ESERCIZIO BONUS 1 (+0.4 pt) – Implementazione di basi di dati relazionali con MySQL

21/10/2025, Consegna: 2/11/2025, ore 23.55

Modalità di consegna → Attraverso la piattaforma di e-learning VIRTUALE (https://virtuale.unibo.it) di UNIBO. Passi da svolgere:

1) Collegarsi a: https://virtuale.unibo.it, utilizzando le proprie credenziali istituzionali UNIBO (email/password) per l'accesso.
2) Scegliere il corso di Basi di dati della Laurea in Informatica per il Management, anno accademico 2025/2026 (codice: 70155).
3) Cliccare su Esercizio Bonus 1 → Consegna File, e procedere con l'upload del file contenente la soluzione dell'esercizio. E' possibile ripetere l'operazione di upload.
4) Quando si vuole consegnare l'elaborato DEFINITIVO, cliccare su "Consegna Compito". Da questo momento, nessun ulteriore upload è possibile. **Se non si clicca su "Consegna Compito", il docente NON vede l'elaborato**.
5) La sottomissione è possibile dal 21/10/2025 al 2/11/2025, ore 23.55.

COSA CONSEGNARE?

- Allegare un singolo file `consegna1_#MATRICOLA.sql`, contenente tutto il codice SQL che implementa le specifiche dell'esercizio descritto a seguire. **NON USARE `sqldump`**!

---

0) Utilizzando il software MySQL, **costruire il database** `PIANI_STUDIO`. Il database tiene traccia di studenti iscritti a corsi di laurea di UNIBO. Nello specifico, il database supporta un'applicazione che gestisce la compilazione dei piani di studio e la verbalizzazione dei voti degli insegnamenti svolti.

Il database `PIANI_STUDIO` è composto dalle seguenti tabelle (usare il table engine: INNODB):

STUDENTE(Matricola, CodiceCL, Nome, Cognome, CFUConseguiti)
CORSOLAUREA(Codice, Nome, NomeDip, NumeroProgrammato)
DIPARTIMENTO(Nome, Sede, NumeroDocenti)
INSEGNAMENTO(Id, Nome, CodiceCL, CFU, SSD, NumIscritti, MaxIscritti)
PIANOSTUDI(Matricola, IdInsegnamento, Data)
RICHIESTE_RIFIUTATE(Matricola, IdInsegnamento, Data)
VERBALIZZAZIONE(Matricola, IdInsegnamento, Data, Voto)

Vincoli sui dati:

- Vincolo di integrità referenziale tra STUDENTE.CodiceCL e CORSOLAUREA.Codice
- Vincolo di integrità referenziale tra INSEGNAMENTO.CodiceCL e CORSOLAUREA.Codice
- Vincolo di integrità referenziale tra CORSOLAUREA.NomeDip e DIPARTIMENTO.Nome
- Vincolo di integrità referenziale tra PIANOSTUDI.Matricola e STUDENTE.Matricola
- Vincolo di integrità referenziale tra PIANOSTUDI.IdInsegnamento e INSEGNAMENTO.Id
- Vincolo di integrità referenziale tra RICHIESTE_RIFIUTATE.Matricola e STUDENTE.Matricola
- Vincolo di integrità referenziale tra RICHIESTE_RIFIUTATE.IdInsegnamento e INSEGNAMENTO.Id
- Vincolo di integrità referenziale tra VERBALIZZAZIONE.Matricola e STUDENTE.Matricola
- Vincolo di integrità referenziale tra VERBALIZZAZIONE.IdInsegnamento e INSEGNAMENTO.Id
- CORSOLAUREA.Nome non ammette duplicati e non può essere NULL

- CORSOLAUREA.NomeDip non può essere NULL
- Tutti i campi di tipo stringa devono essere VARCHAR con lunghezza massima 30 caratteri.
- STUDENTE.CFUConseguiti è un intero, con valore di default pari a 0.
- CORSOLAUREA.NumeroProgrammato (type `ENUM`) può assumere solo due valori: "Attivo" "Nonattivo".
- INSEGNAMENTO.SSD (type `ENUM`) può assumere solo cinque valori: "INF", "ING-INF", "SECS", "IUS", "MAT".
- I campi che si chiamano Data (presenti in tre tabelle) sono di tipo Date.
- VERBALIZZAZIONE.Voto NON può essere NULL.
- Se rimuovo uno STUDENTE, cancello anche tutte le righe nella tabella PIANOSTUDI, RICHIESTE_RIFIUTATE e VERBALIZZAZIONE che fanno riferimento a quello studente.
- Se rimuovo un INSEGNAMENTO, cancello anche tutte le righe nella tabella PIANOSTUDI, RICHIESTE_RIFIUTATE e VERBALIZZAZIONE che fanno riferimento a quell'insegnamento.

Costruire il database `PIANI_STUDIO`, rimuovendo prima (da codice) ogni altro database pre-esistente con lo stesso nome presente nell'installazione corrente di MYSQL.

1) Popolare il contenuto delle tabelle STUDENTE, CORSOLAUREA, INSEGNAMENTO, DIPARTIMENTO caricando i dati dai file `studenti.txt`, `cdl.txt`, `insegnamenti.txt`, `dipartimenti.txt`. I file sono disponibili sulla piattaforma Virtuale. Il caricamento dei dati di una tabella **DEVE** avvenire attraverso istruzioni di `INSERT` ripetute.

2) Implementare le seguenti **Stored Procedure**:

a) `InserisciPartecipazione(IN Mat VARCHAR(30), IN IdIns VARCHAR(30))` → → Verifica se Mat fa riferimento ad una matricola esistente nella tabella STUDENTE, ed InIns ad un insegnamento valido nella tabella INSEGNAMENTO. Se tali condizioni sono soddisfatte:
Se IdIns si riferisce ad un insegnamento di un Corso di Laurea con numero programmato pari ad "Nonattivo", inserisce una riga dentro la tabella PIANOSTUDI, con Matricola pari a Mat, e IdInsegnamento pari a IdIns. Viceversa, Se IdIns si riferisce ad un insegnamento di un Corso di Laurea con numero programmato pari ad "Attivo": verifica se, per l'insegnamento con Id pari a IdIns, il campo NumIscritti è inferiore a MaxIscritti. In tal caso, inserisce una riga dentro la tabella PIANOSTUDI, con Matricola pari a Mat, e IdInsegnamento pari a IdIns. Il campo Data va settato alla data attuale. Vice versa, se, per l'insegnamento con Id pari a IdIns, il campo NumIscritti è uguale o superiore a MaxIscritti, inserire una riga dentro la tabella RICHIESTE_RIFIUTATE, con Matricola pari a Mat, e IdInsegnamento pari a IdIns. Il campo Data va settato alla data attuale

b) `InserisciVerbalizzazione(IN Mat VARCHAR(30), IN Ins VARCHAR(30), Vt INT)` → Verifica se Mat ed Ins fanno riferimento ad una riga presente nella tabella PIANOSTUDI. In tal caso, aggiunge una riga nella tabella VERBALIZZAZIONE, con voto pari a Vt. Il campo Data va settato alla data attuale.

c) `CancellaStorico()` → Rimuove tutte le righe presenti nella tabella RICHIESTE_RIFIUTATE.

d) `RimuoviStudente(IN Mat VARCHAR(30))` → Rimuove dalla tabella STUDENTE la riga relativa allo studente con matricola pari a `Mat`.

3) Implementare i seguenti **trigger**:

a) `IncrementaPartecipanti` → ***Dopo*** *ogni inserimento nella tabella* PIANOSTUDI relativo ad un certo insegnamento con id pari a IdInsegnamento, modifica automaticamente il campo NumIscritti associato a quell'id nella tabella INSEGNAMENTO, incrementandolo di 1 unità.

b) `DecrementaPartecipanti` → ***Dopo*** *ogni rimozione nella tabella* PIANOSTUDI relativo ad un certo insegnamento con id pari a IdInsegnamento, modifica automaticamente il campo NumIscritti associato a quell'id nella tabella INSEGNAMENTO, decrementandolo di 1 unità.

c) `Incremento CFU` → ***Dopo*** *ogni inserimento nella tabella* VERBALIZZAZIONE relativo ad un certo riga con id dell'insegnamento pari a IdInsegnamento e matricola pari a Mat, incrementa automaticamente il campo CFUConseguiti nella tabella STUDENTE nella riga relativa alla matricola uguale a Mat. L'incremento è pari al numero di CFU dell'insegnamento con id pari a IdInsegnamento, presente nella tabella INSEGNAMENTO.

NOTA: Nel file di consegna, il codice dei trigger deve essere collocato **PRIMA** delle istruzioni SQL di INSERT dei dati (punto 1).

4) Implementare le seguenti **viste**:

a) `LISTA_DIP_CON_INFORMATICA`(Nome, Sede) → restituisce nome e sede dei dipartimenti che hanno corsi di laurea che a loro volta contengono insegnamenti con SSD pari a INF.

b) `LISTA_STUDENTI_INATTIVI` (Mat, CodiceCL) → restituisce la lista degli studenti che NON hanno alcun insegnamento nel loro piano di studi.

c) `LISTA_INSEGNAMENTI_TOP` (Id, Nome, CodiceCL) → restituisce l'insegnamento/ insegnamenti che ha / hanno ricevuto più richieste da parte degli studenti. Le richieste si ottengono sommando, per ciascun insegnamento, il numero di volte in cui tale insegnamento appare in un piano di studi (richieste accettate), al numero di volte in cui appare nelle richieste rifiutate.

d) `LISTA_AREE_TOP` (ssd) → restituisce la lista degli SSD che compaiono in almeno 3 insegnamenti presenti nei piani di studio degli studenti con CodiceCL = 8014.

NOTA. Per l'implementazione delle viste, è consentito costruire altre viste o CTE di appoggio.

VINCOLI DI CONSEGNA

- L'implementazione della base di dati deve essere **completa** e **funzionante**.
- **Consegne prodotte mediante strumenti automatici di generazione di testo/codice NON saranno valutate.**
- **NON sono consentite consegne di gruppo. Consegne multiple** (ossia stesso codice da parte di più studenti, o codici molto simili) **NON sono valutate. Sono utilizzati strumenti automatici per il controllo di similarità del codice.**
- Inserire ove possibile dei **commenti** nel codice (in MySQL i commenti si inseriscono tramite i caratteri speciali: #Commento su una riga o `/*` Commento su più righe `*/`).
- **Seguire ALLA LETTERA le specifiche del testo per quanto riguarda i nomi** del database, tabelle, attributi, viste, stored procedure, etc.
- La consegna deve avvenire attraverso la piattaforma indicata nella prima pagina, ed entro la deadline stabilita. **Consegne via email NON saranno valutate**. **Consegne in ritardo o in bozza NON saranno valutate**.
- L'assegnamento del **bonus** è una funzione booleana (assegnato/non assegnato).
- E' possibile testare l'implementazione dei vari punti mediante un programma di test (`test1.sql`) messo a disposizione sulla piattaforma Virtuale.