

Domini

Tipi numerici esatti

Valori interi o con parte decimale di lunghezza prefissata

numeric [(*Precisione* [, *Scala*])]

decimal [(*Precisione* [, *Scala*])]

integer

smallint

Precisione specifica il numero di cifre significative

Scala quante cifre si rappresentano dopo la virgola

decimal(4) numeri da -9999 a 9999

numeric(6,3) numeri da -999.999 a 999.999

La precisione degli interi dipende dall'implementazione

Domini

Tipi numerici approssimati

rappresentano valori reali approssimati

`float` [*Precisione*]

`double precision`

`real`

Precisione specifica il numero di cifre dedicato alla mantissa
(la precisione dell'esponente dipende dall'implementazione)

`real` e `double precision` hanno precisione prestabilita,
una doppia rispetto all'altra.

Domini

Data e ora

tipi utilizzati per rappresentare istanti di tempo

date

time [(*Precisione*)] [with time zone]

timestamp [(*Precisione*)] [with time zone]

Ciascun tipo è divisibile in campi.

date ammette come campi **year**, **month** e **day**

time i campi **hour**, **minute**, **second**

timestamp tutti i campi da **year** a **second**

I campi sono divisi da " : "

20 : 03 : 04 + 1 : 00

Definizione di domini

Come nei linguaggi ad alto livello (es. C) è possibile definire nuovi domini (tipi di dati) a partire da quelli predefiniti, anche se il costruttore è più limitato.

create domain *NomeDominio* **as**

TipodiDato [*ValDefault*] [*Vincolo*]

Non si possono creare array o strutture poiché il modello relazionale impone che gli attributi siano definiti su un dominio elementare.

E' però possibile associare dei vincoli ad un dominio definito dall'utente. Se si modifica la definizione di un dominio, la modifica si propaga a tutte le tabelle.

Definizione di domini: esempio

```
CREATE DOMAIN Voto AS SMALLINT
```

```
DEFAULT NULL
```

```
CHECK ( value >=18 AND value <= 30 )
```

Definizione di schemi

E' possibile definire uno schema di base di dati come collezione di oggetti.

```
create schema[NomeSchema]  
    [[authorization] Autorizzazione]  
    {DefElementoSchema}
```

Autorizzazione rappresenta il nome dell'utente proprietario dello schema

NB In MySQL sintassi diversa per le opzioni

Definizione di tabelle

Una tabella SQL è costituita da una collezione ordinata di attributi e da un insieme di vincoli

```
create table NomeTabella  
( NomeAttributo Dominio [default ValDefault] [Vincoli]  
{ ,NomeAttributo Dominio [default ValDefault] [Vincoli] }  
  [AltriVincoli]  
)
```

Una tabella è inizialmente vuota e chi la crea possiede tutti i diritti su di essa

Definizione di tabelle

Esempio

```
create table Dipartimento
(
    Nome                char(20) primary key,
    Indirizzo           char(50) ,
    Città               char(20)
)
```

Valori di default

Si possono specificare valori di default per un attributo quando, al momento della creazione di una nuova tupla, in un campo non viene inserito alcun valore.

```
default < GenericoValore | user | null >
```

GenericoValore è un valore compatibile col dominio
user è il nome dell'utente che aggiorna la tabella
null corrisponde al valore di default di base

Valori di default

Un nuovo default (es. nella definizione di un tipo a partire da un altro) sostituisce il precedente

Es. `NumeroFigli smallint default 0`

impone il valore 0 quando non viene specificato il valore dell'attributo.

Vincoli intrarelazionali

I più semplici vincoli intrarelazionali predefiniti sono:

not null indica che il valore nullo non è ammesso su uno specifico attributo.

Si impone che sia inserito in ogni caso un valore, salvo che non sia già definito un valore di default.

SQL non distingue i diversi tipi di valore nullo. Se è necessario, bisogna definire opportuni domini.

Vincoli intrarelazionali

unique (*Attributo* {, *Attributo*}) indica che l'insieme di attributi deve essere una superchiave per la tabella.

primary key (*Attributo* {, *Attributo*}) definisce la chiave primaria. Può essere una sola. Tutti gli attributi della chiave primaria sono automaticamente **not null**.

Vincoli intrarelazionali

Nome **character(20) not null,**
Cognome **character(20) not null,**
unique (Nome, Cognome)

impone che non esistano due tuple in cui sia i valori di Nome che di Cognome siano uguali

Nome **character(20) not null unique,**
Cognome **character(20) not null unique**

impone che non esistano due tuple che abbiano uguali valori di Nome o di Cognome

Vincoli interrelazionali

Vincoli di integrità referenziale

Il vincolo di **foreign key** (*chiave esterna*) collega i valori di un insieme di attributi della tabella corrente (*tabella interna*) a un insieme di attributi di un'altra tabella (*tabella esterna*). Per ogni tupla della tabella interna i valori della chiave esterna devono essere presenti nei corrispondenti attributi della tabella esterna.

L'insieme di attributi 'esterni' deve essere **unique**.

Se l'insieme contiene un unico attributo si usa la clausola **references** (*AttribTabExt*) nella riga in cui è definito; altrimenti **foreign key** (*ListaAttributi*)

Vincoli interrelazionali

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

  unique (Cognome, Nome) ,
  foreign key(Nome, Cognome)
    references Anagrafica(Nome, Cognome)
)
```

Vincoli interrelazionali

Per i vincoli visti l'inserimento di un valore che li viola viene semplicemente impedito.

In caso di vincoli di integrità referenziale, ciò vale nel caso di introduzione di dati non validi nella tabella interna.

Se la violazione avviene per una modifica apportata alla tabella esterna, si deve associare una politica alle operazioni che possono causare inconsistenza:

`on < delete | update >`

`<cascade|set null|set default|no action>`

subito dopo la specifica del riferimento

Vincoli interrelazionali

Modifica (comando **update**):

cascade il nuovo valore dell'attributo della tabella esterna viene riportato su tutte le corrispondenti righe della tabella interna.

set null all'attributo referente viene assegnato il valore nullo.

set default all'attributo referente viene assegnato il valore di default.

no action la modifica non viene consentita.

Vincoli interrelazionali

Cancellazione (comando `delete`):

cascade tutte le corrispondenti righe della tabella interna vengono cancellate .

set null all'attributo referente viene assegnato il valore nullo.

set default all'attributo referente viene assegnato il valore di default.

no action la cancellazione non viene consentita.

Modifica degli schemi

E' possibile modificare gli schemi con i comandi **alter** e **drop**.

alter consente di modificare domini e schemi di tabelle.

```
alter domain NomeDominio  
    < set default ValDefault |  
      drop default |  
      add constraint DefVincolo |  
      drop constraint NomeVincolo >
```

Modifica degli schemi

```
alter table NomeTabella <
  alter column NomeAttributo
  <set default NuovoDefault | drop default> |
  add constraint DefVincolo |
  drop constraint NomeVincolo |
  add column NomeAttributo |
  drop column NomeAttributo >
```

NB Quando si inserisce un nuovo vincolo, questo deve essere soddisfatto dai dati già presenti

Modifica degli schemi

Il comando **drop** permette di rimuovere dei componenti

```
drop < schema | domain | table | view |  
assertion > NomeElemento  
    [ restrict | cascade ]
```

restrict specifica di non eseguire il comando in presenza di oggetti non vuoti.

cascade implica che gli oggetti specificati siano rimossi insieme a tutti gli oggetti da essi dipendenti.
Quindi, ATTENZIONE!!!!

Interrogazioni

Le interrogazioni (*query*) in SQL sono formulate in modo *dichiarativo*, descrivono cioè *cosa* si vuole ottenere e non *come* lo si vuole ottenere.

L'interrogazione viene elaborata dall'*ottimizzatore di interrogazioni (query optimizer)* che fa parte del DBMS. Questo la analizza e la traduce nel linguaggio di interrogazione interno al DBMS.

Per questo in SQL si deve cercare di scrivere codice *leggibile e facilmente modificabile*, affidando all'ottimizzatore la formulazione di *query efficienti*.