




**Educazione
Lavoro
Istruzione
Sport**



Database Design e SQL
Christian Finucci

Introduzione al linguaggio SQL: **DDL**



Modelli

MODELLO	TIPO DI MODELLO	COSTRUTTI	DERIVAZIONE
concettuale	Entità-relazioni	E-R 	Algoritmi di derivazione
logico	Relazionale	Relazioni(tabelle)	Costrutti del linguaggio sql
fisico	DBMS Relazionale	Tabelle	



SQL: storia

Originariamente acronimo per "Structured Query Language", ora nome proprio.

Linguaggio con varie funzionalità:

contiene sia il DDL sia il DML

Base comune, i vari DBMS ne ampliano le funzionalità con features specifiche

SQL: storia

- Modello relazionale (1970, Edgar Codd)
- prima proposta implementativa SEQUEL (1974);
- prime implementazioni in SQL/DS e Oracle (1981);
- dal 1983 ca. "standard di fatto";
- Standardizzato dal 1986

mySQL: storia

Aprile 1995: Viene rilasciata la prima release, realizzata con la sponsorizzazione dalla società svedese MySQL AB

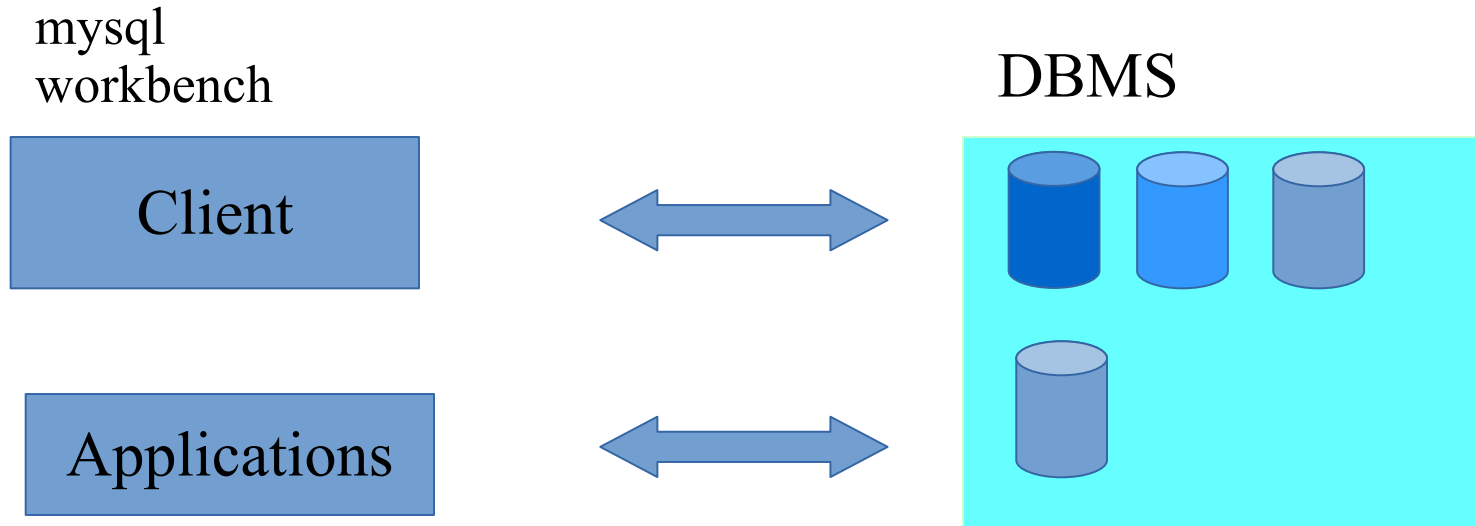
Gennaio 2008: Sun Microsystems acquista la società MySQL AB per un miliardo di dollari

Aprile 2009: Oracle acquisisce Sun Microsystems è stata proposta l'acquisizione da parte di Oracle per 7,4 miliardi di dollari.

Gennaio 2012: diverse distribuzioni Linux e alcuni utenti importanti (ex wikipedia) hanno iniziato a sostituire MySQL con il fork MariaDB

Architettura DBMS MySQL

Processo Server



Gestione attività

File Opzioni Visualizza

Processi Prestazioni Cronologia applicazioni Avvio Utenti Dettagli Servizi

Nome	PID	Descrizione	Stato
MySQL80	6492	MySQL80	In esecuzione

Connessione del client al DBMS

Tutti I comandi seguenti vanno digitati dalla MySQL Command Line Client

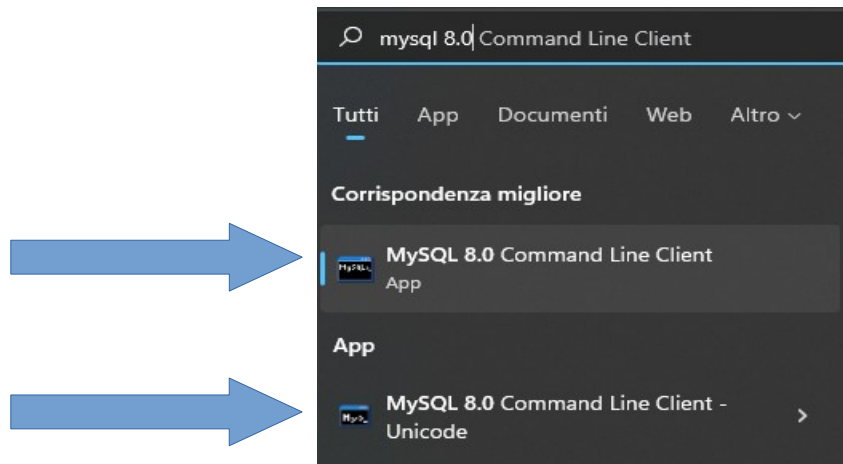
Mac

- Dal Terminale

```
/usr/local/mysql/bin/mysql -u root -p
```

Windows

- Dal menu start



Mysql Troubleshooting

- Access denied:

CAUSA: il server è attivo ma l'utente non è stato riconosciuto

SOLUZIONE: controllare nome utente (root) e password (root)

- Can't connect to server:

CAUSA: il server non è attivo

SOLUZIONE: è necessario avviare il server MySQL80 da Gestione Attività/Servizi

- Comando non riconosciuto:

CAUSA: Il SO non trova il comando

SOLUZIONE: Necessario aggiungere alla variabile di PATH il percorso

`C:\Program Files\mysql\mysqlserver 8.x\bin`

`YOU MUST RESET YOUR PASSWORD (mac)`

`SET PASSWORD FOR root@localhost = PASSWORD('root');`

Data Definition Language

Lista dei databases

Istruzione

```
show databases;
```

Mostra la lista dei database accessibili dall'utente



Tutte
le istruzioni
vanno
completate con un
punto e virgola

Creazione di un database

Istruzione CREATE DATABASE

Crea un'istanza vuota di un database

```
create database prova;
```

Domanda: quante tabelle contiene il db 'prova' appena creato?

Database corrente

Istruzione

```
use nome_database;
```

Il database specificato diventa il database attivo (correntemente in uso): tutti i comandi successivi saranno destinati al database attivo

Ex:

```
use prova;
```

Definizione dei dati in SQL

Istruzione

CREATE TABLE:

- Definisce uno schema di una tabella e ne crea una vuota.
- Specifica attributi, domini e, opzionalmente, vincoli.

```
CREATE TABLE utente (  
    id INT PRIMARY KEY,  
    cognome VARCHAR(30),  
    codice_fiscale CHAR(16) UNIQUE,  
    anno_nascita INT NOT NULL  
);
```

Tabelle di un database

- Istruzione `show tables;`

Mostra la lista delle tabelle del database corrente

Istruzione `show create table nome_tabella;`

Specifica attributi, domini e vincoli della tabella. Rappresenta la sequenza di istruzioni SQL che servono a creare la tabella in oggetto.

Ex: `show create table utente;`

Istruzione `describe nome_tabella;`

Specifica attributi, domini e vincoli della tabella.

Ex: `describe utente;`

Eliminazione di un database

Istruzione DATABASE

DROP

Elimina un database e tutti i dati in esso contenuti

Ex:

```
drop database prova;
```



Connessione del client al DBMS

Tutti I comandi seguenti vanno digitati dalla MySQL Command Line Client

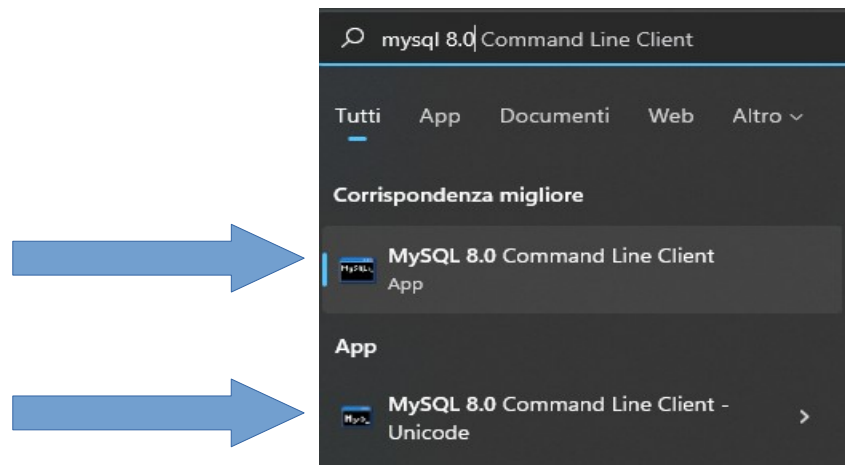
Mac

- Dal Terminale

```
/usr/local/mysql/bin/mysql -u root -p
```

Windows

- Dal menu start



Esercizio

1) Creare il database instagram

```
create database instagram;
```

2) Verifica la creazione del database con il comando

```
Show databases;
```

3) Renderlo attivo

```
use instagram;
```

4) Creare la tabella utente

```
create table utente(  
id int primary key,  
nome varchar(20) not null unique);
```

5) Eseguire i tre comandi e osservare l'output

```
show tables;  
describe utente;  
show create table utente;
```



Esercizio

5) Aggiungere utenti

```
insert into utente(id, nome) values (1, "Rino");  
insert into utente(nome,id) values ('Pino',2);  
insert into utente values (8, 'Cino');
```

6) visualizzare il contenuto della tabella

```
select * from utente;
```

7) provare ad aggiungere un altro utente: perché abbiamo un errore?

```
insert into utente(id, nome) values (1, "Gino");  
insert into utente(id, nome) values (3, "Rino");  
insert into utente(id, nome) values (4, null);
```

8) Inserire voi stessi nel database e visualizzare il contenuto della tabella;

```
select * from utente;
```



Esercizio

9) eseguire le seguenti query e osservare il risultato;

```
select * from utente order by id;  
select * from utente order by nome;  
select * from utente order by id desc;  
select nome from utente;  
select nome,id from utente;  
select * from utente where id>2;  
select nome from utente where id=2;
```

10) Eliminare il database appena creato



```
drop database instagram;
```

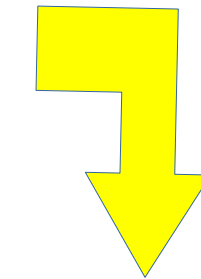
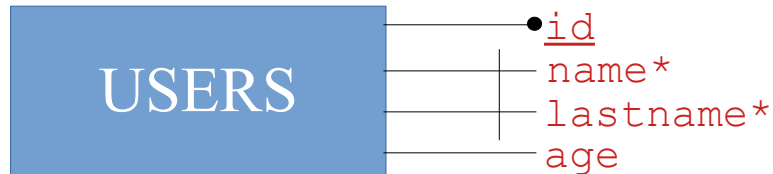
11) Verifica l'avvenuta eliminazione del database con il cc

```
Show databases;
```



Livelli di modellazione

MODELLO CONCETTUALE



MODELLO LOGICO

USERS(id, name*, lastname*, age)

A large yellow arrow pointing to the left, indicating the transition from the Logical Model to the Physical Model.

MODELLO FISICO

```
create table users(  
  id int unsigned primary key auto_increment,  
  name varchar(20) not null,  
  lastname varchar(20) not null,  
  age tinyint unsigned,  
  unique(name, lastname)  
);
```

CREATE TABLE, esempio

L'output di `show create table Impiegato` è

```
CREATE TABLE Impiegato(  
  id INT PRIMARY KEY AUTO_INCREMENT,  
  nome CHAR(20) NOT NULL,  
  cognome VARCHAR(20) NOT NULL,  
  nascita DATE,  
  id_dipart int,  
  stipendio INT DEFAULT 0,  
  ingresso TIME,  
  FOREIGN KEY(id_dipart) REFERENCES Dipartimento(id),  
  UNIQUE (Cognome, Nome)  
);
```

Dichiarazione degli attributi

Nome attributo	Dominio	Valore di Default	Vincoli
----------------	---------	-------------------	---------

- **Nome attributo** (*obbligatorio, al primo posto*)
- **Dominio** elementare di appartenenza (*obbligatorio, al secondo posto*)
- **Valore di default** (*opzionale, è il valore che deve assumere l'attributo quando non ne viene specificato un valore durante l'inserimento di una riga*)
- **Vincoli** (constraint sui valori ammissibili, *opzionali*)

Dichiarazione degli attributi

Nome attributo	Dominio	Valore di Default	Vincoli
----------------	---------	-------------------	---------

Solo “nome attributo” e “Dominio” sono obbligatori

- Stipendio `INT` `DEFAULT 0` `NOT NULL`
- Nascita `DATE`
- Cognome `VARCHAR(20)` `NOT NULL`
- `id int unsigned auto_increment primary key`

Domini elementari

- **I database relazionali supportano solo tipi di dato scalari o elementari, cioè composti da un solo valore**
 - Numerici, esatti e approssimati
 - Stringa
 - Data, ora
 - Booleani
 - Enum
 - Blob

Tipi Interi

Type	Length in Bytes	Minimum Value (Signed)	Maximum Value (Signed)	Minimum Value (Unsigned)	Maximum Value (Unsigned)
TINYINT	1	-128	127	0	255
SMALLINT	2	-32768	32767	0	65535
MEDIUMINT	3	-8388608	8388607 to	0	16777215
INT	4	-2147483648	2147483647	0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807	0	18446744073709551615

Spazio occupato dal singolo valore:

- il numero 32, come tinyint occupa 1 byte
- il numero 32, come bigint occupa 8 byte

Dichiarazione di colonne numeriche intere

```
CREATE TABLE numeri_interi (  
  temperatura tinyint,  
  evento_storico smallint,  
  età tinyint unsigned,  
  contatore_grande bigint unsigned  
);
```

Tipi “float”, “double ”

I tipi FLOAT e DOUBLE contengono tipi di dato decimali (ex 1.24) e in notazione scientifica 1E2300

Type	Length in Bytes	Minimum Value (Signed)	Maximum Value (Signed)	Minimum Value (Unsigned)	Maximum Value (Unsigned)
FLOAT	4	-3.402823466E+38	-1.175494351E-38	1.175494351E-38	3.402823466E+38
DOUBLE	8	-1.7976931348623157E+ 308	-2.2250738585072014E- 308	0, and 2.2250738585072014E- 308	1.7976931348623157E+ 308

Esempio

```
CREATE TABLE galassia (  
  nome_galassia varchar(21),  
  stelle_galassia INT unsigned,  
  media_grandezza FLOAT,  
  distanza_sole DOUBLE  
  );
```

La funzione `ROUND(colonna, numero_decimali)` permetterà di scegliere il numero di cifre decimali da visualizzare

Tipi testuali “*char*” e “*varchar*”

- Sono tipi destinati alla rappresentazione di stringhe di lunghezza fissa (`char`) o variabile (`varchar`)
- La lunghezza massima supportata è di 65,535 caratteri
- Il dato `char` e `varchar` deve essere inserito tra apici doppi o singoli
- Ex "Rino Rano", 'AB123CD'

Tipi testuali: *char*

- Utilizzato quando le stringhe saranno di lunghezza fissa e nota
 - La lunghezza massima è 65,535 caratteri
- E' obbligatorio specificare la lunghezza massima tra parentesi
- Lo storage occupato è **fisso** e non dipende dalla lunghezza della stringa inserita
-

EX definendo un campo

```
targa char(6)
```

e memorizzando in esso la stringa "ciao" mysql riempirà i due byte mancanti con degli spazi vuoti a destra "ciao".

Esercizio char

1) Creare e attivare un database chiamato “prova_char”

2) creare e popolare la tabella seguente

```
create table studenti(nome char(6));
```

1)Popolare la tabella

2) Insert into studenti(nome) values("abcd");

3) insert into studenti values("abcdef");

4) insert into studenti values("abcdefg");

5) select nome from studenti;

Tipi testuali: *varchar*

- Li uso nella dichiarazione di colonne destinate a contenere stringhe di lunghezza variabile;
- La lunghezza massima è 65,535 caratteri
- E' obbligatorio specificare la lunghezza massima tra parentesi
- I valori varchar vanno inseriti tra apici doppi o apici singoli

Ex

```
nome varchar(50)
```

Tipi testuali: varchar

STORAGE

- I varchar occupano una quantità di spazio che dipende dalla lunghezza della stringa inserita
 - un byte in più della lunghezza effettiva se la stringa è lunga meno di 255 caratteri
 - due byte in più della lunghezza effettiva se la stringa è lunga più di 255 caratteri

e memorizzando 'linux' si occuperanno 6 byte (5+1).

Storage di “*char*” e “*varchar*” a confronto

Value	char(4)	char(4) Storage required	VARCHAR(4)	varchar(4) Storage required
"	' '	4 bytes	"	1 byte
'ab'	'ab '	4 bytes	'ab'	3 bytes
'abc'	'abc '	4 bytes	'abc'	4 bytes
'abcd'	'abcd'	4 bytes	'abcd'	5 bytes
'abcdef'	Data too long error	0 bytes	Data too long error	0 bytes

Esercizio Università

1) Convertire nel modello fisico il seguente schema concettuale

1) L'età è positiva

2) QI è un numero con la virgola

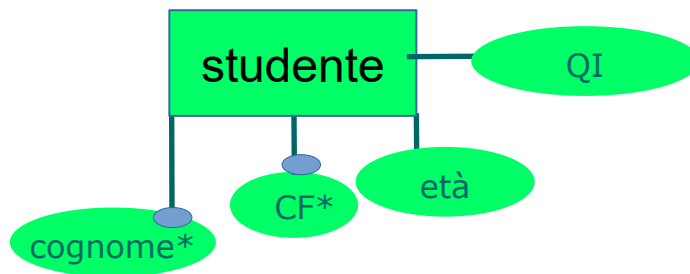
2) Creare il database **università** su mysql

3) Renderlo attivo (use..)

4) Creare la tabella studenti

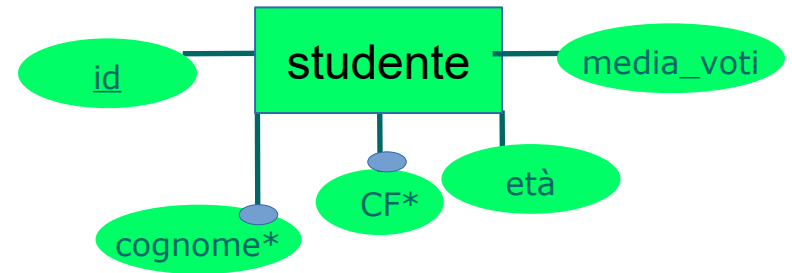
5) Inserire tre studenti

6) Visualizzare il contenuto della tabella



```
create table users(  
  id int unsigned primary key auto_increment,  
  name varchar(20) not null,  
  lastname varchar(20) not null,  
  age tinyint unsigned,  
  unique(name, lastname)  
);
```





//CREAZIONE

```
create table studenti(  
id int unsigned auto_increment primary key,  
cognome varchar(20) unique not null,  
CF char(16) unique not null,  
et  tinyint unsigned,  
qi float  
);
```

//INSERIMENTO VALORI

```
insert into studenti (cognome,CF,et ,qi)  
values ("bianchi","ABC23",22,128.34);
```

//LETTURA VALORI

```
select * from studenti;
```



Tipo “*text*”

TEXT

- è pensato per contenere stringhe di testo senza la limitazione dei 65,535 caratteri imposta da char e varchar.
- Il testo potrà avere una lunghezza molto elevata, fino a 2^{32} byte
- Sui campi di tipo TEXT **non** è possibile porre un vincolo di unicità

Range dei "text"

Data Type	Storage Required (bytes)
TINYTEXT	$L + 1$ bytes, where $L < 2^8$
TEXT	$L + 2$ bytes, where $L < 2^{16}$
MEDIUMTEXT	$L + 3$ bytes, where $L < 2^{24}$
LONGTEXT	$L + 4$ bytes, where $L < 2^{32}$

```
create table libro(  
id int unsigned primary key,  
titolo varchar(20),  
testo mediumtext,  
...
```

Tipi “*blob*”

BLOB (Binary Large Object)

- è pensato per contenere stringhe di dati binari: lo utilizzerò quindi per memorizzare immagini, programmi, mp3.



Range dei ***blob***

Data Type	Storage Required (bytes)
TINYBLOB,	$L + 1$ bytes, where $L < 2^8$
BLOB,	$L + 2$ bytes, where $L < 2^{16}$
MEDIUMBLOB,	$L + 3$ bytes, where $L < 2^{24}$
LOB,	$L + 4$ bytes, where $L < 2^{32}$

```
create table brano(  
id int unsigned primary key,  
autore varchar(20),  
mp3 mediumblob,  
...
```

Datetime

- DATETIME è utilizzato per memorizzare un valore che contiene sia data che ora .
- Il contenuto di una colonna DATETIME viene visualizzato come

AAAA-MM-GG HH:MM:SS

- Le date vanno inserite tra apici doppi

"2018-12-01 12:22:33"

- Esistono anche DATE e TIME per memorizzare rispettivamente date ed orari
- Un valore DATETIME utilizza **5 byte** per l'archiviazione .

FORMATI DATA

DATETIME "2018-12-01 12:22:33"

DATE "2018-12-01"

TIME "12:22:33"

VINCOLI

Vincoli tupla

Sono I vincoli che riguardano i valori di una singola cella, indipendenti dai valori già presenti nella tabella

- CHECK
- NOT NULL

NOT NULL constraint

```
CREATE TABLE Impiegato(  
  id INT PRIMARY KEY AUTO_INCREMENT,  
  Nome CHAR(20) NOT NULL,  
  Cognome VARCHAR(20) NOT NULL,  
  Nascita DATE,  
  id_dipart int,  
  Stipendio INT DEFAULT 0,  
  Ingresso TIME,  
  FOREIGN KEY(id_dipart) REFERENCES Dipartimento(id),  
  UNIQUE (Cognome, Nome)  
);
```

CHECK constraint

```
CREATE TABLE `esami` (  
    nome varchar(20),  
    cognome varchar(20),  
    materia varchar(20),  
    voto tinyint unsigned CHECK (voto >=18 AND voto <= 30)  
);
```

Vincoli chiave

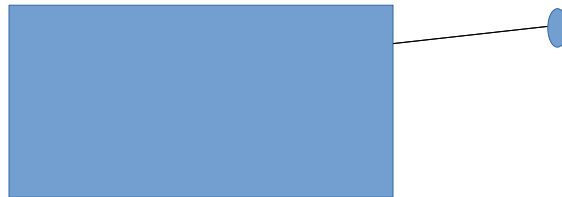
Sono i vincoli che riguardano l'unicità dei valori assunti dagli attributi

- **UNIQUE** definisce chiavi (valori unici, non implica il NOT NULL)
- **PRIMARY KEY**: chiave primaria, un valore unico e non nullo.
 - Nota: può esserne definita una sola per ogni tabella, solitamente il campo `id`)



UNIQUE CONSTRAINTS: singolo campo

```
CREATE TABLE Auto(  
  id INT PRIMARY KEY AUTO_INCREMENT,  
  targa CHAR(7) NOT NULL UNIQUE,  
  marca VARCHAR(20) NOT NULL,  
  modello VARCHAR(20) NOT NULL  
);
```



UNIQUE CONSTRAINTS: su più campi

```
CREATE TABLE Impiegato(  
  id INT PRIMARY KEY AUTO_INCREMENT,  
  Nome CHAR(20) NOT NULL,  
  Cognome VARCHAR(20) NOT NULL,  
  Nascita DATE,  
  id_dipart int,  
  Stipendio INT DEFAULT 0,  
  Ingresso TIME,  
  FOREIGN KEY(id_dipart) REFERENCES Dipartimento(id),  
  UNIQUE (Cognome, Nome) ←  
);
```



Unicità su più attributi

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

≠

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

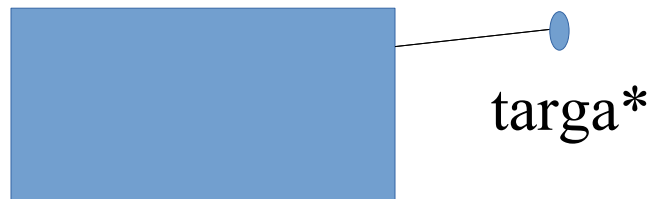


≠



Multiple constraint

```
CREATE TABLE Auto(  
  id INT UNSIGNED PRIMARY KEY AUTO_INCREMENT,  
  targa CHAR(7) NOT NULL UNIQUE CHECK(length(targa)=7),  
  marca VARCHAR(20) NOT NULL,  
  modello VARCHAR(20) NOT NULL  
);
```



PRIMARY KEY CONSTRAINT

```
Matricola int PRIMARY KEY
```

oppure

```
Matricola int,  
.../  
PRIMARY KEY (Matricola)
```

CREARE UNA TABELLA CON CAMPO AUTOINCREMENT

Tabella UTENTI: correggere e migliorare

```
CREATE TABLE utenti(  
    id int primary key AUTO_INCREMENT,  
    nome char(250) NOT NULL,  
    cognome varchar(250) NOT NULL,  
    eta int default '0',  
)ENGINE=InnoDB;
```

CREARE UNA TABELLA CON CAMPO AUTOINCREMENT

Tabella UTENTI: soluzione

```
CREATE TABLE utenti(  
    id int UNSIGNED primary key AUTO_INCREMENT,  
    nome varchar(25) NOT NULL,  
    cognome varchar(25) NOT NULL,  
    eta tinyint unsigned default '0',  
)ENGINE=InnoDB;
```

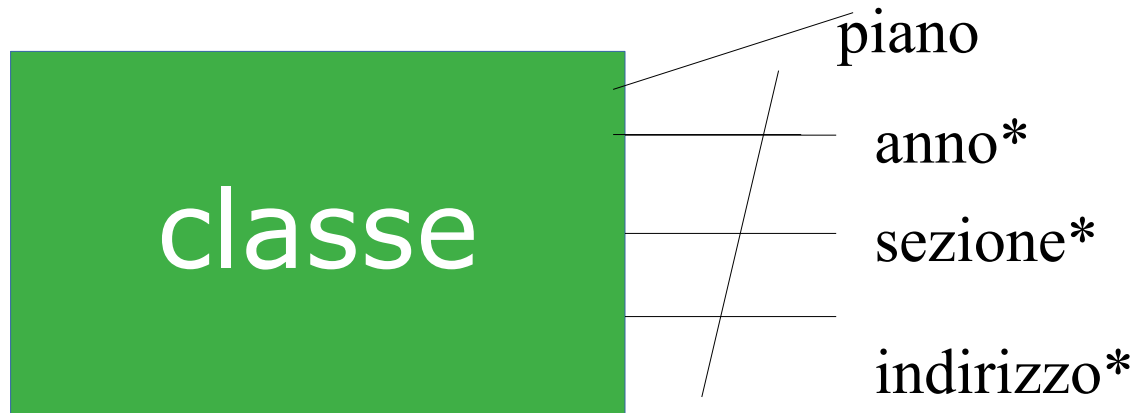
In ogni tabella può esistere un solo campo **AUTO_INCREMENT** e un solo campo **PRIMARY KEY**: nei database in esercizio questo campo sarà **SEMPRE** il campo id

Esercizio

Derivare

- modello logico
- modello fisico

Implementare la tabella in un db chiamato "scuola"

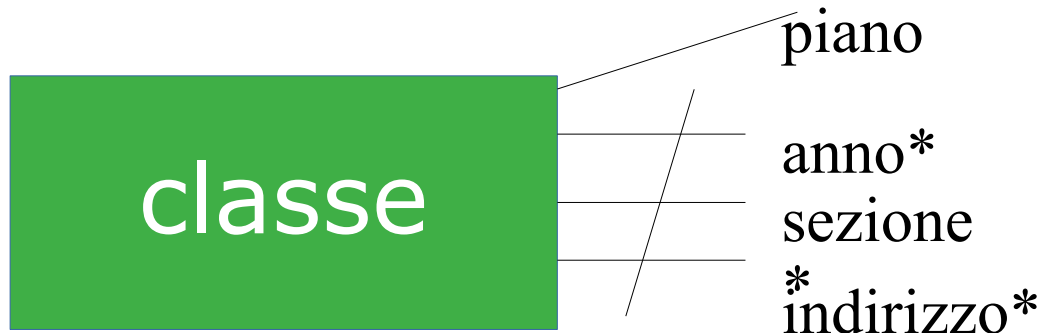


Esercizio

SCHEMA LOGICO

Scuola={classe(id,anno*,sezione*,indirizzo*,piano)}

```
create table classe(  
  id int unsigned primary key auto_increment,  
  anno tinyint unsigned not null check (anno>=1 AND anno <=5),  
  sezione char(1) not null,  
  indirizzo varchar(20) not null,  
  piano tinyint unsigned,  
  unique(anno,sezione,indirizzo)  
);
```



Vincolo di integrità referenziale

Le istruzioni `REFERENCES` e `FOREIGN KEY` permettono di definire vincoli di integrità referenziale

```
//CREAZIONE TABELLA ABITANTI
```

```
CREATE TABLE ABITANTI (
```

```
...
```

```
id_città int unsigned not null,
```

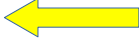

```
...
```

```
FOREIGN KEY(id_città) REFERENCES città(id)
```


FK




PK a cui si riferisce

Vincolo di integrità referenziale

- Stabilire un vincolo di integrità referenziale implica:
 - dichiarare la colonna della foreign key 
 - dichiarare il vincolo 
- Il dominio della FK deve essere dello **STESSO** tipo della PK a cui si riferisce altrimenti il DBMS impedirà la creazione del vincolo:
 - Se la primary key è **int unsigned** allora la foreign key dovrà essere **int unsigned**
 - Se la primary key è **mediumint** allora la foreign key dovrà essere **mediumint**
- il vincolo di FK **non** implica il not null; se la partecipazione dal lato M è obbligatoria, va posto (vedi slide "checklist" della derivazione concettuale- → logico)

CREATE TABLE, esempio

```
CREATE TABLE Impiegato(  
  id INT PRIMARY KEY AUTO_INCREMENT,  
  Nome CHAR(20) NOT NULL,  
  Cognome VARCHAR(20) NOT NULL,  
  Nascita DATE,  
  id_dipart int,  
  Stipendio INT DEFAULT 0,  
  Ingresso TIME,  
  FOREIGN KEY(id_dipart) REFERENCES Dipartimento(id),  
  UNIQUE (Cognome, Nome)  
);
```



DDL in sintesi

TABELLE

- **CREARE**
- **VISUALIZZARE**
- **ELIMINARE**
- **RINOMINARE**

creare, visualizzare

CREAZIONE TABELLA

```
CREATE TABLE Contatti  
(id int(4) NOT NULL,  
cognome varchar(30),  
nome varchar(20) );
```

VISUALIZZAZIONE SCHEMA TABELLA

```
DESCRIBE Contatti;
```

SCRIPT CREAZIONE TABELLA

```
SHOW CREATE TABLE Contatti;
```

eliminare, rinominare



ELIMINAZIONE TABELLA

DROP TABLE Contatti;

RINOMINARE UNA TABELLA

ALTER TABLE Contatti
RENAME TO Rubrica;

- **AGGIUNGERE**
- **ELIMINARE**
- **MODIFICARE**
- **RINOMINARE**

Utilità dell'istruzione ALTER

Modifica dello schema di una tabella:
quando nel database sono presenti dati del cliente ("database in esercizio") non è più possibile eliminare e ricreare una tabella per cambiarne lo schema.

Per preservare i dati si ricorre all'istruzione ALTER

aggiungere, eliminare

AGGIUNGERE UN CAMPO

```
ALTER TABLE Contatti  
ADD COLUMN telefono CHAR(30);
```

```
ALTER TABLE Contatti  
ADD COLUMN indirizzo CHAR(30) NOT NULL AFTER nome;
```



ELIMINARE UN CAMPO

```
ALTER TABLE Contatti  
DROP COLUMN Nome;
```

Modificare (dominio, vincoli)

MODIFICA DI UN CAMPO

```
ALTER TABLE Contatti  
MODIFY COLUMN indirizzo VARCHAR(50);  
//indirizzo da char(30) NOT NULL è  
diventato varchar(50) senza vincoli
```

Il MODIFY consente di

- cambiare dominio
- aggiungere/ togliere vincoli di NOT NULL)
- aggiungere vincoli di chiave (UNIQUE, PRIMARY KEY)
- NOTA il campo va ridichiarato interamente con nome, dominio, vincoli e default

Aggiunta/Rimozione del vincolo NOT NULL

Modifica dominio e aggiunge il vincolo NOT NULL e il vincolo UNIQUE

```
ALTER TABLE PERSONE  
MODIFY  nome varchar(20) NOT NULL UNIQUE
```

Modifica dominio ed elimina NOT NULL ma non il vincolo UNIQUE

```
ALTER TABLE PERSONE  
MODIFY  nome varchar(30)
```

E' necessario inserire nome, dominio e, se lo si desidera aggiungere o conservare, il vincolo.

modificare e rinominare (nome, dominio, vincoli)

RINOMINARE (E MODIFICARE)

```
ALTER TABLE Contatti  
CHANGE indirizzo residenza VARCHAR(50);
```

Il **CHANGE** è simile al **modify** ma in più mi permette di **rinominare** la colonna da `indirizzo` a `residenza`

- cambiarne dominio
- aggiungere/ togliere vincoli di tupla (ex NOT NULL)
- aggiungere vincoli di chiave (UNIQUE, PRIMARY KEY)
- **NOTA** il campo va ridichiarato interamente con nome, dominio, vincoli e default

Eliminare vincoli di chiave (unicità)

ELIMINAZIONE DI UN VINCOLO DI
UNICITA' (dalla colonna `indirizzo`)

```
ALTER TABLE Contatti  
DROP INDEX indirizzo;
```

- Un vincolo di tupla viene creato/rimosso mediante il **MODIFY**
- Un vincolo di unicità viene creato mediante il **MODIFY**
- Un vincolo di unicità viene rimosso mediante il **DROP INDEX**

CHECK

Per aggiungere CHECK a colonne già esistenti:

```
alter table auto add constraint CHECK  
(length(targa) = 7);
```


Foreign keys

Per aggiungere foreign key a tabelle già esistenti:

```
ALTER TABLE esami  
ADD FOREIGN KEY (id_studente)  
REFERENCES studente(id);
```

Cambiare lo schema

Creare nel database zalandò la tabella clienti

```
CREATE TABLE clienti (  
  id int,  
  nome char(30) NOT NULL,  
  cognome varchar(30),  
  codice_fiscale varchar(40) NOT NULL,  
  age int UNIQUE  
);
```

Esercizio
avanzato

Utilizzando esclusivamente l'istruzione ALTER TABLE trasformare la tabella clienti nella seguente tabella customers così strutturata

```
CREATE TABLE customers (  
  id int unsigned PRIMARY KEY AUTO_INCREMENT ,  
  name varchar(30) NOT NULL,  
  lastname varchar(30) NOT NULL,  
  address varchar(40),  
  fiscal_code char(16) NOT NULL UNIQUE CHECK (length(fiscal_code) = 16),  
  age tinyint unsigned CHECK ((age >= 18) and (age <= 70))  
)
```



Cambiare lo schema

- `alter table clienti modify id int unsigned primary key auto_increment;`
- `alter table clienti change cognome lastname varchar(30) not null;`
- `alter table clienti change nome name varchar(30) not null;`
- `alter table clienti change codice_fiscale fiscal_code char(16) not null unique;`
- `alter table clienti add constraint CHECK (length(fiscal_code) = 16);`
- `alter table clienti modify age tinyint unsigned;`
- `alter table clienti add constraint CHECK (age>=18 AND age<=70);`
- `alter table clienti drop key age;`
- `alter table clienti rename to customers;`
- `alter table customers add column address varchar(40) after lastname;`



Riepilogo

- Con quale comando creo un nuovo database di nome 'Autonoleggio'?
- Quale tra questi quattro non è obbligatorio nella definizione di uno schema di una tabella:
 - Nome tabella
 - Nome attributo
 - Dominio
 - vincolo
- Cosa significa rendere attivo un database?
- Con quale comando rendo attivo un database?
- Elencare tutti i tipi numerici:
 - Interi
 - Approssimati
- Differenze tra char e varchar
- Istruzioni per
 - Creare una tabella
 - Eliminare una tabella
 - Modificare una tabella