Università degli Studi di Milano Bicocca
Corso di Laurea in Informatica

Data Definition in MySQL

LABORATORIO DI BASI DI DATI
A.A. 2019/2020

Dott. Marco Savi

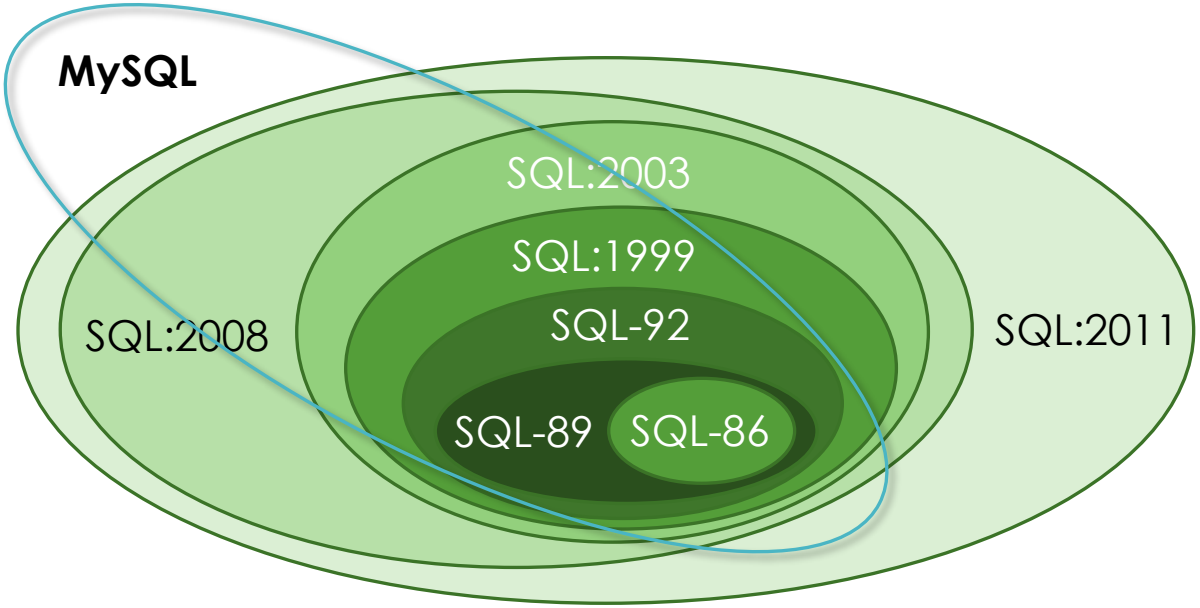
Contenuti riadattati a partire da slide gentilmente concesse
dai **Dott. Paolo Napoletano** e **Claudio Venturini**

SQL

- SQL è un **linguaggio standard** per la definizione e l'interrogazione di database relazionali
- Inizialmente sviluppato da IBM negli anni '70, e implementato per la prima volta nel 1979 da Oracle.
- Diventa uno standard ANSI nel 1986, e uno standard ISO nel 1987
- Diverse versioni:
  - SQL-86 (SQL 1), SQL-89, SQL-92 (SQL 2), SQL:1999 (SQL 3), SQL:2003, SQL:2008, SQL:2011
- Solitamente i DBMS implementano in parte lo standard e lo estendono per supportare sintassi differenti e funzionalità specifiche

MySQL e SQL

- MySQL implementa parte degli standard SQL, fino a SQL:2008
- MySQL estende SQL aggiungendo alcune funzionalità



- Per approfondire:
  - http://dev.mysql.com/doc/refman/5.7/en/compatibility.html
  - https://dev.mysql.com/doc/refman/5.7/en/differences-from-ansi.html

SQL DDL, DML, DCL

DDL – Data Definition Language

- Definizione e modifica dello schema del DB (db, tabelle, colonne, viste, …)
- Operazioni CREATE, ALTER, DROP

```
mysql> create table studente (matricola int, nome varchar(100));
mysql> drop table esame;
```

DML – Data Manipulation Language

- Interrogazione e modifica dei dati
- Operazioni **CRUD**: Create, Read, Update, Delete

```
mysql> select * from studente;
mysql> update studente set name = "Mario";
```

DCL – Data Control Language

- Controllo del DBMS e dei database

```
mysql> use univ;
mysql> show databases;
```

MySQL DCL

- Elenco dei database

```
> SHOW DATABASES;
> SHOW SCHEMAS;
```

- Selezione di un database

```
> USE univ;
```

- Elenco delle tabelle

```
> SHOW TABLES FROM univ;
> SHOW TABLES;
```

- Ispezione della struttura di una tabella

```
> DESCRIBE univ.studente;
> DESCRIBE studente;
> SHOW COLUMNS FROM studente;
> SHOW COLUMNS FROM studente FROM univ;
```

- Indici definiti su una tabella

```
> SHOW INDEX FROM univ.studente;
> SHOW INDEX FROM studente FROM univ;
> SHOW INDEX FROM studente;
```

MySQL DDL

- Statement `CREATE`, `ALTER` e `DROP` per database e tabelle

- Creazione di un database

```
> CREATE DATABASE univ;
> CREATE SCHEMA univ;

> CREATE DATABASE IF NOT EXISTS univ;
```

- Eliminazione di un database

```
> DROP DATABASE univ;
> DROP SCHEMA univ;
```

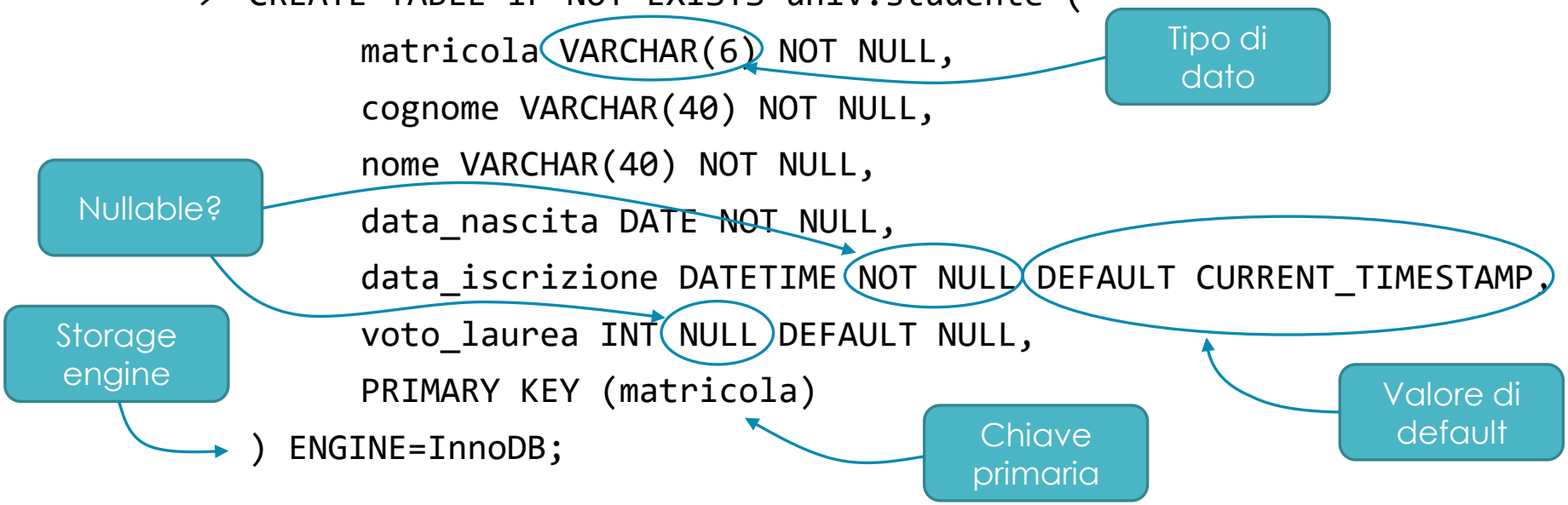
- Specificare il *character set* e la *collation*

```
> CREATE SCHEMA IF NOT EXISTS univ CHARSET='utf8';
> CREATE SCHEMA IF NOT EXISTS univ CHARSET='utf8' COLLATE='utf8_general_ci';
```

MySQL DDL – Creazione di tabelle

- Statement `CREATE TABLE`

```
> CREATE TABLE IF NOT EXISTS univ.studente (
      matricola VARCHAR(6) NOT NULL,
      cognome VARCHAR(40) NOT NULL,
      nome VARCHAR(40) NOT NULL,
      data_nascita DATE NOT NULL,
      data_iscrizione DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
      voto_laurea INT NULL DEFAULT NULL,
      PRIMARY KEY (matricola)
  ) ENGINE=InnoDB;
```



- Sintassi completa: http://dev.mysql.com/doc/refman/5.7/en/create-table.html
- **NB:** MySQL non è case-sensitive in Windows mentre lo è nella maggior parte dei sistemi operativi UNIX. Il consiglio è comunque di scrivere sempre dichiarazioni e clausole in maiuscolo, come da specifica SQL

MySQL – Tipi di dati principali [1]

- Tipi stringa
  - Lunghezza fissa: CHAR(N) (0-255 caratteri)
  - Lunghezza variabile: VARCHAR(N) (0-65.536 caratteri), TEXT, MEDIUMTEXT, LONGTEXT

- Tipi interi (*signed* e *unsigned*)
  - TINYINT (8 bit), SMALLINT (16 bit), MEDIUMINT (24 bit), INT (32 bit), BIGINT (64 bit)
  - UNSIGNED è usata come opzione per indicare numeri non negativi

- Tipi decimali (*signed* e *unsigned*)
  - Virgola mobile: FLOAT (32 bit), DOUBLE (64 bit)
  - Virgola fissa: DECIMAL(M,D)
    - M è la *precisione* e D la *scala*

MySQL – Tipi di dati principali [2]

- Tipi temporali
  - DATE, DATETIME, TIMESTAMP (data minima 1/1/1970, data massima 19/01/2038), YEAR
- Tipi binari
  - BIT (1-64 bit), BINARY(N) (0-255 byte), VARBINARY(N) (0-65536 byte), BLOB, LONGBLOB
- Enumerazioni
  - ENUM('valore1', 'valore2', 'valore3', …)

- **E i booleani?**
  - Non sono supportati, benché lo standard SQL definisca il tipo BOOLEAN
  - MySQL converte BOOLEAN in TINYINT(1)

MySQL – Auto increment

- Una colonna di tipo intero può essere dichiarata ad *incremento automatico* con la keyword **AUTO_INCREMENT**
  - Il suo valore viene automaticamente generato all'inserimento del record, tramite un contatore incrementale interno al DBMS associato alla tabella

```
> CREATE TABLE IF NOT EXISTS univ.corso (
        codice INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        nome VARCHAR(255) NOT NULL,
        …
  ) ENGINE=InnoDB;
```

  - Il primo corso avrà codice 1, il secondo 2, ecc...
  - Il valore iniziale del contatore può essere modificato

```
> ALTER TABLE univ.corso AUTO_INCREMENT = 100;
```

  - I codici che tornano disponibili dopo l'eliminazione di corsi *non vengono riusati*, a meno che il contatore non venga reimpostato al valore iniziale

MySQL – Vincoli di tupla

- Vincoli di unicità
  - Supponiamo che non possano esistere due studenti con lo stesso nome e cognome (ovviamente nella realtà non è così…):

```
> CREATE TABLE IF NOT EXISTS univ.studente (
        matricola VARCHAR(6) NOT NULL,
        cognome VARCHAR(40) NOT NULL,
        nome VARCHAR(40) NOT NULL,
        PRIMARY KEY (matricola),
        CONSTRAINT un_studente UNIQUE (cognome, nome)
    ) ENGINE=InnoDB;
```

  - Consentono di definire le chiavi (non primarie)
- MySQL non supporta altri vincoli di tupla né vincoli di dominio
  - I costrutti `DOMAIN` e `CHECK` di SQL non sono supportati

MySQL – Vincoli di integrità referenziale

- Ogni esame sostenuto da uno studente è relativo ad uno specifico corso

```
> CREATE TABLE IF NOT EXISTS univ.esame (
        codice INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        matricola VARCHAR(6) NOT NULL,
        codice_corso VARCHAR(40) NOT NULL,
        data VARCHAR(40) NOT NULL,
        voto TINYINT NOT NULL,
        CONSTRAINT un_esame UNIQUE (matricola, codice_corso, data),
        CONSTRAINT fk_studente
                FOREIGN KEY (matricola)
                REFERENCES univ.studente (matricola),
        CONSTRAINT fk_corso
                FOREIGN KEY (codice_corso)
                REFERENCES univ.corso (codice_corso)
  ) ENGINE=InnoDB;
```

MySQL – Propagation constraints

- Consentono di definire l'azione con cui propagare gli aggiornamenti riguardanti valori oggetto di un vincolo di integrità referenziale verso la tabella in cui il vincolo è definito (tabella dipendente)
  - Cosa succede alla tabella esame se lo studente cambia matricola?
  - Cosa succede alla tabella esame se lo studente viene eliminato?

Azione	In caso di modifica	In caso di eliminazione
`CASCADE`	Valore automaticamente aggiornato	Record automaticamente eliminato
`RESTRICT`	Se l'operazione viola l'integrità, viene negata	Se l'operazione viola l'integrità, viene negata
`NO ACTION`	Secondo lo standard SQL equivale a RESTRICT, ma l'integrità è verificata solo al termine dell'esecuzione della transazione (verifica ritardata). In realtà in MySQL ciò avviene immediatamente, per cui si comporta in maniera identica a RESTRICT	
`SET DEFAULT`	Valore impostato al valore di default della colonna	Valore impostato al valore di default della colonna
`SET NULL`	Valore impostato a NULL	Valore impostato a NULL

MySQL – Propagation constraints

- L'azione da eseguire è definita dai costrutti **ON UPDATE** e **ON DELETE**

```
…
CONSTRAINT fk_studente
        FOREIGN KEY (matricola)
        REFERENCES univ.studente (matricola)
        ON UPDATE CASCADE
        ON DELETE RESTRICT,
…
```

- In questo esempio…
  - se la matricola dello studente cambia, il valore della matricola nella tabella esame viene automaticamente aggiornato
  - se si tenta di eliminare lo studente prima di aver eliminato anche i suoi esami, il sistema impedisce l'operazione

MySQL DDL – Modifica di tabelle [1]

- Statement `ALTER TABLE`
  - Sintassi completa: http://dev.mysql.com/doc/refman/5.7/en/alter-table.html

- Aggiunta di colonne

```
> ALTER TABLE univ.studente
        ADD COLUMN codice_fiscale CHAR(16) NOT NULL AFTER cognome,
        ADD COLUMN laureato TINYINT(1) NOT NULL DEFAULT 0;
```

- Rimozione di colonne

```
> ALTER TABLE univ.studente DROP COLUMN codice_fiscale;
```

- Modifica di colonne (nome, tipo di dato, valore di default, ….)

```
> ALTER TABLE univ.studente
        CHANGE COLUMN matricola numero_matricola CHAR(7) NOT NULL,
        MODIFY COLUMN cognome VARCHAR(100) NOT NULL;
```

MySQL DDL – Modifica di tabelle [2]

- Impostazione ed eliminazione del valore di default

```
> ALTER TABLE univ.studente
        ALTER COLUMN laureato DROP DEFAULT,
        ALTER COLUMN codice_fiscale SET DEFAULT "000000000000";
```

- Definizione della chiave primaria

```
> ALTER TABLE univ.studente ADD PRIMARY KEY (matricola);
```

- Eliminazione della chiave primaria

```
> ALTER TABLE univ.studente DROP PRIMARY KEY;
```

- Aggiunta di un vincolo di unicità e di integrità referenziale

```
> ALTER TABLE univ.studente
        ADD CONSTRAINT un_codice_fiscale UNIQUE (codice_fiscale),
        ADD CONSTRAINT fk_citta
                FOREIGN KEY (cod_citta)
                REFERENCES citta (codice)
                ON UPDATE CASCADE ON DELETE SET NULL;
```

MySQL DDL – Modifica di tabelle [3]

- Eliminazione di un vincolo di unicità

```
> ALTER TABLE univ.studente DROP un_codice_fiscale;
```

- Eliminazione di un vincolo di integrità referenziale

```
> ALTER TABLE univ.studente DROP FOREIGN KEY fk_citta;
```

- Ridenominazione di una tabella

```
> ALTER TABLE univ.corso RENAME TO univ.insegnamento;
```

… oppure utilizzando lo statement `RENAME` …

```
> RENAME TABLE univ.corso TO univ.insegnamento;
```

MySQL DDL – Eliminazione tabelle e indici

- Eliminazione di una tabella

```
> DROP TABLE univ.insegnamento;
```

- Creazione di **indici**

I dettagli nel corso di ***Basi di Dati Complementi****!*

```
> ALTER TABLE univ.studente
        ADD INDEX idx_studente_codice_fiscale
        ON univ.studente (codice_fiscale);
```

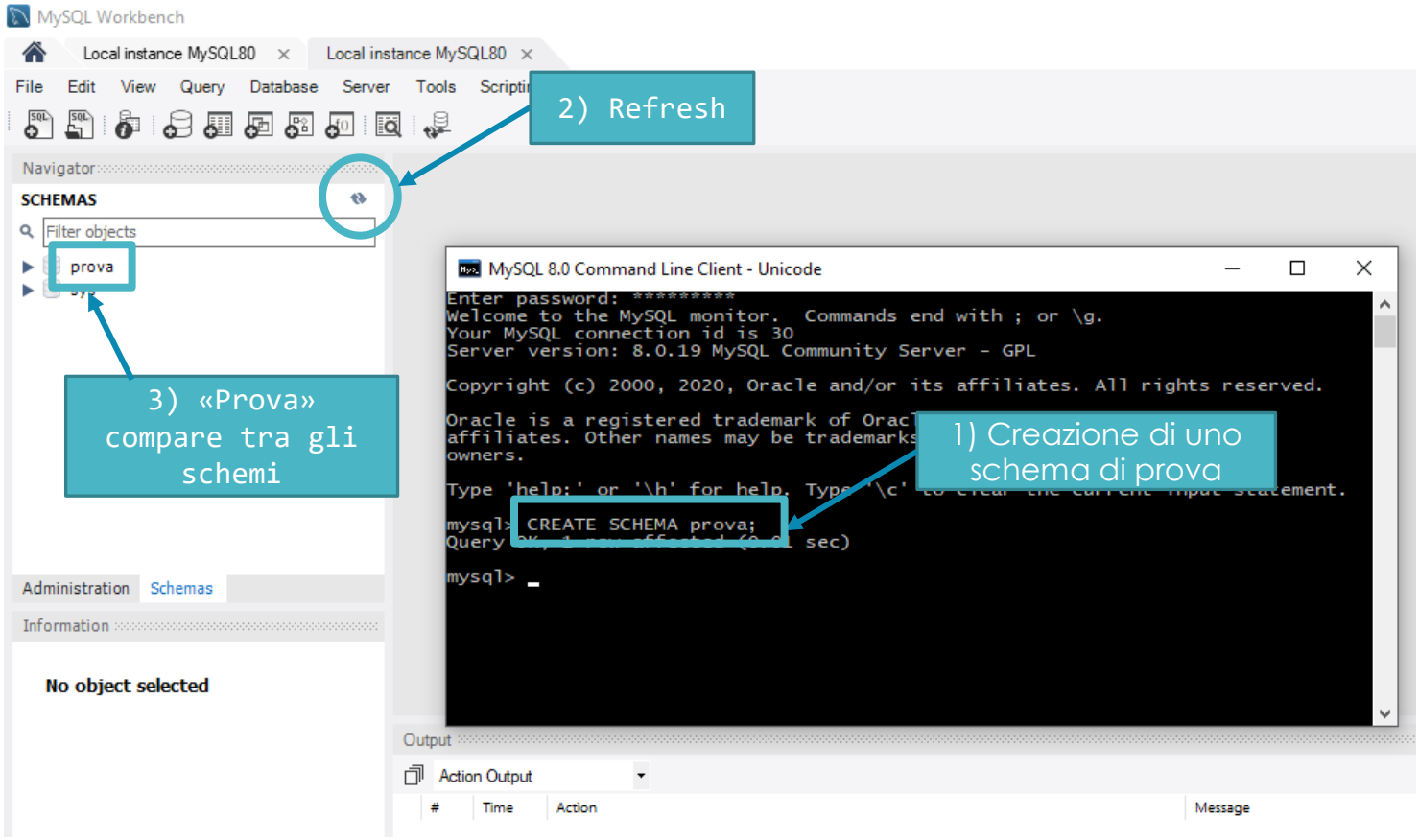
… oppure utilizzando lo statement `CREATE INDEX` …

```
> CREATE INDEX idx_studente_codice_fiscale
        ON univ.studente (codice_fiscale);

> CREATE UNIQUE INDEX idx_un_esame
        ON univ.esame (matricola ASC, codice_corso ASC, data ASC)
        USING BTREE
        LOCK DEFAULT;
```

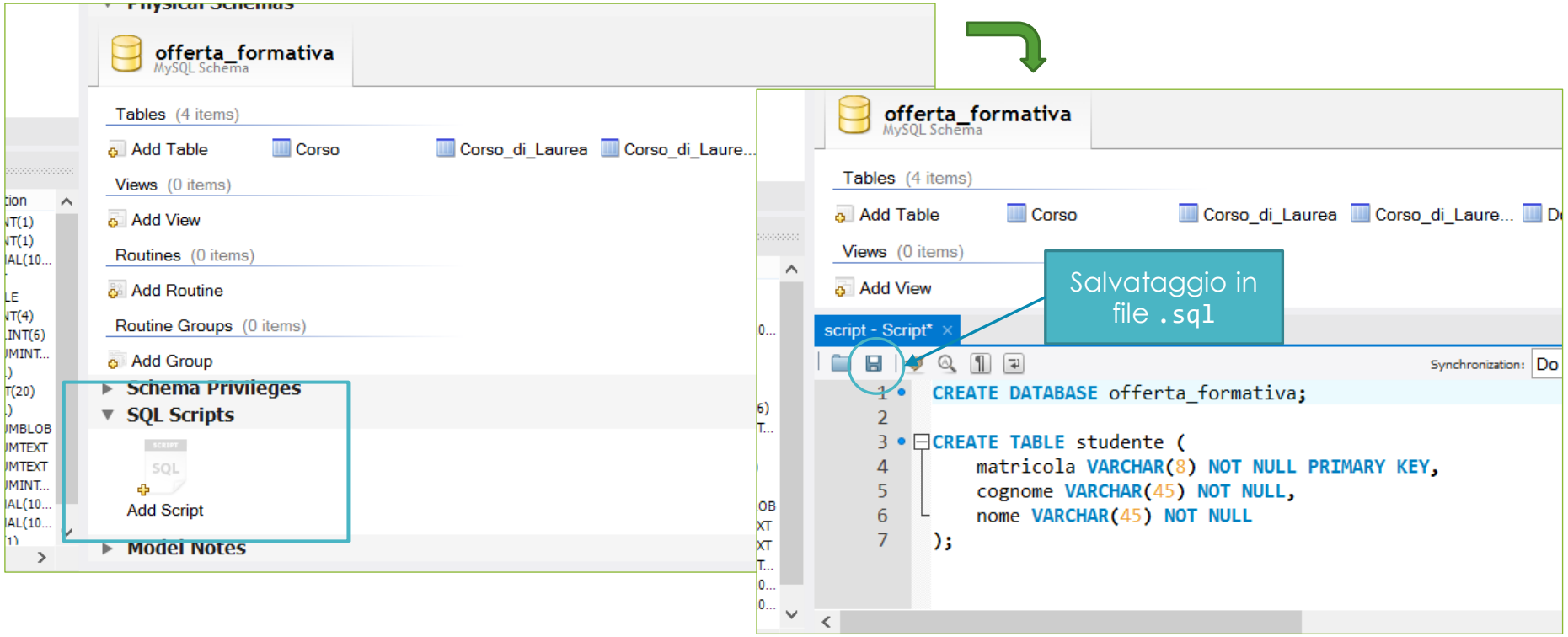
MySQL – Command Line Client

- Permette di accedere a un Server MySQL (locale o remoto) e di eseguire comandi per mezzo di Command Line Interface
- E' possibile verificare con Workbench la corretta esecuzione delle istruzioni eseguite tramite CLI



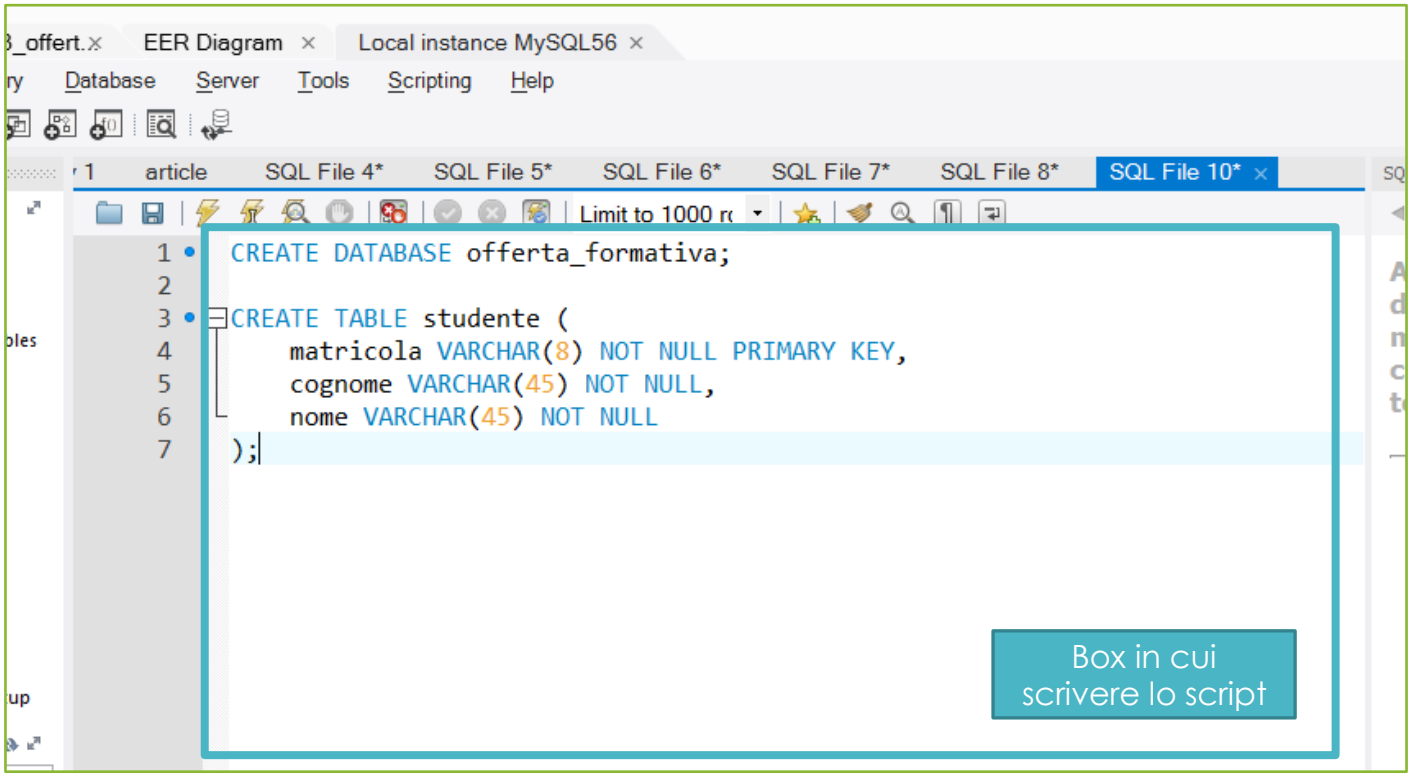
MySQL – SQL Script

- Uno script è un file di testo che contiene statement SQL
  - Utili per creare la struttura di un database o per manipolare i dati in batch
  - Uno script di creazione di un database contiene solo statement DDL
- MySQL Workbench consente di creare script nel file di modello



MySQL – SQL Script

- …oppure si possono creare script dopo aver stabilito una connessione ad un DBMS (locale o remoto)
  - File > New Query Tab



Esercizi

Esercizio preliminare

- Utilizzando il **client CLI** di MySQL, creare e manipolare lo schema del database progettato nell'Esercizio 1 del laboratorio precedente

- Effettuare almeno le seguenti operazioni:
  1. Creare il database, dando un nome a piacere
  2. Eseguire gli statement `CREATE TABLE` necessari per creare ogni tabella, *definendo le colonne, la chiave primaria, e i vincoli di integrità referenziale* (specificando anche le modalità di propagazione degli aggiornamenti).
  3. Utilizzando lo statement `ALTER TABLE`, aggiungere alla tabella *studente*:
     - la *data di laurea*, impostandola di default a `NULL`
     - Il *titolo della tesi*, impostandola di default a `NULL`
  4. Aggiungere alla tabella *esame* un campo booleano che indichi *se lo studente ha ottenuto un voto con lode* impostandolo di default a *falso*
  5. Rinominare a piacere una delle tabelle create
  6. Eliminare la tabella *esame*
  7. Eliminare l'intero database

Esercizio 3

- Progettare la seguente base di dati: **Offerta Formativa**
  - I dati rappresentati riguardano i corsi erogati e i docenti che li insegnano

  - Ogni **corso** può essere insegnato da un solo professore, ma un professore può erogare più corsi. Ogni professore insegna almeno un corso.
  - I corsi sono descritti con un codice, un nome, il numero di ore di lezione, il numero di ore di esercitazione, il numero di crediti di lezione, il numero di crediti di esercitazione, il docente e il corso di laurea (o i corsi di laurea) a cui afferiscono.
  - I **docenti** sono descritti con la matricola, il nome, il cognome, la città di residenza e possono avere il ruolo di professore ordinario, professore associato, o ricercatore.
  - I **corsi di laurea** sono descritti da un codice, un nome, una tipologia (triennale o magistrale) e dal professore che lo presiede. Un professore può presiedere solo un corso di laurea.

Esercizio 3

- **Progettazione**
  1. Progettazione concettuale: modello E-R
  2. Progettazione logica: modellare con il modello relazionale i dati rappresentati dal diagramma E-R (tabelle, relazioni, e attributi, chiavi e vincoli di integrità)
- **DDL**
  1. Provare a scrivere uno script SQL per costruire le tabelle appena progettate
  2. Tramite un diagramma di MySQL Workbench modellare le tabelle appena progettate

- Tramite il **forward engineering**, generare automaticamente uno script SQL di costruzione del database modellato
  - Confrontare lo script generato in automatico con quello scritto manualmente
- Ricostruire il modello del database dallo script scritto manualmente tramite **reverse engineering**
  - Verificare la correttezza dello script confrontando il modello ottenuto con il modello disegnato in MySQL Workbench