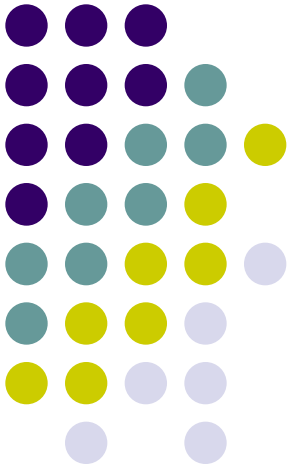
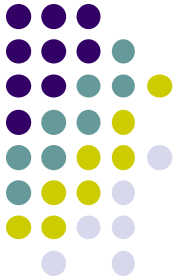
SQL

Descrizione del linguaggio SQL
Definizione del database

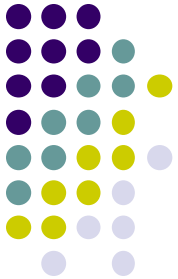


SQL



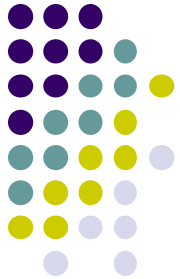
- Structured Query Language
- è un linguaggio con varie funzionalità:
  - contiene sia il DDL (Data Definition Language) che il DML (Data Manipulation Language)
- ne esistono varie versioni
- Noi utilizzeremo la versione di MySQL

Breve storia dell'SQL



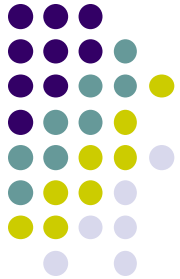
- SQL (pronunciato "Sequel"): Structured Query Language
- SQL sviluppato alla IBM nel 1973
  - Dal 1983 *standard de facto*
  - Primo standard nel 1986 rivisto nel 1989 (SQL-89)
  - Secondo standard nel 1992 (SQL-2 o SQL-92)
  - Terzo standard nel 1999 (SQL-3 o SQL-99)
- Quasi tutti i DBMS commerciali adottano lo standard SQL più estensioni proprie (non-standard)
- Alcuni sistemi commerciali
  - Oracle, Informix, Sybase, DB2, SQL-Server, etc.
- Alcuni sistemi open-source:
  - MySQL, Postgres
- Esistono sistemi commerciali che utilizzano interfacce tipo QBE (Query by Example): ACCESS
  - Tuttavia hanno sistemi per la traduzione automatica in SQL

Definizione degli oggetti in SQL



Introduciamo il Data Definition Language (DDL) SQL, che consiste nell' insieme delle istruzioni SQL che permettono la creazione, modifica e cancellazione delle tabelle, dei domini e degli altri oggetti del database, al fine di definire il suo schema logico.

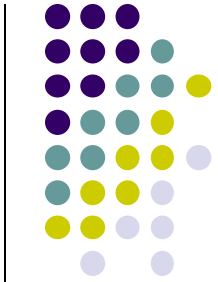
Definizione del database



Una volta loggati, mediante il comando show databases vengono mostrati i database presenti nell'ambiente in cui vi trovate. Di solito se siete nella root vedrete vari database di sistema che è bene non toccare.

Per creare il vostro database utilizzare il comando
Create database <nome_database>
A questo punto show databases evidenzierà la presenza del nuovo database

Accesso al database



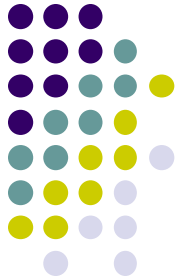
A questo punto è possibile utilizzare il database mediante L' istruzione use <nome_database>

Esempio:

Create database universita;

Use universita;

Definizione delle tabelle

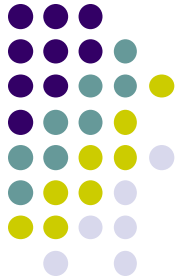


Le tabelle (corrispondenti alle relazioni dell'algebra relazionale) vengono definite in SQL mediante l`Istruzione CREATE TABLE.

Questa istruzione

- definisce uno schema di relazione e ne crea un'istanza vuota
- specifica attributi, domini e vincoli

CREATE TABLE, sintassi



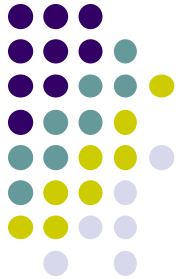
```
CREATE TABLE <nome_tabella>
(   nome_colonna_1  tipo_colonna_1  clausola_default_1 vincolo_di_colonna_1,
    nome_colonna_2  tipo_colonna_2  clausola_default_2 vincolo_di_colonna_2,
      ........
    nome_colonna_k  tipo_colonna_k clausola_default_k vincolo_di_colonna_k,
    vincoli di tabella
     )
```

L'istruzione che crea la tabella è CREATE TABLE, seguito da un nome che la caratterizza, e dalla lista delle colonne (attributi), di cui si specificano le caratteristiche. Alla fine si possono anche specificare eventuali vincoli di tabella, di cui parleremo in seguito

CREATE TABLE, esempio

```
CREATE TABLE IMPIEGATO (
   Matricola CHAR(6) PRIMARY KEY,
   Nome CHAR(20) NOT NULL,
   Cognome CHAR(20) NOT NULL,
   Dipart CHAR(15),
   Stipendio NUMERIC(9) DEFAULT 0,
   FOREIGN KEY(Dipart) REFERENCES
        Dipartimento(NomeDip),
   UNIQUE (Cognome,Nome)
 )
```

CREATE TABLE, effetto



L'effetto del comando CREATE TABLE definisce uno schema di relazione e ne crea un' istanza vuota, specificandone attributi, domini e vincoli.
Una volta creata, la tabella è pronta per l'inserimento dei dati che dovranno soddisfare i vincoli imposti.
Nel caso dell'esempio dato nella slide precedente, si ottiene

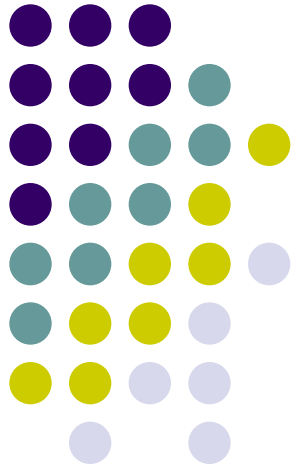
IMPIEGATO

Matricola	Nome	Cognome	Dipart	Stipendio

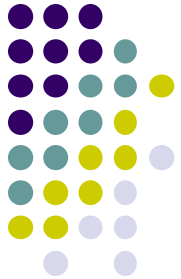
La visualizzazione dello schema di una tabella, dopo che è stata creata, può essere ottenuta mediante il comando SQL DESCRIBE. Nel caso specifico:

DESCRIBE impiegato

I domini



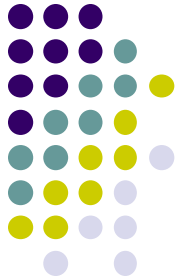
Domini



I domini sono i tipi di dato che possono assumere le colonne. Corrispondono ai tipi nei linguaggi di programmazione procedurali. Essi possono essere:

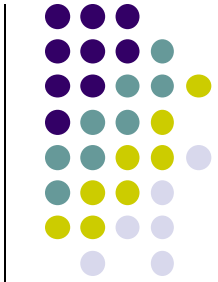
- Domini elementari (predefiniti)
- Domini definiti dall'utente (semplici, ma riutilizzabili)

Domini elementari



- Carattere: singoli caratteri o stringhe, anche di lunghezza variabile
- Numerici, esatti e approssimati
- Date
- Sistemi diversi estendono il set di base con domini non standard (vettori, periodi, ecc.)

Char e varchar

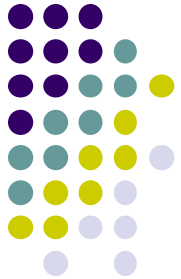


Il dominio di tipo carattere permette di definire singoli caratteri o stringhe di caratteri. Si possono usare i seguenti formati:

character (n) (o char(n)) definisce una stringa di n caratteri. Se manca il parametro n, il default è n=1, ossia un singolo carattere. Questa definizione è rigida, nel senso che viene destinato a quell'attributo lo spazio di esattamente n caratteri.

character varying (n) (o varchar(n)), più flessibile del precedente è il formato che è in grado di contenere stringhe fino a lunghezza n, destinando a quel particolare valore lo spazio che effettivamente utilizza, purchè non ecceda la taglia n. Il parametro n è in questo caso obbligatorio.

Tipi text e enum



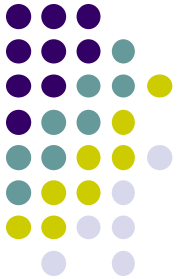
Char e varchar possono contenere fino a 255 caratteri. Per testi più lunghi si può utilizzare il tipo text che può memorizzare fino a 65535 caratteri

Infine il tipo enum permette alla variabile di assumere valori che fanno parte di una lista specificata

Esempio:

Genere enum ('m' , 'f')

Tipi numerici esatti



- Numeric (precisione, scala)
- Decimal o dec (precisione, scala)
- Integer o Int (4 byte)
- Tinyint (1 byte) Smallint (2 byte) Mediumint (3 byte) Bigint(8 byte)

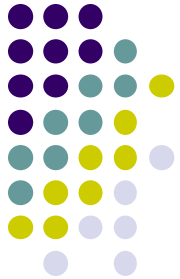
Precisione indica il numero massimo di cifre (intere e decimali).

Scala indica il numero di cifre dopo la virgola.

Ovviamente deve essere sempre scala<precisione.

La precisione di decimal è in genere maggiore o uguale alla precisione di numeric. In alcuni DBMS come MySQL sono sinonimi

Tipi numerici approssimati



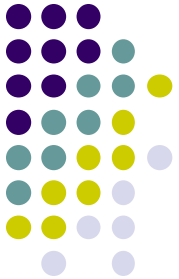
- Float(precisione)
- Double precision

Rappresentano i numeri reali. Sono rappresentati attraverso mantissa (m) ed esponente (n)

mEn indica $m*10^n$

In float precisione indica il massimo numero di cifre della mantissa. In real e double precision la precisione è fissata dall'implementazione. Ovviamente la precisione per double precision è maggiore che per real

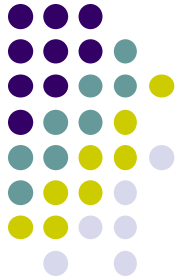
Date



Il tipo date in SQL rappresenta le date come sequenze di tre numeri, corrispondenti rispettivamente a year, month e day. Le costanti di tipo date vengono dichiarate e visualizzate come segue

'AAAA.MM.GG'

Time e timestamp

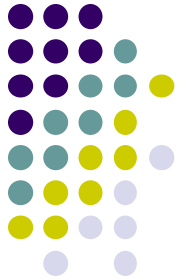


Il tipo time in SQL rappresenta le date come sequenze di tre numeri, corrispondenti rispettivamente a ore, minuti e secondi HH:MM:SS.

Il tipo timestamp invece include tutti i campi da year a second. AAAA:MM:GG:HH:MM:SS

Se in una colonna timestamp per una particolare riga non viene specificato nessun valore, allora viene memorizzato il momento in cui la riga è stata inserita o modificata per l'ultima volta

Definizione di domini

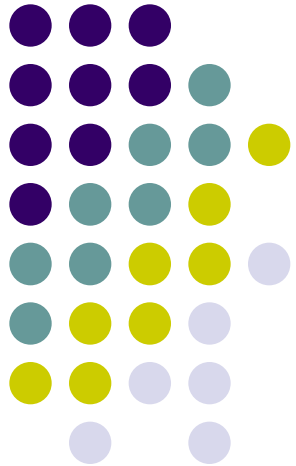


Istruzione CREATE DOMAIN:
definisce un dominio (semplice), utilizzabile in definizioni di relazioni, anche con vincoli e valori di default.

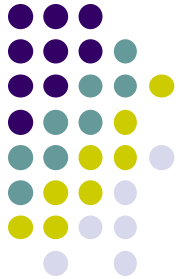
Esempio:

```
CREATE DOMAIN Voto
AS SMALLINT DEFAULT NULL
CHECK ( value >=18 AND value <= 30 )
```

I valori di default



Valori di default



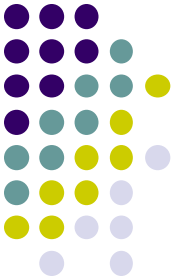
La clausola di default relativamente a un attributo assegna un valore specifico a una colonna quando per un certo dato non è assegnato esplicitamente un valore .
Si assegna mediante il comando

Default valore_default

inserito nella definizione di colonna dopo avere dichiarato il dominio relativo a quella colonna

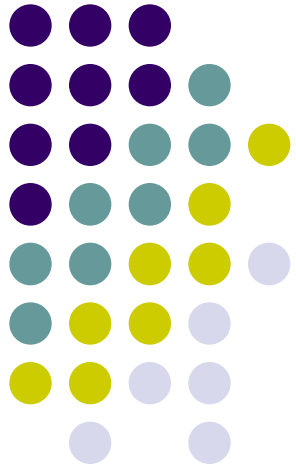
Quando il valore di default non è specificato, i valori non assegnati esplicitamente assumono automaticamente valore NULL.

Valori di default, esempio

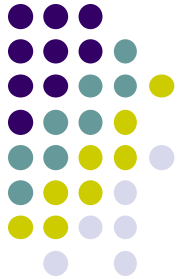


```
CREATE TABLE Impiegato(
        Matricola CHAR(6) PRIMARY KEY,
        codice_fiscale CHAR(16) UNIQUE,
        Nome VARCHAR(20) NOT NULL,
        Cognome VARCHAR(20) NOT NULL,
        Dipart VARCHAR(15),
        Stipendio NUMBER(9) DEFAULT 0,
        FOREIGN KEY(Dipart) REFERENCES
            Dipartimento(NomeDip),
        UNIQUE (Cognome,Nome)
 )
```

I vincoli



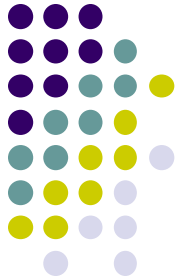
Vincoli intrarelazionali



I vincoli di integrità consentono di limitare i valori ammissibili per una determinata colonna della tabella in base a specifici criteri. I vincoli di integrità intrarelazionali (ossia che non fanno riferimento ad altre relazioni) sono:

- NOT NULL
- UNIQUE definisce chiavi
- PRIMARY KEY: chiave primaria (una sola, implica NOT NULL)
- CHECK, vedremo più avanti

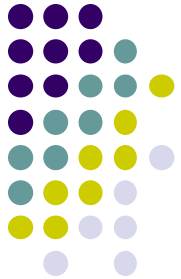
UNIQUE



- Può essere espresso in due forme:
  - nella definizione di un attributo, se forma da solo la chiave
  - come elemento separato

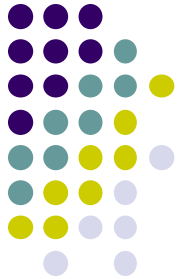
Il vincolo unique utilizzato nella definizione dell'attributo indica che non ci possono essere due valori uguali in quella colonna. E' una chiave della relazione, ma non una chiave primaria.

Esempio di vincolo unique



```
CREATE TABLE Impiegato(
    Matricola CHAR(6) PRIMARY KEY,
    Codice_fiscale CHAR(16) UNIQUE,
    Nome VARCHAR(20) NOT NULL,
    Cognome VARCHAR(20) NOT NULL,
    Dipart VARCHAR(15),
    Stipendio NUMBER(9) DEFAULT 0,
    FOREIGN KEY(Dipart) REFERENCES
          Dipartimento(NomeDip),
    UNIQUE (Cognome,Nome)
 )
```

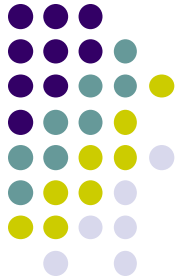
Vincolo unique per insiemi di attributi



Il vincolo di unicità può anche essere riferito a coppie o insiemi di attributi. Ciò non significa che per gli attributi dell'insieme considerato ogni singolo valore deve apparire una sola volta, ma che non ci siano due dati (righe) per cui l'insieme dei valori corrispondenti a quegli attributi siano uguali.
In questo caso il vincolo viene dichiarato dopo aver dichiarato tutte le colonne mediante un vincolo di tabella, utilizzando il comando

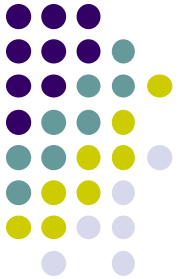
Unique (lista attributi)

Esempio vincolo unique per insiemi di attributi



```
CREATE TABLE Impiegato(
   Matricola CHAR(6) PRIMARY KEY,
   codice_fiscale CHAR(16) UNIQUE,
   Nome VARCHAR(20) NOT NULL,
   Cognome VARCHAR(20) NOT NULL,
   Dipart VARCHAR(15),
   Stipendio NUMBER(9) DEFAULT 0,
   FOREIGN KEY(Dipart) REFERENCES
        Dipartimento(NomeDip),
   UNIQUE (Cognome,Nome)
 )
```

Vincolo di unicità su più attributi, attenzione!



```
Nome        VARCHAR(20) NOT NULL,
Cognome     VARCHAR(20) NOT NULL,
UNIQUE (Cognome,Nome),
```

```
Nome        VARCHAR(20) NOT NULL UNIQUE,
Cognome  VARCHAR(20) NOT NULL UNIQUE,
```

Non è la stessa cosa!

La prima espressione indica che non ci possono essere due persone con lo stesso nome e cognome

La seconda indica che non ci possono essere due persone con lo stesso nome né due persone con lo stesso cognome.

Primary key

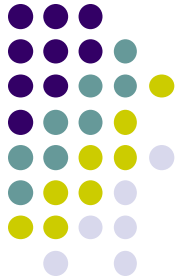
- due forme:
  - nella definizione di un attributo, se forma da solo la chiave
  - come elemento separato

Il vincolo PRIMARY KEY è simile a unique, ma definisce la chiave primaria della relazione, ossia un attributo che individua univocamente un dato. Implica sia il vincolo UNIQUE che il vincolo NOT NULL (non è ammesso che per uno degli elementi della tabella questo valore sia non definito). Serve ad indentificare univocamente i soggetti del dominio. Questo vincolo permette spesso il collegamento fra due tabelle (vedremo in seguito)

Esempio chiave primaria

```
CREATE TABLE Impiegato(
   Matricola CHAR(6) PRIMARY KEY,
   Codice_fiscale CHAR(16) UNIQUE,
   Nome VARCHAR(20) NOT NULL,
   Cognome VARCHAR(20) NOT NULL,
   Dipart VARCHAR(15),
   Stipendio NUMBER(9) DEFAULT 0,
   FOREIGN KEY(Dipart) REFERENCES
        Dipartimento(NomeDip),
   UNIQUE (Cognome,Nome)
 )
```

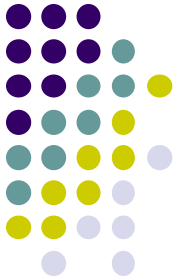
Chiave primaria con insiemi di attributi



Analogamente al vincolo unique, anche il vincolo di chiave primaria può essere definito su un insieme di elementi.
In tal caso la sintassi è simile a quella di unique

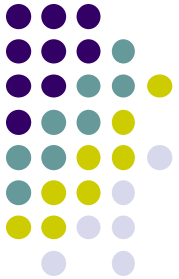
Primary key (lista di attributi)

Esempio chiave primaria con insiemi di attributi



```
CREATE TABLE Studente(
   Nome VARCHAR(20),
   Cognome VARCHAR(20),
   nascita DATE,
   Corso_Laurea VARCHAR(15),
   Facolta VARCHAR (20)
   PRIMARY KEY(Cognome,Nome, Nascita)
 )
```

Esercitazione



Costruzione di un database di un mobilificio, relativo alla vendita di mobili componibili.
Costruiremo via via le tabelle che ci servono

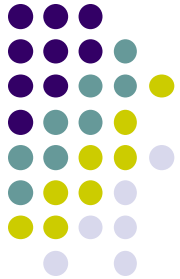
Esercizio 1

Costruire una tabella Categorie per descrivere le categorie di mobili presenti nel mobilificio.
Di tali categorie definire il codice (cat_cod) di tre caratteri e la descrizione (cat_descrizione) che è una stringa di al più 30 caratteri. Scegliere un'opportuna chiave primaria, e vincoli di unicità, se servono.

Soluzione:

```
Create table Categorie
(cat_cod char(3) primary key,
 Cat_descrizione varchar (30) unique)
```

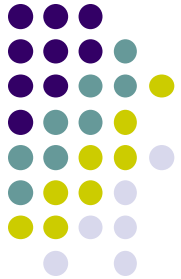
Avete fatto errori?



In tal caso usate il comando DROP TABLE (cancella la tabella) e createla correttamente

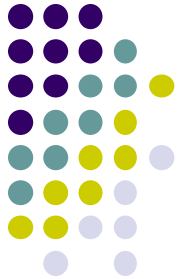
```
DROP TABLE categorie
```

```
CREATE TABLE Categorie
(cat_cod char(3) primary key,
 Cat_descrizione varchar (30) unique)
```



Categorie
Cat_cod
Cat_descrizione

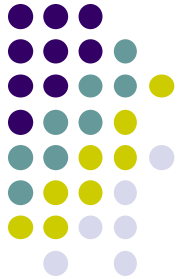
Esercizio 2



Creare una tabella dei laboratori in cui vengono prodotte le componenti utilizzate negli articoli prodotti dal mobilificio.
Tale tabella deve contenere un codice del laboratorio (lab_cod) di 4 caratteri, l' indirizzo (lab_indirizzo), la città (lab_città),il numero di telefono (lab_telefono).
Determinare chiave e vincoli di unicità.

Soluzione

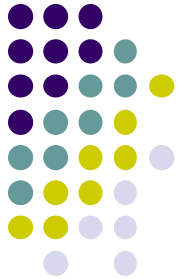
```
CREATE TABLE Laboratori
( Lab_cod char(4) PRIMARY KEY,
  Lab_indirizzo varchar(50),
  Lab_citta varchar (20),
  Lab_telefono varchar (12) UNIQUE)
```



Categorie
Cat_cod
Cat_descrizione

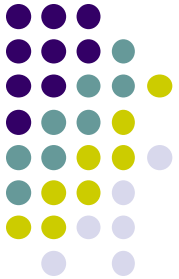
Laboratori
Lab_cod
Lab_indirizzo
Lab_città
Lab_telefono

Vincolo not null



Il vincolo NOT NULL obbliga un certo attributo a non avere valori non definiti. Spesso il valore NULL è il valore di default che viene attribuito quando c'è un dato mancante, a meno che non venga specificato un diverso valore di default. Come gli altri vincoli, il vincolo NOT NULL viene dichiarato subito dopo la dichiarazione di un attributo.

Esempio not null



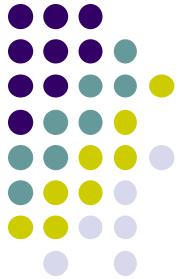
```
CREATE TABLE Impiegato(
    Matricola CHAR(6) PRIMARY KEY,
    Nome VARCHAR(20) NOT NULL,
    Cognome VARCHAR(20) NOT NULL,
    Dipart VARCHAR(15),
    Stipendio NUMBER(9) DEFAULT 0,
    FOREIGN KEY(Dipart) REFERENCES
          Dipartimento(NomeDip),
    UNIQUE (Cognome,Nome)
 )
```

Esercizio 3

Costruire la tabella Negozi, che contiene i codici, le denominazioni, gli indirizzi, le città e i numeri di telefono dei negozi che il mobilificio rifornisce.

Soluzione

```
CREATE TABLE Negozi
(neg_cod Char(4) primary key,
  neg_nome varchar (50) NOT NULL,
  neg_indirizzo Varchar(50),
  neg_citta Varchar(15),
  neg_telefono Varchar(12) UNIQUE)
```

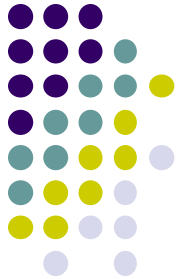


Negozi
Neg_cod
Neg_nome
Neg_indirizzo
Neg_città
Neg_Telefono

Categorie
Cat_cod
Cat_descrizione

Laboratori
Lab_cod
Lab_indirizzo
Lab_città
Lab_telefono

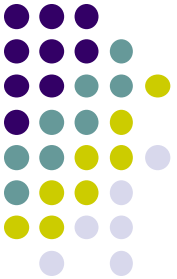
Vincoli interrelazionali



I vincoli interrelazionali sono quei vincoli che vengono imposti quando gli attributi di due diverse tabelle devono essere messi in relazione.

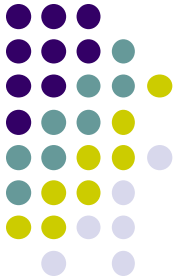
Questo è fatto per soddisfare l'esigenza di un database di non essere ridondante e di avere i dati sincronizzati.
Se due tabelle gestiscono gli stessi dati, è bene che di essi non ce ne siano più copie, sia allo scopo di non occupare troppa memoria, sia affinchè le modifiche fatte su dati uguali utilizzati da due tabelle siano coerenti.

Vincoli interrelazionali



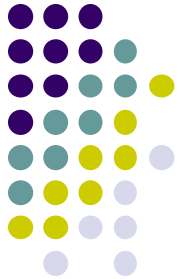
- CHECK, vedremo più avanti
- REFERENCES e FOREIGN KEY permettono di definire vincoli di integrità referenziale
- di nuovo due sintassi
  - per singoli attributi (come vincolo di colonna)
  - su più attributi (come vincolo di tabella)
- E' possibile definire politiche di reazione alla violazione (ossia stabilire l'azione che il DBMS deve compiere quando si viola il vincolo)

Vincolo di chiave esterna come vincolo di colonna



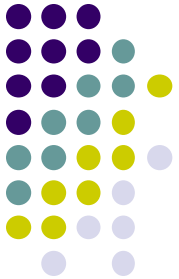
```
CREATE TABLE <nome tabella>
(attributo_1…,
 attributo_2…,
  …
 attributo_k REFERENCES tabella_riferita(colonna_riferita)
  …
  )
```

Vincolo di chiave esterna come vincolo di tabella



```
CREATE TABLE <nome_tabella>
(attributo_1 …,
 attributo_2 …,
…
attributo_n …,
FOREIGN KEY (col_referenti) REFERENCES tab_riferita(col_riferite)
…
)
```

Vincolo referenziale, esempio



Auto

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

Come vincolo di tabella

Infrazioni

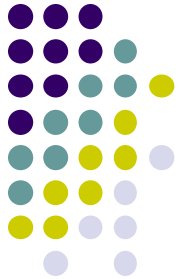
Codice	Data	Vigile	Prov	Numero
34321	1/2/95	3987	MI	39548K
53524	4/3/95	3295	TO	E39548
64521	5/4/96	3295	PR	839548
73321	5/2/98	9345	PR	839548

Come vincolo di colonna

Vigili

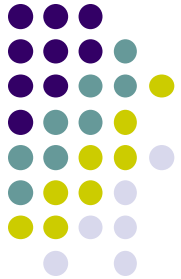
Matricola	Cognome	Nome
3987	Rossi	Luca
3295	Neri	Piero
9345	Neri	Mario
7543	Mori	Gino

Vincolo references, foreign key, esempio



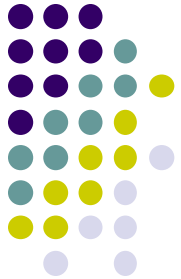
```
Create TABLE Vigili (
    Matricola INTEGER PRIMARY KEY,
    Nome VARCHAR (15),
    Cognome VARCHAR (15))

CREATE TABLE Auto (
    Prov CHAR(2),
    Numero CHAR (6),
    Cognome VARCHAR(15),
    Nome VARCHAR(15),
    Primary key (prov, numero))

CREATE TABLE Infrazioni(
    Codice CHAR(6) PRIMARY KEY,
    Data  DATE NOT NULL,
    Vigile  INTEGER NOT NULL REFERENCES Vigili(Matricola),
    Provincia  CHAR(2),
    Numero     CHAR(6) ,
    FOREIGN KEY(Provincia, Numero) REFERENCES Auto(Prov, Numero)  )
```

Esercizio 4

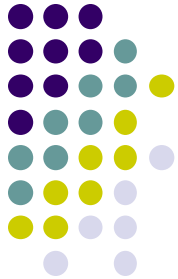


Costruire una tabella Articoli che contiene gli articoli venduti dal mobilificio.
Ciascuno di tali articoli è caratterizzato da un codice (art_cod) di 4 caratteri, appartiene a una categoria (cat_cod), viene data una descrizione(art_descrizione) che ha al piu' 30 caratteri, il suo prezzo (art_prezzo), l'aliquota IVA (art_IVA) e le spese di trasporto (art_spese_trasporto).

Esercizio 4, soluzione



```
CREATE TABLE Articoli
   ( art_cod char (4) primary key,
     cat_cod char (3) references Categorie(cat_cod),
     art_descrizione varchar(30) not null,
     art_prezzo number (8,2),
     art_IVA integer,
     art_spese_trasporto number (8,2)
    )
```



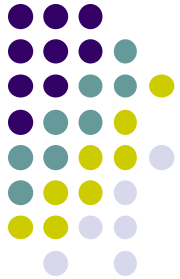
Negozi
Neg_cod
Neg_nome
Neg_indirizzo
Neg_città
Neg_Telefono

Categorie
Cat_cod
Cat_descrizione

Articoli
Art_cod
Cat_cod
Art_descrizione
Art_prezzo
Art_IVA
Art_spese_trasporto

Laboratori
Lab_cod
Lab_indirizzo
Lab_città
Lab_telefono

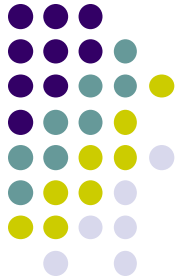
Reazione alla violazione



Quando si crea un vincolo foreign key in una tabella, in SQL standard si può specificare l'azione da intraprendere quando delle righe nella tabella riferita vengono cancellate o modificate.
Tali reazioni alla violazione vengono dichiarate al momento della definizione dei vincoli di foreign key rispettivamente mediante i comandi

ON DELETE
ON UPDATE

Sintassi



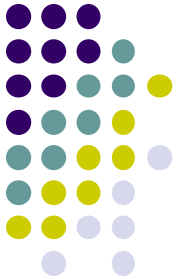
Per vincoli foreign key di colonna:

```
SELECT TABLE nome_tabella
( attr_1…,
  attr_2…,
  …
  attr_k … REFERENCES tab_riferita(attr_riferito)
                              ON DELETE/ON UPDATE reazione
  …
  attr_n …
  vincoli tabella)
```

Per vincoli foreign key di tabella:

```
SELECT TABLE nome_tabella
( attr_1…,
  attr_2…,
  …
  attr_n …,
 FOREIGN KEY tab_referente(attr_referenti) REFERENCES
        tab_riferita(attr_riferito) ON DELETE/ON UPDATE reazione
```

Reazioni alla violazione on delete



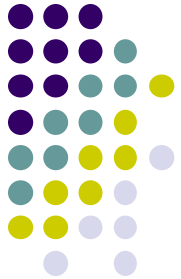
Impedire il delete (NO ACTION): Blocca il delete delle righe dalla tabella riferita quando ci sono righe che dipendono da essa. Questa è l'azione che viene attivata per default.

Generare un delete a catena (CASCADE): Cancella tutte le righe dipendenti dalla tabella quando la corrispondente riga è cancellata dalla tabella riferita.

Assegnare valore NULL (SET NULL): Assegna NULL ai valori della colonna che ha il vincolo foreign key nella tabella quando la riga corrispondente viene cancellata dalla tabella riferita.

Assegnare il valore di default (SET DEFAULT): Assegna il valore di default ai valori della colonna che ha il vincolo foreign key nella tabella quando la riga corrispondente viene cancellata dalla tabella riferita.

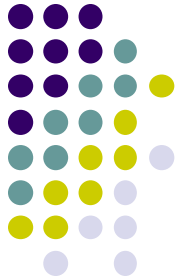
Reazioni alla violazione on update



Nello standard SQL la reazione alla violazione può anche essere attivata quando i dati della tabella riferita vengono aggiornati.
Viene attivato mediante il comando ON UPDATE seguito da:

CASCADE : Le righe della tabella referente vengono impostati ai valori della tabella riferita
SET NULL : i valori della tabella referente vengono impostati a NULL
SET DEFAULT : i valori della tabella referente vengono impostati al valore di default
NO ACTION : rifiuta gli aggiornamenti che violino l' integrità referenziale

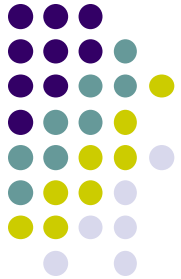
Eliminazione



L' istruzione per l' eliminazione di un database o di una Tabella è drop. In particolare:
Per eliminare un database l' istruzione è
Drop database <nome_database>

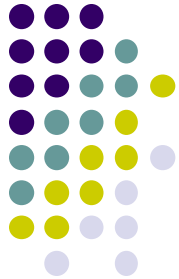
Per eliminare una tabella l' istruzione è
Drop table <nome_tabella>

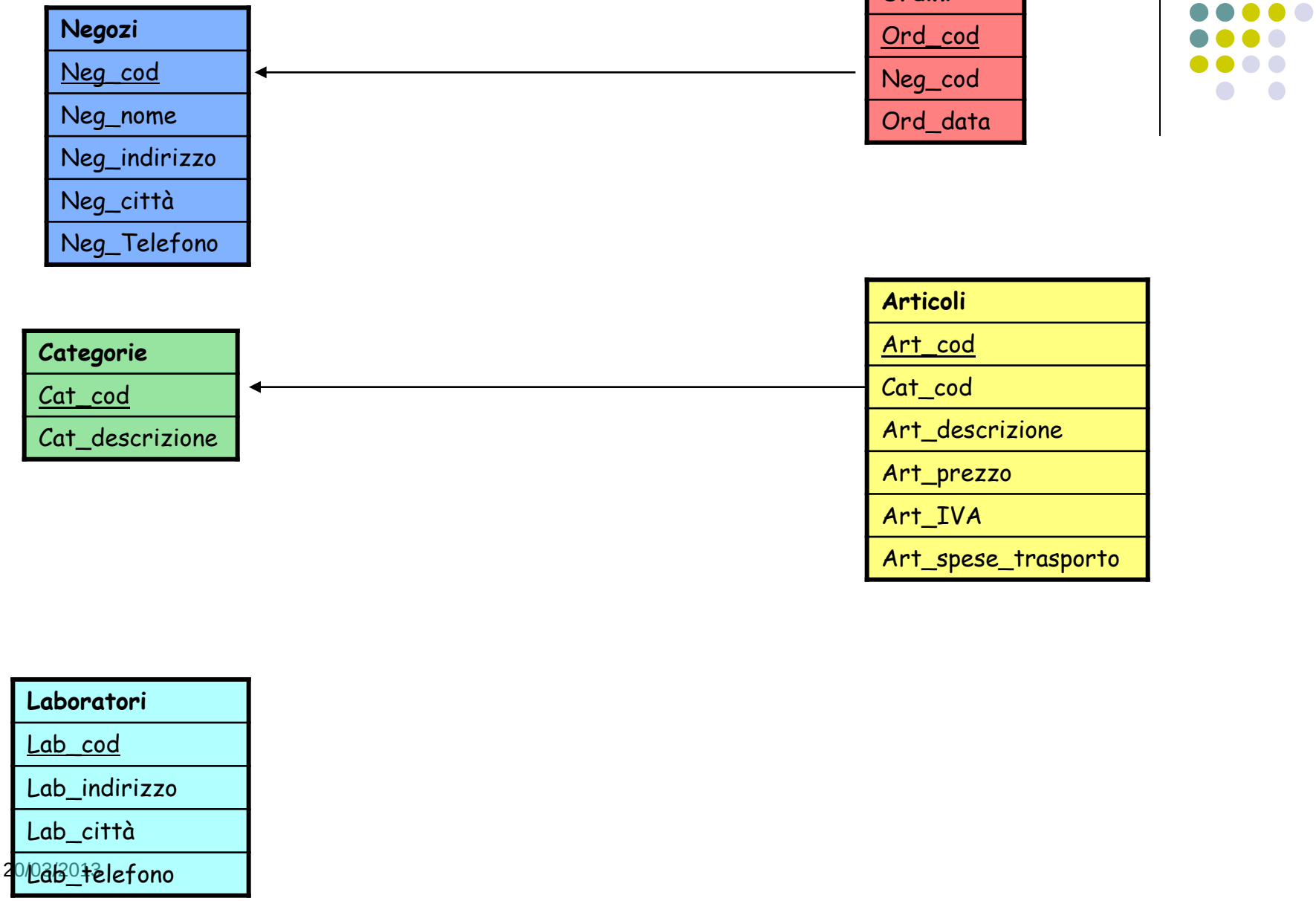
Esercizio



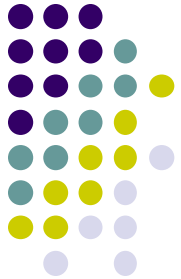
Costruire una tabella Ordini per gestire gli ordini ricevuti dal mobilificio da parte dei negozi. Memorizzare nella tabella il codice dell' ordine ord_cod (di 6 caratteri), il codice del negozio neg_cod che effettua l' ordine e la data dell' ordine ord_data. Si specifichino, se necessario, le reazioni alla violazione in caso di cancellazione delle eventuali tabelle riferite

```
CREATE TABLE Ordini
(Ord_cod char(6) primary key,
  Neg_cod char(4) REFERENCES Negozi(Neg_cod)
            ON DELETE CASCADE,
Ord_data date)
```



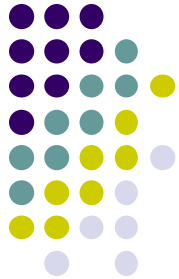

Negozi
Neg_cod
Neg_nome
Neg_indirizzo
Neg_città
Neg_Telefono
Ordini
Ord_cod
Neg_cod
Ord_data
Articoli
Art_cod
Cat_cod
Art_descrizione
Art_prezzo
Art_IVA
Art_spese_trasporto
Categorie
Cat_cod
Cat_descrizione
Laboratori
Lab_cod
Lab_indirizzo
Lab_città
Lab_telefono

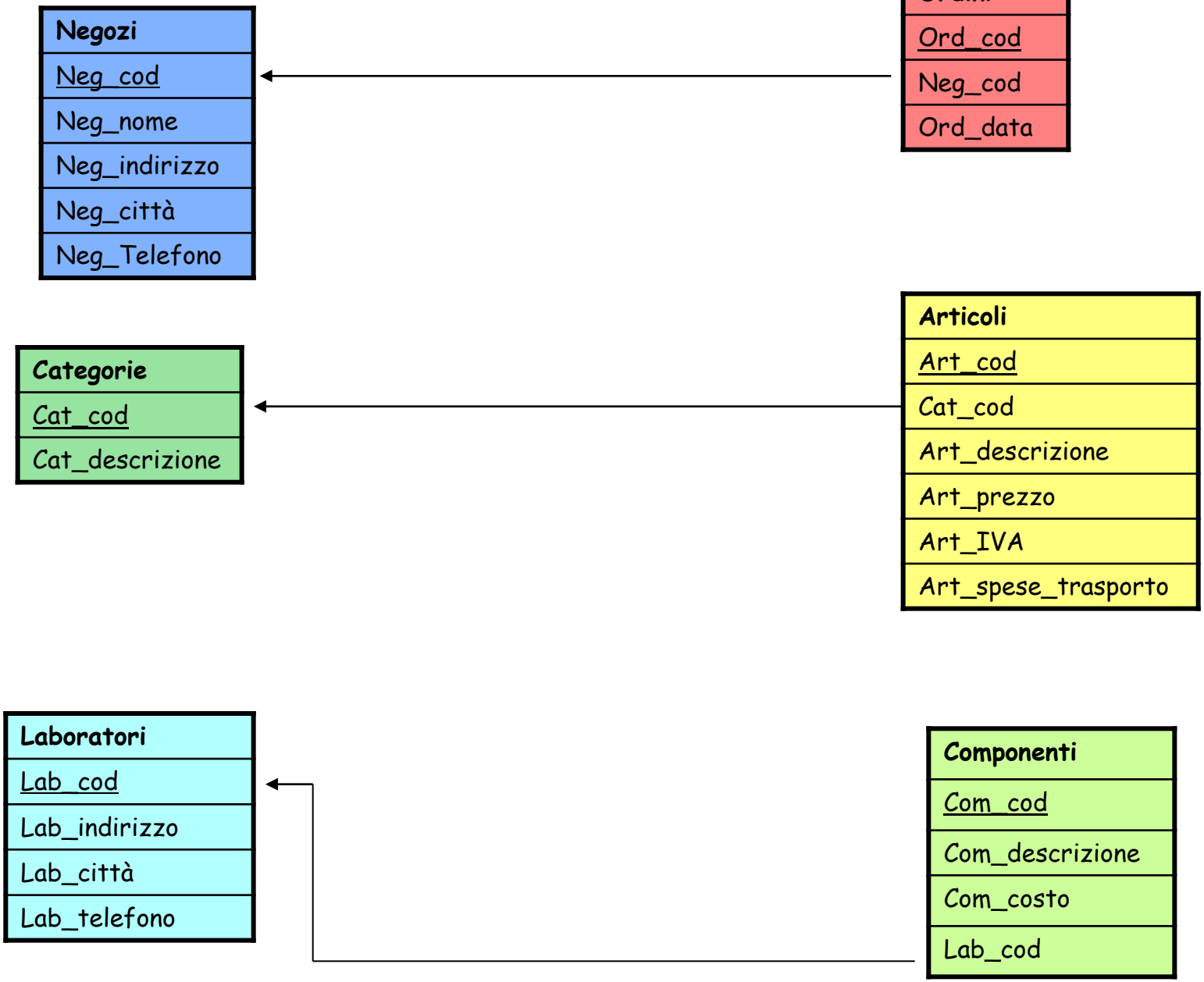
Esercizio



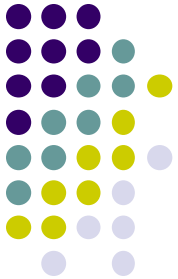
Definire lo schema della tabella Componenti, che descrive i componenti utilizzati nella costruzione dei mobili del Mobilificio.Tale tabella deve contenere un codice del componente com_cod, 4 caratteri, la descrizione del componente com_descrizione, max 30 caratteri, il suo costo com_costo e il codice del laboratorio lab_cod in cui viene prodotto. Si specifichino se necessario le reazioni alla violazione in caso di cancellazione delle eventuali tabelle riferite

```
CREATE TABLE Componenti
(com_cod CHAR(4) Primary key,
com_descrizione varchar(30),
Com_costo number(8,2),
Lab_cod char(4) REFERENCES Laboratori(lab_cod) ON DELETE
SET NULL)
```




Negozi
Neg_cod
Neg_nome
Neg_indirizzo
Neg_città
Neg_Telefono
Ordini
Ord_cod
Neg_cod
Ord_data
Categorie
Cat_cod
Cat_descrizione
Articoli
Art_cod
Cat_cod
Art_descrizione
Art_prezzo
Art_IVA
Art_spese_trasporto
Laboratori
Lab_cod
Lab_indirizzo
Lab_città
Lab_telefono
Componenti
Com_cod
Com_descrizione
Com_costo
Lab_cod

CHECK

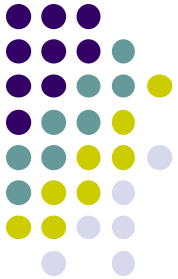


Un vincolo di CHECK richiede che una colonna, o una combinazione di colonne, soddisfi una condizione per ogni riga della tabella. Il vincolo CHECK deve essere una espressione booleana che è valutata usando i valori della colonna che vengono inseriti o aggiornati nella riga.

Può essere espresso sia come vincolo di riga che come vincolo di tabella.

Se è espresso come vincolo di riga, può coinvolgere solo l'attributo su cui è definito, mentre se serve eseguire un check che coinvolge due o più attributi, si deve definire come vincolo di tabella

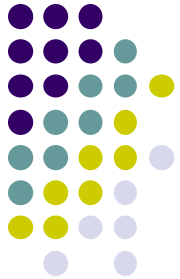
CHECK, osservazioni



Si usa il vincolo CHECK quando si ha bisogno di rinforzare le regole di integrità basate su espressioni logiche, come confronti.
Non usare mai il vincolo CHECK quando qualcuno degli altri tipi di vincoli di integrità forniscono il checking necessario.

Questo perchè gli altri vincoli di integrità si esprimono in maniera più immediata e inoltre consentono di definire la reazione alla violazione, non consentita dalla sintassi del check

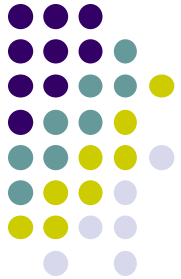
CHECK, esempio



Nessuno stipendio degli impiegati può avere valore minore di 0 (espresso come vincolo di riga).

```
CREATE TABLE IMPIEGATO (
        Matricola CHAR(6) PRIMARY KEY,
        Nome CHAR(20) NOT NULL,
        Cognome CHAR(20) NOT NULL,
        Dipart CHAR(15),
        Stipendio NUMERIC(9) DEFAULT 0 CHECK (Stipendio>=0),
        FOREIGN KEY(Dipart) REFERENCES
                        Dipartimento(NomeDip),
        UNIQUE (Cognome,Nome)
)
```

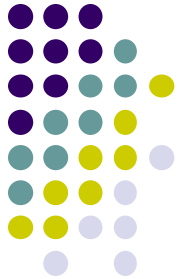
CHECK, esempio



Nessuno stipendio degli impiegati può avere valore minore di 0 (espresso come vincolo di tabella).

```
CREATE TABLE IMPIEGATO (
        Matricola CHAR(6) PRIMARY KEY,
        Nome CHAR(20) NOT NULL,
        Cognome CHAR(20) NOT NULL,
        Dipart CHAR(15),
        Stipendio NUMERIC(9) DEFAULT 0,
        FOREIGN KEY(Dipart) REFERENCES
                Dipartimento(NomeDip),
        UNIQUE (Cognome,Nome) ,
        CHECK (Stipendio>=0)
)
```

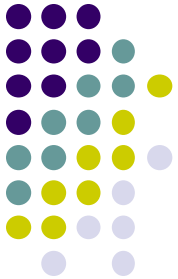
Check, esempio



I dipartimenti si possono trovare solo nelle locazioni di Boston, New York e Dallas.

```
CREATE TABLE Dipartimenti
(dip_cod char(4) primary key,
 dip_nome varchar2(20) not null,
 dip_citta varchar2(15) not null,
 CHECK (dip_citta ='Boston'
       or dip_citta='New York'
       or dip_citta='Dallas')
)
```

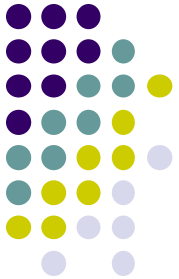
Esempio Check



Un vincolo check sullo stipendio e la commissione per evitare che la commissione sia più alta del salario.

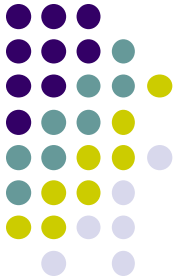
```
Create table pagamenti(
pag_cod char(6),
pag_codicef char(16) REFERENCES dipendenti(dipe_codicef),
pag_stipendio number(8,2),
pag_commissione number (8,2),
check (pag_stipendio>pag_commissione) )
```

Esercizio

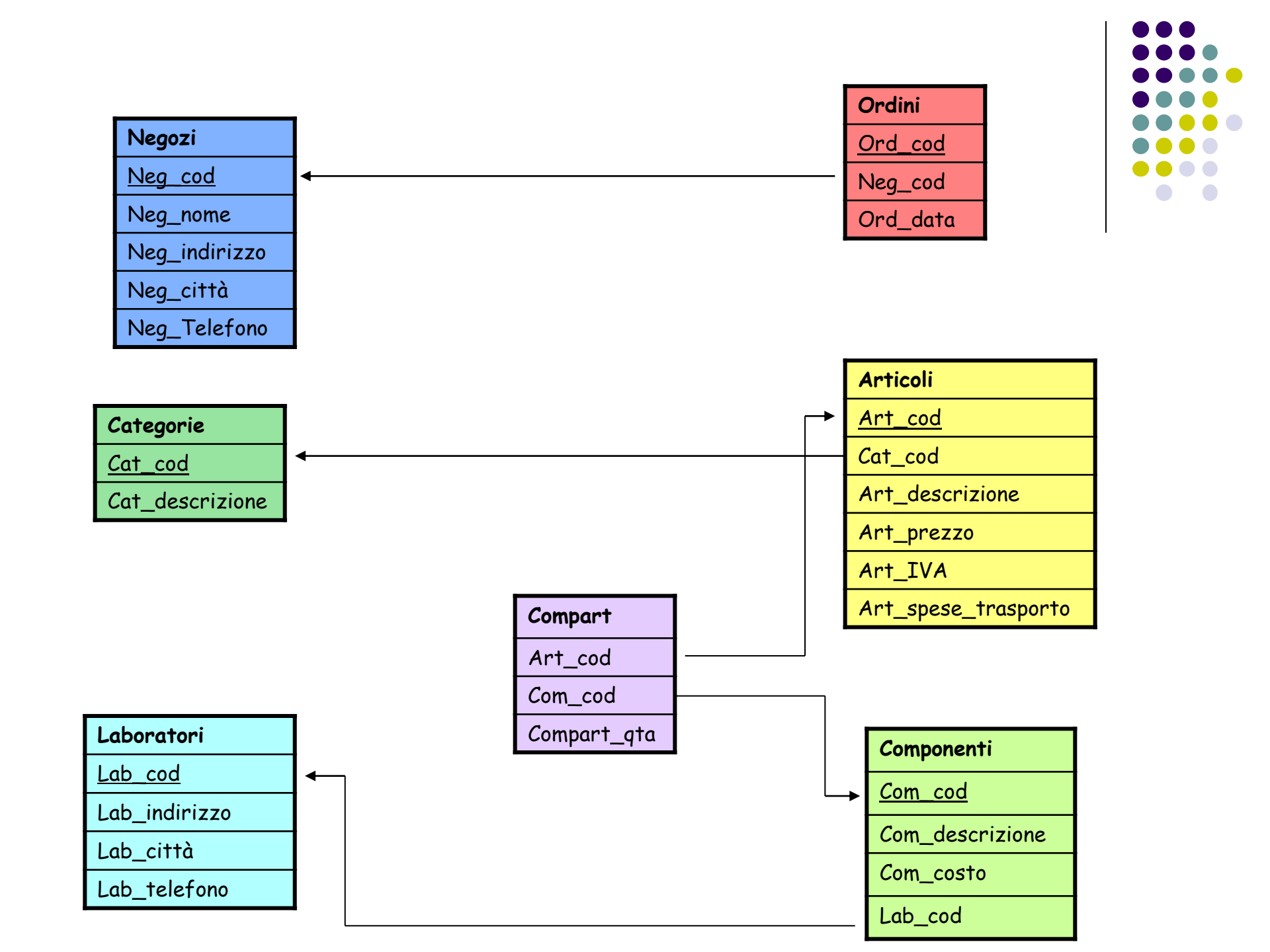


La tabella Compart serve a stabilire la quantità di ogni componente che serve per la composizione di ogni articolo.
Una tale tabella deve contenere il codice dell' articolo (art_cod), il codice del componente (com_cod) e la quantità del componente nell' articolo (Compart_Qta).
Creare la tabella con i rispettivi vincoli referenziali ed eventuali reazioni alla violazione dei vincoli di chiave esterna.

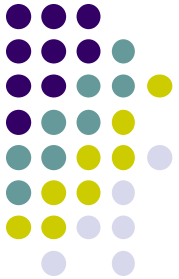
Esercizio, Soluzione



```
CREATE TABLE Compart
( Art_cod CHAR(4) References Articoli(Art_cod),
  Com_cod CHAR(4) References Componenti(Com_cod),
  Compart_qta NUMBER(4),
    PRIMARY KEY (Art_cod, Com_cod))
```


Negozi
Neg_cod
Neg_nome
Neg_indirizzo
Neg_città
Neg_Telefono
Ordini
Ord_cod
Neg_cod
Ord_data
Categorie
Cat_cod
Cat_descrizione
Articoli
Art_cod
Cat_cod
Art_descrizione
Art_prezzo
Art_IVA
Art_spese_trasporto
Compart
Art_cod
Com_cod
Compart_qta
Laboratori
Lab_cod
Lab_indirizzo
Lab_città
Lab_telefono
Componenti
Com_cod
Com_descrizione
Com_costo
Lab_cod

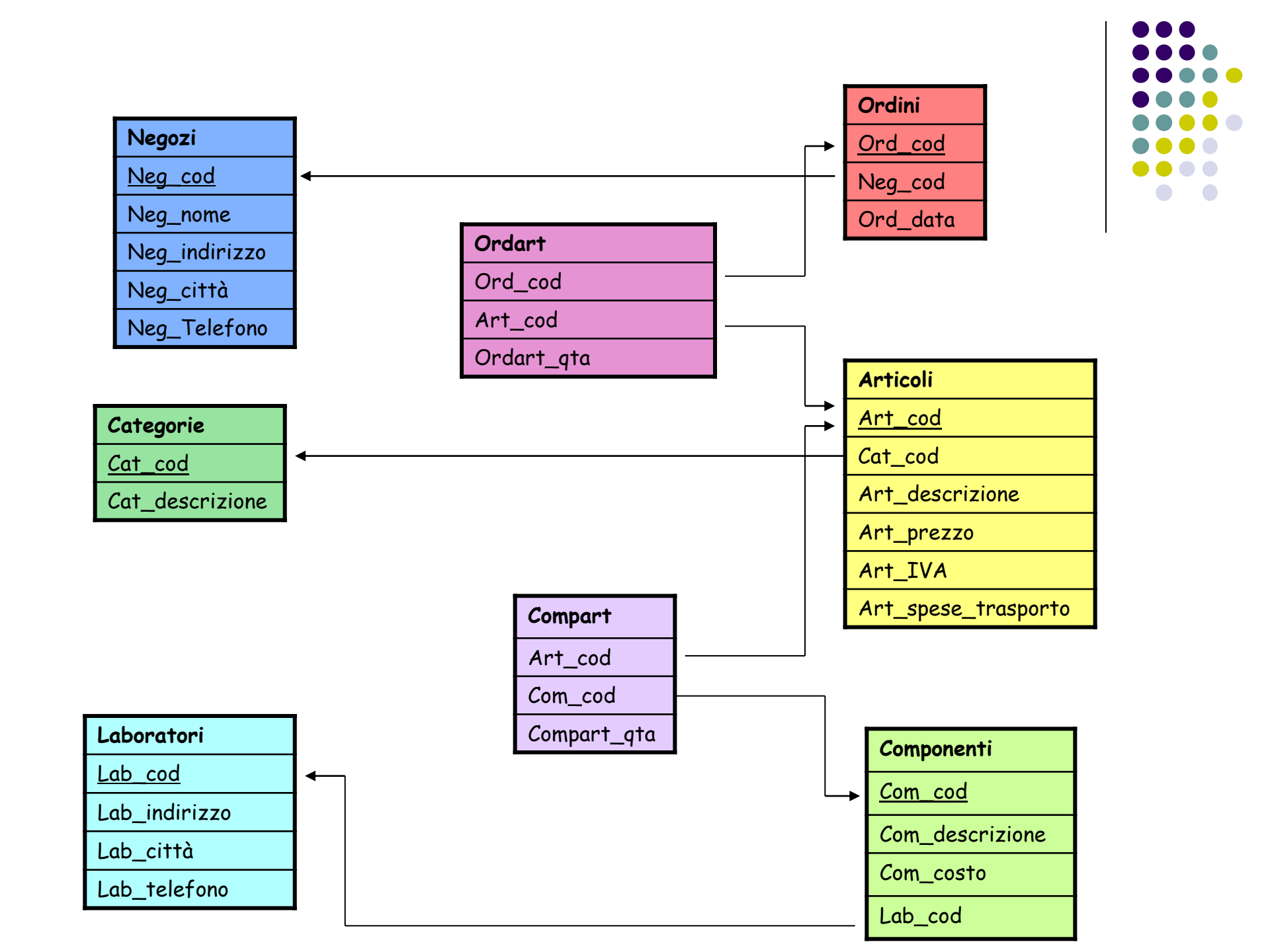
Esercizio



La tabella Ordart descrive per ogni articolo la quantità di pezzi che viene richiesta in ogni ordine.
È dunque costituita da una colonna ord_cod che contiene il codice di un ordine, la colonna art_cod che contiene il codice dell' articolo considerato e la quantità di pezzi ordart_qta dell' articolo, richiesti in quell' ordine.
Costruire questa tabella specificando i vincoli necessari e le eventuali reazioni alla violazione dei vincoli.

Esercizio, soluzione

```
CREATE TABLE Ordart
(ord_cod char(4) references Ordini(ord_cod),
 art_cod char(4) references Articoli(art_cod),
 ordart_qta number(4) not null,
Primary key (ord_cod, art_cod)
)
```


Negozi
Neg_cod
Neg_nome
Neg_indirizzo
Neg_città
Neg_Telefono
Ordini
Ord_cod
Neg_cod
Ord_data
Ordart
Ord_cod
Art_cod
Ordart_qta
Articoli
Art_cod
Cat_cod
Art_descrizione
Art_prezzo
Art_IVA
Art_spese_trasporto
Categorie
Cat_cod
Cat_descrizione
Compart
Art_cod
Com_cod
Compart_qta
Laboratori
Lab_cod
Lab_indirizzo
Lab_città
Lab_telefono
Componenti
Com_cod
Com_descrizione
Com_costo
Lab_cod