VINCOLI

Ripasso: I PRINCIPALI TIPI DI DATO

CHAR(n) ……..: definisce una stringa di **n** caratteri a lunghezza fissa.

VARCHAR(n) ..: definisce una stringa di lunghezza **variabile** fino ad un massimo di **n** caratteri.

DECIMAL(n,v) : definisce un numero che puo' contenere al massimo n cifre di cui v cifre decimali.
Esempio : NUMERIC(8,2) → 999,999.99

INTEGER ……: definisce un numero intero da 16 o 32 bit.

FLOAT
REAL ……: Definiscono numeri a virgola mobile.

DATE ……..: Definisce una data.

TIME ……..: Definisce l'orario.

Ripasso: COME CREARE UNA TABELLA

Comando DDL

```
CREATE TABLE nome_tabella
(
    nome_attributo_1     TIPO_DATO,
    nome_attributo_2     TIPO_DATO,
        . . .                . . .
);
```

Esempio

```
CREATE TABLE studenti
(
    matricola    CHAR(6),
    cognome      VARCHAR(30),
    nome         VARCHAR(30),
    data_nascita DATE
);
```

Ripasso: COME INSERIRE DATI ALL'INTERNO DI UNA TABELLA

Comando DML

```
INSERT INTO nome_tabella VALUES (dato_1, dato_2, …);
```

Esempio

```
INSERT INTO studenti VALUES ('200768', 'Verdi', 'Fabio', 12-FEB-1792);
INSERT INTO studenti VALUES ('937653', 'Rossi', 'Luca',  10-OCT-1971);
INSERT INTO studenti VALUES ('937653', 'Bruni', 'Mario', 01-DEC-1971);
```

ESEMPIO: Creazione di un Data Base (senza vincoli)

STUDENTI

Matricola	Cognome	Nome	Data di nascita
200768	Verdi	Fabio	**12/02/1792**
937653	Rossi	Luca	10/10/1971
937653	Bruni	Mario	01/12/1971

CORSI

Codice	Materia	Docente
01	Analisi	Giani
03	Chimica	Melli
04	Chimica	Belli

ESAMI

Studente	Voto	Lode	Corso
200768	**36**		**05**
937653	28	**Lode**	01
937653	30	Lode	01
276545	25		04

```
CREATE TABLE studenti
(
     matricola      CHAR(6),
     cognome        VARCHAR(20),
     nome           VARCHAR(20),
     data_nascita   DATE
);

CREATE TABLE corsi
(
     codice         CHAR(2),
     materia        VARCHAR(20),
     docente        VARCHAR(20)
);

CREATE TABLE esami
(
     studente       CHAR(6),
     voto           NUMERIC(2),
     lode           CHAR(4),
     corso          CHAR(2)
);
```

A questo punto e' possibile inserire i dati :

```
INSERT INTO studenti VALUES ('200768','Verdi','Fabio','12-FEB-1792');
INSERT INTO studenti VALUES ('937653','Rossi','Luca', '10-OCT-1971');
INSERT INTO studenti VALUES ('937653','Bruni','Mario','01-DEC-1971');

INSERT INTO corsi    VALUES ('01','Analisi','Giani');
INSERT INTO corsi    VALUES ('03','Chimica','Melli');
INSERT INTO corsi    VALUES ('04','Chimica','Belli');

INSERT INTO esami    VALUES ('200768', 36,'',    '05');
INSERT INTO esami    VALUES ('937653', 28,'lode','01');
INSERT INTO esami    VALUES ('937653', 30,'lode','01');
INSERT INTO esami    VALUES ('276545', 25,'',    '04');
```

Questa istanza di Data Base è coerente???

COME CREARE LA CHIAVE PRINCIPALE DI UNA TABELLA

La chiave primaria puo' essere indicata in fase di creazione della tabella:

```
CREATE TABLE studenti
(
    matricola    CHAR(6)      PRIMARY KEY,
    cognome      VARCHAR(30),
    nome         VARCHAR(30),
    data_nascita DATE
);
```

NOTA BENE → In ogni tabella e' possibile definire **una sola chiave primaria**.
→ Se un campo e' stato definito come chiave primaria **non** puo' mai accettare valori **nulli**.

```
INSERT INTO studenti VALUES ('', 'Verdi', 'Fabio', 12-FEB-1792);
```

Questo statement viene rifiutato dal motore SQL segnalando il relativo errore.

Se si eseguono in successione i seguenti statement

```
INSERT INTO studenti VALUES ('937653', 'Rossi', 'Luca',  10-OCT-1971);
INSERT INTO studenti VALUES ('937653', 'Bruni', 'Mario', 01-DEC-1971);
```

il primo viene accettato mentre il secondo viene respinto con la relativa segnalazione di errore (*duplicate key*).

E' possibile anche scrivere lo statement di creazione nel seguente modo:

```
CREATE TABLE studenti
(
    matricola    CHAR(6),
    cognome      VARCHAR(30),
    nome         VARCHAR(30),
    data_nascita DATE,
    PRIMARY KEY  matricola
);
```

E’ possibile creare una tabella senza chiave principale e, aggiungerla successivamente:

```
CREATE TABLE studenti
(
    matricola    CHAR(6),
    cognome      VARCHAR(30),
    nome         VARCHAR(30),
    data_nascita DATE
);
```

In questo modo la tabella viene creata senza chiave. Se si inseriscono dei record non viene eseguito alcun controllo.

Per creare quindi la chiave primaria della tabella occorre eseguire il seguente statement:

```
ALTER TABLE studenti
      ADD PRIMARY KEY  (matricola);
```

Se lo statement **ALTER TABLE** viene eseguito quando nella tabella sono gia’ presenti dei record , il motore SQL controlla che nel campo matricola non siano presenti dei valori nulli o delle chiavi doppie. In caso contrario l’esecuzione del comando viene sospesa e viene segnalato l’errore.

La creazione di una chiave primaria si traduce, in pratica, nella creazione di un indice.

COME CREARE CHIAVI PRIMARIE FORMATE DA PIU' CAMPI

Se chiave primaria è formata da più campi occorre eseguire il seguente statement:

```
CREATE TABLE esami
(
     studente       CHAR(6),
     voto           NUMERIC(2),
     lode           CHAR(4),
     corso          CHAR(2),
     PRIMARY KEY    (studente, corso)
);
```

o in alternativa:

```
CREATE TABLE esami
(
     studente       CHAR(6),
     voto           NUMERIC(2),
     lode           CHAR(4),
     corso          CHAR(2)
);

ALTER TABLE studenti
       ADD PRIMARY KEY  (studente, corso);
```

ESEMPIO: Creazione di un Data Base con le chiavi primarie

STUDENTI

Matricola	Cognome	Nome	Data di nascita
200768	Verdi	Fabio	**12/02/1792**
937653	Rossi	Luca	10/10/1971
937867	Bruni	Mario	01/12/1971

CORSI

Codice	Materia	Docente
01	Analisi	Giani
03	Chimica	Melli
04	Chimica	Belli

ESAMI

Studente	Voto	Lode	Corso
200768	**36**		**05**
937653	28	**Lode**	01
937653	30	Lode	04
276545	25		01

```
CREATE TABLE studenti
(
     matricola      CHAR(6)         PRIMARY KEY,
     cognome        VARCHAR(20),
     nome           VARCHAR(20),
     data_nascita   DATE
);

CREATE TABLE corsi
(
     codice         CHAR(2)         PRIMARY KEY,
     materia        VARCHAR(20),
     docente        VARCHAR(20)
);

CREATE TABLE esami
(
     studente       CHAR(6),
     voto           NUMERIC(2),
     lode           CHAR(4),
     corso          CHAR(2),
     PRIMARY KEY    (studente, corso)
);
```

A questo punto è possibile inserire i dati :

```
INSERT INTO studenti VALUES (‘200768’,’Verdi’,’Fabio’,’12-FEB-1792’);
INSERT INTO studenti VALUES (‘937653’,’Rossi’,’Luca’, ’10-OCT-1971’);
INSERT INTO studenti VALUES (‘937867’,’Bruni’,’Mario’,’01-DEC-1971’);

INSERT INTO corsi    VALUES (‘01’,’Analisi’,’Giani’);
INSERT INTO corsi    VALUES (‘03’,’Chimica’,’Melli’);
INSERT INTO corsi    VALUES (‘04’,’Chimica’,’Belli’);

INSERT INTO esami    VALUES (‘200768’, 36,’’,    ’05’);
INSERT INTO esami    VALUES (‘937653’, 28,’lode’,’01’);
INSERT INTO esami    VALUES (‘937653’, 30,’lode’,’04’);
INSERT INTO esami    VALUES (‘276545’, 25,’’,    ’01’);
```

Questa istanza di Data Base è coerente???

COME CREARE DI UNA CHIAVE ALTERNATIVA

Si consideri la seguente tabella:

STUDENTI

Matricola	**Cognome**	**Nome**	**Codice fiscale**
200768	Verdi	Fabio	VRDFBB72T12G535G
345778	Merli	Enzo	***null***
937653	Rossi	Luca	**RSSLCC71R10H236G**
845633	Sarti	Ivan	*null*
937867	Bruni	Mario	**RSSLCC71R10H236G**

L'attributo **Matricola** rappresenta sempre la chiave principale e pertanto, avendolo dichiarato tale, ne viene garantita l'univocità. Non e' possibile quindi che due studenti diversi abbiano lo stesso numero di matricola.
Si osservi, però, che anche per l'attributo **Codice Fiscale** occorre garantirne l'univocità perchè, come e' noto, non e possibile assegnare a due persone diverse lo stesso codice fiscale.
Nel nostro esempio Rossi e Bruni hanno lo stesso codice fiscale e questo e' palesemente sbagliato. Si osservi inoltre che il codice fiscale di Merli e Sarti non e' stato ancora inserito e quindi, in entrambi i casi, assume il valore NULL.
Come e' già stato spiegato ogni tabella può avere, al massimo, una sola chiave primaria. Per garantire anche l'univocità del campo Codice Fiscale occorre usare la clausola **`UNIQUE`**.

```
CREATE TABLE studenti
(
     matricola      CHAR(6)         PRIMARY KEY,
     cognome        VARCHAR(20),
     nome           VARCHAR(20),
     codice_fiscale CHAR(16)        UNIQUE
);
```

In questo modo non sarà più possibile inserire due codici fiscali uguali. Si osservi, però, che , a differenza della clausola PRIMARY KEY, **è possibile inserire valori nulli** e pertanto è possibile avere più righe con valore nullo. In altre parole il controllo dell'univocità non viene effettuato quando si inseriscono dei valori nulli ma solo quando si inseriscono valori diversi da NULL.

L'assegnazione di una chiave alternativa si traduce, in pratica, nella creazione di un indice.

L’esempio che segue risulta pertanto coerente con la definizione della tabella:

STUDENTI

Matricola	**Cognome**	**Nome**	**Codice fiscale**
200768	Verdi	Fabio	VRDFBB72T12G535G
345778	Merli	Enzo	***null***
937653	Rossi	Luca	RSSLCC71R10H236G
845633	Sarti	Ivan	*null*
937867	Bruni	Mario	BRNMRR71T10J432H

ESEMPIO: Creazione di un Data Base con vincoli di tupla (riga)

STUDENTI

Matricola	Cognome	Nome	Data di nascita
200768	Verdi	Fabio	12/02/1972
937653	Rossi	Luca	10/10/1971
937867	Bruni	Mario	01/12/1971

CORSI

Codice	Materia	Docente
01	Analisi	Giani
03	Chimica	Melli
04	Chimica	Belli

ESAMI

Studente	Voto	Lode	Corso
200768	30		**05**
937653	28		01
937653	30	Lode	04
276545	***null***		01

```
CREATE TABLE studenti
(
     matricola      CHAR(6)         PRIMARY KEY,
     cognome        VARCHAR(20),
     nome           VARCHAR(20),
     data_nascita   DATE,
     CHECK          (data_nascita  > '01-JAN-1900')
);

CREATE TABLE corsi
(
     codice         CHAR(2)         PRIMARY KEY,
     materia        VARCHAR(20),
     docente        VARCHAR(20)
);

CREATE TABLE esami
(
     studente       CHAR(6),
     voto           NUMERIC(2),
     lode           CHAR(4),
     corso          CHAR(2),
     PRIMARY KEY    (studente, corso),
     CHECK          (voto BETWEEN 18 AND 30),
     CHECK          (NOT(lode='lode') OR (voto = 30))
);
```

A questo punto e' possibile inserire i dati :

```
INSERT INTO studenti VALUES ('200768','Verdi','Fabio','12-FEB-1972');
INSERT INTO studenti VALUES ('937653','Rossi','Luca', '10-OCT-1971');
INSERT INTO studenti VALUES ('937867','Bruni','Mario','01-DEC-1971');

INSERT INTO corsi    VALUES ('01','Analisi','Giani');
INSERT INTO corsi    VALUES ('03','Chimica','Melli');
INSERT INTO corsi    VALUES ('04','Chimica','Belli');

INSERT INTO esami    VALUES ('200768', 30,'',    '05');
INSERT INTO esami    VALUES ('937653', 28,'',    '01');
INSERT INTO esami    VALUES ('937653', 30,'lode','04');
INSERT INTO esami    VALUES ('276545', '','',    '01');
```

NOTA BENE : **Si osservi che i valori NULL soddisfano sempre il constraint *CHECK*. In altre parole il controllo della clausola *CHECK* NON viene attivato quando il valore dell'attributo da controllare e' NULL.**

Questa istanza di Data Base è coerente???

ESEMPIO: Creazione di un Data Base con controllo dei valori nulli

Si consideri la seguente istanza di Data Base:

STUDENTI

Matricola	**Cognome**	**Nome**	**Data di nascita**
200768	***null***	*null*	12/02/1972
937653	Rossi	Luca	10/10/1971
937867	Bruni	Mario	***null***

CORSI

Codice	**Materia**	**Docente**
01	***null***	Giani
03	Chimica	Melli
04	Chimica	***null***

ESAMI

Studente	**Voto**	**Lode**	**Corso**
200768	***null***		**05**
937653	28		01
937653	30	Lode	04
276545	25		01

Osserviamo:

- Gli attributi Matricola (*Studenti*), Codice (*Corsi*), Studente e Corso (*Esami*) **NON** possono contenere valori nulli in quanto **sono stati definiti come chiavi primarie**.
- Tutti gli altri campi possono contenere il valore NULL.

Dal punto di vista della gestione del Data Base possiamo fare le seguenti osservazioni:

Tabella STUDENTI:

- Gli attributi **Cognome** e **Nome** devono sempre contenere un valore in quanto non ha senso attribuire un numero di matricola ad una persona di cui non si conosce neanche il nome.
- L'attributo **Data di Nascita** può presentare il valore NULL in quanto si può ipotizzare che questa informazione possa essere aggiunta in un secondo tempo.

Tabella CORSI:
- L'attributo **Materia** deve essere sempre definito e quindi non e' ammesso il valore NULL.
- L'attributo **Docente** può invece assumere il valore NULL a significare che la cattedra e' al momento vacante.

Tabella ESAMI:
- L'attributo **Voto** deve sempre essere presente in quanto non ha senso registrare un esame di cui non si conosce il voto.
- L'attributo **Lode** può di fatto presentare soltanto due valori: "**lode**" (*ma solo se il voto e' 30*) e NULL quando la lode non viene assegnata.

Per creare un Data Base che contenga anche queste regole di salvaguardia dell'integrità occorre eseguire i seguenti statement contenenti la clausola
NOT NULL
che previene l'inserimento di valori nulli indesiderati:

```
CREATE TABLE studenti
(
     matricola      CHAR(6)         PRIMARY KEY,
     cognome        VARCHAR(20)     NOT NULL,
     nome           VARCHAR(20)     NOT NULL,
     data_nascita   DATE,
     CHECK          (data_nascita  > '01-JAN-1900')
);

CREATE TABLE corsi
(
     codice         CHAR(2)         PRIMARY KEY,
     materia        VARCHAR(20)     NOT NULL,
     docente        VARCHAR(20)
);

CREATE TABLE esami
(
     studente       CHAR(6)
     voto           NUMERIC(2)      NOT NULL,
     lode           CHAR(4),
     corso          CHAR(2)
     PRIMARY KEY    (studente, corso),
     CHECK          (voto BETWEEN 18 AND 30),
     CHECK          (NOT(lode='lode') OR (voto = 30))
);
```

A questo punto è possibile inserire i dati :

```
INSERT INTO studenti VALUES ('200768','Verdi','Fabio','12-FEB-1972');
INSERT INTO studenti VALUES ('937653','Rossi','Luca', '10-OCT-1971');
INSERT INTO studenti VALUES ('937867','Bruni','Mario','')           );

INSERT INTO corsi    VALUES ('01','Analisi','Giani');
INSERT INTO corsi    VALUES ('03','Chimica','Melli');
INSERT INTO corsi    VALUES ('04','Chimica','')     );

INSERT INTO esami    VALUES ('200768', 30,'',    '05');
INSERT INTO esami    VALUES ('937653', 28,'',    '01');
INSERT INTO esami    VALUES ('937653', 30,'lode','04');
INSERT INTO esami    VALUES ('937854', 25,'',    '01');
```

da cui si ricava la seguente istanza:

STUDENTI

Matricola	**Cognome**	**Nome**	**Data di nascita**
200768	Verdi	Fabio	12/02/1972
937653	Rossi	Luca	10/10/1971
937867	Bruni	Mario	***null***

CORSI

Codice	**Materia**	**Docente**
01	Analisi	Giani
03	Chimica	Melli
04	Chimica	***null***

ESAMI

Studente	**Voto**	**Lode**	**Corso**
200768	30		**05**
937653	28		01
937653	30	Lode	04
276545	25		01

Questa istanza di Data Base è coerente???

VINCOLI DI INTEGRITA' REFERENZIALE

Nell'istanza precedente sono evidenti due anomalie presenti sulla tabella ESAMI:

- **Lo studente con matricola '276545' non e' presente sulla tabella STUDENTI** e quindi non si capisce chi abbia effettivamente sostenuto l'esame.
- **Lo studente Verdi (Matr. 200768) ha sostenuto l'esame con codice '05'** che pero' non e presente sulla tabella corsi. Non si capisce quindi che esame abbia sostenuto.

In realtà i codici potrebbero essere sbagliati per una svista dell'operatore che ha inserito i dati.
Per prevenire questo tipo di errore occorre fissare le seguenti regole di integrità che devono essere verificate ogni qualvolta si inseriscono i dati di un nuovo esame:
- il valore dell'attributo **Studente** deve essere presente nel campo **Matricola** della tabella STUDENTI.
- Il valore dell'attributo **Corso** deve essere presente nel campo **Codice** della tabella CORSI.

Si osservi che gli attributi **Matricola** e **Codice** sono **chiavi primarie** rispettivamente delle tabelle STUDENTI e CORSI.

Gli attributi **Studente** e **Corso** della tabella ESAMI sono **FOREIGN KEY**.
Gli attributi **Matricola** della tabella STUDENTI e **Codice** della tabella CORSI sono **REFERENCED KEY** rispetto ai campi **Studente** e **Corso** della tabella ESAMI.

La clausola

```
REFERENCES nome_tabella (nome_chiave)
```

Consente di definire questo tipo di controllo. Si veda l'esempio seguente.

ESEMPIO: Creazione di un Data Base con vincoli di integrita' referenziale

STUDENTI

Matricola	Cognome	Nome	Data di nascita
200768	Verdi	Fabio	12/02/1972
937653	Rossi	Luca	10/10/1971
937867	Bruni	Mario	01/12/1971

CORSI

Codice	Materia	Docente
01	Analisi	Giani
03	Chimica	Melli
04	Chimica	Belli

ESAMI

Studente	Voto	Lode	Corso
200768	30		03
937653	28		01
937653	30	Lode	04
200768	25		01

```
CREATE TABLE studenti
(
     matricola      CHAR(6)         PRIMARY KEY,
     cognome        VARCHAR(20)     NOT NULL,
     nome           VARCHAR(20)     NOT NULL,
     data_nascita   DATE,
     CHECK          (data_nascita  > '01-JAN-1900')
);

CREATE TABLE corsi
(
     codice         CHAR(2)         PRIMARY KEY,
     materia        VARCHAR(20)     NOT NULL,
     docente        VARCHAR(20)
);

CREATE TABLE esami
(
     studente       CHAR(6)         REFERENCES studenti (matricola),
     voto           NUMERIC(2)      NOT NULL,
     lode           CHAR(4),
     corso          CHAR(2)         REFERENCES corsi (codice),
     PRIMARY KEY    (studente, corso),
     CHECK          (voto BETWEEN 18 AND 30),
     CHECK          (NOT(lode='lode') OR (voto = 30))
);
```

A questo punto è possibile inserire i dati :

```
INSERT INTO studenti VALUES (‘200768’,’Verdi’,’Fabio’,’12-FEB-1972’);
INSERT INTO studenti VALUES (‘937653’,’Rossi’,’Luca’, ’10-OCT-1971’);
INSERT INTO studenti VALUES (‘937867’,’Bruni’,’Mario’,’01-DEC-1971’);

INSERT INTO corsi    VALUES (‘01’,’Analisi’,’Giani’);
INSERT INTO corsi    VALUES (‘03’,’Chimica’,’Melli’);
INSERT INTO corsi    VALUES (‘04’,’Chimica’,’Belli’);

INSERT INTO esami    VALUES (‘200768’, 30,’’,    ’03’);
INSERT INTO esami    VALUES (‘937653’, 28,’’,    ’01’);
INSERT INTO esami    VALUES (‘937653’, 30,’lode’,’04’);
INSERT INTO esami    VALUES (‘200768’, 25,’’,    ’01’);
```

NOTA BENE Le tabelle STUDENTI - ESAMI sono legate tra di loro da una **relazione 1/N** (*uno a N*) in quanto ad ogni singolo studente possono corrispondere più esami.
Lo stesso si dica per le tabelle CORSI – ESAMI in quanto ad ogni singolo corso possono corrispondere più esami.

Questa istanza di Data Base è coerente??? Finalmente Sì!