

17. Definizione delle tabelle in SQL

database

definizione delle tabelle in SQL

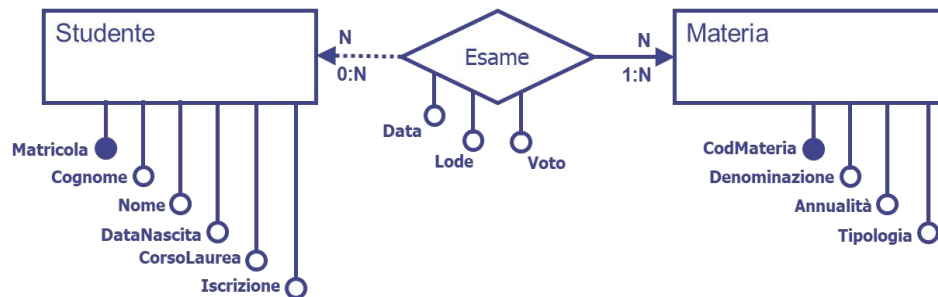
Il modello relazionale ci consente la creazione delle tabelle che costituiscono il nostro database.

Cmq, prima di iniziare la costruzione effettiva delle tabelle, è necessario definire per ogni attributo il **tipo di dato** (dominio) a cui esso fa riferimento: stringa di caratteri, valore numerico (intero, decimale), data, orario, ecc. con eventuale indicazione del numero di caratteri o di cifre impiegate per la rappresentazione.

È necessario anche fare riferimento ai **vincoli di integrità** sull'ammissibilità dei valori da assegnare ai campi sia al livello di vincoli intrarelazionali (vincoli di tupla e di integrità sull'entità) che al livello di integrità referenziale.

definizione delle tabelle in SQL

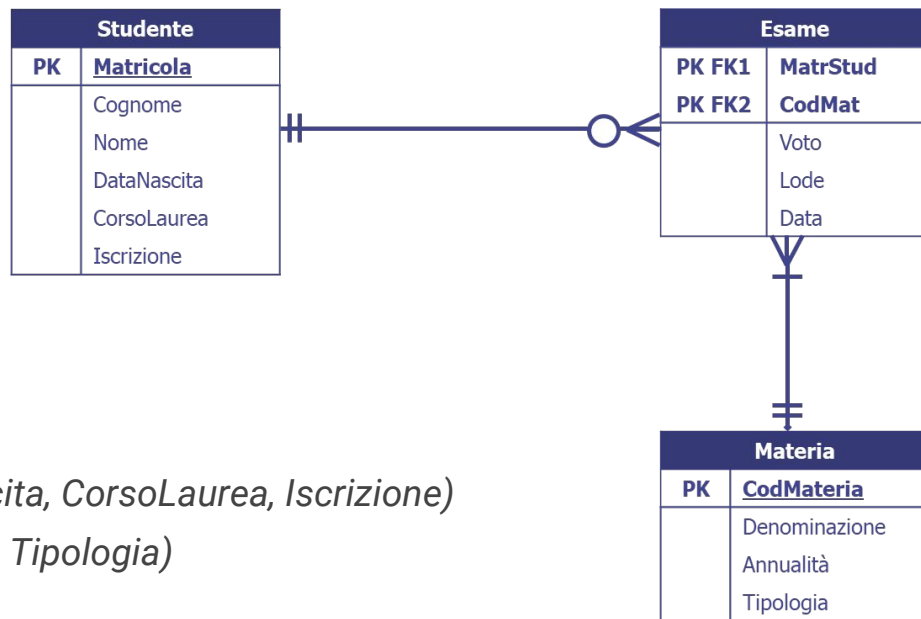
Supponiamo di dover realizzare un semplice sistema per la registrazione degli esami sostenuti dagli studenti di un ateneo universitario; considerato che ogni studente può essere esaminato in diverse materie e ciascuna materia può essere oggetto d'esame da parte di più studenti, possiamo realizzare il diagramma E/R seguente:



definizione delle tabelle in SQL

In figura, il diagramma UML delle tabelle ed il modello relazionale corrispondente:

NB: Stiamo supponendo che, come è ovvio, ogni studente possa sostenere lo stesso esame una volta soltanto (se non viene bocciato) quindi è ammessa un'unica registrazione di ciascun esame superato sul libretto di ciascuno studente.



Studente(Matricola, Cognome, Nome, DataNascita, CorsoLaurea, Iscrizione)

Materia(CodMateria, Denominazione, Annualità, Tipologia)

Esame(MatrStud, CodMat, Voto, Lode, Data)

definizione delle tabelle in SQL

Realizziamo le tabelle con le specifiche riportate:

*Tabella **Esame**, contiene i dati degli esami sostenuti dagli studenti*

Campo		Tipo	Descrizione
MatrStud	PK FK	Testo (10 caratteri)	Matricola dello studente che sostiene l'esame
CodMat	PK FK	Testo (5 caratteri)	Codice della materia oggetto dell'esame
Voto		Intero	Deve essere compreso tra 18 e 30
Lode		Booleano	1 (<i>true</i>) solo se 30 con lode, 0 (<i>false</i>) altrimenti
Data		Testo (32 caratteri)	Data dell'esame

*Tabella **Studente**, contiene l'anagrafica degli studenti*

Campo		Tipo	Descrizione
Matricola	PK	Testo (10 caratteri)	Numero di matricola
Cognome		Testo (24 caratteri)	
Nome		Testo (24 caratteri)	
DataNascita		Data	
CorsoLaurea		Testo (32 caratteri)	Denominazione del corso di laurea
Iscrizione		Decimale	4 cifre intere e due decimali

*Tabella **Materia**, informazioni sulla materia d'esame*

Campo		Tipo	Descrizione
CodMateria	PK	Testo (5 caratteri)	Identifica la materia d'esame
Denominazione		Testo (32 caratteri)	Nome della materia
Annualità		Enumerativo	Valori: 1T, 2T, 3T, 1M, 2M
Tipologia		Enumerativo	Valori: 'Semestrale', 'Annuale'

definizione delle tabelle in SQL

Il linguaggio SQL (acronimo di *Structured Query Language*) nasce e viene utilizzato fondamentalmente come query language.

Cmq SQL incorpora funzionalità di DDL (*Data Definition Language*) per la definizione di uno schema di database e dei corrispondenti schemi di relazione, oltre a funzionalità di DML (*Data Manipulation Language*) per le operazioni di inserimento, modifica e cancellazione dei dati memorizzati nelle varie tabelle.

Ad es. per creare un nuovo database possiamo usare il comando:

CREATE DATABASE <NomeDatabase> oppure: **CREATE SCHEMA** <NomeDatabase>

Il nostro database si chiamerà RegistroEsami, quindi:

```
CREATE DATABASE RegistroEsami
```

definizione delle tabelle in SQL

Una volta creato il nuovo database, possiamo costruire le tabelle di cui si compone, rispettando le specifiche del nostro modello relazionale.

Il comando principale che SQL mette a disposizione per la costruzione di una nuova tabella, definendone la struttura, gli indici ed i principali vincoli di integrità, è:

```
CREATE TABLE <NomeTabella> (  
    <NomeAttributo> <Dominio> [ValoreDiDefault] [Vincoli]  
    {, <NomeAttributo> <Dominio> [ValoreDiDefault] [Vincoli]}  
    AltriVincoli  
)
```

Dove sostanzialmente vengono indicati il nome della tabella e l'elenco di tutti i suoi attributi con la specifica del dominio di appartenenza.

definizione delle tabelle in SQL

Analizziamo ad es. il seguente comando:

```
CREATE TABLE Studente(  
  Matricola INT PRIMARY KEY AUTO_INCREMENT,  
  Cognome CHAR(24) NOT NULL,  
  Nome CHAR(24) NOT NULL,  
  DataNascita DATE DEFAULT NULL,  
  CorsoLaurea VARCHAR(32) DEFAULT '',  
  Iscrizione DECIMAL(6,2) DEFAULT 0.00  
)
```

Nome della tabella

Definizione della chiave primaria: il campo `Matricola` di tipo `INT` (numero intero) è chiave primaria.

La parola chiave `AUTO_INCREMENT` indica che il campo (di tipo intero) in questione verrà automaticamente incrementato per ogni nuovo record.

I campi `Cognome` e `Nome`, di tipo `CHAR(24)` (stringa di esattamente 24 caratteri) non ammettono valore nullo.

Elenco dei campi

Studente(Matricola, Cognome, Nome, DataNascita, CorsoLaurea, Iscrizione)

definizione delle tabelle in SQL

Analizziamo ad es. il seguente comando:

```
CREATE TABLE Studiante(
```

Nome della tabella

```
    Matricola INT PRIMARY KEY AUTO_INCREMENT,
```

```
    Cognome CHAR(24) NOT NULL,
```

```
    Nome CHAR(24) NOT NULL,
```

```
    DataNascita DATE DEFAULT NULL,
```

```
    CorsoLaurea VARCHAR(32) DEFAULT '',
```

```
    Iscrizione DECIMAL(6,2) DEFAULT 0.00
```

```
)
```

Elenco dei campi

Il campo DataNascita è di tipo DATE, per default assume valore nullo.

Il campo CorsoLaurea VARCHAR(32) (cioè varying character, stringa a lunghezza variabile al massimo di 32 caratteri) se non specificato, assume la stringa vuota come valore di default.

Il campo Iscrizione è di tipo DECIMAL con 4 cifre intere e 2 decimali; per default assume valore 0.00.

Studiante(Matricola, Cognome, Nome, DataNascita, CorsoLaurea, Iscrizione)

Questo comando crea una nuova tabella *Studiante* con sei campi, imposta la chiave primaria e stabilisce alcuni vincoli *intrarelazionali* (cioè all'interno della stessa tabella).

definizione delle tabelle in SQL

La parola chiave `DEFAULT` permette di stabilire il valore di default per un campo specifico.

Il valore di default per un campo è quel valore che deve assumere il campo quando viene inserita una riga nella tabella, senza che sia specificato un valore per l'attributo stesso.

Quando il valore di default non è specificato, si assume come default il valore nullo.

```
...  
DataNascita DATE DEFAULT NULL,  
CorsoLaurea VARCHAR(32) DEFAULT '',  
Iscrizione DECIMAL(6,2) DEFAULT 0.00  
...
```

definizione delle tabelle in SQL

Il comando `CREATE TABLE` può definire anche alcuni *vincoli intrarelazionali* tra i quali:

- **PRIMARY KEY**: associato ad un solo campo, imposta una chiave primaria su quel campo; tale vincolo implica anche il vincolo **NOT NULL**, che può essere omesso.
- **UNIQUE**: impone che i valori dell'attributo in questione siano diversi per ogni riga della tabella (costituendo di fatto una superchiave).
- **NOT NULL**: impone che i valori dell'attributo siano diversi dal valore nullo.

```
CREATE TABLE Materia(  
    CodMateria CHAR(5) PRIMARY KEY,  
    Denominazione VARCHAR(32),  
    Annualità ENUM('1T','2T','3T','1M','2M'),  
    Tipologia ENUM('Sem','Ann')  
)
```

Chiave primaria impostata sul campo `CodMateria`.

Il tipo **ENUM** elenca un insieme di opzioni alternative.

Materia(CodMateria, Denominazione, Annualità, Tipologia)

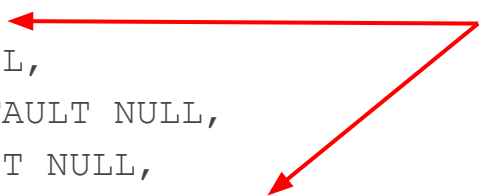
definizione delle tabelle in SQL

La definizione della chiave primaria può avvenire anche dopo la definizione di tutti i campi, se questa è costituita da più campi, utilizzando la parola chiave `PRIMARY KEY` seguita dall'elenco dei campi che ne fanno parte, tra parentesi tonde.

Esame(**MatrStud**, **CodMat**, Voto, Lode, Data)

```
CREATE TABLE Esame (  
    MatrStud INT,  
    CodMat CHAR(5),  
    Voto INT NOT NULL,  
    Lode BOOLEAN DEFAULT NULL,  
    Data DATE DEFAULT NULL,  
    PRIMARY KEY(MatrStud, CodMat)  
)
```

La chiave primaria viene impostata su due campi:
MatrStud e CodMat.

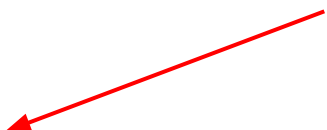


definizione delle tabelle in SQL


Oltre alla chiave primaria, che ponendo a NOT NULL i campi che ne fanno parte, impone anche un controllo di *integrità sull'entità*, possiamo anche impostare vincoli di tupla sui valori assegnati ai campi di ciascuna riga.

```
CREATE TABLE Esame (  
    MatrStud INT,  
    CodMat CHAR(5),  
    Voto INT NOT NULL CHECK (Voto>=18 AND Voto<=30),  
    Lode BOOLEAN DEFAULT NULL,  
    Data DATE DEFAULT NULL,  
    PRIMARY KEY(MatrStud, CodMat),  
    CONSTRAINT ChkLode CHECK ((Voto=30 AND Lode=1)  
        OR (Voto>=18 AND Voto<=30 AND Lode=0))  
    )
```

Il voto dell'esame deve essere compreso tra 18 e 30



Per dare un nome ad un vincolo (ad es. ChkLode) o definire un vincolo su più colonne, si può usare la parola chiave CONSTRAINT.



definizione delle tabelle in SQL

Per creare un indice secondario invece bisogna utilizzare il comando:

```
CREATE [UNIQUE] INDEX <NomeIndice>  
ON <NomeTabella> (<NomeAttributo1>, <NomeAttributo2>, ...)
```

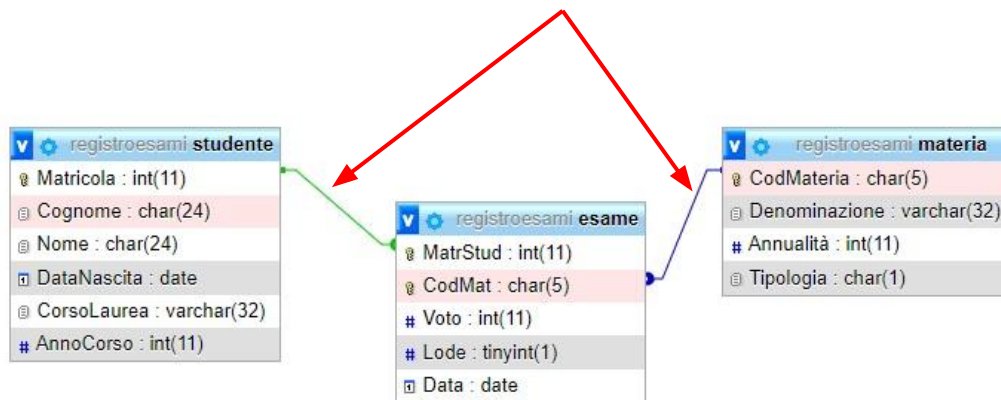
Un indice secondario può servire per velocizzare le operazioni di ricerca sui valori dei campi che ne fanno parte e cmq sono indispensabili nella definizione delle relazioni, in quanto bisogna impostare un indice secondario sui campi che costituiscono le chiavi esterne e che devono essere legate alle corrispondenti chiavi primarie.

definizione delle tabelle in SQL

Ad es. per inserire una relazione tra le tabelle *Studente* ed *Esame* sui campi *Matricola* (chiave primaria per *Studente*) e *MatrStud* (chiave esterna per *Esame*), bisogna prima impostare un indice secondario su *MatrStud*.

Stessa cosa per il campo *CodMat*, in modo da inserire una relazione anche tra le tabelle *Esame* e *Materia* sui campi *CodMat* e *CodMateria*.

Due relazioni tra *Studente* ed *Esame* sui campi *Matricola* e *MatrStud* e tra *Esame* e *Materia* dai campi: *CodMat* e *CodMateria*.



definizione delle tabelle in SQL

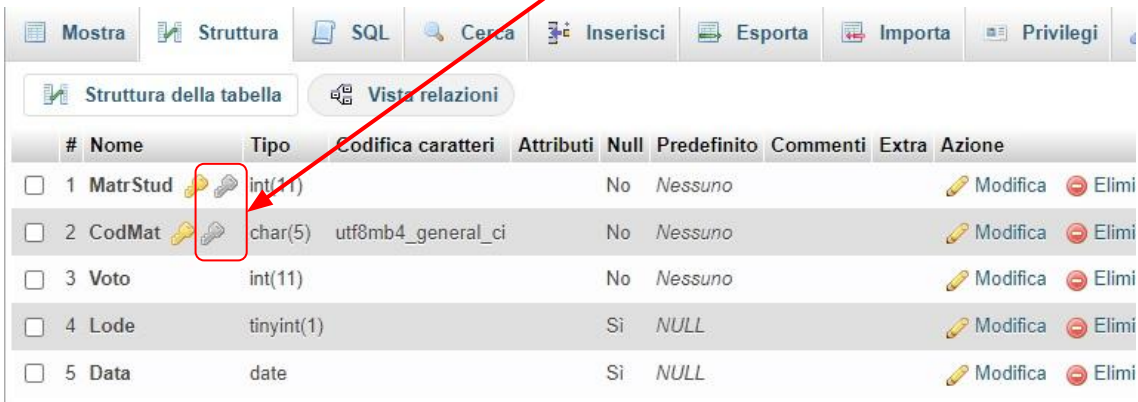
Possiamo impostare due indici secondari sui campi *MatrStud* e *CodMat* della tabella *Esame* tramite le seguenti istruzioni:

```
CREATE INDEX FK_Studente ON Esame (MatrStud);
```

```
CREATE INDEX FK_Esame ON Esame (CodMat);
```

NB: In questo caso specifico non sarebbe necessario, in quanti i campi chiave esterne fanno già parte della chiave primaria.

Indici secondari



#	Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito	Commenti	Extra	Azione
<input type="checkbox"/>	1 MatrStud	int(11)			No	Nessuno			Modifica Elimina
<input type="checkbox"/>	2 CodMat	char(5)	utf8mb4_general_ci		No	Nessuno			Modifica Elimina
<input type="checkbox"/>	3 Voto	int(11)			No	Nessuno			Modifica Elimina
<input type="checkbox"/>	4 Lode	tinyint(1)			Sì	NULL			Modifica Elimina
<input type="checkbox"/>	5 Data	date			Sì	NULL			Modifica Elimina

definizione delle tabelle in SQL

Per quanto riguarda i **vincoli interrelazionali** (tra più tabelle), il più importante è senza dubbio il vincolo di **integrità referenziale**, l'impostazione del quale prevede anche la definizione di una chiave esterna (*foreign key*).

In una relazione *uno-a-molti* tra due tabelle, il vincolo di integrità referenziale crea un legame tra i valori di uno o più campi (in genere la chiave primaria) della tabella *dominio* (lato *uno* della relazione), ed i valori di altrettanti campi (in genere la chiave esterna) dell'altra tabella (*codominio*, dal lato *molti*).

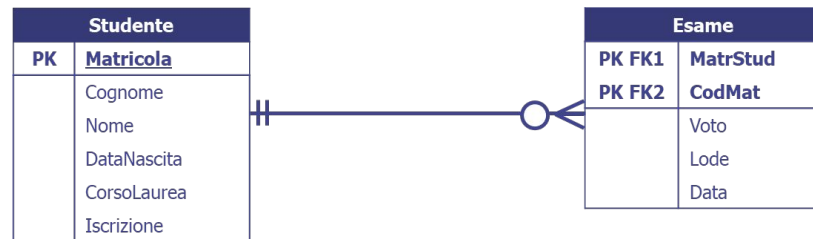
Studente(Matricola, Cognome, Nome, DataNascita, CorsoLaurea, Iscrizione)

Esame(MatrStud, CodMat, Voto, Lode, Data)

Materia(CodMateria, Denominazione, Annualità, Tipologia)

definizione delle tabelle in SQL

Il vincolo di integrità referenziale impone che, data la relazione *uno-a-molti* tra *Studente* ed *Esame*, per ogni riga della tabella codominio (*Esame*), il valore del campo *MatrStud* (quando diverso da `NULL`) sia presente nelle righe della tabella dominio (*Studente*) tra i valori del campo corrispondente (*Matricola*).



L'unico requisito che la sintassi impone è che il campo cui si fa riferimento nel dominio sia soggetto ad un vincolo `UNIQUE`, quindi sia un identificatore (superchiave) per il dominio (normalmente si sceglie infatti la chiave primaria).

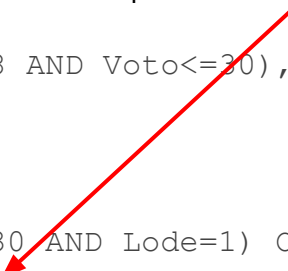
definizione delle tabelle in SQL

Per impostare il vincolo di integrità referenziale, si può usare il costrutto `FOREIGN KEY` che indica la chiave esterna, insieme con la parola chiave `REFERENCES` con il quale si specificano la tabella dominio ed il campo corrispondente da legare.

Leghiamo il campo *Matricola* chiave primaria per la tabella *Studente*, con il campo *MatrStud* chiave esterna per la tabella *Esame*.

Stessa cosa per i campi *CodMateria* chiave primaria per *Materia* e *CodMat* chiave esterna per *Esame*.

```
CREATE TABLE Esame(  
    MatrStud INT,  
    CodMat CHAR(5),  
    Voto INT NOT NULL CHECK (Voto>=18 AND Voto<=30),  
    Lode BOOLEAN DEFAULT NULL,  
    Data DATE DEFAULT NULL,  
    PRIMARY KEY(MatrStud, CodMat),  
    CONSTRAINT ChkLode CHECK ((Voto=30 AND Lode=1) OR (Voto>=18 AND Voto<=30 AND Lode=0)),  
    FOREIGN KEY(MatrStud) REFERENCES Studente(Matricola),  
    FOREIGN KEY(CodMat) REFERENCES Materia(CodMateria)  
)
```



NB: La definizione delle chiavi esterne impone automaticamente per esse (quando necessario) anche un indice secondario.

definizione delle tabelle in SQL

Abbiamo un sistema che deve registrare i titoli accademici (ognuno con un punteggio) presentati dagli studenti di diverse università (identificati da matricola e università) per stilare una graduatoria per l'assegnazione di una borsa di studio; ovviamente ogni studente può presentare più titoli.

Riportiamo per semplicità solo il diagramma UML del progetto:

La chiave primaria della tabella *Studente* è formata da due campi *Matricola* e *CodUniversità*, per cui anche la chiave esterna è formata da due campi.



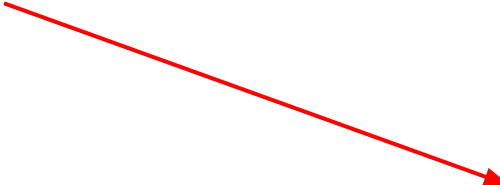
definizione delle tabelle in SQL

Possiamo creare le due tabelle con i seguenti comandi SQL:

Da notare il vincolo di integrità referenziale impostato contemporaneamente sui due campi `MatrStudiante` e `CodUniversità`.

```
CREATE TABLE Studente(  
    Matricola INT,  
    CodUniversità CHAR(5),  
    Nominativo VARCHAR(36) NOT NULL,  
    DataNascita DATE DEFAULT NULL,  
    CorsoLaurea VARCHAR(32) DEFAULT '',  
    AnnoCorso INT DEFAULT 0,  
    PRIMARY KEY(Matricola, CodUniversità)  
);
```

```
CREATE TABLE TitoloAccademico(  
    NumRegistrazione INT PRIMARY KEY AUTO_INCREMENT,  
    Descrizione VARCHAR(36) NOT NULL,  
    Data DATE NOT NULL,  
    Punteggio INT DEFAULT 0,  
    MatrStudiante INT NOT NULL,  
    CodUniversità CHAR(5) NOT NULL,  
    FOREIGN KEY(MatrStudiante, CodUniversità)  
    REFERENCES Studente(Matricola, CodUniversità)  
);
```



definizione delle tabelle in SQL

Operazioni di modifica del valore della chiave primaria nella tabella dominio (o cmq nel campo riferito) o cancellazioni di righe, possono portare a violazioni del vincolo di integrità referenziale.

Cosa succede ad es. agli esami registrati relativamente ad uno specifico studente se modifichiamo la sua matricola o se addirittura eliminiamo del tutto lo studente?

In fase di dichiarazione dell'integrità referenziale, la politica di reazione nella tabella codominio può essere controllata dalle seguenti parole chiave: **ON UPDATE/DELETE**

CASCADE: modifica il valore della chiave esterna (o elimina la riga) corrispondente;

SET NULL: la chiave esterna viene impostata a `null`;

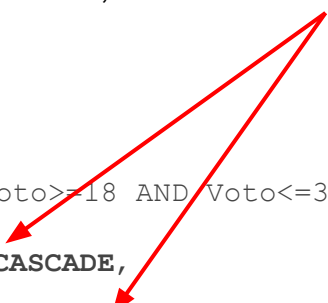
NO ACTION: la modifica/cancellazione non viene consentita.

definizione delle tabelle in SQL

Per non consentire modifiche alle matricole degli studenti se questi sono collegati ad esami registrati e fare in modo che, in caso di eliminazione di uno studente, anche tutti gli esami registrati a suo nome vengano eliminati, possiamo costruire la tabella *Esame* come segue:

```
CREATE TABLE Esame (  
    MatrStud INT,  
    CodMat CHAR(5),  
    Voto INT NOT NULL CHECK (Voto>=18 AND Voto<=30),  
    Lode BOOLEAN DEFAULT NULL,  
    Data DATE DEFAULT NULL,  
    PRIMARY KEY(MatrStud, CodMat),  
    CONSTRAINT ChkLode CHECK ((Voto=30 AND Lode=1) OR (Voto>=18 AND Voto<=30 AND Lode=0)),  
    FOREIGN KEY(MatrStud) REFERENCES Studente(Matricola)  
        ON UPDATE NO ACTION ON DELETE CASCADE,  
    FOREIGN KEY(CodMat) REFERENCES Materia(CodMateria)  
        ON UPDATE NO ACTION ON DELETE CASCADE  
)
```

In caso di aggiornamento del valore della chiave primaria, non consentire nessuna azione; nel caso di eliminazione della riga corrispondente, elimina in cascata tutte le righe collegate.



definizione delle tabelle in SQL

Vediamo i tipi di dato principali; tipi *numerici*:

Tipo di dato	Descrizione
TINYINT	Un byte, può contenere 256 valori, da -128 a +127, oppure da 0 a 255 se UNSIGNED.
SMALLINT	2 byte, può contenere 65536 valori da -32768 a 32767, oppure da 0 a 65535 se UNSIGNED.
MEDIUMINT	3 byte, può contenere 16.777.216 valori da -8388608 a 8388607, oppure da 0 a 16777215 se UNSIGNED.
INT	4 byte, può contenere oltre 4 miliardi di valori da -2147483648 a 2147483647, oppure da 0 a 4294967295 se UNSIGNED.
BIGINT	8 byte, può contenere valori da -9223372036854775808 a 9223372036854775807 oppure da 0 a 18446744073709551615 se UNSIGNED.
FLOAT/DOUBLE	È rappresentato in virgola mobile a «precisione singola»/«precisione doppia».
DECIMAL (p, s)	Numero decimale con esattamente p cifre di cui s decimali, ad es. decimal (5, 2) è un numero con 3 cifre intere e 2 decimali.

definizione delle tabelle in SQL

Vediamo i tipi di dato principali; tipi *date* e *orari*:

Tipo di dato	Descrizione
DATE	Rappresenta date comprese tra 1000-01-01 (1 gennaio 1000) e 9999-12-31 (31 dicembre 9999).
DATETIME	Rappresenta una data e un'ora, con lo stesso range visto per <code>DATE</code> , nel formato AAAA-MM-GG HH:MM:SS,
TIME	Rappresenta un valore di orario (ore, minuti e secondi) che va da -838:59:59 a 838:59:59.
YEAR [(2 4)]	Rappresenta, su quattro cifre, un anno compreso tra 1901 e 2155, oppure 0000, su due cifre invece i valori vanno da 70 (1970) a 69 (2069).
TIMESTAMP [(M)]	Può essere usato per memorizzare i valori corrispondenti al timestamp Unix, corrispondente al numero di secondi trascorsi dalla mezzanotte del 1 gennaio 1970.

NB: I valori relativi al tempo possono essere inseriti sia come stringhe sia come numeri. MySQL/MariaDB consente di utilizzare, nel caso delle stringhe, diversi tipi di caratteri come separatori; l'importante però è che l'ordine dei valori sia sempre anno-mese-giorno-ore-minuti-secondi.

definizione delle tabelle in SQL

Vediamo i tipi di dato principali; tipi *stringhe di caratteri*:

Tipo di dato	Descrizione
CHAR (M)	È una stringa di lunghezza fissa M completata con spazi a destra al momento della memorizzazione che sono eliminati in fase di lettura; la lunghezza va da 0 a 255 caratteri.
VARCHAR (M)	È una stringa a lunghezza variabile con lunghezza massima dichiarabile di 65535 caratteri.
TINYTEXT TEXT [(M)] MEDIUMTEXT LONGTEXT	Memorizzano campi di testo: la lunghezza massima è 255 caratteri per TINYTEXT , 65 535 per TEXT , 16 777 215 per MEDIUMTEXT , 4 294 967 296 per LONGTEXT .
ENUM ('V1 ' , 'V2 ' . . .)	Oggetto stringa che può assumere un solo valore scelto da una lista di valori predefiniti.
BLOB	Sono <i>Binary Long Objects</i> (TINY , MEDIUM , LONG) gestiti come stringhe binarie (ad es. per memorizzare un'immagine).

definizione delle tabelle in SQL

NB: La funzione `DATE_FORMAT()` può risolvere il problema della rappresentazione di date e orari in un formato facilmente interpretabile dall'utente.

La funzione riceve due argomenti: il primo è la data da formattare, il secondo una stringa composta da alcuni caratteri speciali, preceduti dal simbolo «%», che indicano come tale data debba essere formattata, secondo la tabella mostrata.

```
SELECT Nominativo,  
       DATE_FORMAT(DataNascita, '%d %M %Y') AS 'Nato il'  
FROM Dipendente  
WHERE Stipendio > 30000
```

<div>← T →</div>						Nominativo	Nato il	
<input type="checkbox"/>		Modifica		Copia		Elimina	Giada Perri	03 April 2002
<input type="checkbox"/>		Modifica		Copia		Elimina	Pinco Pallino	08 March 1993
<input type="checkbox"/>		Modifica		Copia		Elimina	Ornella Gemelli	04 February 2001

Carattere	Descrizione
%d	giorno del mese numerico 01...31
%M	nome del mese January...December
%m	mese numerico 01...12
%H	ora 00...23
%i	minuti 00...59
%s	secondi 00...59
%Y	anno di quattro cifre
%y	anno di due cifre

definizione delle tabelle in SQL

Alcune funzioni utili su date e orari sono presentate nella tabella seguente:

Funzione	Descrizione
NOW () CURRENT_TIMESTAMP () SYSDATE ()	Forniscono la data e l'ora corrente nel formato AAAA-MM-GG HH:MM:SS o AAAAMMGGHHMMSS a seconda che si tratti di un contesto stringa o numerico.
CURRDATE () CURRENT_DATE ()	Forniscono la data corrente nel formato AAAA-MM-GG o AAAAMMGG a seconda che si tratti di un contesto stringa o numerico.
CURRTIME () CURRENT_TIME ()	Forniscono l'ora corrente nel formato HH:MM:SS o HHMMSS a seconda che si tratti di un contesto stringa o numerico.
UNIX_TIMESTAMP ()	Se non viene specificato alcun argomento <i>UNIX_TIMESTAMP()</i> fornisce lo <i>Unix timestamp</i> corrente, se invece si specifica un valore di tipo DATE , DATETIME , TIME-STAMP fornisce lo <i>Unix timestamp</i> corrispondente alla particolare data.

definizione delle tabelle in SQL

Alcune funzioni utili su date e orari sono presentate nella tabella seguente:

Funzione	Descrizione
YEAR () MONTH () DAY () WEEK ()	Estraggono da un valore o campo data rispettivamente l'anno, il mese, il giorno, la settimana in formato numerico.
HOURL () MINUTE () SECOND ()	Estraggono da un valore orario rispettivamente l'ora, i minuti, i secondi in formato numerico.
ADDDATE ()	Aggiunge un intervallo temporale espresso in giorni a una data.
SUBDATE ()	Sottrae un intervallo temporale espresso in giorni a una data.
DATEDIFF ()	Effettua la differenza tra due date espressa in giorni.