2. I COMANDI DDL DEL LINGUAGGIO SQL: CREATE, ALTER, DROP

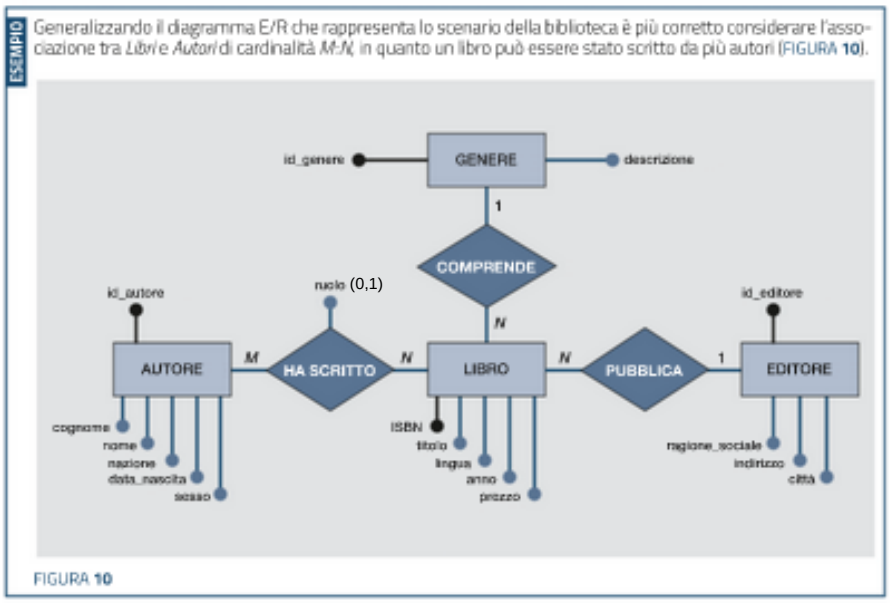
I comandi DDL (Data Definition Language) di SQL sono i comandi che consentono di "creare" lo schema della base di dati: il database, le tabelle, i campi, vincoli di integrità, gli indici. Gli indici sono assegnati a determinati campi per rendere più veloci le ricerche di dati che coinvolgono tali campi. I campi per i quali sono definiti degli indici vengono detti indicizzati. Metaforicamente l'indice è proprio come l'indice di un libro. Così come l'indice del libro permette di ricercare velocemente un argomento all'interno di un libro, così un indice permette al motore del DBMS di ricercare più velocemente un dato.

Per comprendere i comandi DDL utilizziamo il database della biblioteca.

ESEMPIO.

Applicando le regole di derivazione, generiamo le relazioni dal diagramma E/R dell'esempio "libreria".

Partiamo da diagramma E/R:



Lo **schema logico completo della base di dati** è dato quindi dai seguenti **schemi**:

libri (**ISBN**, titolo, lingua, anno, prezzo, *id_genere*, *id_editore*)

autori (**id_autore**, cognome, nome, nazione, data_nascita, sesso)

editori (**id_editore**, ragione_sociale, indirizzo, città)

generi (**id_genere**, descrizione)

autori_libri (**id_autore**, **ISBN**, ruolo)

CAP 1: COMANDI PER CREARE IL DATBASE E LE TABELLE

Vediamo i comandi DDL per la creazione delle tabelle e dei relativi vincoli:

- Comando per creare il database:

  create database 5b_libreria

- Iniziamo sempre dalle relazioni che non hanno chiavi esterne, ad esempio da "autori"

  autori (**id_autore**, cognome, nome, nazione, data_nascita, sesso)

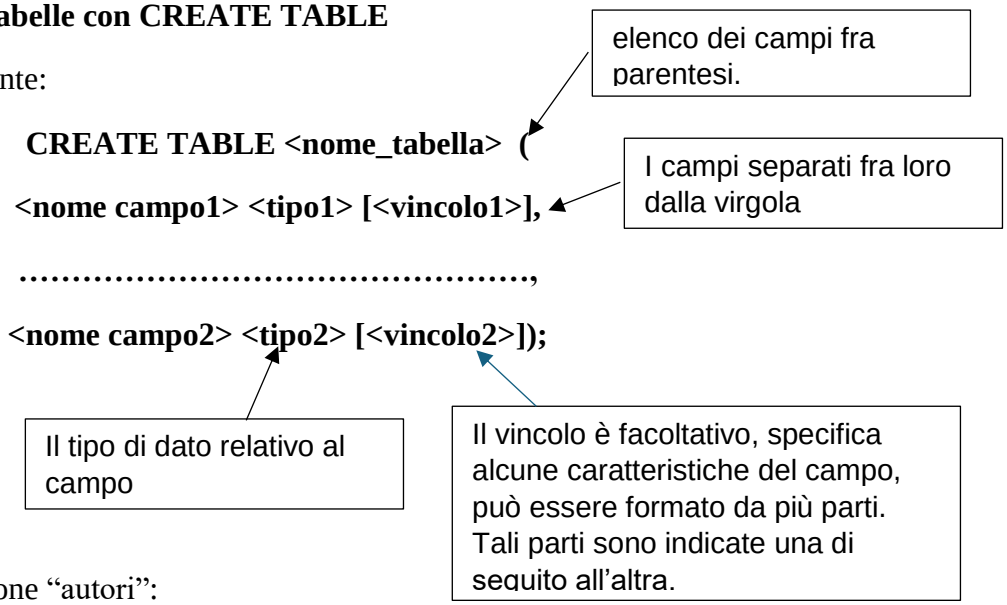
NOME	CHIAVE	TIPO	DIMENSIONE	NOT NULL
id_autore	PK	int auto_increment		X
cognome		varchar	30	X
nome		varchar	30	X
nazione		varchar	30	X
data_nascita		date		X
sesso		M o F		X

varchar è un tipo di dato string da utilizzare quando il valore assunto dall'attributo ha lunghezza variabile (ad esempio per i nomi). Quando invece la lunghezza è fissa (ad esempio per CF) è meglio usare il tipo di dato char (più veloce nella ricerca)

auto_increment indica un numero intero che si incrementerà automaticamente ogni volta che si aggiungerà un nuovo elemento (autore) alla tabella

La creazione delle tabelle con CREATE TABLE

La sintassi è la seguente:



Comando DDL per la relazione "autori":

```
create table autori (
  id_autore INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
  cognome VARCHAR(30) NOT NULL,
  nome VARCHAR (30) NOT NULL,
  nazione VARCHAR (30) NOT NULL,
  data_nascita DATE NOT NULL,
  sesso ENUM ('M','F') NOT NULL);
```

Attenzione: se copi e incolli questo codice in phpMyAdmin gli apici devono essere riscritti a mano

- Relazione "editori"

  editori (**id_editore**, ragione_sociale, indirizzo, città)

NOME	CHIAVE	TIPO	DIMENSIONE	NOT NULL
id_editore	PK	int auto_increment		X
ragione_sociale		varchar	30	X
indirizzo		varchar	30	X
citta		varchar	30	X

Comando DDL per la relazione "editori":

```
create table editori (
  id_editore int AUTO_INCREMENT NOT NULL PRIMARY KEY,
```

ragione_sociale VARCHAR(30) NOT NULL,

indirizzo VARCHAR (30) NOT NULL,

citta VARCHAR (30) NOT NULL);

- Relazione “generi”

  generi (**id_genere**, descrizione)

NOME	CHIAVE	TIPO	DIMENSIONE	NOT NULL
id_genere	PK	int auto_increment		X
descrizione		varchar	30	X

Comando DDL per la relazione “generi”:

create table generi

(id_genere INT AUTO_INCREMENT NOT NULL PRIMARY KEY,

descrizione VARCHAR(30) NOT NULL)

- Relazione “libri”

  libri (**ISBN**, titolo, lingua, anno, prezzo, *id_genere*, *id_editore*)

NOME	CHIAVE	TIPO	DIMENSIONE	NOT NULL
ISBN	PK	char	13	X
titolo		varchar	50	X
lingua		varchar	15	X
anno		int		X
prezzo		float		X
id_genere	FK	int		X
id_editore	FK	int		X

FK: id_genere → generi.id_genere

id_editore → editori.id_editore

Attenzione: la FK deve essere **dello stesso tipo** della PK a cui è associata. In questo caso “int” **MA NON DEVE ESSERE AUTO_INCREMENT**

Comando DDL per la relazione “libri”:

```
create table libri (
   ISBN CHAR(13) NOT NULL PRIMARY KEY,
   titolo VARCHAR(50) NOT NULL,
   lingua VARCHAR(15) NOT NULL,
   anno INT NOT NULL,
   prezzo INT NOT NULL,
   id_genere INT NOT NULL,
   id_editore INT NOT NULL,
   CONSTRAINT fk1_libri FOREIGN KEY (id_genere) REFERENCES generi(id_genere),
   CONSTRAINT fk2_libri FOREIGN KEY (id_editore) REFERENCES editori(id_editore));
```

Aggiunta del vincolo di integrità referenziale (Foreign key). Questo vincolo si aggiunge in seguito all’elenco dei campi ed ha la seguente sintassi:

CONSTRAINT: parola chiave che consente di aggiungere qualsiasi vincolo,

fk1: nome assegnato al vincolo (può essere qualsiasi nome ma **i nomi dei vincoli non possono ripetersi all’interno dello stesso database**)

FOREIGN KEY: indica il tipo di vincolo assegnato, in questo caso vincolo di chiave esterna

(id_genere): nome dell’attributo di questa tabella che diventerà la chiave esterna

REFERENCES: parola chiave per indicare la PK associata alla FK

editori(id_editori): relazione e attributo PK a cui è associata la FK

- Relazione “autori_libri”

  autori_libri (**id_autore**, **ISBN**, ruolo)

NOME	CHIAVE	TIPO	DIMENSIONE	NOT NULL
id_autore	PK, FK	int		X
ISBN	PK, FK	char	13	X
ruolo		varchar	15	

FK: id_autore → autori.id_autore

ISBN → libri.ISBN

OSSERVAZIONE: in questo caso la PK è composta quindi non è possibile indicare tale vincolo direttamente sugli attributi, anche il vincolo di chiave primaria va assegnato con la parola chiave CONSTRAINT

Comando DDL per la relazione “autori_libri”:

create table autori_libri(

id_autore INT NOT NULL,

ISBN CHAR(13) NOT NULL,

ruolo VARCHAR(15),

CONSTRAINT pk_autori_libri PRIMARY KEY (id_autore,ISBN),

altro modo per definire il vincolo di CHIAVE PRIMARIA. Si deve usare questa sintassi quando la PK è composta

CONSTRAINT fk1_autori_libri FOREIGN KEY (id_autore) REFERENCES autori(id_autore),

CONSTRAINT fk2_autori_libri FOREIGN KEY (ISBN) REFERENCES libri(ISBN));

AGGIUNTA DI ALTRI VINCOLI

Vincolo UNIQUE

Per ogni singolo campo può essere definito il vincolo di unicità UNIQUE. Questo vincolo non consente che il contenuto del campo possa ripetersi anche se l’attributo non è chiave primaria.

Esempio: se vogliamo che il campo “descrizione” della relazione “generi” non abbia mai due valori uguali si può aggiungere a tale campo il vincolo “UNIQUE”.

La tabella andrà dunque definita nel seguente modo (**se la tabella è già presente nel database dovrà essere eliminata prima di poterne creare un’altra con lo stesso nome**. Per eliminare la tabella si veda il comando **DROP. Una relazione non può essere eliminata se la sua PK è utilizzata come FK in un’altra relazione poiché ciò potrebbe violare il vincolo di integrità referenziale**):

create table generi

(id_genere INT AUTO_INCREMENT NOT NULL PRIMARY KEY,

descrizione VARCHAR(30) UNIQUE NOT NULL)

Il vincolo UNIQUE può essere aggiunto ad un **insieme** di attributi, in questo caso il valore di ogni singolo attributo può ripetersi, ciò che non si può ripetere è il valore di un **insieme** di attributi. In questo caso la sintassi utilizza il CONSTRAINT.

Esempio: si vuole che nella relazione “libri” non possano essere presenti più libri che abbiano lo stesso titolo e la stessa lingua. Deve però essere possibile inserire più libri che hanno lo stesso titolo e più volte libri che hanno la stessa lingua.

La tabella andrà dunque definita nel seguente modo:

create table libri (

ISBN CHAR(13) NOT NULL PRIMARY KEY,

titolo VARCHAR(50) NOT NULL,

lingua VARCHAR(15) NOT NULL,

anno INT NOT NULL,

prezzo INT NOT NULL,

id_genere INT NOT NULL,

id_editore INT NOT NULL,

CONSTRAINT fk1_libri FOREIGN KEY (id_genere) REFERENCES generi(id_genere),

CONSTRAINT fk2_libri FOREIGN KEY (id_editore) REFERENCES editori(id_editore),

CONSTRAINT vincolo_titolo_lingua UNIQUE (titolo, lingua));

Vincolo CHECK

Il vincolo **CHECK** viene utilizzato per garantire che i valori di una colonna soddisfino una condizione specificata. Può essere usato per limitare i valori che possono essere inseriti in una colonna. La condizione deve essere espressa con un predicato dell'Algebra di Boole.

Esempio: si vuole che sia possibile inserire un libro solo se il suo prezzo è maggiore o uguale a 0.

In tal caso la relazione "libri" va creata nel seguente modo:

create table libri (

ISBN CHAR(13) NOT NULL PRIMARY KEY,

titolo VARCHAR(50) NOT NULL,

lingua VARCHAR(15) NOT NULL,

anno INT NOT NULL,

prezzo INT NOT NULL,

id_genere INT NOT NULL,

id_editore INT NOT NULL,

CONSTRAINT fk_libri1 FOREIGN KEY (id_genere) REFERENCES generi(id_genere),

CONSTRAINT fk_libri2 FOREIGN KEY (id_editore) REFERENCES editori(id_editore),

CONSTRAINT prezzo_libro CHECK(prezzo>=0));

ATTENZIONE: l'interfaccia phpMyAdmin potrebbe mostrare la presenza di un errore quando si utilizza il vincolo CHECK ma esegue comunque il comando.

Il vincolo CHECK può essere definito anche coinvolgendo più campi della relazione.

Esempio, se si vuole vincolare l'inserimento di libri in lingua spagnola solo se il loro prezzo è uguale a 0, la tabella "libri" deve essere definita nel seguente modo:

```
create table libri (
   ISBN CHAR(13) NOT NULL PRIMARY KEY,
   titolo VARCHAR(50) NOT NULL,
   lingua VARCHAR(15) NOT NULL,
   anno INT NOT NULL,
   prezzo INT NOT NULL,
   id_genere INT NOT NULL,
   id_editore INT NOT NULL,
   CONSTRAINT fk_libri10 FOREIGN KEY (id_genere) REFERENCES generi(id_genere),
   CONSTRAINT fk_libri11 FOREIGN KEY (id_editore) REFERENCES editori(id_editore),
   CONSTRAINT spagnolo_gratis CHECK(NOT(lingua="spagnolo" AND prezzo!=0)));
```

Vincolo DEFAULT

Per ogni singolo campo può essere definito il vincolo DEFAULT. Questo non è un vero e proprio vincolo ma consente di assegnare un valore di DEFAULT ad un campo non valorizzato durante l'inserimento del dato.

Esempio: nella relazione autori si vuole poter inserire il valore "pseudonimo" come valore del campo "nome" quando l'autore è noto con uno pseudonimo. In tal caso è opportuno anche che il sesso possa assumere valore NULL (poiché potrebbe non essere noto tale dato) e che abbia come default il valore NULL. In tal caso la tabella dovrebbe essere definita nel seguente modo:

```
create table autori (
   id_autore INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
```

cognome VARCHAR(30) NOT NULL,

nome VARCHAR (30) DEFAULT "pseudonimo" NOT NULL,

nazione VARCHAR (30) NOT NULL,

data_nascita DATE NOT NULL,

sesso ENUM ('M','F') DEFAULT NULL);

Attenzione: se copi e incolli questo codice in phpMyAdmin gli apici devono essere riscritti a mano

Attenzione: se un campo non viene valorizzato durante l'inserimento, a tale campo non verrà assegnato il valore NULL ma uno fra i seguenti valori (sia che che ci sia il vincolo sia che non ci sia):

- 0 se il campo è di tipo numerico
- stringa vuota se il campo è di tipo char o varchar
- 0000-00-00 se il campo è di tipo date

L'assegnamento di un valore NULL per un campo va specificato esplicitamente, come vedremo con i comandi DML (Data Manipulation Language)

Aggiungere un indice con INDEX

Anche l'indice non è un vero e proprio vincolo. Creare un indice su un campo di una tabella significa "segnalare" tale campo al motore del database (**database engine**). Il database engine è la parte del DBMS che si occupa della memorizzazione e ricerca fisica dei dati. La creazione di un indice su un attributo consente di velocizzare le ricerche dei dati basate su quel campo. Il prezzo da pagare è che rallentano le operazioni di inserimento che coinvolgono quel campo, quindi gli indici vanno utilizzati con accortezza e prevalentemente sui campi che hanno tante ricerche e pochi inserimenti.

Esempio: se vogliamo aggiungere un indice sull'attributo "titolo" della relazione "libri" la sintassi è la seguente:

CREATE INDEX indice_titolo ON libri(titolo)

nome della tabella

nome dell'attributo

Per visualizzare gli indici presenti in una tabella la sintassi è la seguente:

SHOW INDEX FROM libri

CAP 2: COMANDI PER MODIFICARE LE TABELLE GIA' ESISTENTI

Per apportare modifiche a una tabella si utilizzano i comandi **RENAME, ALTER, DROP**

1. Il comando **RENAME** permette di rinominare una tabella **(rinominare un database non si può, ne devi creare uno nuovo)**

   RENAME TABLE libri TO volumi

2. Il comando **ALTER** permette di modificare una tabella in combinazione con i comandi **ADD, MODIFY, DROP:**

   I. **ALTER TABLE <nome tabella> ADD …..**
   per aggiungere ad una relazione un attributo, un indice, un vincolo.

   Esempio: Aggiungere un attributo "pagine" di tipo intero alla relazione "libri"
   ALTER TABLE libri ADD pagine INT NOT NULL
   (ai libri già presenti verrà assegnato pagine=0)

   Esempio: Aggiungere il vincolo CHECK che l'attributo "pagine" della relazione "libri" debba essere >=0
   ALTER TABLE libri ADD CONSTRAINT CHECK(pagine>=0)
   (se il vincolo non è rispettato da alcuni dati presenti il DBMS non esegue il comando e segnala un errore)

   II. **ALTER TABLE <nome tabella> MODIFY …..**
   per modificare il tipo di dato e le caratteristiche di un attributo esistente (tipo di dato, auto_increment, not null, ecc…, ma non il nome)

   Esempio: modificare l'attributo "pagine" della tabella "libri" in modo che si possa assegnare il valore null

   ALTER TABLE libri MODIFY pagine INT

   (manca il NOT NULL quindi sarà possibile assegnare il valore NULL all'attributo pagine)

   ALTER TABLE <nome tabella> CHANGE COLUMN …..

   funziona come MODIFY per modificare gli attributi ma in più consente anche di modificare il nome di un attributo. Attenzione, in DBMS diversi da MySQL/MariaDB la sintassi è diversa quindi questo comando non funziona.

Esempio: modificare il nome dell'attributo "pagine" in "numero_pagine"

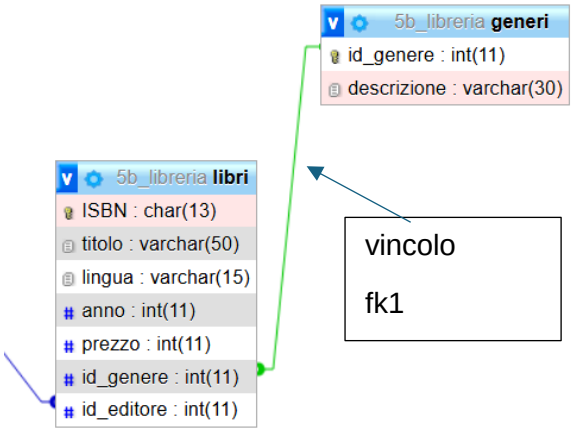
ALTER TABLE libri CHANGE COLUMN pagine numero_pagine INT

III. **ALTER TABLE <nome tabella> DROP …..**
per eliminare un attributo, un indice, un vincolo.

Esempio: eliminiamo dalla tabella "libri" l'attributo "pagine"
ALTER TABLE libri DROP numero_pagine

3. Il comando **DROP rimuove** un elemento da un database (una tabella, un indice, il database stesso). Può agire sull'intero database, non sulla singola tabella. (Per eliminare elementi dalla singola tabella si usa il comando già visto ALTER TABLE tabella DROP …)

Esempio: Se si volesse rimuovere la tabella "generi" dal database non sarebbe possibile poiché il vincolo di integrità referenziale la lega alla tabella libri. Questo accade anche se non vi sono libri presenti. **Una tabella non può essere eliminata se la sua PK è FK in un'altra tabella**.



Per eliminare la tabella "generi" sarà dunque necessario agire nel seguente modo:

a. prima eliminare il vincolo di integrità referenziale (il quale ha un proprio nome che consente di eliminarlo) dalla tabella "libri".
b. successivamente eliminare la relazione "generi"

comando 1: ALTER TABLE libri DROP CONSTRAINT fk1

comando2: DROP TABLE generi

Per visualizzare i vincoli presenti in una tabella di un database con il DBMS MySql il comando è il seguente (attenzione, con altri DBMS la sintassi è diversa):

```
SELECT
   CONSTRAINT_NAME,
   CONSTRAINT_TYPE
FROM
   INFORMATION_SCHEMA.TABLE_CONSTRAINTS
WHERE
   TABLE_SCHEMA = 'nome_del_tuo_database'
   AND TABLE_NAME = 'nome_della_tua_tabella';
```