

Basi di Dati

Esercitazione #1

Tutor

Mauro Farina

- Dottorando in ingegneria industriale e dell'informazione
- 1° anno
- Cybersecurity e misurazioni di Internet
- mauro.farina@phd.units.it

Materiale su GitHub: github.com/mauro-farina/database-exercises

Feedback: forms.gle/yAYRAvC3iHarHJyL6

Esercitazioni

Tra marzo e maggio, faremo 4 o 5 esercitazioni:

1. **mercoledì 26 marzo**
2. venerdì 11 aprile
3. TBD
4. TBD
5. [TBD]

Creazione Database

- Problema
- Modello logico
- Implementazione
- Popolamento

Formulazione problema

Vogliamo modellare la base di dati per **tenere traccia degli esami** sostenuti dagli studenti all'interno dell'università

Entità principali...?

Formulazione problema

Vogliamo modellare la base di dati per **tenere traccia degli esami** sostenuti dagli studenti all'interno dell'università

Entità principali:

- Studenti
- Esami
- Corsi
- Professori

... e quali attributi?

Studenti

Esami

Corsi

Professori

Modello logico

“sottintende una specifica rappresentazione dei dati” → **Tabelle**

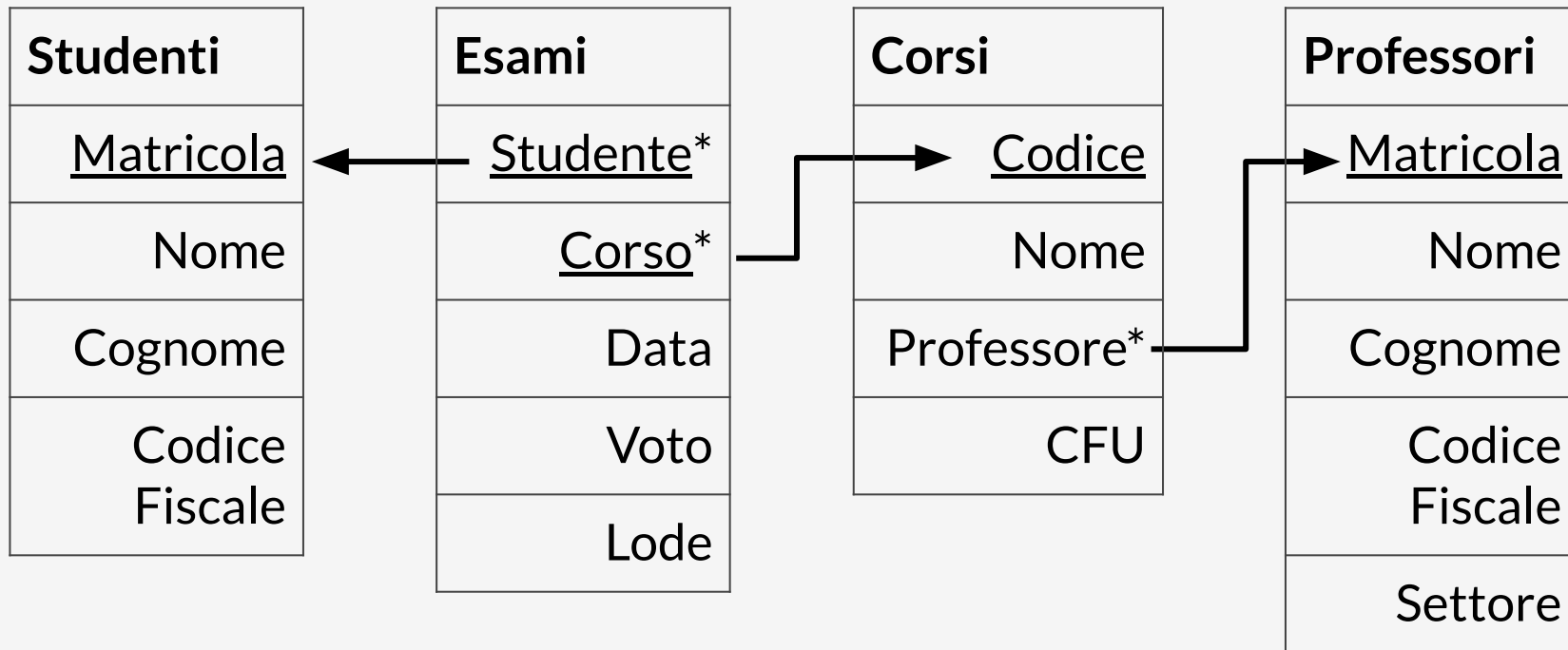
Studenti
Matricola
Nome
Cognome
Codice Fiscale

Esami
Studente
Corso
Data
Voto
Lode

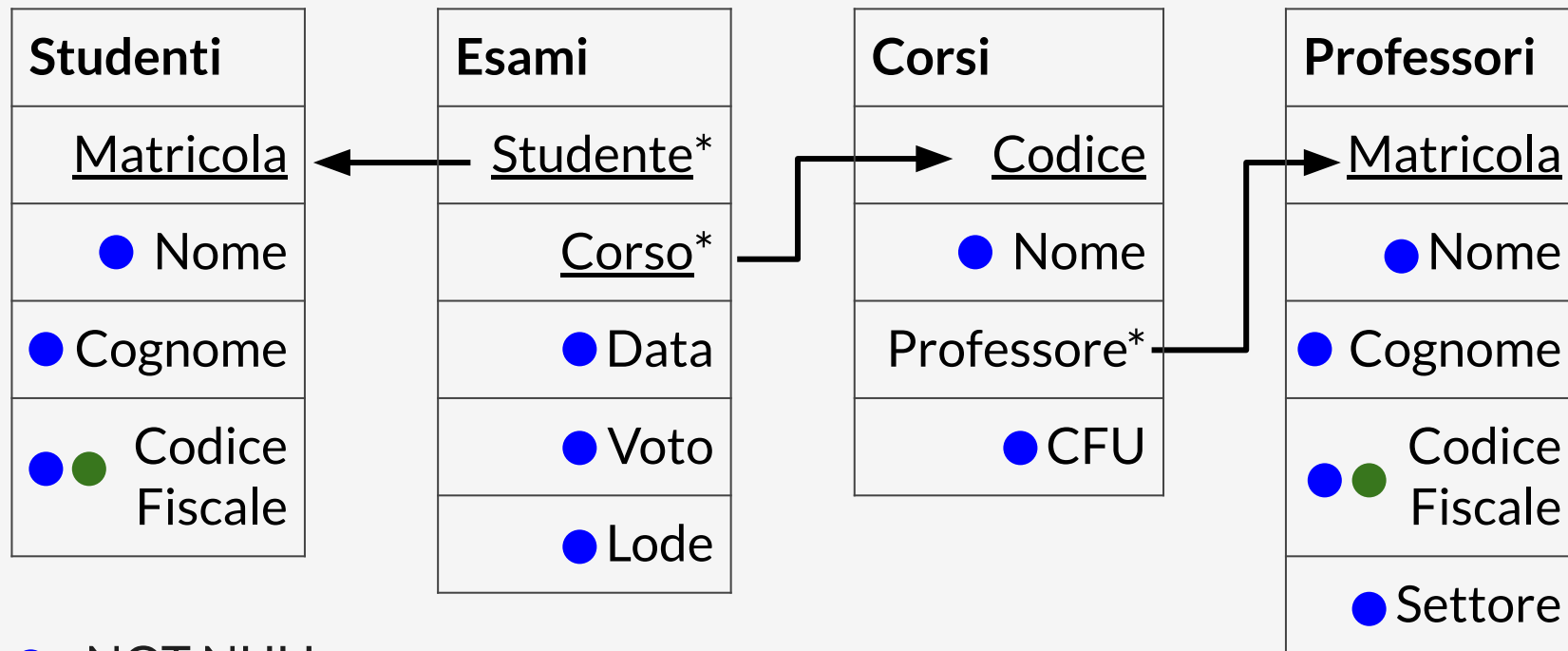
Corsi
Codice
Nome
Professore
CFU

Professori
Matricola
Nome
Cognome
Codice Fiscale
Settore

... mancano le chiavi!



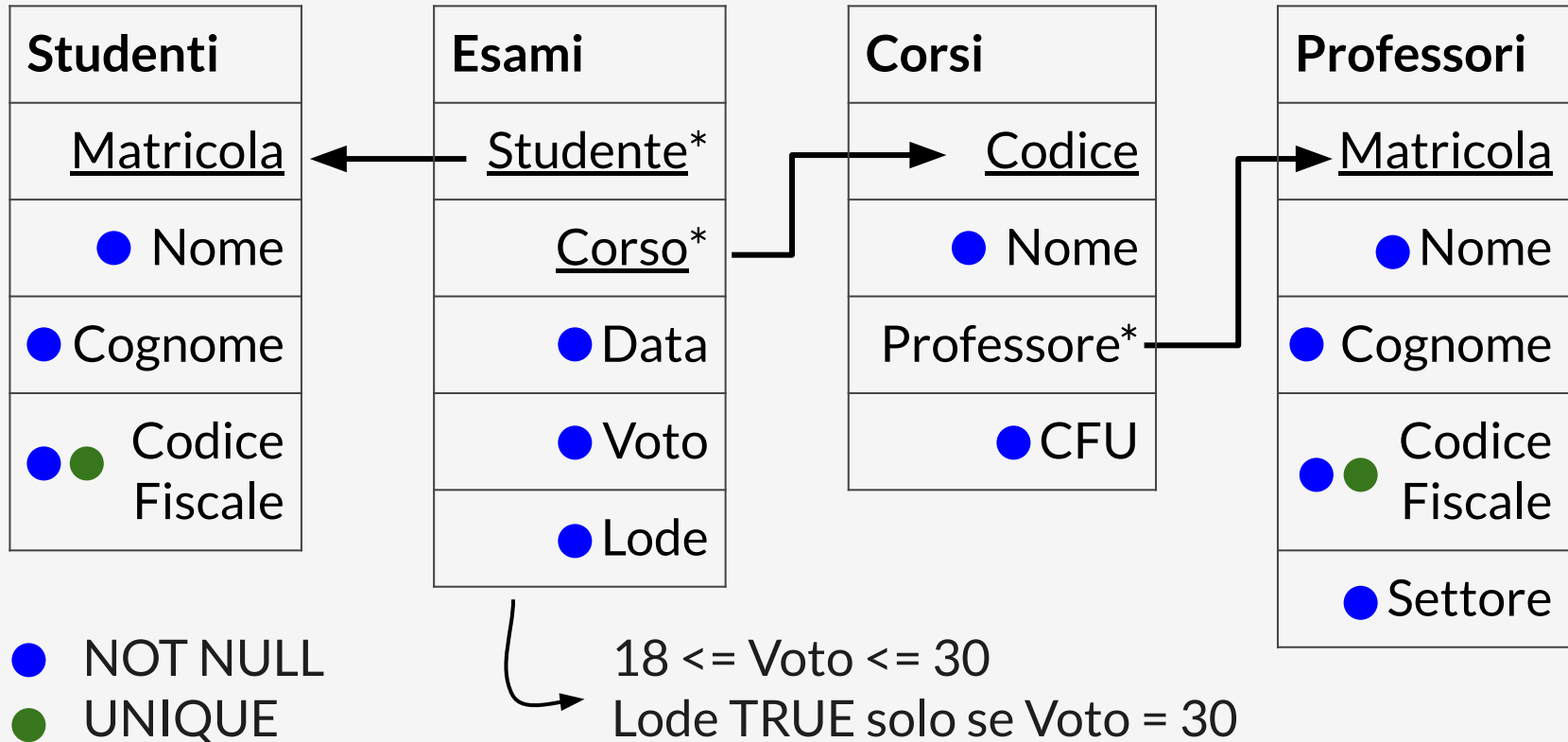
... e i vincoli!



● NOT NULL
● UNIQUE

Altri vincoli?

... e i vincoli!



Implementazione MySQL

1. Creazione del database

```
CREATE DATABASE unidb;
```

2. Selezione del database

```
USE unidb;
```

3. Creazione tabelle

4. Popolamento del database

Creazione tabelle: Studenti

<u>Matricola</u>	Nome	Cognome	Codice Fiscale
------------------	------	---------	----------------

- Non vogliamo valori NULL
- Codice Fiscale deve essere UNIQUE

Creazione tabelle: Studenti

<u>Matricola</u>	Nome	Cognome	Codice Fiscale
------------------	------	---------	----------------

```
CREATE TABLE studenti(  
    matricola CHAR(9) PRIMARY KEY,  
    nome VARCHAR(45) NOT NULL,  
    cognome VARCHAR(45) NOT NULL,  
    cf CHAR(16) NOT NULL UNIQUE  
);
```

Creazione tabelle: Professori

<u>Matricola</u>	Nome	Cognome	Codice Fiscale	Settore
------------------	------	---------	----------------	---------

- Non vogliamo valori NULL
- Codice Fiscale deve essere UNIQUE

Creazione tabelle: Professori

<u>Matricola</u>	Nome	Cognome	Codice Fiscale	Settore
------------------	------	---------	----------------	---------

```
CREATE TABLE professori(  
    matricola INT(4) PRIMARY KEY AUTO_INCREMENT,  
    nome VARCHAR(45) NOT NULL,  
    cognome VARCHAR(45) NOT NULL,  
    cf CHAR(16) NOT NULL UNIQUE,  
    settore VARCHAR(12) NOT NULL  
);
```


Creazione tabelle: Corsi

<u>Codice</u>	Nome	CFU	Professore*
---------------	------	-----	-------------

- Nome e CFU non devono essere NULL
- Professore fa riferimento all'attributo matricola della tabella **professori**

Creazione tabelle: Corsi

<u>Codice</u>	Nome	CFU	Professore*
---------------	------	-----	-------------

```
CREATE TABLE corsi(  
    codice CHAR(5) PRIMARY KEY,  
    nome VARCHAR(45) NOT NULL,  
    cfu TINYINT NOT NULL,  
    professore INT(4) ,  
    FOREIGN KEY (professore)  
        REFERENCES professori(matricola)  
);
```

Creazione tabelle: Corsi

<u>Codice</u>	Nome	CFU	Professore*
---------------	------	-----	-------------

```
CREATE TABLE corsi(  
    codice CHAR(5) PRIMARY KEY,  
    nome VARCHAR(45) NOT NULL,  
    cfu TINYINT NOT NULL,  
    professore INT(4),  
    FOREIGN KEY (professore)  
        REFERENCES professori(matricola)  
);
```


... e se eliminiamo dalla
tabella **professori**
un'istanza a cui
facciamo riferimento
nella tabella **corsi**?

Creazione tabelle: Corsi

<u>Codice</u>	Nome	CFU	Professore*
---------------	------	-----	-------------

```
CREATE TABLE corsi(  
  codice CHAR(5) PRIMARY KEY,  
  nome VARCHAR(45) NOT NULL,  
  cfu TINYINT NOT NULL,  
  professore INT(4),  
  FOREIGN KEY (professore)  
    REFERENCES professori(matricola)  
);
```

... e se eliminiamo dalla
tabella **professori**
un'istanza a cui
facciamo riferimento
nella tabella **corsi**?



Comportamento
default: **rifiuto**
dell'operazione

Creazione tabelle: Corsi

<u>Codice</u>	Nome	CFU	Professore*
---------------	------	-----	-------------

```
CREATE TABLE corsi(  
    codice CHAR(5) PRIMARY KEY,  
    nome VARCHAR(45) NOT NULL,  
    cfu TINYINT NOT NULL,  
    professore INT(4) ,  
    FOREIGN KEY (professore)  
        REFERENCES professori(matricola)  
        ON DELETE SET NULL  
);
```

Creazione tabelle: Esami

<u>Corso</u> *	<u>Studente</u> *	Data	Voto	Lode
----------------	-------------------	------	------	------

- Data, Voto e Lode non devono essere NULL
- Lode di default FALSE
- Voto compreso tra 18 e 30
- Se Voto < 30, Lode **non** può essere TRUE

Creazione tabelle: Esami

<u>Corso</u> *	<u>Studente</u> *	Data	Voto	Lode
----------------	-------------------	------	------	------

```
CREATE TABLE esami(  
    corso CHAR(5), studente VARCHAR(9),  
    data DATE NOT NULL, voto TINYINT NOT NULL,  
    lode BOOL DEFAULT FALSE,  
    PRIMARY KEY (corso, studente),  
    FOREIGN KEY (studente) REFERENCES studenti(matricola),  
    FOREIGN KEY (corso) REFERENCES corsi(codice),  
    CHECK (voto BETWEEN 18 AND 30 AND ((voto<=30 AND  
        lode=FALSE) OR (voto=30 AND lode=TRUE)))  
);
```

Creazione tabelle: Esami

<u>Corso</u> *	<u>Studente</u> *	Data	Voto	Lode
----------------	-------------------	------	------	------

```
CREATE TABLE esami(  
  corso CHAR(5), studente VARCHAR(9),  
  data DATE NOT NULL, voto TINYINT NOT NULL,  
  lode BOOL DEFAULT FALSE,  
  PRIMARY KEY (corso, studente),  
  FOREIGN KEY (studente) REFERENCES studenti(matricola),  
  FOREIGN KEY (corso) REFERENCES corsi(codice),  
  CHECK (voto BETWEEN 18 AND 30 AND ((voto<=30 AND  
    lode=FALSE) OR (voto=30 AND lode=TRUE)))  
);
```

Come gestiamo le
operazioni DELETE
sulle tabelle **studenti**
e **corsi**?

Creazione tabelle: Esami

<u>Corso</u> *	<u>Studente</u> *	Data	Voto	Lode
----------------	-------------------	------	------	------

```
CREATE TABLE esami(  
    corso CHAR(5), studente VARCHAR(9),  
    data DATE NOT NULL, voto TINYINT NOT NULL,  
    lode BOOL DEFAULT FALSE,  
    PRIMARY KEY (corso, studente),  
    FOREIGN KEY (studente) REFERENCES  
        studenti(matricola) ON DELETE CASCADE,  
    FOREIGN KEY (corso) REFERENCES corsi(codice),  
    CHECK (voto BETWEEN 18 AND 30 AND ((voto<=30 AND  
        lode=FALSE) OR (voto=30 AND lode=TRUE)))  
);
```

Verifica delle tabelle create

1. Tutte le tabelle nel database

```
SHOW TABLES;
```

2. Dettagli di una tabella → colonne, PK, default, unique, ...

```
DESCRIBE nomeTabella;
```

3. Codice usato per creare la tabella → tutti i **vincoli** imposti

```
SHOW CREATE TABLE nomeTabella;
```

Inserimento dati

Promemoria: possiamo inserire dati in due modi:

1. Se inserisco solo un **sottoinsieme** di attributi

```
INSERT INTO nomeTabella(colonna1, ..., colonnaN)  
VALUES (valore1, ..., valoreN);
```

2. Se invece specifico **tutti** gli attributi, si semplifica

```
INSERT INTO nomeTabella  
VALUES (valore1, ..., valoreN);
```

Inserimento dati: studenti

<u>Matricola</u>	Nome	Cognome	Codice Fiscale
------------------	------	---------	----------------

Opzioni equivalenti?

```
1. INSERT INTO studenti(matricola, nome, cognome, cf)
VALUES ("IN0500123", "Marco", "Rossi", "RSSMRC99A11L424K") ;
```

```
2. INSERT INTO studenti
VALUES ("IN0500123", "Marco", "Rossi", "RSSMRC99A11L424K") ;
```

Inserimento dati: studenti

<u>Matricola</u>	Nome	Cognome	Codice Fiscale
------------------	------	---------	----------------

Opzioni equivalenti?

```
1. INSERT INTO studenti(matricola, nome, cognome, cf)  
VALUES ("IN0500123", "Marco", "Rossi", "RSSMRC99A11L424K") ;
```

```
2. INSERT INTO studenti  
VALUES ("IN0500123", "Marco", "Rossi", "RSSMRC99A11L424K") ;
```

Sì! → Tutti gli attributi sono specificati

Inserimento dati: professori

<u>Matricola</u>	Nome	Cognome	Codice Fiscale	Settore
------------------	------	---------	----------------	---------

Opzioni equivalenti?

1. `INSERT INTO professori (nome, cognome, cf, settore)
VALUES ("Marco", "Rossi", "RSSMRC75A24L424K") ;`
2. `INSERT INTO professori
VALUES (1, "Marco", "Rossi", "RSSMRC75A24L424K") ;`

Inserimento dati: professori

<u>Matricola</u>	Nome	Cognome	Codice Fiscale	Settore
------------------	------	---------	----------------	---------

Opzioni equivalenti?

```
1. INSERT INTO professori (nome, cognome, cf, settore)
   VALUES ("Marco", "Rossi", "RSSMRC75A24L424K") ;
```

```
2. INSERT INTO professori
   VALUES (1, "Marco", "Rossi", "RSSMRC75A24L424K") ;
```

Dipende → `matricola INT(4) PRIMARY KEY AUTO_INCREMENT`

Inserimento dati: professori

<u>Matricola</u>	Nome	Cognome	Codice Fiscale	Settore
------------------	------	---------	----------------	---------

Opzioni equivalenti

```
1. INSERT INTO professori (nome, cognome, cf, settore)
   VALUES ("Marco", "Rossi", "RSSMRC75A24L424K") ;
```

```
2. INSERT INTO professori
   VALUES (NULL, "Marco", "Rossi", "RSSMRC75A24L424K") ;
```

Sì! → `matricola INT(4) PRIMARY KEY AUTO_INCREMENT`

Inserimento dati: professori

<u>Matricola</u>	Nome	Cognome	Codice Fiscale	Settore
------------------	------	---------	----------------	---------

```
matricola INT(4) PRIMARY KEY AUTO_INCREMENT
```

...

E se facciamo queste operazioni?

```
INSERT INTO professori  
VALUES (35, "Marco", "Rossi", "RSSMRC75A24L424K", "INF/01");
```

```
INSERT INTO professori(nome, cognome, cf, settore)  
VALUES ("Carlo", "Verdi", "VRDCRL68A13L424Y", "INF/01");
```

Inserimento dati: professori

<u>Matricola</u>	Nome	Cognome	Codice Fiscale	Settore
0035	Marco	Rossi	RSSMRC75A24L424K	INF/01
0001	Carlo	Verdi	VRDCRL68A13L424Y	INF/01
0036	Carlo	Verdi	VRDCRL68A13L424Y	INF/01

```
INSERT INTO professori
```

```
VALUES (35, "Marco", "Rossi", "RSSMRC75A24L424K", "INF/01") ;
```

```
INSERT INTO professori(nome, cognome, cf, settore)
```

```
VALUES ("Carlo", "Verdi", "VRDCRL68A13L424Y", "INF/01") ;
```

Inserimento dati: professori

<u>Matricola</u>	Nome	Cognome	Codice Fiscale	Settore
0035	Marco	Rossi	RSSMRC75A24L424K	INF/01
0001	Carlo	Verdi	VRDCRL68A13L424Y	INF/01
0036	Carlo	Verdi	VRDCRL68A13L424Y	INF/01

```
INSERT INTO professori
```

```
VALUES (35, "Marco", "Rossi", "RSSMRC75A24L424K", "INF/01") ;
```

```
INSERT INTO professori(nome, cognome, cf, settore)
```

```
VALUES ("Carlo", "Verdi", "VRDCRL68A13L424Y", "INF/01") ;
```

Inserimento dati: professori

<u>Matricola</u>	Nome	Cognome	Codice Fiscale	Settore
0035	Marco	Rossi	RSSMRC75A24L424K	INF/01
0036	Carlo	Verdi	VRDCRL68A13L424Y	INF/01

E se poi facciamo...

```
INSERT INTO professori (matricola, nome, cognome, cf, settore)  
VALUES (30, "Marta", "Casali", "CSLMRT92A65L424Y", "INF/01");
```

```
INSERT INTO professori (nome, cognome, cf, settore)  
VALUES ("Ferdinando", "Grigi", "GRGFRD87A19L424Z", "INF/01");
```

Inserimento dati: professori

<u>Matricola</u>	Nome	Cognome	Codice Fiscale	Settore
0035	Marco	Rossi	RSSMRC75A24L424K	INF/01
0036	Carlo	Verdi	VRDCRL68A13L424Y	INF/01
0030 ?	Marta	Casali	CSLMRT92A65L424Y	INF/01
0031? 0037?	Fernando	Grigi	GRGFRN87A19L424Z	INF/01

```
INSERT INTO professori (matricola,nome,cognome,cf,settore)
VALUES (30,"Marta","Casali","CSLMRT92A65L424Y","INF/01");
```

```
INSERT INTO professori (nome,cognome,cf,settore)
VALUES ("Fernando","Grigi","GRGFRN87A19L424Z","INF/01");
```

Inserimento dati: professori

<u>Matricola</u>	Nome	Cognome	Codice Fiscale	Settore
0035	Marco	Rossi	RSSMRC75A24L424K	INF/01
0036	Carlo	Verdi	VRDCRL68A13L424Y	INF/01
0030	Marta	Casali	CSLMRT92A65L424Y	INF/01
0031 0037	Fernando	Grigi	GRGFRN87A19L424Z	INF/01

Popolamento del database

[unidb.sql](https://github.com/mauro-farina/database-exercises) → <https://github.com/mauro-farina/database-exercises>

- 301 studenti
- 52 professori
- 31 corsi
- 2203 esami

Per caricare lo script...

1. Copia-incolla, oppure
2. Da MySQL Workbench: File > Open SQLScript

Query sul Database

Soluzioni

Una volta risolto l'esercizio, inviate la vostra soluzione

- <https://forms.gle/uNJYeCpu2wjUhJeX9>



Esercizio 1

Ritornare il numero di ragazze iscritte a ingegneria

- La matricola degli studenti di ingegneria inizia con IN
- Nel codice fiscale delle donne, la data di nascita viene modificata aggiungendo 40 al **giorno di nascita**
 - e.g., nata il 12 maggio → 52

Esercizio 1: soluzione

Ritornare il numero di ragazze iscritte a ingegneria

```
SELECT count(*) FROM studenti  
WHERE matricola LIKE 'IN%' AND  
substring(cf, 10, 1) IN ('4', '5', '6', '7');
```

Esercizio 1: soluzione

Ritornare il numero di ragazze iscritte a ingegneria

```
SELECT count(*) FROM studenti  
WHERE matricola LIKE 'IN%' AND  
substring(cf, 10, 1) BETWEEN '4' AND '7';
```

Esercizio 1: soluzione

Ritornare il numero di ragazze iscritte a ingegneria

```
SELECT count(*) FROM studenti
WHERE matricola LIKE 'IN%' AND (
    cf LIKE '_____4%' OR cf LIKE '_____5%'
    OR cf LIKE '_____6%' OR cf LIKE '_____7%'
);
```

Esercizio 2

Aggiungere la colonna *genere* alla tabella **studenti**. Questa colonna può assumere solo uno dei seguenti valori: "M", "F". **Non può essere NULL.**

- Promemoria: **ALTER TABLE <nomeTabella> <azione>**

Esercizio 2: soluzione

Aggiungere la colonna *genere* alla tabella **studenti**. Questa colonna può assumere solo uno dei seguenti valori: "M", "F". Non può essere NULL.

1. `ALTER TABLE studenti ADD COLUMN genere CHAR(1) NOT NULL DEFAULT 'M';`
2. `ALTER TABLE studenti ADD CONSTRAINT CHECK(genere IN ('M', 'F'));`

Esercizio 3

Assegnare a ciascuno studente e professore il rispettivo attributo *genere*

Esercizio 3

Assegnare a ciascuno studente il rispettivo attributo *genere*

1. `SET SQL_SAFE_UPDATES=0;`

Dobbiamo **disabilitare** i safe updates poiché andremo ad aggiornare i valori delle istanze senza fare riferimento a esse tramite la chiave primaria.

Esercizio 3

Assegnare a ciascuno studente il rispettivo attributo *genere*

1. `SET SQL_SAFE_UPDATES=0;`
2. `UPDATE studenti`
`SET genere="F"`
`WHERE substring(cf, 10, 1) BETWEEN '4' AND '7';`

Esercizio 3

Assegnare a ciascuno studente e professore il rispettivo attributo *genere*

1. `SET SQL_SAFE_UPDATES=0;`
2. `UPDATE studenti`
`SET genere="F"`
`WHERE substring(cf, 10, 1) BETWEEN 4 AND 7;`
3. `SET SQL_SAFE_UPDATES=1;`

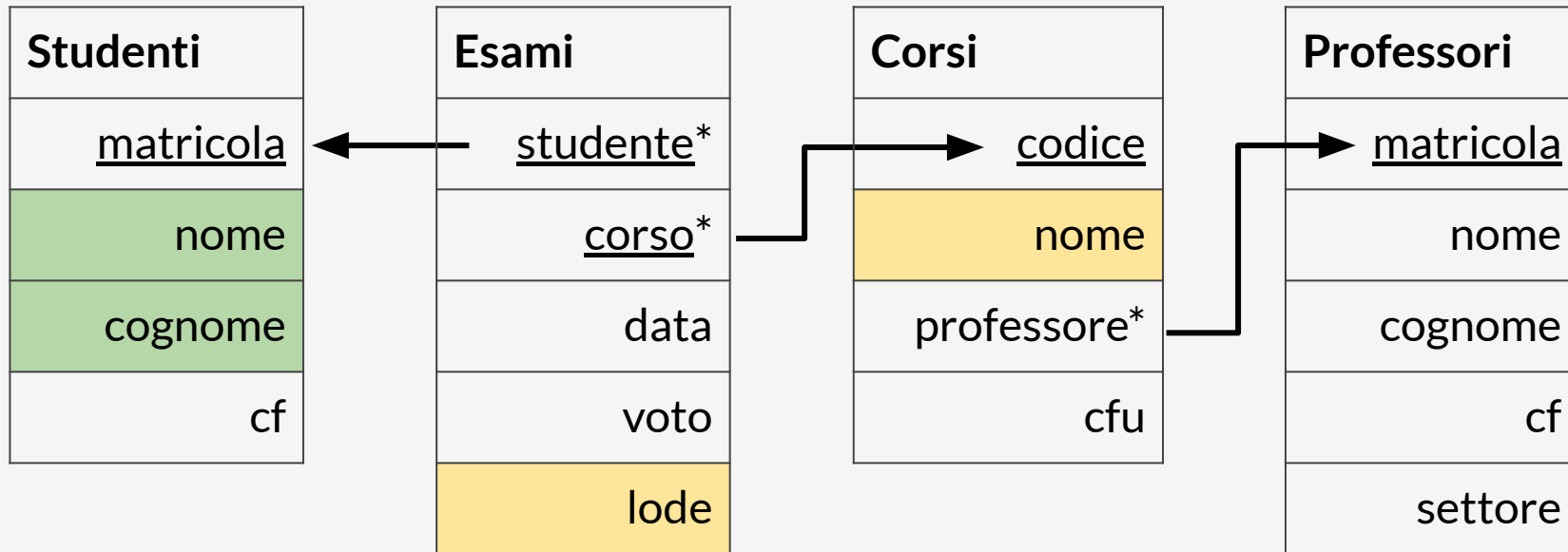
Ri-abilitiamo i safe updates

Esercizio 4

Elencare nome e cognome degli studenti che hanno preso 30L nell'esame di Basi di Dati

Esercizio 4

Elencare nome e cognome degli studenti che hanno preso 30L nell'esame di Basi di Dati



Esercizio 4: soluzione

Elencare nome e cognome degli studenti che hanno preso 30L nell'esame di Basi di Dati

```
SELECT s.nome, s.cognome
FROM studenti s
INNER JOIN esami e ON s.matricola = e.studente
INNER JOIN corsi c ON e.corso = c.codice
WHERE c.nome="Basi di Dati" AND e.lode IS TRUE;
```