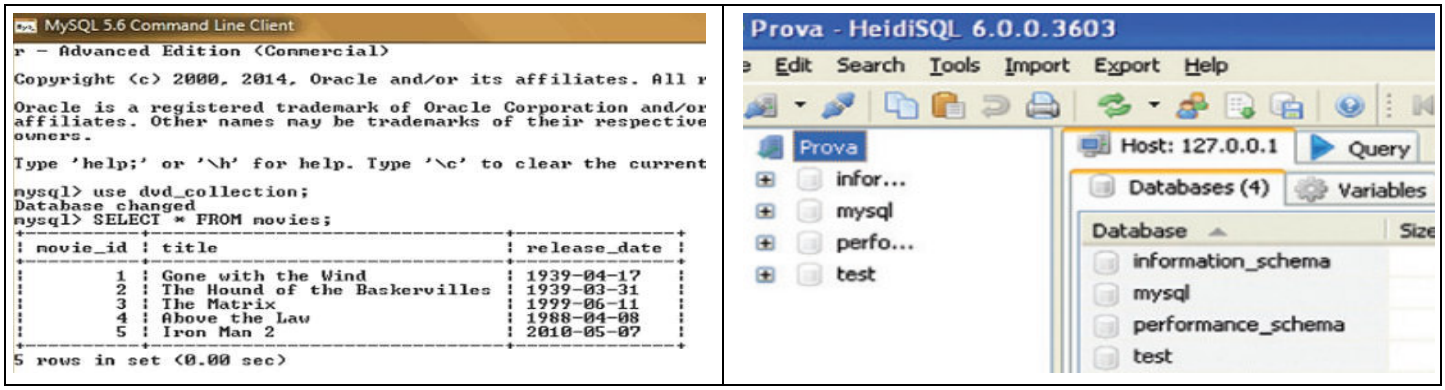
Client MySQL: a riga di comando, desktop e client web

Ricordiamo che un **DBMS server dei database** è un programma che riceve le richieste da parte di un DBMS client e accede **c**onseguentemente ai contenuti del database per fornire al client, se autorizzato, le informazioni che esso richiede. **Un es. di DBMS server è Mysql.** Esso è un dbms di tipo relazionale e permette, quindi, di creare database costituiti da tabelle connesse tra loro da relazioni logiche.

Per motivi di sicurezza l'accettazione delle connessioni dei client può essere limitata a solo:
1) il **localhost** (http://127.0.0.1): i client sono localizzati sulla stessa macchina del server;
2) ai **client** appartenenti allo **stesso dominio di rete del server** stesso.

Per accedere a MySql, essendo esso **multipiattaforma** (disponibile per windows, linux e mac osx), esistono in commercio vari client DMBS: opensource, shareware e a pagamento suddivisi in tre categorie:

1) **a riga di comando** (CUI, *Character User Interface)* in genere usati dal dba (database administrator) per effettuare una manutenzione locale. I client a riga di comando vengono eseguiti dal sistema operativo che ospita il DBMS; MySQL possiede un client a riga di comando sia per Windows sia per Linux. In Windows possiamo utilizzare il client a riga di comando denominato **MySql Command Line Client.** Per usarlo bisogna conoscere i comandi del DBMS MySql



2) **desktop GUI** (*Graphics User Interface*), cioè applicazioni visuali che consentono una manutenzione interattiva. Alcuni esempi sono i seguenti: **heidiSQL**, semplice client per win32 e win64 opensource, **SQLWave e Navicat**: client visuale multipiattaforma, **dbdesigner:** ambiente multipiattaforma di sviluppo visuale dello schema.

3) **client web online**, cioè pagine dinamiche scritte con linguaggi di scripting lato server (ad es. PHP). Ad esempio nelle **suite Easyphp, Xampp e Wamp** (per Windows) **e Lamp** (per Linux) è presente un'applicazione (**phpMyAdmin**) che permette la connessione per l'amministrazione remota del DBMS e la manipolazione dei database attraverso un'interfaccia grafica intuitiva disponibile sul Web.

L'utente root di un database o dba

Esso è quello che crea un database e che è in grado di concedere diritti di lettura, inserimento, cancellazione, modifica, concessione ad altri utenti connessi al database eventualmente identificati da utente e password.

Operazioni preliminari su un DBMS MySql con il pacchetto Xamp 5.6.24 del 2016

Il pacchetto Xamp permette di installare sul nostro computer locale, un server DBMS MySql e un server Web, Apache oltre ad altri server. Il server MySql è necessario per poter creare un database (quindi tabelle e campi),

riempirlo e creare delle query su esso. Il server Apache è necessario per gestire il database tramite l'applicazione web: phpMyAdmin.

1) Scaricato Xampp dal sito personale www.ascuoladi.eu o dal sito del proprietario www.apachefrinds.com[1] il pacchetto xampp, bisogna installarlo nel percorso C://xampp. Aperta la cartella Xampp, eseguire l'applicazione **xampp control panel** (fig.1) attraverso la quale è possibile avviare (**start**) il server MySql e il server Apache (fig.2)

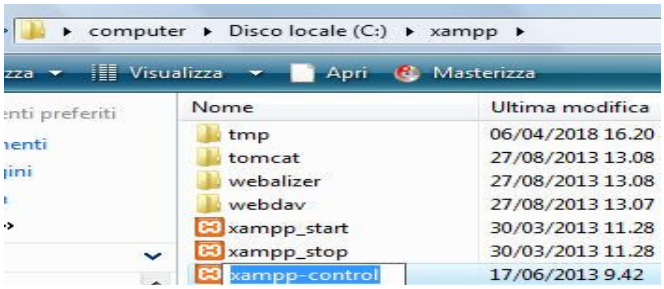


Fig.1

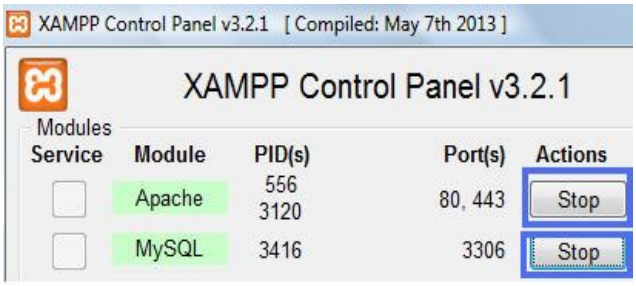
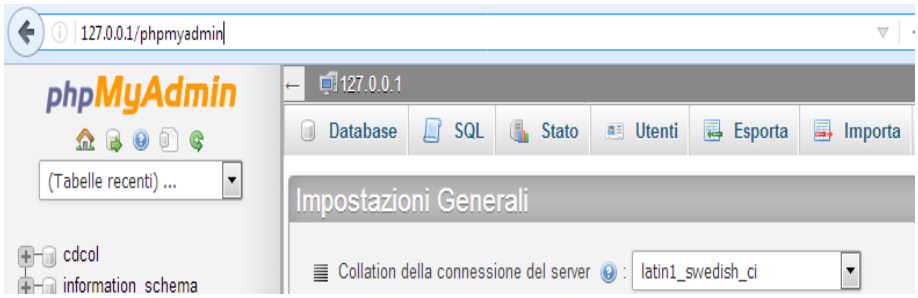


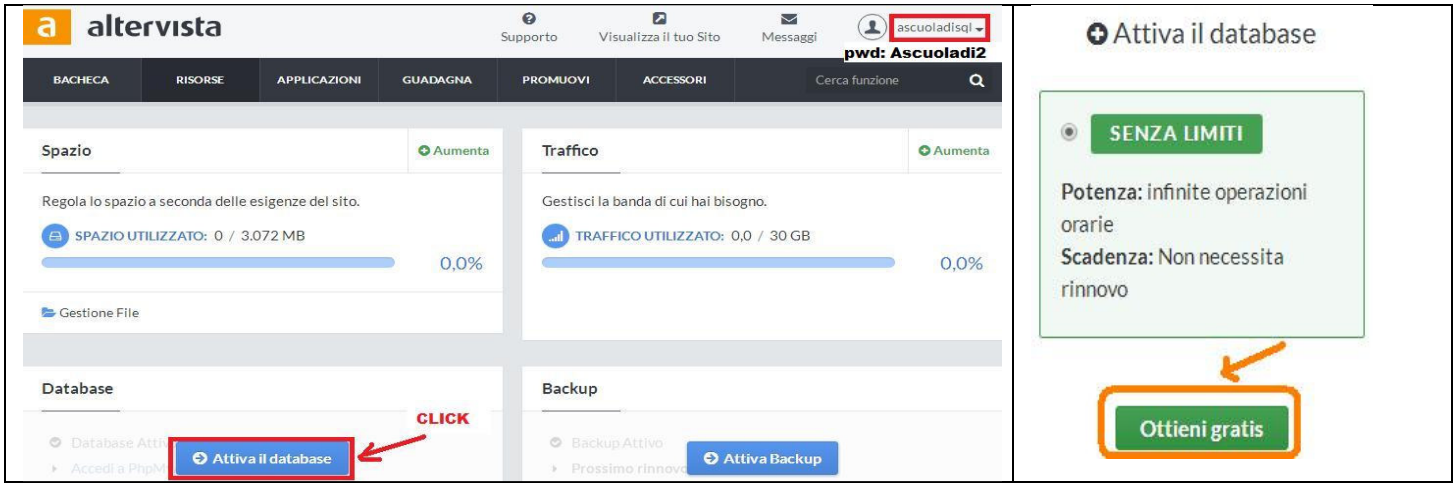
Fig. 2

2) Aprire il browser è collegarsi all'IP locale http://127.0.0.1/phpmyadmin in modo che appaia la finestra seguente:



Uso dei server MySql e Apache on line del sito www.altervista.org

A volte, dopo aver scaricato il pacchetto xampp o similari, se non si dovesse riuscire a fare avviare i server mysql e apache in locale, poichè le porte usate da loro sono già occupate e usate da altri servizi (ad. la porta 80 in genere usata da mysql, già usata da skype) si puo usare il servizio on line www.altervista.org che dopo opportuna registrazione, fornisce la possibilità di attivare un database e conseguentemente di usare l'applicazione phpmyadmin on line (vedi fig.). **Creato l'account ascuoladisql, pwd: Ascuoladisql2, basta fare click sul pulsante "Attiva il database", confermare "ottieni gratis" e appare il link a phpmyadmin.**



[1] o dal sito https://sourceforge.net/projects/xampp/files/XAMPP/Windows/5.6.24/

Facendo click sul link phpMyAdmin che appare si accede all'interfaccia tipica di phpMyadmin. **Per iniziare bisogna: selezionare il dababase unico assegnato, in tal caso my_ascuoladisql e, successivamente scrivere nella TAB sql, cerchiata di blu:**

Per spiegare l'uso di phpMyAdmin e il linguaggio SQL, partiremo da un modello logico relazionale costituito da due semplici tabelle legate da una relazione 1: N[2], ricavato dal seguente modello E/R

Primi passi con l'applicazione web, phpMyAdmin

Aperto phpMyAdmin, bisogna creare: un database e le tabelle di cui esso è costituito.

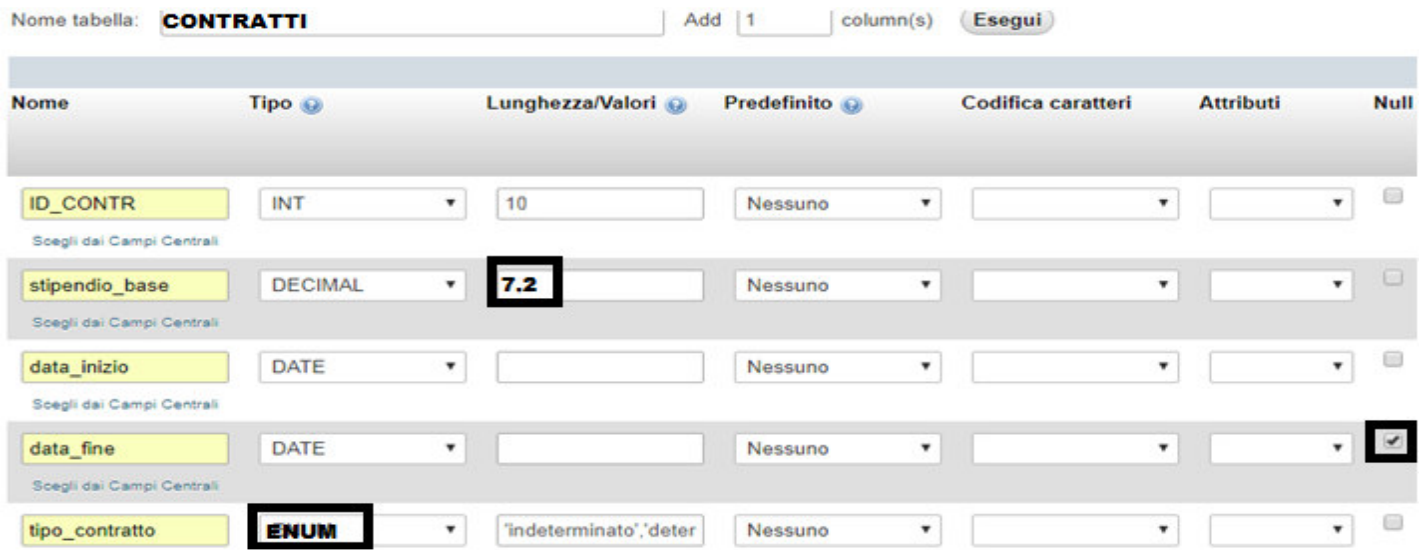
Per creare il database, come mostra la fig. basta selezionare la **"tab database"** e inserire un nome a scelta nel rigo bianco, ad es. DB-PROVA. Poi bisogna scegliere la codifica dei caratteri[3] (ad es. un latin) e infine cliccare sul pulsante crea, in modo che appaia la figura mostrata su a sinistra[4].

In tale figura, appare nel frame sinistro il db-prova appena creato e cliccandoci sopra, appare il frame di destra in figura in cui possiamo definire il nome della prima tabella denominata, **contratti**, e anche il numero dei campi che essa dovrà avere: **cinque** (in riferimento al modello logico ricavato dal modello e/r suddetto.) Facendo click su "**esegui"** appare la visualizzazione struttura della tabella da compilare seguente:

[2] La chiave esterna è sempre inserita nella tabella lato N ed è collegata alla chiave primaria inserita nella tabella lato 1.

[3] rappresentazione dei caratteri che possiamo usare nell'inserimento dei dati nel database, noi useremo la "latin_swedish_ci, poiché La codifica *Latin* gestisce i caratteri dell'alfabeto latino, che sono quelli predominanti nel mondo occidentale. Usando questa codifica, chiaramente un utente russo, non riuscirà a decifrare i caratteri usati da noi.

[4] Per creare il database con il codice sql bastera scrivere nella tab sql: **create database nome_database;**



Osservazioni:
- ID_CONTR è di tipo intero con **10 cifre** che poi sarà modificata per diventare chiave primaria
- stipendio_base è di tipo **decimal 7.2** ovvero 7 cifre in totale con 5 per la parte intera e 2 per la parte decimale
- data_fine è di tipo date ma con **l'opzione null attiva** (poiché la data fine rapporto può essere mancante)
- tipo_contratto essendo un insieme di valori finiti è **tipo enum** (cioè enumerativo o insiemistico). Dichiarato tipo_contratto di tipo ENUM è necessario inserire i valori dell'insieme enum, cliccando sul link "modifica i valori ENUM" altrimenti da errore o inserendo i valori tra apici e separati da virgole.

Adesso basta digitare il pulsante "salva" e si avrà, nel tab struttura, la tabella creata ovvero la figura seguente:

Adesso non rimane che modificare il campo ID_CONTR, facendolo diventare chiave primaria, basterà cliccare sul link "primaria" corrispondente al campo ID_CONTR. Ma noi vogliamo creare questa tabella e quelle seguenti con il codice SQL, pertanto prima di procedere, ricordiamo quanto segue:

Caratteristiche generali del linguaggio SQL

Il linguaggio SQL è il linguaggio usato per la gestione dei database relazionali, cioè dei database creati con un DBMS di tipo relazionale. Esso nacque nella seconda metà degli anni '70 ad opera di IBM e successivamente nacquero altre versioni dell'SQL ad opera di altre aziende e tutte diverse le une dalle altre. Successivamente nel 1986 l'**ANSI** (**A**merican **N**ational **S**tandard **I**stitute) definì un linguaggio standard[1] che prese il nome di SQL/86 e successivamente gli standard SQL/89 e SQL/92. **Tale linguaggio standard** ha come obiettivo quello di fornire alle diverse aziende un linguaggio comune da usare per creare database compatibili con i diversi DBMS in circolazione come: Access SQL, MySQL, SQL Informix, DBII SQL, Postgre SQL, Oracle, ecc.

A che serve il linguaggio SQL?
Esso serve a definire la **struttura delle tabelle** di un database, sfruttando i comandi del linguaggio: **DDL** (Data Definition Language); a **manipolare i dati** contenuti nel database, sfruttando i comandi del **DML** (Data Manipulation Language)*;* a porre **interrogazioni** al database sfruttando i comandi del **QL** (Query Language) e, infine, a **gestire gli accessi e i permessi** per gli utenti, sfruttando i comandi del **DCL** (Data Control Language). Esso, quindi, sembrerebbe che serve a fare le stesse cose che ad es. il DBMS Access ci permette di fare. In realtà non è così.

Motori di MySql usati: Innodb e MyIsam
1) InnoDB è un motore per il salvataggio di dati (Storage Engine) per MySQL, fornito in tutte le sue distribuzioni. La sua caratteristica principale è quella di supportare le transazioni

2) MyISAM utilizza la struttura ISAM e deriva da un tipo più vecchio, oggi non più utilizzato, che si chiamava appunto ISAM. È un motore di immagazzinamento dei dati estremamente veloce e richiede poche risorse, sia in termini di memoria RAM, sia in termini di spazio su disco. Il suo limite principale rispetto ad alcuni altri SE, come InnoDB, consiste nel mancato supporto delle transazioni

Identificatori.
Essi sono i nomi delle tabelle e i nomi dei campi delle tabelle del database. Un identificatore è costituito da una sequenza di caratteri che deve iniziare con una lettera e può anche contenere il carattere _ ("trattino basso" o underline), come ad es. **:** Nome_Cognome**,** ecc.

Il tipo di un campo è l'insieme dei possibili valori che un campo di una tabella può assumere.

Tipo	Descrizione		Dimensione massima/format
BIGINT	*8 byte = intero lunghissimo*	**INTERI**	*da -2^{63} a $2^{63}-1$*
INT	*4 byte = intero lungo*		*da -2^{31} a $2^{31}-1$*
TINYINT	*intero ridotto*		*da –128 a +127*
DOUBLE	*8 byte = reale a doppia precisione*	**REALI**	*da ±2.225*10308 a ± 1.798*10308*
FLOAT	*reale a singola precisione*		*da ±1.176*10-38 a ± 3.403*1038*
DECIMAL	*4 byte = decimale memorizzato come stringa*		*da ±2.225*10-308 a ± 1.798*10308*
DATE	*una data in formato US (aaaa-mm-gg)*	**DATA**	*dal clic '1000-01-01' a '9999-12-31*
TIME	*un orario*		*formato HH:MM:SS*
DATETIME	una data con ora		formato aaaa-mm-gg hh:mm:ss
YEAR	*un anno*		formato AAAA
CHARACTER	*stringa a lunghezza fissa*	**STRINGA**	*da 0 a 255 caratteri*
VARCHAR	*stringa a lunghezza variabile*		*da 0 a 255 caratteri*
TEXT	campo testo a lunghezza fissa		da 0 a 65535 caratteri
MEDIUMTEXT	*campo testo (memo)*		*16 MB di caratteri –1*
BLOB	*immagini jpeg, bmp, gif (Binary Large*	**OGGETTI**	*fino a 64 KB*
MEDIUM BLOB	*Object), oppure file binary*		*fino a 16 MB*
LONG BLOB	*in esadecimale o di testo*		*circa 4 GB*

I comandi del linguaggio DDL (Data Definition Language): CREATE E ALTER

Il comando CREATE
In SQL le tabelle sono definite con il comando CREATE TABLE seguito dal nome della tabella e dall'elenco dei campi. Per ogni campo occorre specificare il nome e il suo tipo. La sua sintassi è la seguente:

CREATE TABLE 'nome_tabella'
('nome_campo1' tipo_campo1 clausola,
……
'nome_campon' tipo_campon clausola

Qui vicino all'ultimo campo non ci vuole la virgola

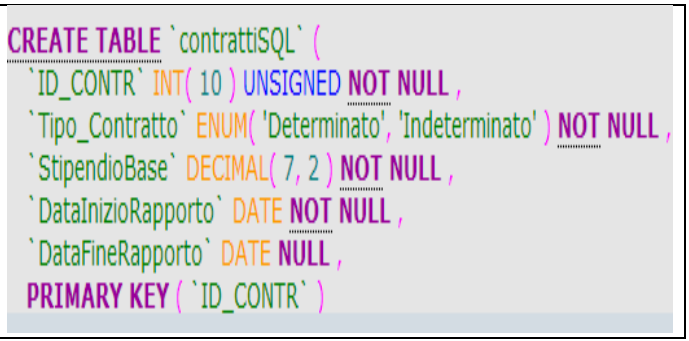
dove la coppia (nome-campo1 tipo-campo1), …., (nome-campon tipo-campon) sono i nomi dei campi e dei loro tipi e pertanto sono obbligatori, mentre la scritta "**clausola**" sottintende un valore facoltativo scelto tra i seguenti: **NULL, NOT NULL, DEFAULT, PRIMARY KEY, FOREGN KEY, UNIQUE O REFERENCE**.

Le clausole in dettaglio
- la clausola NULL è da attribuire ad un campo che può essere anche lasciato vuoto senza valore.
- la clausola NOT NULL definisce un campo che obbligatoriamente deve essere riempito.
- la clausola DEFAULT definisce il valore da attribuire al campo al momento della creazione.
- la clausola PRIMARY KEY definisce un campo chiave primaria
- la clausola UNIQUE vieta la presenza in un campo di valori duplicati.[5]
- la clausola REFERENCE definisce un vincolo di integrità referenziale.
- la clausola AUTO_INCREMENT definisce un numero progressivo attribuito automaticamente dal DBMS.
- la clausola PREDEFINITO definisce un valore iniziale che avrà il campo

Alcune precisazioni:
- Le stringhe sono delimitate da apice singolo oppure doppio quindi "ciao" e 'ciao' sono equivalenti.
- Il valore NULL è differente da 0. Esso attesta che il campo può avere valore mancante
- Le costanti numeriche possono essere precedute o meno dal segno (+ o –).
- Le costanti di tipo data e ora vengono considerate come stringhe, quindi sempre racchiuse tra apici singoli
- MySql di default specifica la data cosi: anno-mese-giorno. Per avere il formato data compatibile con quello europeo (giorno-mese-anno) anziché quello americano (anno-mese-giorno) è sufficiente intervenire nel file My.ini modificando la variabile date_format come segue: date_format=%d-%m-%y

Esercizio1: Creazione tabella ContrattiSQL, usando il linguaggio SQL e l'applicazione phpMyAdmin

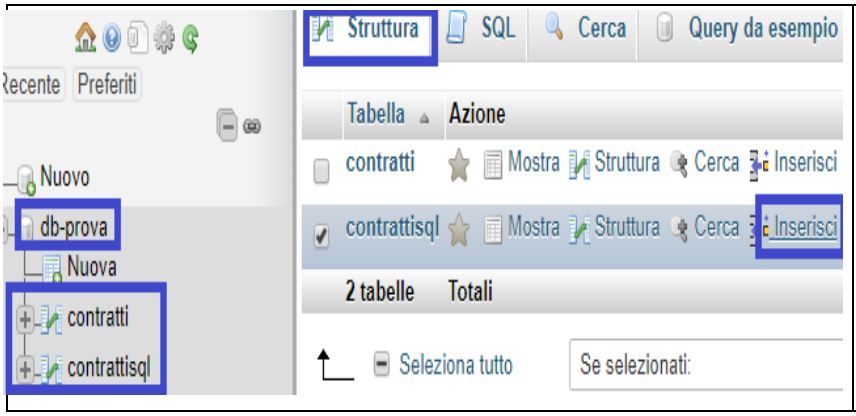
```
CREATE TABLE `contrattiSQL` (
`ID_CONTR` INT(10) UNSIGNED NOT NULL,
`Tipo_Contratto` ENUM('Determinato','Indeterminato')
 NOT NULL,
`StipendioBase` DECIMAL(7,2) NOT NULL,
`DataInizioRapporto` DATE NOT NULL,
`DataFineRapporto` DATE NULL,
 PRIMARY KEY (`ID_CONTR`)
);
```



[5] Ad esempio nel campo username, poiché le username sono tutte diverse tra loro chiediamo che i valori di tale campo siano unici. In questo modo non diamo la possibilità a due utenti che si registrano al database di avere due username uguali, mostrando un messaggio di errore se accadesse.

Eseguiamo tale query con phpMyAdmin, quindi, apriamo la struttura della tabella contrattiSql creata seguente:

Notiamo che: è stata definita il campo chiave primaria e un campo con valore possibile mancante "DataFineRapporto". Se non si dovesse vedere la colonna primaria, essa può essersi nascosta sotto la voce "Piu" per mancanza di spazio sullo schermo (in alternativa si può anche allargare il frame di destra)
Nell'elenco a sinistra della figura sotto, contenente l'elenco dei database e, precisamente nel database db-prova, ci sono le due tabelle create: contratti e contrattiSql come mostra la figura seguente:



La tabella contratti è stata creata in modo visuale

La tabella contrattiSQL con il codice SQL

Per popolare la tabella contrattiSQL basterà fare click sul link "inserisci" oppure usare l'opportuno comando SQL: INSERT INTO

Esercizio2: Creazione della tabella Dipendenti con SQL:

Cod_Dip	Cognome	Nome	Data_Nascita	Cod_Contratto

```
CREATE TABLE `Dipendenti` (
   `Cod_Dip` INT(10) NOT NULL,
   `NomeCognome` VARCHAR(60) NOT NULL,
   `Data_Nascita` DATE NOT NULL,
   `Qualifica`ENUM('Operaio','Impiegato','Dirigente')
    NOT NULL,
   `Cod_Contratto` INT(10) UNSIGNED NOT NULL,

   PRIMARY KEY AUTO_INCREMENT (`Cod_Dip`),
   FOREIGN KEY (`Cod_Contratto`) REFERENCES
   `contrattiSQL` (`ID_CONTR`) );
```

```
CREATE TABLE `Dipendenti` (
  `Cod_Dip` INT( 10 ) NOT NULL ,
  `NomeCognome` VARCHAR( 60 ) NOT NULL ,
  `Data_Nascita` DATE NOT NULL ,
  `Qualifica` ENUM( 'Operaio', 'Impiegato', 'Dirigente' ) NOT NULL ,
  `Cod_Contratto` INT( 10 ) UNSIGNED NOT NULL ,
  PRIMARY KEY AUTO_INCREMENT ( `Cod_Dip` ) ,
  FOREIGN KEY ( `Cod_Contratto` ) REFERENCES `contrattiSQL` (
     `ID_CONTR`
  )
```

Osservazione:
- Tra le tabelle contratti e dipendenti c'è una associazione 1: N, cioè un contratto della tabella Contratti e associato ad uno o più dipendenti della tabella Dipendenti e viceversa.

- Nella creazione della tabella Dipendente, oltre a definire i campi della tabella Dipendente (associato al modello E/R di pag. 2), dobbiamo inserire anche un campo aggiuntivo, detto Cod_Contratto che permetta di realizzare la relazione 1:N tra esse e definito chiave esterna.

Dalla barra delle tab, andando, sulla tab "Più" e cliccando su Designer appare il modello relazionale:

Altre osservazioni sulla query Sql CREATE TABLE Dipendenti
- la chiave esterna (in inglese "foreign key") è collegata (in inglese "references") alla tabella ContrattiSQL tramite il campo chiave ID_CONTR.
- il campo chiave primaria Cod_Dip della tabella Dipendenti, avrà la clausola auto_increment cioè avrà un numero progressivo che partirà da 1 e si auto inserirà.

Per avere un ulteriore conferma che la query eseguita per creare dipendenti, ha effettivamente creato la struttura della tabella Dipendenti, basta andare nel tab Struttura e vedere se esiste la tabella dipendenti creata:

Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito	Extra	Azione
Cod_Dip	int(10)			No	*Nessuno*		Modifica Elimina Primaria Più
NomeCognome	varchar(60)	latin1_swedish_ci		No	*Nessuno*		Modifica Elimina Primaria Più
Data_Nascita	date			No	*Nessuno*		Modifica Elimina Primaria Più
Qualifica	enum('Operaio', 'Impiegato', 'Dirigente')	latin1_swedish_ci		No	*Nessuno*		Modifica Elimina Primaria Più
Cod_Contratto	int(10)		UNSIGNED	No	*Nessuno*		Modifica Elimina Primaria

Dalla figura precedente si può notare come solo il campo ID_Dip ha l'attributo chiave primaria abilitato, poiché esso è l'unico campo definito chiave primaria in fase di creazione della tabella, inoltre tutti i campi

ID_Dip, NomeCognome, Data_Nascita, Qualifica e Cod_Contratto hanno valore No nell'attributo NULL, confermando la richiesta NON NULL nella stringa SQL per tali campi.

Il comando ALTER

Esso permette di modificare ("ALTERare") la struttura di una tabella in un secondo tempo. Tramite il comando ALTER potremo effettuare diverse modifiche come: rinominare una tabella (RENAME); cambiare nome ad un campo (CHANGE); aggiungere ai campi già presenti indici come: index, unique, primary key, ecc (ADD); cancellare campi e tabelle (DROP); modificare le caratteristiche dei campi (come il tipo di dato o la dimensione del tipo) (MODIFY); ecc.

La sintassi è: ALTER TABLE 'nome tabella' CLAUSOLA 'nome_campo' tipo_campo;	**Esempio: per aggiungere una campo in fondo** **ALTER TABLE `dipendenti`** **ADD `e_mail` VARCHAR(30) NOT NULL ;**

ESERCIZIO 3

Per aggiungere una campo in testa alla tabella ALTER TABLE 'nome_tabella' ADD 'campo' VARCHAR(10) FIRST;	**Esempio: per aggiungere un campo in testa** **ALTER TABLE `dipendenti`** **ADD `cap` VARCHAR(5) FIRST;**
Per aggiungere una campo dopo un altro ALTER TABLE 'nome_tabella' ADD 'campo1' VARCHAR(5) AFTER 'campo2';	**Esempio: per aggiungere un campo dopo un altro** **ALTER TABLE `dipendenti`** **ADD `cap` VARCHAR(5) AFTER 'DataNascita;**
Per aggiungere piu di un campo in fondo ALTER TABLE 'nome_tabella' ADD 'campo1' VARCHAR(6), ADD 'campo2' INT(10);	**Esempio: per aggiungere piu campi in fondo** **ALTER TABLE `dipendenti`** **ADD `e_mail` VARCHAR(30) NOT NULL ,** **ADD `indirizzo` VARCHAR (25) NOT NULL ;**
Per eliminare un campo dalla tabella ALTER TABLE 'nome_tabella' DROP 'nome_campo';	**Esempio: per eliminare un campo dalla tabella** **ALTER TABLE `dipendenti`** **DROP `e_mail`;**

Per rinominare una tabella ALTER TABLE 'nome_tabella' RENAME TO 'nuovo_nome_tabella';	**Esempio: per rinominare una tabella** **ALTER TABLE `dipendenti`** **RENAME `impiegati `;**
Per rinominare un campo di una tabella	**Esempio: per rinominare il campo**
ALTER TABLE 'nome_tabella' CHANGE 'campo' 'nuovo_campo' TIPO ;	**ALTER TABLE `dipendenti`** **CHANGE `Cod_Dip ` 'ID_Dip' INT (10) ;**
Per aggiungere l'indice primary key ALTER TABLE 'nome_tabella' ADD PRIMARY KEY ('nome_campo');	**Esempio: aggiungere l'indice Primary Key** **ALTER TABLE** `dipendenti` **ADD PRIMARY KEY** (`Cod_Dip`) ;
Per aggiungere l'indice UNIQUE ALTER TABLE 'nome_tabella' ADD UNIQUE ('nome_campo');	**Esempio: aggiungere l'indice UNIQUE** **ALTER TABLE** `dipendenti` **ADD UNIQUE** (`E-MAIL `) ;

I comandi del linguaggio DML: INSERT, UPDATE E DELETE

I valori dei campi nei record della tabella possono essere inseriti, aggiornati o cancellati rispettivamente con i comandi **INSERT, UPDATE E DELETE**.

Data la struttura a fianco della tabella contrattiSQL, in phpMyAdmin, selezionato il dabase DB-prova, basta scrivere nella TAB SQL, il codice:

```
INSERT INTO `db-prova`.`contrattisql`
(`ID_CONTR`,`Tipo_Contratto`,`StipendioBase`,`DataInizioRapporto`,
`DataFineRapporto`) VALUES
('1', 'indeterminato', '1600.35', '2005-09-01', NULL);
```

Tale metodo consiste nello scrivere in ordine i campi definiti nella tabella e in corrispondenza di tali campi, bisogna inserire i valori corrispondenti. Comunicando a MySql: il database, la tabella del database su cui stiamo lavorando, MySql viene a sapere i nomi dei campi che costituiscono la tabella e la loro posizione, quindi, per l'inserimento di nuovi record, basterà utilizzare il seguente metodo piu veloce:

```
INSERT INTO `db-prova`.`contrattisql`
VALUES ('2', 'determinato', '1250.45', '2017-10-03', '2017-06-30');
```

Avendo definito la tabella contrattiSql con solo due tipi di contratti, essa conterrà solo i due record inseriti.

Dopo questi due inserimenti la tabella contrattiSql diventa cosi:

ID_CONTR	Tipo_Contratto	StipendioBase	DataInizioRapporto	DataFineRapporto
1	Indeterminato	1600.35	2005-09-01	*NULL*
2	Determinato	1250.45	2017-10-03	2017-06-30

Per il contratto a tempo determinato essendo a termine, ha senso prevedere la data di fine rapporto. Per il contratto a tempo indeterminato no.

Esercizio:4 Inseriamo alcuni record nella tabella Dipendenti

```
INSERT INTO `db-prova`.`dipendenti`
(`Cod_Dip` , `NomeCognome` , `Data_Nascita` , `Qualifica` , `Cod_Contratto`)
VALUES ('1', 'Mario Rossi', '2000-09-01', 'Operaio', '1');
```

```
INSERT INTO `db-prova`.`dipendenti`
(`Cod_Dip` , `NomeCognome` , `Data_Nascita` ,`Qualifica` , `Cod_Contratto`)
VALUES ( '2', 'Giuseppe Festa', '2001-09-01', 'Impiegato', '1');
```

```
INSERT INTO `db-prova`.`dipendenti`
(`Cod_Dip` , `NomeCognome` , `Data_Nascita` ,`Qualifica` , `Cod_Contratto`)
VALUES ( '3', 'Lucia Narducci', '1972-05-02', 'Dirigente', '1');
```

```
INSERT INTO `db-prova`.`dipendenti`
(`Cod_Dip` , `NomeCognome` , `Data_Nascita` , `Qualifica` , `Cod_Contratto`)
VALUES ('4', 'Maria Scirocco', '1992-03-23', 'Impiegato', '2');
```

```
INSERT INTO `db-prova`.`dipendenti`
(`Cod_Dip` , `NomeCognome` , `Data_Nascita` , `Qualifica` , `Cod_Contratto`)
VALUES ('5', 'Giovanni Di Domenico', '2000-06-22', 'Operaio', '2');
```

```
INSERT INTO `db-prova`.`dipendenti`
(`Cod_Dip` , `NomeCognome` , `Data_Nascita` , `Qualifica` , `Cod_Contratto`)
VALUES ('6', 'Giancarlo Giaquinto', '2001-04-12', 'Impiegato', '1');
```

POSSIBILI ERRORI NELL'INSERIMENTO DEI DATI

#1062 - Duplicate entry '5' for key 'PRIMARY'

Esso è l'errore che si ha quando andiamo ad inserire un record nella tabella Dipendenti, individuato da un valore di chiave primaria già usato nella tabella, ovvero duplicato. Il tal caso la clausola Primary Key stabilita per il campo Cod_Dip, nella stringa SQL che crea la tabella Dipendenti, interviene segnalandoci questo errore commesso nell'inserimento di un nuovo record.

#1452 - Cannot add or update a child row: a foreign key constraint fails (`db-prova`.`dipendenti`, CONSTRAINT dipendenti_ibfk_1` FOREIGN KEY (`Cod_Contratto`) REFERENCES `contrattisql` (`ID_CONTR`))

Esso è l'errore che si ha quando andiamo ad inserire un record nella tabella Dipendenti, il cui valore del campo chiave esterna Cod_Contratto non è associato a nessun valore del corrispondete campo ID_CONTR della tabella Contratti. In pratica se la tabella Contratti è la seguente:

ID_CONTR	Tipo_Contratto	StipendioBase	DataInizioRapporto	DataFineRapporto
1	Indeterminato	1600.35	2005-09-01	NULL
2	Determinato	1250.45	2017-10-03	2017-06-30

Noi non possiamo inserire nella tabella Dipendenti un record avente un valore, in tal caso, 3, del campo chiave esterna Cod_Contratto, diverso da 1 e 2 del corrisponde campo ID_CONTR**.**

Per quanto su detto la stringa SQL che genera l'errore suddetto è il seguente:

```
INSERT INTO `db-prova`.`dipendenti` (`Cod_Dip`,`NomeCognome`, `Data_Nascita`,`Qualifica`,`Cod_Contratto`)
VALUES ('7', 'GianMarco Tognazzi', '1973-04-12', 'Impiegato', '3');
```

Inseriti i sei record nella tabella Dipendenti, se non ci sono errori, essa diventa cosi popolata:

db-prova » dipendenti

Struttura | SQL | Cerca | Inserisci | Esporta | Importa | Pi

	Cod_Dip	NomeCognome	Data_Nascita	Qualifica	Cod_Contratto
Copia Elimina	1	Mario Rossi	2000-09-01	Operaio	1
Copia Elimina	2	Giuseppe Festa	2001-09-01	Impiegato	1
Copia Elimina	3	Lucia Narducci	1972-05-02	Dirigente	1
Copia Elimina	4	Maria Scirocco	1992-03-23	Impiegato	2
Copia Elimina	5	Giovanni Di Domenico	2000-06-22	Operaio	2
Copia Elimina	6	Sandra Cafiero	2001-04-12	Impiegato	1

Ci sono valori duplicati nel campo Cod_Contratto

Osservazione:

Nella tabella dipendenti il campo chiave primaria è solo Cod_Dip quindi i suoi valori sono tutti diversi (vedi fig.), mentre nel campo Cod_Contratto i valori possono essere duplicati. Ad es. I record 1, 2, 3 e 6 hanno tutti un valore 1 nel campo Cod_Contratto e ciò significa solo che tutti e quattro i dipendenti selezionati hanno un contratto a "tempo indeterminato".

Osservazione:
Chiaramente, il popolamento di una tabella è possibile farla più velocemente attraverso l'interfaccia grafica di phpMyAdmin. In tal caso basta selezionare il nome del database nel frame di sinistra di phpmyadmin, selezionare nel frame a destra, la tabella dipendenti e poi la tab inserisci. Riempire tutti i campi e poi digitare il pulsante esegui.[6]

Il comando UPDATE del linguaggio DML
Esso permette di aggiornare il valore di un campo appartenente ad un record. **La sintassi SQL è:**
UPDATE 'nome_database'. 'nome_tabella'
SET 'nome_campo' = 'valore' WHERE 'nome_tabella'. 'nomecampo' = valore;

Esercizio 5.
1) Aggiornare a 'Sandra Cafiero' il campo NomeCognome del record con chiave primaria Cod_Dip=6.

```
UPDATE `db-prova`.`dipendenti`
SET `NomeCognome` = 'Sandra Cafiero'  WHERE`dipendenti`.`Cod_Dip` =6;
```

2) Aggiornare a 'Operaio' il campo Qualifica del record con chiave primaria Cod_Dip=1

```
UPDATE `db-prova`.`dipendenti`
SET `Qualifica` = 'Impiegato' WHERE `dipendenti`.`Cod_Dip` =1
```

Se serve aggiornare piu campi di un record conviene farlo graficamente, con l'interfaccia phpMyAdmin.

1) modo: cliccando all'interno della cella "cerchio blu" contenente il valore da cambiare **2) modo**: cliccando sul link modifica cerchio rosso	

Il comando UPDATE del linguaggio DML
Esso permette di aggiornare un record di una tabella. La sintassi SQL è la seguente:
DELETE FROM `nome_database`.`nome_tabella` **WHERE** `nome_tabella. `.`nome_campoID ` = 7

Esercizio 6.
Cancellare da db-prova, tabella Dipendenti, il record identificato dal campo chiave =6.
DELETE FROM **`db-prova`.`dipendenti`** WHERE **`dipendenti`.`Cod_Dip` =7**

Se serve cancellare piu record conviene farlo con phpMyAdmin, selezionando il link Elimina

I comandi del linguaggio QL
Essi sono i comandi usati per porre delle interrogazioni ad un database dopo che è stato definito e popolato.

Il comando SELECT
Esso permette di selezionare i record di una tabella che soddisfano a nessuna, una o piu condizioni. La sintassi SQL è la seguente: **SELECT** 'nome_campo' **FROM** 'nome_tabella' **WHERE** condizione=valore[7]

[6] Il canale youtube WeBSCHOOL condivide dei video su PhpMyAdmin: https://www.youtube.com/watch?v=6fjY- GaEvM&t=59s per l'introduzione a phpmyadmin e i successivi video #2 e #3 per la creazione e l'inserimento dei dati nelle tabelle in modo visuale

[7] se il valore è intero non prende gli apici, altrimenti se il valore è di testo prende gli apici

Esercizio 7

1) Seleziona solo il campo NomeCognome dalla tabella dipendenti[8]

nessuna condizione

```
SELECT NomeCognome  FROM dipendenti;
```

2) Seleziona NomeCognome da Dipendenti che sono impiegati

```
SELECT NomeCognome  FROM `dipendenti`
where  `Qualifica` = 'Impiegato';
```

(una condizione)

NomeCognome
Giuseppe Festa
Maria Scirocco
Sandra Cafiero

3) Selezione record che soddisfano a due condizioni (AND)

```
SELECT NomeCognome, StipendioBase
FROM Dipendenti, Contrattisql
WHERE Dipendenti.Cod_Contratto = Contrattisql.ID_CONTR
AND StipendioBase >1200 ;
```

NomeCognome	StipendioBase
Mario Rossi	2050.45
Giuseppe Festa	2050.45
Lucia Narducci	2050.45
Sandra Cafiero	2050.45
Maria Scirocco	1250.45

4) Seleziona tutti i dipendenti che hanno un contratto a termine[9]

```
SELECT NomeCognome FROM Dipendenti, Contrattisql
WHERE Dipendenti.Cod_Contratto = Contrattisql.ID_CONTR
AND DataFineRapporto IS NOT NULL;
```

NomeCognome
Maria Scirocco
Giovanni Di Domenico

5) Seleziona tutti i dipendenti nati prima del 1 gennaio 2000

```
SELECT *  FROM Dipendenti
WHERE Data_Nascita < '2000-01-01';
```

NomeCognome	Data_Nascita	Qualifica
Lucia Narducci	1972-05-02	Dirigente
Maria Scirocco	1992-03-23	Impiegato

6) Selezionare tutti i contratti con stipendio base tra 500€ e 1500€

```
SELECT DataFineRapporto, StipendioBase
FROM Contrattisql
WHERE StipendioBase BETWEEN 500 AND 1500;
```

DataFineRapporto	StipendioBase
2017-06-30	1250.45

Le funzioni di aggregazione in SQL

Esso sono funzioni standard native di SQL che permettono di ottenere valori numerici e/o effettuare calcoli su un insieme di tuple (record di una tabella). Esempi di esse sono:

Funzione	Descrizione
AVG( )	Restituisce la media tra due valori specificati
COUNT( )	Restituisce un intero che indica il numero di record trovati.
MAX( )	Restituisce il valore massimo tra due valori
MIN( )	Restituisce il valore minimo tra due valori
SUM( )	Restituisce la somma tra più record dello stesso campo

[8] Non essendoci condizioni, saranno selezionati i nomi e cognomi di tutti dipendenti inseriti nel database.
[9] essi sono quei dipendenti che hanno nella tabella contrattiSql, il valore non nullo della data di fine rapporto.

Nell'utilizzo di una funzione di aggregazione è importante (consigliato, anche se non obbligatorio) specificare un *campo* alias per il risultato, con l'utilizzo della clausola **AS**. I nostri alias li personalizzeremo in base al calcolo da effettuare oppure li potremmo chiamare **temp,** per sottolineare che sono campi temporanei.

Esercizio 8

1) Calcola la somma di tutti gli stipendi erogati

```
SELECT SUM( StipendioBase ) AS TotaleStipendi
FROM Dipendenti, ContrattiSql
WHERE Dipendenti.Cod_Contratto = Contrattisql.ID_CONTR;
```

TotaleStipendi
10702.70

Esso è quel campo temporaneo detto in Sql Alias, (in Access campo calcolato)

2) Calcola la somma degli stipendi dei dipendenti con qualifica impiegato

```
SELECT SUM( StipendioBase ) AS TotaleStipendiImpegato
FROM Dipendenti, ContrattiSql
WHERE Dipendenti.Cod_Contratto=ID_CONTR AND Qualifica='Impiegato';
```

TotaleStipendiImpiegato
5351.35

Esso è un altro campo Alias, creato al momento per contenere il risultato della query.

3) Calcola il valore massimo del campo cod_dipendente

```
SELECT MAX( Cod_dip ) AS temp  FROM Dipendenti;
```

temp
6

usato come campo Alias, il nome temp. Esce 6 poiché nella tabella sono 6 i dipendenti inseriti e 6 è il valore massimo.

4) Calcola la media degli stipendi erogati

```
SELECT AVG (StipendioBase ) AS MediaStipendi
FROM Dipendenti, ContrattiSql
WHERE Dipendenti.Cod_Contratto = Contrattisql.ID_CONTR;
```

MediaStipendi
1783.783333

Esso è un altro campo alias temporaneo. E' la media di 6 stipendi base.

5) Calcola quanti impiegati ci sono tra i dipendenti

```
SELECT COUNT( * ) AS NumeroImpiegati
FROM DIPENDENTI
WHERE QUALIFICA = 'impiegato';
```

NumeroImpiegati
3

Infatti, basta controllare che nella tabella dipendenti ci siano solo 3 con la qualifica di impiegato