Università degli Studi di Milano Bicocca
Corso di Laurea in Informatica

Creazione di database e interrogazioni 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**

Riepilogo: 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

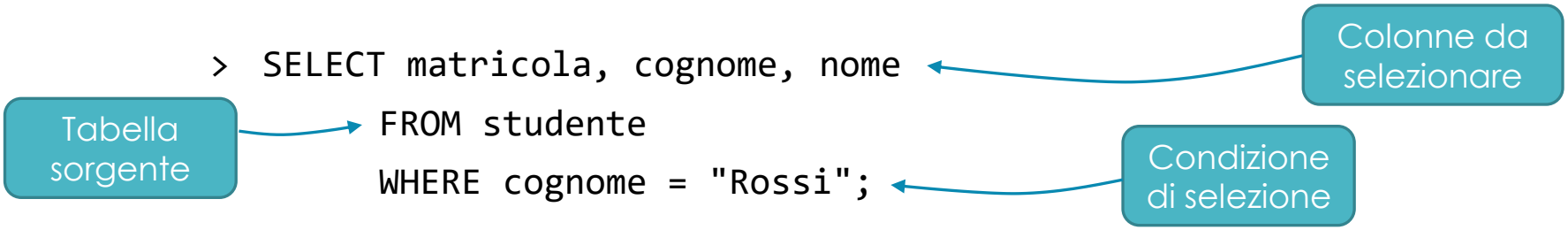
- Controllo del DBMS e dei database

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

MySQL Select

- Lo statement SQL **SELECT** consente di estrarre un insieme di record dal database
  - Oggi vedremo solo interrogazioni di base
  - Sintassi completa: https://dev.mysql.com/doc/refman/5.7/en/select.html

- Selezione di un insieme di record da una tabella



- … se si vogliono selezionare tutti le colonne è possibile utilizzare "*"

```
> SELECT * FROM studente WHERE cognome = "Rossi";
```

MySQL Select – Condizioni

- La clausola `WHERE` consente di **ridurre l'insieme di record selezionati** includendo unicamente quelli che rendono vera una specifica espressione booleana
  - L'espressione può contenere **operatori** e **predicati**, applicare **funzioni** ed eseguire **operazioni logiche, binarie e algebriche**

- Operatori logici: AND, OR, XOR, NOT, IS NULL, IS NOT NULL
  - **Attenzione alla logica a 3 valori: il confronto con il valore NULL produce NULL!**
- Operatori di confronto: =, >, <, >=, <=, <> (oppure !=)

- Predicati tra espressioni:
  - `expr IN(expr1, expr2, expr3, …)`: true se il valore di `expr` è uno di quelli elencati
  - `expr BETWEEN min AND max`: equivalente a `expr >= min AND expr <= max`
  - `expr LIKE "pattern"`: true se `expr` rispetta il pattern dato
    - Il carattere `%` rappresenta una qualsiasi sottostringa

MySQL Select – Condizioni (esempi)

- *Selezionare gli studenti che hanno una matricola compresa tra 80000 e 89999 oppure un cognome che inizia per "R"*

```
> SELECT matricola, cognome, nome
        FROM studente
        WHERE
                matricola BETWEEN 80000 AND 89999 OR
                cognome LIKE "R%";
```

Equivalente a `matricola >= 80000 AND matricola <= 89999`

Razzi, Righi, Rizzi, Rossi, …

- *Selezionare gli ex-studenti laureati con cognome "Rossi", "Verdi" o "Gialli"*

```
> SELECT matricola, cognome, nome
        FROM studente
        WHERE
                data_laurea IS NOT NULL AND
                cognome IN ("Rossi", "Verdi", "Gialli");
```

Supponiamo che gli studenti che non hanno una data di laurea siano quelli non ancora laureati

MySQL Select – Rimozione dei duplicati

- Poiché la `SELECT` consente di selezionare un sottoinsieme delle colonne di una tabella, l'insieme di risultati può contenere duplicati
- La keyword `DISTINCT` consente di rimuovere i duplicati dal risultato

- *Selezionare le città di residenza degli studenti*

```
SELECT
citta_residenza
FROM studenti
```

```
SELECT DISTINCT
citta_residenza
FROM studenti
```

studente

matricola	cognome	nome	citta_residenza
111	Rossi	Mario	Milano
112	Verdi	Paolo	Bergamo
113	Esposito	Maria	Napoli
114	Brambilla	Giovanni	Milano
115	Ferrari	Alice	Bergamo

citta_residenza
Milano
Bergamo
Napoli
Milano
Bergamo

citta_residenza
Milano
Bergamo
Napoli

MySQL Select – Rimozione dei duplicati

- La keyword `DISTINCT` considera diversi due record se differiscono per almeno un valore di una colonna
- *Selezionare i nomi e i cognomi degli studenti*
  - Attenzione alla presenza di **omonimi**!

studente

matricola	cognome	nome	citta_residenza
111	Rossi	Mario	Milano
112	Verdi	Paolo	Bergamo
113	Esposito	Maria	Napoli
114	Brambilla	Giovanni	Milano
115	Ferrari	Alice	Bergamo
116	Rossi	Mario	Bergamo
117	Carli	Alice	Napoli
118	Verdi	Paolo	Milano

```
SELECT
cognome, nome
FROM studente
```

cognome	nome
Rossi	Mario
Verdi	Paolo
Esposito	Maria
Brambilla	Giovanni
Ferrari	Alice
Rossi	Mario
Carli	Alice
Verdi	Paolo

```
SELECT DISTINCT
cognome, nome
FROM studente
```

cognome	nome
Rossi	Mario
Verdi	Paolo
Esposito	Maria
Brambilla	Giovanni
Ferrari	Alice
Carli	Alice

MySQL Select – Operazioni e funzioni

- È possibile applicare **operazioni** e **funzioni** sia ai valori da selezionare sia nelle condizioni di selezione

- Operatori aritmetici: `+`, `-`, `*`, `/`, `%` (oppure `MOD`), `DIV` (divisione intera)
- Operatori bit a bit: `&` (and), `|` (or), `^` (xor), `~` (not)
- Funzioni matematiche:
  - `ABS(v)`: valore assoluto di `v`
  - `CEIL(v)`: minor valore intero `>= v`
  - `FLOOR(v)`: maggior valore intero `<= v`
  - `SIN(v)`, `COS(v)`, `TAN(v)`, …: funzioni trigonometriche seno, coseno, tangente, …
  - `LOG(v)`, `LN(v)`, `LOG10(v)`, `LOG2(v)`: logaritmo naturale, in base 10 e in base 2 di `v`
  - `RAND()`: valore casuale floating-point tra 0.0 e 1.0
  - `ROUND(v, d)`: arrotonda `v` a `d` cifre decimali (d di default è 0)
  - … e altre: https://dev.mysql.com/doc/refman/5.7/en/numeric-functions.html

MySQL Select – Operazioni e funzioni

- *Selezionare gli studenti laureati con il voto di laurea convertito in 30-esimi, arrotondato per eccesso all'intero successivo*

$$voto\_30esimi = \left\lceil voto\_laurea \times \frac{30}{110} \right\rceil$$

Nome della colonna calcolata

```
> SELECT matricola, cognome, nome, CEIL(voto_laurea * (30/110)) AS voto_30esimi
      FROM studente
      WHERE voto_laurea IS NOT NULL;
```

studente

matricola	cognome	nome	voto_laurea
111	Rossi	Mario	NULL
112	Verdi	Paolo	110
113	Esposito	Maria	98
114	Brambilla	Giovanni	103

matricola	cognome	nome	voto_30esimi
112	Verdi	Paolo	30
113	Esposito	Maria	27
114	Brambilla	Giovanni	29

MySQL Select – Funzioni su stringhe

- `LENGTH(s)`: numero di byte della stringa `s`
  - Corrisponde al numero di caratteri tranne nel caso in cui la stringa contenga caratteri multi-byte (ad esempio caratteri accentati nel caso del character set UTF-8)
- `CHAR_LENGTH(s)`: numero di caratteri della stringa `s`
  - Eventuali caratteri multi-byte vengono contati come un solo carattere
- `CONCAT(s1, s2, …)`: concatenazione delle stringhe `s1`, `s2`, …
- `LCASE(s)`: converte la stringa `s` in minuscolo (lowercase)
- `UCASE(s)`: converte la stringa `s` in maiuscolo (uppercase)
- `REPLACE(s, from, to)`: rimpiazza ogni occorrenza della stringa `from` presente nella stringa `s`, con la stringa `to`
- `LEFT(s, n)`: restituisce i primi `n` caratteri della stringa `s`
- `RIGHT(s, n)`: restituisce gli ultimi `n` caratteri della stringa `s`

MySQL Select – Funzioni su stringhe

- `LTRIM(s)`, `RTRIM(s)`, `TRIM(s)`: rimuovono eventuali blank rispettivamente all'inizio, alla fine e da entrambe le estremità della stringa `s`
- `SUBSTRING(s, p, l)` oppure `SUBSTRING(s FROM p FOR l)`: estrae dalla stringa `s` una sottostringa di lunghezza `l`, partendo dal carattere in posizione `p`
  - Se `l` è negativo, `p` indica la posizione a partire dalla fine della stringa
- … e altre: https://dev.mysql.com/doc/refman/5.7/en/string-functions.html

- *Selezionare le iniziali del cognome e del nomi degli studenti*

```
> SELECT matricola, CONCAT(LEFT(cognome, 1), LEFT(nome, 1)) AS iniziali
  FROM studente
```

studente

matricola	cognome	nome
111	Rossi	Mario
112	Verdi	Paolo
113	Esposito	Maria

matricola	iniziali
112	RM
113	VP
114	EM

MySQL Select – Funzioni temporali

- `NOW()`, `CURRENT_TIMESTAMP()`: data e ora attuale
- `CURTIME()`, `CURRENT_TIME()`: ora attuale
- `CURDATE()`, `CURRENT_DATE()`: data attuale
- `DATE(dt)`: estrae la data dal valore dell'espressione temporale `dt`
- `TIME(dt)`: estrae l'ora dal valore dell'espressione temporale `dt`
- `YEAR(dt)`, `MONTH(dt)`, `DAY(dt)`, `HOUR(dt)`, `MINUTE(dt)`, `SECOND(dt)`, `MICROSECOND(dt)`: estraggono dall'espressione temporale `dt` rispettivamente l'anno, il mese, il giorno nel mese, l'ora, i minuti, i secondi e i microsecondi.
- `DAYNAME(dt)`: nome del giorno della settimana rappresentato dall'espressione temporale `dt`
- `DATE_FORMAT(dt, format)`: formatta la data `dt` secondo il formato `format`
  - `DATE_FORMAT('2009-10-04 22:23:00', '%W %M %D %Y %H:%i:%s')` → `"Sunday October 10th 2009 22:23:00"`

MySQL Select – Funzioni temporali

- `ADDDATE(dt, INTERVAL expr unit)`, `SUBDATE(dt, INTERVAL expr unit)`: rispettivamente aggiunge o sottrae dalla data `dt` il valore `expr` di unità temporali `unit`
  - `ADDDATE('2015-05-12', INTERVAL 10 DAYS)` → `'2015-05-22'`
  - `SUBDATE('2015-03-08', INTERVAL 2 YEARS)` → `'2013-03-08'`
- `ADDTIME(dt1, dt2)`, `SUBTIME(dt1, dt2)`: rispettivamente somma o sottrae le espressioni temporali `dt1` e `dt2`
  - `ADDTIME('01:00:00.999999', '02:00:00.999998')` → `'03:00:01.999997'`
  - `SUBTIME('01:00:00.999999', '02:00:00.999998')` → `'-00:59:59.999999'`
- `DATEDIFF(dt1, dt2)`: numero di giorni di differenza tra `dt1` e `dt2`
  - Se `dt1 < dt2` il numero di giorni è negativo
- … e altre: http://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html

MySQL Select – Funzioni temporali

- *Selezionare la matricola e l'anno di laurea degli studenti laureati, con la durata degli studi (numero di giorni intercorsi tra l'iscrizione e la laurea)*

```
SELECT
      matricola,
      YEAR(data_laurea) AS anno_laurea,
      DATEDIFF(data_laurea, data_iscrizione) AS durata_studi
FROM studente
WHERE data_laurea IS NOT NULL
```

studente

matricola	cognome	nome	data_iscrizione	data_laurea
111	Rossi	Mario	2005-08-03	2008-12-03
112	Verdi	Paolo	2006-09-05	2010-04-30
113	Esposito	Maria	2007-09-22	2010-09-15

matricola	anno_laurea	durata_studi
111	2008	1218
112	2010	1333
113	2010	1089

MySQL Select – Controllo di flusso

- Funzioni di controllo di flusso:
  - `CASE v WHEN v1 THEN r1 WHEN v2 THEN r2 … ELSE r END`
    - Confronta `v` con i valori (`v1`, `v2`, …) indicati nelle clausole `WHEN`, e ritorna il risultato indicato nella corrispondente clausola `THEN` (`r1`, `r2`, …). Se `v` è diverso da tutti i valori elencati ritorna il risultato `r` indicato nella clausola `ELSE`.
  - `CASE WHEN cond1 THEN r1 WHEN cond2 THEN r2 … ELSE r END`
    - Valuta le condizioni (`cond1`, `cond2`, …) indicati nelle clausole `WHEN`, e restituisce il valore (`r1`, `r2`, …) indicato dalla clausola `THEN` corrispondente alla prima condizione che produce `TRUE`. Se tutte le condizioni producono `FALSE` restituisce il valore r indicato dalla clausola `ELSE`.
  - `IF(expr, exprTrue, exprElse)`
    - Valuta l'espressione `expr`: se produce TRUE ritorna il valore dell'espressione `exprTrue`, altrimenti (se FALSE o NULL) restituisce il valore dell'espressione `exprElse`.
  - `IFNULL(expr, exprNull)`
    - Se l'espressione `expr` ha valore NULL ritorna il valore dell'espressione `exprNull`, altrimenti ritorna il valore di `expr`.
  - `NULLIF(expr1, expr2)`
    - Ritorna NULL se `expr1 = expr2`, altrimenti ritorna il valore dell'espressione `expr1`.

MySQL Select – Controllo di flusso

- *Selezionare l'età degli studenti*
  - Attenzione: non basta considerare l'anno, bisogna considerare anche il giorno di nascita rispetto alla data attuale!
  - Se il giorno della data di nascita è maggiore del giorno della data di oggi, non ho ancora compiuto gli anni.

```
> SELECT
        IF(
            MONTH(data_nascita) < MONTH(NOW()) OR
                (MONTH(data_nascita) = MONTH(NOW()) AND
                DAY(data_nascita) <= DAY(NOW())),
            YEAR(NOW()) - YEAR(data_nascita),
            YEAR(NOW()) - YEAR(data_nascita) - 1
        ) AS eta
    FROM studente
```

Condizione

Se TRUE, anni già compiuti

Se FALSE, anni ancora non compiuti!

MySQL Select – Selezione da più tabelle

- Due (o più) tabelle possono essere **combinate attraverso l'operazione di join** (prodotto cartesiano)
  - Produce **tutte le possibili combinazioni** tra le tuple delle tabelle coinvolte

- Lo statement `SELECT` effettua il join delle tabelle elencate nella clausola `FROM`
  - In questo caso si parla di **join implicito**
  - L'eventuale condizione di join è definita nella clausola `WHERE`

- *Selezionare gli studenti con la matricola e il nome del corso di laurea a cui sono iscritti*

```
> SELECT st.matricola, cl.nome AS corso_laurea
      FROM studente AS st, corso_laurea AS cl
      WHERE st.id_corso_laurea = cl.id;
```

Condizione di join

Alias

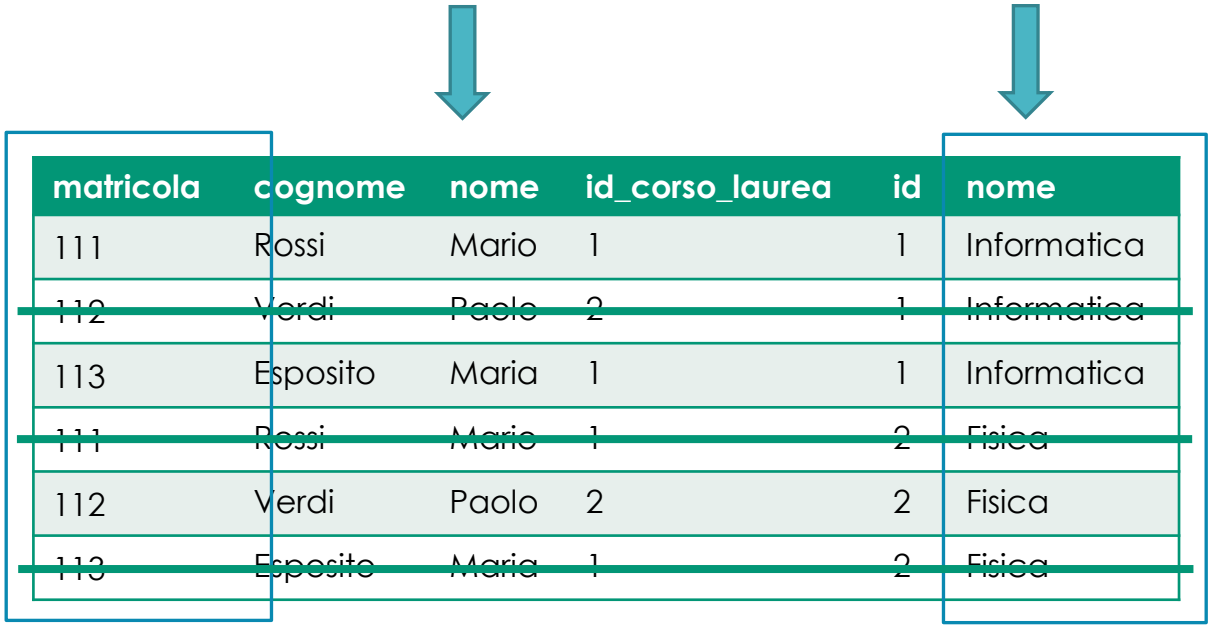
MySQL Select – JOIN

studente

matricola	cognome	nome	id_corso_laurea
111	Rossi	Mario	1
112	Verdi	Paolo	2
113	Esposito	Maria	1

corso_laurea

id	nome
1	Informatica
2	Fisica



matricola	cognome	nome	id_corso_laurea	id	nome
111	Rossi	Mario	1	1	Informatica
112	Verdi	Paolo	2	1	Informatica
113	Esposito	Maria	1	1	Informatica
111	Rossi	Mario	1	2	Fisica
112	Verdi	Paolo	2	2	Fisica
113	Esposito	Maria	1	2	Fisica

matricola	corso_laurea
111	Informatica
112	Fisica
113	Informatica

MySQL Select – JOIN

- Il join in SQL è esprimibile tramite la sintassi `JOIN table ON condition`
  - In questo caso si parla di **join esplicito**
  - La clausola `ON` definisce la condizione di join

- Riprendendo l'esempio precedente…
  Selezionare gli studenti con la matricola e il nome del corso di laurea a cui sono iscritti

```
> SELECT st.matricola, cl.nome AS corso_laurea
    FROM studente AS st
    JOIN corso_laurea AS cl ON cl.id = st.id_corso_laurea;
```

Condizione di join

Tabella in join

studente

matricola	cognome	nome	id_corso_laurea
111	Rossi	Mario	1
112	Verdi	Paolo	2
113	Esposito	Maria	1

corso_laurea

id	nome
1	Informatica
2	Fisica

matricola	corso_laurea
111	Informatica
112	Fisica
113	Informatica

MySQL Select – JOIN

- Cosa succede se la condizione di join restituisce un valore NULL?

- Esempio: *selezionare gli studenti con la matricola e la regione di nascita*

studente

matricola	cognome	nome	straniero	id_comune_nascita
111	Rossi	Mario	0	2
112	Verdi	Paolo	0	1
113	Doe	John	1	NULL

comune

id	nome	regione
1	Milano	Lombardia
2	Napoli	Campania

```
SELECT s.matricola, c.regione
FROM studente AS s
JOIN comune AS c
ON s.id_comune_nascita = c.id
```

matricola	regione
111	Campania
112	Lombardia

Sbagliato!
Abbiamo perso lo studente 113 poiché non soddisfa la condizione di join!

MySQL Select – Tipologie di join

- Il join visto fin qui è detto **join interno** (*inner join*) perché esclude tutte le tuple per le quali la condizione di join restituisce valori NULL
  - `JOIN` e `INNER JOIN` sono equivalenti

- SQL prevede anche alcune forme di **join esterno** (*outer join*), che preserva le tuple la cui condizione di join produce un valore NULL
  - `LEFT OUTER JOIN` (o `LEFT JOIN`): mantiene solo le tuple della tabella alla sinistra dell'operatore di join
  - `RIGHT OUTER JOIN` (o `RIGHT JOIN`): mantiene solo le tuple della tabella alla destra dell'operatore di join
  - `FULL OUTER JOIN`: mantiene le tuple di entrambe le tabelle coinvolte nel join
    - **Non è supportato da MySQL** (e anche da molti altri DBMS), ma emulabile tramite `UNION ALL`

MySQL Select – LEFT JOIN

- Riprendendo l'esempio precedente…
  Selezionare gli studenti con la matricola e la regione di nascita

studente

matricola	cognome	nome	straniero	id_comune_nascita
111	Rossi	Mario	0	2
112	Verdi	Paolo	0	1
113	Doe	John	1	NULL

comune

id	nome	regione
1	Milano	Lombardia
2	Napoli	Campania

```
> SELECT s.matricola, c.regione
  FROM studente AS s LEFT JOIN comune AS c ON s.id_comune_nascita = c.id;
```

- Tabella a *sinistra*: **studente**
- Tabella a *destra*: **comune**

matricola	regione
111	Campania
112	Lombardia
113	NULL

Grazie al LEFT JOIN **lo studente 113 viene preservato** anche se non soddisfa la condizione di join

Esercizi

Esercizio 7

- **Scrivere ed eseguire le query riportate nelle slide successive**
- Utilizzare il database `universita.sql` (reso disponibile tramite piattaforma di eLearning)
- Caricare lo script e creare il DB tramite MySQL Workbench
  - Se vi disconnettete dal DBMS e vi riconnettete, dovete selezionare il DB con

```
USE universita;
```

- Per ottenere uno schema della struttura del database potete utilizzare il reverse engineering di MySQL Workbench
- Le query possono essere scritte sia tramite MySQL Workbench che tramite Command Line Client (per i più temerari ☺)
- A fianco della query da eseguire è reso disponibile il numero di entry (righe) che sono selezionate se avete scritto la query corretta

Esercizio 7 – Query

1. Si selezionino tutte le città (tutti i campi) [370 righe]
2. Si selezionino le città che iniziano con la lettera "B" (tutti i campi) [39 righe]
3. Si selezionino gli studenti il cui cognome finisce con la "o" (tutti i campi) [125 righe]
4. Si selezionino gli studenti che nel loro nome contengono una "r" (tutti i campi) [353 righe]
5. Si selezionino le città che iniziano per "B" e che appartengono alla regione Piemonte (tutti i campi) [11 righe]
6. Si selezionino gli studenti che hanno una matricola tra 1000000 e 1999999 (tutti i campi) [129 righe]
7. Si selezionino i corsi che hanno un numero di ore di lezione compreso tra 10 e 20 [9 righe]
8. Si selezioni il nome e il CAP delle città che si trovano in Lombardia o in Piemonte [185 righe]
9. Si selezioni la denominazione del Corso di Laurea e nome e cognome del Presidente [2 righe]
10. Si selezionino il nome e il cognome dei docenti che insegnano in corsi in cui le ore di lezione sono più di 20 [53 righe]
11. Selezionare tutte le città di residenza, con la regione, dei docenti ordinari [29 righe]
12. Evidenziare da quali città provengono gli studenti [298 righe]

Esercizio 7 – Query (continua…)

13. Evidenziare da quali regioni provengono gli studenti [18 righe]
14. Si selezioni lo stipendio mensile, nome e cognome dei docenti ordinari, e i corsi che essi insegnano [12 righe]
15. Si selezionino tutti i corsi (nome) e il totale delle ore, appartenenti a corsi di laurea triennali [31 righe]
16. Selezionare gli esami svolti dallo studente "1492601" con le relative votazioni e date [6 righe]
17. Selezionare gli esami svolti nei corsi di laurea triennale, e le relative votazioni [1000 righe]
18. Selezionare gli esami, e i relativi crediti totali, sostenuti nei corsi di laurea triennale, senza ripetizioni di record e senza votazioni [31 righe]
19. Si selezionino il nome, il cognome, la matricola e la citta di residenza degli studenti che hanno il corso "Sistemi distribuiti" nel piano di studi [316 righe]
20. Si selezionino il nome, il cognome, la matricola e la citta di residenza dei docenti ordinari e associati che insegnano alla triennale e che guadagnano almeno 40000 [14 righe]
21. Si selezionino gli studenti triennali (nome, cognome, matricola e citta di residenza) che hanno sostenuto l'esame per il corso "Sistemi distribuiti" e appartengono alla regione Piemonte [37 righe]

Esercizio 8

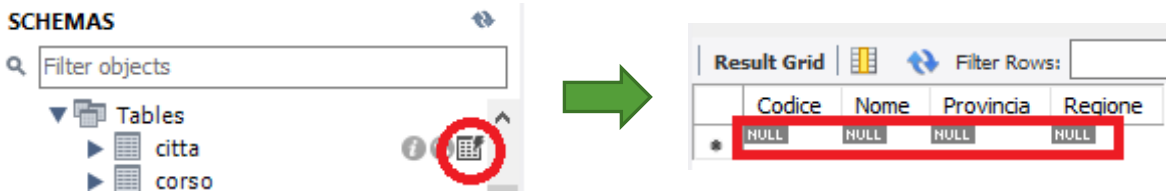
- Progettare la seguente base di dati: **Scuola Secondaria**
  - Si vuole rappresentare la **base dati di una scuola secondaria**, ad esempio un liceo.
  - Gli **insegnanti** hanno un codice fiscale, un nome e un cognome. Alcuni insegnano anche in altre **scuole**. Di questi si vogliono rappresentare tali scuole, con codice e nome della scuola, e comune e regione in cui questa si trova. Di tutti gli insegnanti si vuole rappresentare il comune di residenza e di nascita, con un codice del comune, il nome e la regione.
  - Ogni docente insegna in diverse **classi**, in genere più di una **materia**. Le classi sono descritte da un anno e da una lettera (esempio *Seconda A*). Le materie sono descritte da un codice e un nome. Si vuole inoltre rappresentare il numero di ore settimanali di insegnamento di ogni materia insegnata da ogni docente per ognuna delle classi in cui insegna (ad esempio Mario Rossi insegna *Italiano* alla *Terza B* per tre ore alla settimana).

Esercizio 8 (continua…)

- Le materie hanno diversi **argomenti**, descritti da un codice progressivo (ad es. argomento 1, 2, …) univoco relativamente alla specifica materia, da un nome, e da un numero di ore di insegnamento richiesto. Degli argomenti interessa conoscere anche le propedeuticità (ad esempio il modello relazionale è propedeutico all'SQL). Si assume che la propedeuticità coinvolga solo argomenti della stessa materia.
- Gli **studenti** sono descritti dal codice fiscale, dal nome, dal cognome, e dal comune di residenza. Gli studenti stranieri sono descritti anche dalla nazione di origine. Per ogni studente si vuole rappresentare la classe di appartenenza e il voto (solo uno per semplicità) conseguito (se conseguito) in ogni argomento di ogni materia.
- Si è interessati anche a rappresentare eventuali **legami di parentela** tra gli studenti, con il grado di parentela.

Esercizio 8

- **Progettazione**
  1. Progettazione concettuale: modello E-R
  2. Progettazione logica: modellare con il modello relazionale i dati rappresentati dal diagramma E-R (tabelle, relazioni, attributi, chiavi e vincoli di integrità)
- **DDL**
  1. Modellare le tabelle appena progettate tramite MySQL Workbench
- **Creazione del database**
  1. Tramite il **forward engineering**, generare automaticamente uno script SQL che definisca il database modellato
  2. Eseguire lo script generato al fine di creare il database modellato
  3. Popolare il DB con dei dati a piacere per mezzo di Workbench (è possibile farlo cliccando, per la tabella da popolare, l'icona cerchiata qui sotto in rosso e aggiungere le entry manualmente in «Result Grid»



- **Interrogazione** – scrivere ed eseguire le interrogazioni seguenti:
  1. Selezionare i docenti il cui cognome termina con la lettera "i"
  2. Selezionare gli studenti con nome, cognome e classe a cui appartengono