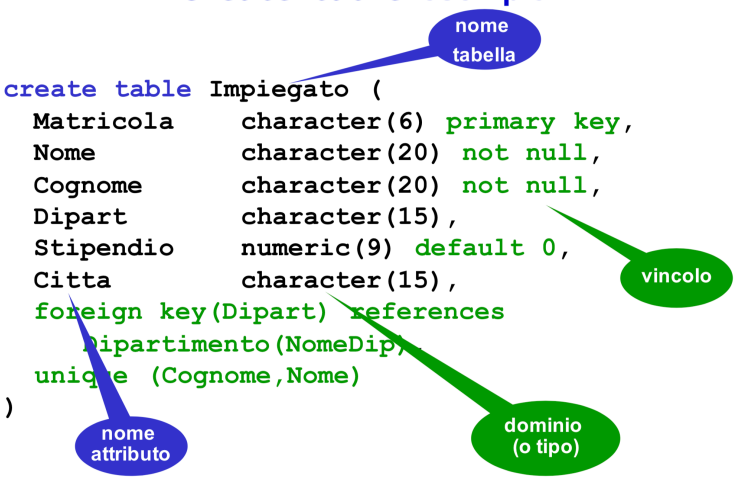
Create table

Permette di creare istanza vuota di una tabella

sintassi

```
CREATE TABLE NomeTabella (
        NomeAttributo Dominio [vincoli]
        NomeAttributo Dominio [vincoli]
        ….
        NomeAttributo Dominio [vincoli]
        [AltriVincoli]
)
```

create table: esempio




```
create table Impiegato (
  Matricola     character(6) primary key,
  Nome          character(20) not null,
  Cognome       character(20) not null,
  Dipart        character(15),
  Stipendio     numeric(9) default 0,
  Citta         character(15),
  foreign key(Dipart) references
     Dipartimento(NomeDip),
  unique (Cognome,Nome)
)
```

Tramite DEFAULT è possibile applicare valore di default all'attributo.
WARNING È importante definire una primary key o attributi unique altrimenti la tabella potrà avere ennuple completamente uguali.

Create domain

Permette di creare domini personalizzati da domini predefiniti tramite check

sintassi

```
CREATE DOMAIN NomeDominio
        AS DominioPresistente [Default][Vincoli]
```

Esempio:

```
create domain Voto
as smallint default null
check ( value >=18 and value <= 30 )
```

Alter table

Permette di modificare una tabella, essenziale per creare vincoli foreign key

sintassi

```
ATLER TABLE NomeTabella
        ADD Attributo|Vincolo
```

Esempio:

```
create table Infrazioni (
  Codice          character(6) not null primary key,
  Data            date not null,
  Vigile          integer not null
                  references Vigili(Matricola),
  Provincia       character(2),
  Numero          character(6),
)

alter table Infrazioni
 add constraint MioVincolo foreign key(Provincia, Numero)
 references Auto(Provincia,Numero)
```

CONSTRAINT permette di dare un nome al vincolo

Drop table

Permette di eliminare una tabella

sintassi

DROP TABLE NomeTabella RESTRICT|CASCADE

I parametri RESTRICT e CASCADE specificano come eliminare le tabelle collegate:

- **RESTRICT** => elimina solo se non ci sono riferimenti ad essa
- **CASCADE** => elimina tabella e tutte le tabelle che si riferiscono ad essa

---

Vincoli intrarelazionali

Esprimono condizioni sui valori di ogni tupla

- **PRIMARY KEY** => assegna chiave primaria, valore univoco identificativo (NOT NULL)
- **NOT NULL** => il valore non può essere NULL
- **UNIQUE** => permette di creare una superchiave per l'attributo
  OSS Attributo1 UNIQUE… attribuito2 UNIQUE =/= UNIQUE(attributo1, attributo2)
- **CHECK** => vincoli sulle tuple (ex. Permette di assegnare valori predefiniti nelle quali possono rientrare i valori delle tuple)

Vincoli interrelazionali

Permettono di associare informazioni su relazioni differenti

- **CHECK** => permette di creare vincoli complessi tra tabelle, tra cui vincoli di inclusione.
- **REFERENCES** e **FOREIGN KEY** => definiscono vincoli di integrità referenziali, gli attributi referenziati devono essere delle primary key (o unique) nella tabella di arrivo.
  Può essere creato solo se tabella da referenziare esiste già, richiede utilizzo di alter table ciclci per doppia referenziazione.
  sintassi

  - REFERENCES Attributo
  - FOREIGN KEY (Attributo, Attributo, …) REFERENCES NomeTabella(Attr, Attr, …)

Infrazioni

Codice	Data	Vigile	Prov	Numero
34321	1/2/95	3987	MI	39548K
53524	4/3/95	3295	TO	E39548
64521	5/4/96	3295	PR	839548
73321	5/2/98	9345	PR	839548

Auto

Prov	Numero	Cognome	Nome
MI	39548K	Rossi	Mario
TO	E39548	Rossi	Mario
PR	839548	Neri	Luca

Infrazioni

Codice	Data	Vigile	Prov	Numero
34321	1/2/95	3987	MI	39548K
53524	4/3/95	3295	TO	E39548
64521	5/4/96	3295	PR	839548
73321	5/2/98	9345	PR	839548

Vigili

Matricola	Cognome	Nome
3987	Rossi	Luca
3295	Neri	Piero
9345	Neri	Mario
7543	Mori	Gino

```
create table Infrazioni (
  Codice    character(6) not null primary key,
  Data      date not null,
  Vigile    integer not null
            references Vigili(Matricola),
  Provincia      character(2),
  Numero         character(6),
  foreign key(Provincia, Numero)
            references Auto(Provincia,Numero)
)
```

OSS il dominio di un attributo referenziato corrisponde con il dominio dell'attributo nell'altra tabella

Extra foreign key

Il vincolo di foreign key permette inoltre di settare le azioni nel caso di aggiornamento (ON UPDATE) o cancellazione (ON DELETE) della tabella.

sintassi

```
ON UPDATE  NO ACTION|RESTRICT|CASCADE|SET DEFAULT|SET NULL
ON DELETE  NO ACTION|RESTRICT|CASCADE|SET DEFAULT|SET NULL
```

- **NO ACTION** => genera errore al momento della verifica del vincolo
- **RESTRICT** => genera errore immediato, non aspetta fine transazione
  WARNING anche se **DEFERRED**[1]
- **CASCADE** => cancella tupla che referenzia e tupla referenziata
- **SET DEFAULT** => sostituisce referenziata con valore default
- **SET NULL** => sostituisce referenziata con NULL

Le transazioni

- Atomiche => le operazioni non si possono suddividere
- Consistenti => rispettano i vincoli
  OSS I vincoli DEFERRED possono essere violati durante le operazioni ma devono essere rispettati alla fine della transazione, altrimenti viene annulata per intero
- Isolate => gestiscono sezione critica nel caso di transazioni simultanee
- Durevoli => una volta effettuate sono definitive

sintassi

```
BEGIN;
    BEGIN|UPDATE|DELETE SintassiOperazioneManipolazioneDati
COMMIT;
```

```
begin;
    update ContoCorrente
       set Saldo = Saldo - 10
       where NumeroConto = 12345;
    update ContoCorrente
       set Saldo = Saldo + 10
       where NumeroConto = 55555;
commit;
```

[1] DEFERRED => applicabile sui vincoli di foreign key, permette di controllare il vincolo solo alla fine della transazione

MANIPOLAZIONE DEI DATI

Insert

Permette di inserire tuple nella tabella

sintassi

```
INSERT INTO Tabella [(Attributi)]
        VALUES
        (valori),
        (valori),
        ….
        valori;
```

```
insert into persone values('Mario',25,52)

insert into persone(nome, eta, reddito)
   values('Pino',25,52)

insert into persone(nome, reddito)
   values('Lino',55)

insert into persone (nome)
   select padre
   from paternita
   where  padre not in (select nome from persone)
```

Se non specifico attributi prendo tutti, è possibile inserire elementi tramite select

Delete

Permette di eliminare tuple

sintassi

```
DELETE FROM Tabella
        [WHERE Condizione]
```

Fare attenzione a vicoli cascade

```
delete from persone
where eta < 35

delete from paternita
where figlio not in
              (select nome from persone)
```

Update

Permetti di modificare tuple

sintassi

```
UPDATE NomeTabella
SET Attributo = Espressione|select…|null|default
[WHERE Condizione]
```

```
update persone set reddito = 45
where  nome = 'Piero'

update persone set reddito = reddito * 1.1
where  eta < 30
```

Interrogazioni

La select

La versione base permette di effettuare le interrogazioni sugli attributi (o le espressioni sugli attributi) inseriti nella clausola select appartenenti alle tabelle della clausola from, aventi le condizioni indicate nella where.

sintassi

```
SELECT ListaAttributi o Espressioni
FROM ListaTabelle
[WHERE CondizioniSemplici]
[GROUP BY ListaAttributiRaggruppamento]
[HAVING CondizioniAggregate]
[ORDER BY ListaAttributiOrdinamento]
[LIMIT numero]
```

Nome e reddito delle persone con meno di 30 anni

```
select persone.nome, persone.reddito
from   persone
where  persone.eta < 30
```

nome	reddito
Andrea	21
Aldo	15
Filippo	30

- Gli attributi si scrivono come "*NomeTabella.NomeAttributo*"
- Tramite l'espressione * si possono selezionare tutti gli attributi della tabella

Distinct

Permette di eliminare le ennuple ripetute, essendo le tabelle SQL un multinsieme.

sintassi

```
SELECT DISTINCT ListaAttributi o Espressioni
…
```

As

Permette di effettuare ridenominazione su attributi

sintassi

```
SELECT NomeAttributo AS NuovoNomeAttributo
…
```

```
select p.nome as nome, p.reddito as reddito
from   persone as p
where  p.eta < 30
```

o anche:

```
select p.nome as nome, p.reddito as reddito
from   persone p
where  p.eta < 30
```

OSS

La proiezione si effettua utilizzando s le clausole select e from;
la selezione tramite la clausola where;
il join implicito[2] tramite la from;

Like

Permette di creare espressioni regolari da confrontare con stringhe degli attributi

sintassi

```
SELECT ListaAttributi o Espressioni
FROM ListaTabelle
WHERE attributo LIKE 'espressione'
…
```

```
select *
from   persone
where  nome like 'A_d%'
```

[2] join implicito => equivalente ad operazione di prodotto cartesiano

Is null

Permette di gestire valori nulli

sintassi

```
SELECT ListaAttributi o Espressioni
FROM ListaTabelle
WHERE attributo IS NULL|IS NOT NULL
…
```

```
select *
from   impiegati
where  eta > 40 or eta is null
```

Join esplicito

Equivalente a join implicito (theta-join), permette però di esplicitare le condizioni di join nella from separandole dalle condizione della where

sintassi

```
SELECT Attributi
FROM tabella1 JOIN tabella2 ON CondizioniJoin
…
```

```
select paternita.figlio, padre, madre
from   maternita, paternita
where  paternita.figlio = maternita.figlio

select madre, paternita.figlio, padre
from   maternita join paternita on
       paternita.figlio = maternita.figlio
```

Join naturale

Gli attributi uguali delle tabelle vengono messi direttamente in corrispondenza

sintassi

```
SELECT Attributi
FROM tabella1 NATURAL JOIN tabella2
…
```

```
select paternita.figlio, padre, madre
from maternita natural join paternita
```

Outer join

Include anche valori che della tabella che non verrebbero aggiunti a causa di valori null

sintassi

```
SELECT Attributi
FROM tabella1 LEFT|RIGHT|FULL OUTER JOIN tabella2
…
```

```
select paternita.figlio, padre, madre
from   maternita join paternita
       on maternita.figlio = paternita.figlio

select paternita.figlio, padre, madre
from   maternita left outer join paternita
       on maternita.figlio = paternita.figlio

select paternita.figlio, padre, madre
from   maternita right outer join paternita
       on maternita.figlio = paternita.figlio

select nome, padre, madre
from   persone full outer join maternita on
      persone.nome = maternita.figlio
       full outer join paternita on
       persone.nome = paternita.figlio
```

Operatori aggreganti

Permettono di calcolare valori a partire da insiemi di ennuple, il DISTINCT permette di non riusare valori uguali provenienti da tuple differenti

Sintassi

```
SELECT FUNZIONE([DISTINCT] EspressioneSuAttributi)
…
```

funzioni principali:

- **COUNT** => conta ennuple nella tabella
  OSS può usare '*' come espressione che prende tutte le ennuple della tabella
- **SUM** => somma tutti i valori dell'attributo
- **AVG** => media valori dell'attributo
- **MAX** => valore massimo dell'attributo
- **MIN** => valore minimo dell'attributo

OSS gli attributi con valori NULL non vengono considerati

paternita

padre	figlio
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

```
select count(*) as NumFigliDiFranco
from   paternita
where  padre = 'Franco'
```

NumFigliDiFranco
2

Group by

permette di creare sottoinsiemi di ennuple nella tabella prendendo tutti gli attributi con valori uguali (per rendere attributo omogeneo con operatore)

sintassi

```
SELECT ListaAttributi o Espressioni
FROM ListaTabelle
…
[GROUP BY ListaAttributiRaggruppamento]
…
```

WARNING se è presente nella SELECT un attributo contenuto nella GROUP BY oppure un operatore aggregante, allora anche tutti gli altri attributi dovranno far parte della GROUP BY oppure essere operatori aggreganti

```
select padre, count(*) as NumFigli
from   paternita
group by padre
```

paternita

padre	figlio
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

padre	NumFigli
Sergio	1
Luigi	2
Franco	2

Having

Permette di imporre condizioni sui gruppi, differenti da quelle della where

sintassi

```
SELECT ListaAttributi o Espressioni
FROM ListaTabelle
…
[GROUP BY ListaAttributiRaggruppamento]
[HAVING CondizioniAggregate]
…
```

Esempio: i padri i cui figli hanno un reddito medio maggiore di 25.

```
select padre, avg(f.reddito)
from   persone f join paternita
       on figlio = nome
group by padre
having avg(f.reddito) > 25
```

Order by

Permette di ordinare tabella output in base ad un attributo (DESC per ordine decrescente)

sintassi

```
SELECT ListaAttributi o Espressioni
FROM ListaTabelle
…
[ORDER BY ListaAttributiOrdinamento [DESC]]
…
```

```
select nome, reddito
from   persone
where  eta < 30
order by nome
```

Limit

Permette di limitare il numero di tuple da mostrare

sintassi

```
SELECT ListaAttributi o Espressioni
FROM ListaTabelle
…
[LIMIT numero]
```

```
select nome, reddito
from   persone
where  eta < 30
order by nome
limit 2
```

Union|Except|Intersect

Permettono di effettuare unione, differenza e intersezione tra i risultati di 2 select, al contrario della select le tuple duplicate vengono eliminate => bisogna usare ALL per mantenerle

sintassi

```
SELECT ListaAttributi o Espressioni
FROM ListaTabelle
…
UNION|EXCEPT|INTERSECT [ALL]
SELECT ListaAttributi o Espressioni
FROM ListaTabelle
…
```

```
select padre, figlio
from   paternita
union
select madre, figlio
from   maternita
```

WARNING gli attributi delle 2 select devono essere uguali e nello stesso ordine

Interrogazioni nidificate

È possibile inserire nelle condizioni atomiche di una select una select nidificata

sintassi

```
SELECT ListaAttributi o Espressioni
FROM ListaTabelle
WHERE condizione <operatore> (SELECT ListaAttributi o Espressioni
                              FROM ListaTabelle
                              … )
```

Gli operatori che mettono in relazione le select possono essere:

- Se interrogazione nidificata ha risultato unico
  - **UGUAGLIANZA O CONFRONTO**
- Se interrogazione nidificata ha più risultati
  - **ANY** => se confronto è valido per una qualunque tupla del risultato
  - **ALL** => se confronto è valido per tutte le tuple del risultato
  - **IN** => equivalente a “=ANY” (intersezione)
  - **NOT IN** => equivalente a “<>ALL” (differenza)
  - **EXIST** => vera se risultato non è vuoto

OSS è possibile applicare le condizione su più attributi tramite il costrutto:
(Attributo1, Attributo2, …) <operatore> (SELECT ListaAttributi …)

OSS è inoltre possibile inserire interrogazioni nidificate anche nella clausola FROM, in questo modo si creano “viste in line” delle tabelle (altrimenti creabili tramite WITH).

OSS si possono inserire anche nella HAVING

Nome ed età delle madri che hanno almeno un figlio minorenne.

```
select nome, eta
from   persone, maternita
where  nome = madre and
       figlio in (select nome
                  from   persone
                  where  eta < 18)
```

Esempio: le persone che hanno almeno un figlio.

```
select *
from   persone p
where  exists (select *
               from  paternita
               where padre = p.nome)
       or
       exists (select *
               from  maternita
               where madre = p.nome)
```

La visibilità

Una variabile X è visibile in un blocco M se:

- (passo base) X è definito in M
- X è definito in un blocco che contiene M e non è ridefinito in M

Questo implica che nelle interrogazioni nidificate non è possibile fare riferimento a variabili definite nella select interna ma la questa può far riferimento alle variabili della select che la contiene.
Nel secondo caso si parla di interrogazioni nidificate e correlate.

Viste

Tabelle virtuali le cui istanze sono derivate da altre tabelle mediante una interrogazione, si creano come normali tabelle scrivendo una query separata e vengono salvate dal DBMS. Si utilizzano esattamente come le altre tabelle.

sintassi

```
CREATE VIEW NomeVista [(ListaAttributi)] AS
        SELECT ListaAttributi o Espressioni
        FROM ListaTabelle
        …
```

- *Esempio*:

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

OSS utilizzando WITH al posto di CREATE VIEW è possibile creare una “vista in line” da usare direttamente sopra la query

Cast

Effettuare cast su attributi

sintassi

```
SELECT CAST( Attributo o Espressione AS NuovoTipo [ ( Numcifre.Numcifre ) ] )
…
```

Trucchetti

A. Per evitare ripetizione stessa coppia <Attributo1,Attributo2> con ordine inverso:

```
…
where Attributo1 < Attributo2
```

B. Mettere in comune una coppia <attributo1, attributo1> con una coppia <attributo2, attributo2> della stessa tabella:

```
select distinct  T1.Attributo1, T3.Attributo1

from tabella1 T1, tabella1 T2, tabella1 T3, tabella1 T4

where T1.Attributo1 = T2.Attributo1 AND
      T3.Attributo1 = T4.Attributo1 AND
      T1.Attributo2 = T3.Attributo2 AND
      T2.Attributo2 = T4.Attributo2 AND
      T1.Attributo2 <> T2.Attributo2 AND
      T1.Attributo1 < T3.Attributo1
```