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 preliminare – Possibile soluzione

- Effettuare almeno le seguenti operazioni:
  1. Creare il database, dando un nome a piacere

     ```
     > CREATE SCHEMA IF NOT EXISTS es1_esami;
     ```

  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).

     ```
     > CREATE TABLE IF NOT EXISTS es1_esami.studente (
     -> matricola INT NOT NULL,
     -> cognome VARCHAR(45) NOT NULL,
     -> nome VARCHAR(45) NOT NULL,
     -> citta_nascita VARCHAR(45) NOT NULL,
     -> citta_residenza VARCHAR(45) NOT NULL,
     -> corso_laurea VARCHAR(45) NOT NULL,
     -> PRIMARY KEY (matricola))
     -> ENGINE = InnoDB;
     ```

NB: i caratteri `->` non fanno parte del comando: indicano il cursore della CLI quando si preme invio (operazione consigliabile per migliorare la leggibilità del comando che si sta scrivendo)

Esercizio preliminare – Possibile soluzione

2. *(continua)* 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).

```
> CREATE TABLE IF NOT EXISTS es1_esami.corso (
-> codice INT NOT NULL,
-> nome VARCHAR(45) NOT NULL,
-> ore_lezione INT NOT NULL,
-> ore_esercitazione INT NULL,
-> crediti_lezione INT NOT NULL,
-> crediti_esercitazione INT NULL,
-> docente VARCHAR(45) NOT NULL,
-> PRIMARY KEY (codice))
-> ENGINE = InnoDB;
```

Esercizio preliminare – Possibile soluzione

2. *(continua)* 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).

```
> CREATE TABLE IF NOT EXISTS es1_esami.esame (
-> matricola_studente INT NOT NULL,
-> codice_corso INT NOT NULL,
-> data DATETIME NOT NULL,
-> voto INT NOT NULL,
-> PRIMARY KEY (matricola_studente, codice_corso),
-> CONSTRAINT fk_studente
->    FOREIGN KEY (matricola_studente)
->    REFERENCES es1_esami.studente (matricola)
->    ON UPDATE CASCADE
->    ON DELETE RESTRICT,
-> CONSTRAINT fk_corso
->    FOREIGN KEY (codice_corso)
->    REFERENCES es1_esami.corso (codice)
->    ON UPDATE CASCADE
->    ON DELETE RESTRICT)
-> ENGINE = InnoDB;
```

Esercizio preliminare – Possibile soluzione

3. Utilizzando lo statement `ALTER TABLE`, aggiungere alla tabella *studente*:
   - la *data di laurea*, impostandola di default a `NULL`

```
> ALTER TABLE es1_esami.studente
->     ADD COLUMN data_laurea DATETIME NULL DEFAULT NULL;
```

   - Il *titolo della tesi*, impostandola di default a `NULL`

```
> ALTER TABLE es1_esami.studente
->     ADD COLUMN titolo_tesi VARCHAR(45) NULL DEFAULT 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*

```
> ALTER TABLE es1_esami.esame
->     ADD COLUMN lode BOOLEAN NOT NULL DEFAULT FALSE;
```

   - In alternativa può essere direttamente usato `TINYINT(1)` (il tipo di dato viene convertito in TINYINT(1) in ogni caso)

```
> ALTER TABLE es1_esami.esame
->     ADD COLUMN lode TINYINT(1) NOT NULL DEFAULT 0;
```

Esercizio preliminare – Possibile soluzione

5. Rinominare a piacere una delle tabelle create

```
> ALTER TABLE es1_esami.studente RENAME TO es1_esami.studente_immatricolato;
```

   - Oppure (equivalente)

```
> RENAME TABLE es1_esami.studente TO es1_esami.studente_immatricolato;
```

6. Eliminare la tabella *esame*

```
> DROP TABLE es1_esami.esame;
```

   - **NB:** se avete settato in un vincolo di integrità referenziale "ON DELETE RESTRICT" non vi è giustamente possible cancellare la tabella referenziata (*studente* o *corso*, in questo esempio)

5. Eliminare l'intero database

```
> DROP SCHEMA es1_esami;
```

Esercizio 3

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

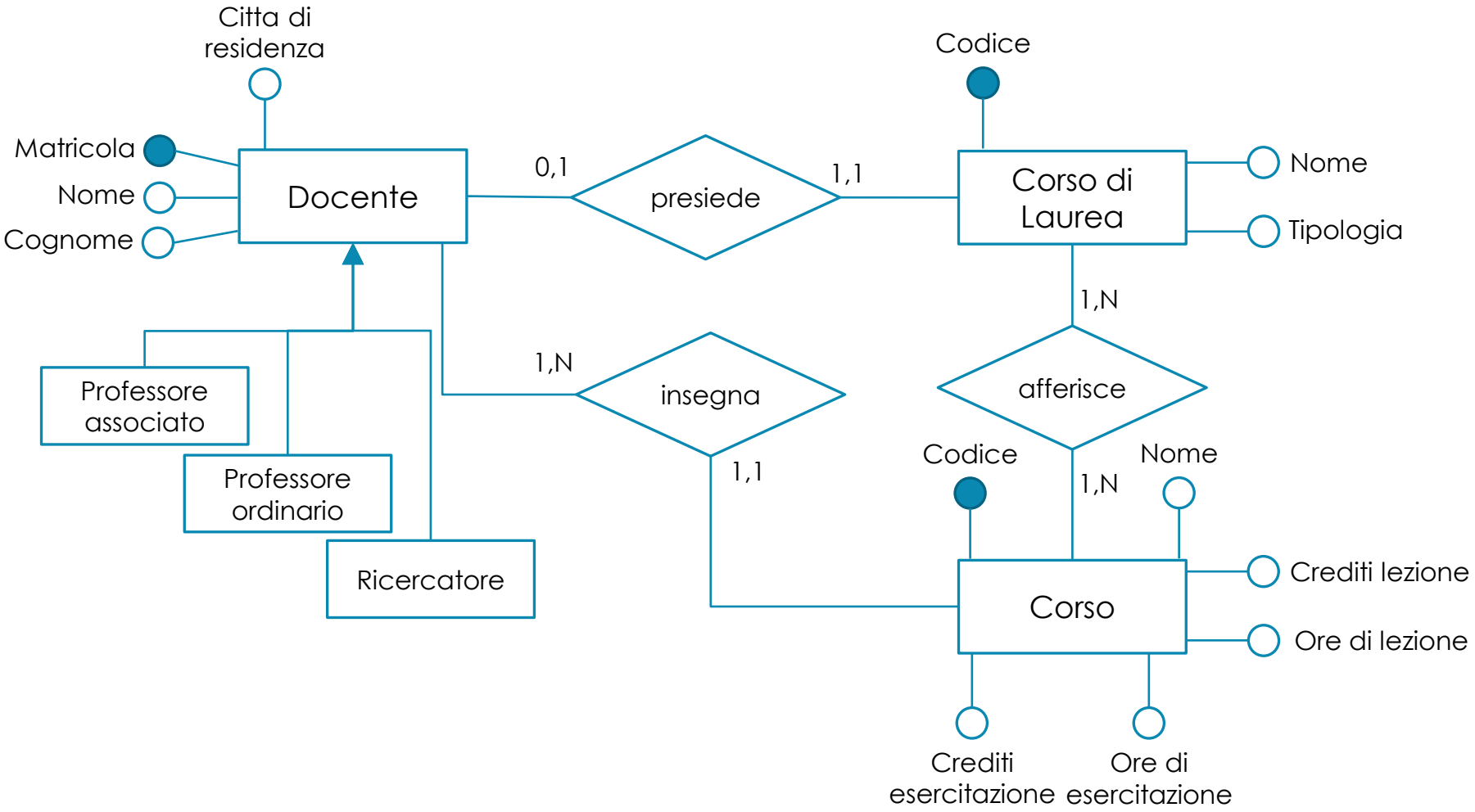
  - Ogni **corso** può essere insegnato da un solo professore, ma un professore può erogare più corsi. Ogni professore insegna almeno un corso.
  - I corsi sono descritti con un codice, un nome, il numero di ore di lezione, il numero di ore di esercitazione, il numero di crediti di lezione, il numero di 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

Esercizio 3 – Modello ER



Esercizio 3 – Modello relazionale

- **docente** (matricola, cognome, nome, citta_residenza, ruolo)
  - *ruolo* può assumere i valori (*professore associato*, *professore ordinario*, *ricercatore*)

- **corso_laurea** (codice, nome, tipologia, presidente)
  - *presidente* è una chiave esterna verso *matricola* in *docente*
  - *tipologia* può assumere i valori (*triennale*, *magistrale*)

- **corso_di_laurea_corsi** (codice_corso_laurea, codice_corso)
  - *codice_corso_laurea* è una chiave esterna verso *codice* in *corso_laurea*
  - *codice_corso* è una chiave esterna verso *codice* in *corso*

- **corso** (codice, nome, ore_lezione, crediti_lezione, ore_esercitazione, crediti_esercitazione, docente)
  - *docente* è una chiave esterna verso *matricola* in *docente*

Esercizio 3 – Modellazione in Workbench