Esercitazione 4
SQL

c.vallati@iet.unipi.it

DB Riferimento

Il database di riferimento è il database UNIPI, creare una nuova istanza di database sul server e caricare i dati

(se si usa le postazioni del laboratorio creare il database UNIPI con un nome univoco)

CREATE TABLE

1. Creare una tabella studenti nel database unipi per rappresentare gli studenti immatricolati con la seguente struttura:
   studenti(Matricola, Nome, Cognome, DataIscrizione)
   Rappresentare la matricola tramite una stringa a lunghezza fissa di 6 caratteri, Nome e Cognome tramite stringhe a lunghezza variabile (lunghezza massima 20) e DataIscrizione come una data. Imporre il vincolo di chiave primaria sull'attributo Matricola e imporre il vincolo di unicita sugli attributi Nome e Congnome. Imporre un controllo sulla data di iscrizione successiva al 1980-01-01.
2. Creare una tabella corsi nel database unipi per rappresentare i corsi disponibili presso l'ateneo:
   corsi(Codice, NomeCorso, MatricolaDocente)
   Rappresentare il Codice attraverso un Integer, il NomeCorso attraverso una stringa a lunghezza variabile (lunghezza massima 20) e la MatricolaDocente come una stringa a lunghezza variabile (lunghezza massima 6). Imporre il vincolo di chiave su Codice, vincolo di unique su Nome e vincolo NOT NULL e un vincolo di chiave esterna su MatricolaDocente. La reference deve essere riferita al campo Matricola della relazione docenti.
3. Creare una tabella esami nel database unipi per rappresentare gli esami sostenuti dagli studenti:
   esami(CodiceCorso, MatricolaStudente, DataEsame, Annotazioni)
   Imporre gli appropriati vincoli di chiave esterna. Imporre il vincolo NOT NULL sulla Data. Imporre un controllo sulla data in cui l'esame e' sostenuto che non deve essere precedente alla data di iscrizione dello stundente.

CREATE TABLE Es1

studenti(Matricola, Nome, Cognome, DataIscrizione)

```
CREATE TABLE studenti(
Matricola CHAR(6),
Nome VARCHAR(20),
Cognome VARCHAR(20),
DataIscrizione DATE,
CONSTRAINT studenti_pk PRIMARY KEY (Matricola),
CONSTRAINT studenti_un UNIQUE(Nome, Cognome),
CONSTRAINT studenti_ch CHECK(DataIscrizione > '1980-01-01')
)
```

CREATE TABLE Es2

corsi(Codice, Nome, MatricolaDocente)

```
CREATE TABLE corsi(
Codice INTEGER,
NomeCorso VARCHAR(20),
MatricolaDocente VARCHAR(6) NOT NULL,
CONSTRAINT corsi_pk PRIMARY KEY (Codice),
CONSTRAINT corsi_un UNIQUE(NomeCorso),
CONSTRAINT corsi_fk FOREIGN KEY(MatricolaDocente)
REFERENCES docenti(Matricola)
)
```

CREATE TABLE Es3

esami(CodiceCorso, MatricolaStudente, DataEsame)

```
CREATE TABLE esami(
CodiceCorso Integer,
MatricolaStudente VARCHAR(6),
DataEsame Date NOT NULL,
Annotazioni TEXT,
CONSTRAINT esami_fk1 FOREIGN KEY(CodiceCorso)
REFERENCES corsi(Codice),
CONSTRAINT esami_fk2 FOREIGN KEY(MatricolaStudente)
REFERENCES studenti(Matricola),
CONSTRAINT esami_ch CHECK( studenti.DataIscrizione <
DataEsame )
)
```

TEXT è un tipo di dato alternativo a VARCHAR usato per memorizzare testi di grandi dimensioni di cui non è possibile sapere la lunghezza a priori

NOTA: Un attributo al quale viene imposto il vincolo di chiave esterna deve avere lo stesso dominio di quello riferito.

INSERT INTO

Inserire i valori seguenti dentro le tabelle studenti:

```
insert into studenti (Matricola, Nome, Cognome, DataIscrizione) values ('369871', 'Brandon', 'Graham', '2015-08-04');
insert into studenti (Matricola, Nome, Cognome, DataIscrizione) values ('515140', 'Heather', 'Holmes', '2015-09-27');
insert into studenti (Matricola, Nome, Cognome, DataIscrizione) values ('090456', 'Kevin', 'Mills', '2016-01-16');
insert into studenti (Matricola, Nome, Cognome, DataIscrizione) values ('579555', 'Stephanie', 'Snyder', '2015-08-12');
insert into studenti (Matricola, Nome, Cognome, DataIscrizione) values ('018701', 'Bobby', 'Gibson', '2015-08-15');
insert into studenti (Matricola, Nome, Cognome, DataIscrizione) values ('582320', 'Craig', 'Burton', '2016-01-20');
```

INSERT INTO

Cosa succede se inseriamo il seguente dato?

insert into studenti (Matricola, Nome, Cognome, DataIscrizione) values ('582321', ’Craig', 'Burton', '1989-01-20');

E questa?

insert into studenti (Matricola, Nome, Cognome, DataIscrizione) values ('582321', 'Jessie', 'Burton', '1979-01-20');

MySQL e CHECK

- “The CHECK clause is parsed but ignored by all storage engines.”, MySQL manual (versione 5.5)
- http://dev.mysql.com/doc/refman/5.5/en/create-table.html

- Il controllo CHECK e’ invece controllato effettivamente dalla maggior parte degli altri sistemi DBMS

INSERT INTO

Inserire i valori seguenti dentro le tabelle corsi:

```
insert into corsi (Codice, NomeCorso, MatricolaDocente) values (1, ’Matematica', '014500');
insert into corsi (Codice, NomeCorso, MatricolaDocente) values (2, ’Informatica', '004178');
insert into corsi (Codice, NomeCorso, MatricolaDocente) values (3, ’Letteratura Greca', '000485');
insert into corsi (Codice, NomeCorso, MatricolaDocente) values (4, ’Medicina I', '010277');
insert into corsi (Codice, NomeCorso, MatricolaDocente) values (5, ’Fisica I', '020150');
insert into corsi (Codice, NomeCorso, MatricolaDocente) values (6, ’Costruzioni', '002473');
```

INSERT INTO

Provare ad aggiungere:

insert into corsi (Codice, NomeCorso, MatricolaDocente) values (7, 'Basi Di Dati', '001111');

Oppure

insert into corsi (Codice, NomeCorso, MatricolaDocente) values (5, ’Costruzioni II’, '002473');

Qual’e’ il risultato?

INSERT INTO

1. Aggiungere un esame di matematica a tutti gli studenti che si sono iscritti dopo il 1 Settembre 2015, l'esame e' verbalizzato il 1 Giugno 2016
2. Aggiungere un esame di Fisica I a tutti quelli che hanno la lettera 'h' nel nome. L'esame viene verbalizzato con la data di oggi. A tal fine usare il costrutto CURDATE() che ritorna la data odierna.

INSERT INTO Es1

```
INSERT INTO esami(CodiceCorso, MatricolaStudente, DataEsame)
SELECT 1, Matricola, '2016-06-01' FROM studenti
WHERE DataIscrizione > '2015-09-01';
```

Oppure

```
INSERT INTO esami(CodiceCorso, MatricolaStudente, DataEsame)
SELECT (SELECT Codice FROM corsi
        WHERE NomeCorso = 'Matematica'),
        Matricola, '2016-06-01'
FROM studenti
WHERE DataIscrizione > '2015-09-01';
```

INSERT INTO Es2

```
INSERT INTO esami(CodiceCorso, MatricolaStudente, DataEsame)
SELECT 5, Matricola, CURDATE() FROM studenti
WHERE Nome LIKE '%h%';
```

Oppure

```
INSERT INTO esami(CodiceCorso, MatricolaStudente, DataEsame)
SELECT (SELECT Codice FROM corsi
        WHERE NomeCorso = 'Fisica I'),
        Matricola, CURDATE() FROM studenti
WHERE Nome LIKE '%h%';
```

ALTER TABLE

1. Aggiungere la colonna Voto alla tabella esame, il tipo del nuovo valore deve essere Integer e NOT NULL. Impostare come valore di default 18.
2. Aggiungere una colonna Lode alla tabella esame, il nuovo tipo deve essere un solo carattere che puo' assumere 'L' o NULL.
3. Aggiungere un controllo imponendo che la lode possa essere 'L' solo se il voto e' pari a 30.
4. Rimuovere tale controllo e poi rimuovere la colonna Lode

ALTER TABLE

ALTER TABLE esami ADD COLUMN Voto Integer NOT NULL DEFAULT 18

ALTER TABLE esami ADD COLUMN Lode char(1) DEFAULT NULL CHECK(Lode IN ('L', NULL))

ALTER TABLE esami ADD CONSTRAINT ch CHECK (Voto = 30 AND Lode = 'L' OR Voto < 30 AND Lode IS NULL)

ALTER TABLE esami DROP INDEX ch

ALTER TABLE esami DROP COLUMN Lode

NOTA: in alcune versioni la ADD CONSTRAINT ch CHECK e la DROP INDEX ch possono generare un errore.

UPDATE/DELETE

1. Alzare il voto di 10 punti a tutti gli esami sostenuti da studenti con matricola dispari
2. Alzare il voto di 5 punti a tutti gli studenti con cognome che comincia con la letter ‘H’
3. Cancellare gli esami a tutti gli studenti che hanno sostenuto l’esame con la professoressa Scherbatsky
4. Aggiornare la data di assunzione di tutti I docenti portandola avanti di 15 giorni

NOTA: Per abilitare gli update e le delete di questo tipo (che non contengono una chiave primaria nella condizione where) dentro il workbench andare in Preferences -> SQL Editor e togliere la spunta sulla voce “Safe Updates” e poi chiudere e riaprire.

UPDATE/DELETE

UPDATE esami SET Voto = Voto + 10 WHERE MatricolaStudente % 2 = 1

UPDATE esami SET Voto = Voto + 10 WHERE MatricolaStudente IN (SELECT Matricola FROM studenti WHERE Nome LIKE 'H%')

DELETE FROM esami WHERE CodiceCorso IN
(SELECT Codice FROM corsi INNER JOIN docenti ON Matricola = MatricolaDocente WHERE Cognome = 'Scherbatsky')

NOTA: In questo caso la Matricola in docenti e MatricolaDocente in corsi hanno nome diverso, quindi non puo' essere usato un NATURAL JOIN.

UPDATE docenti SET DataAssunzione = DataAssunzione + INTERVAL 15 DAY

ON DELETE/ON UPDATE

1. Aggiungere una colonna Tutor alla tabella studenti. La colonna deve contenere la matricola del docente che fa tutoraggio nei confronti dello studente.
2. Aggiungere il vincolo di chiave esterna facendo in modo che se il docente viene cancellato la matricola del Tutor viene impostata a null.
3. Impostare come Tutor di tutti gli studenti il Prof. Underwood.
4. Cancellare il Prof. Underwood.

ON DELETE/ON UPDATE

1. ALTER TABLE studenti ADD COLUMN Tutor VARCHAR(6)
2. ALTER TABLE studenti ADD CONSTRAINT fk FOREIGN KEY (Tutor) REFERENCES docenti(Matricola) ON UPDATE SET NULL ON DELETE SET NULL
3. UPDATE studenti SET Tutor = '030405’
   Oppure
   UPDATE studenti SET Tutor = (SELECT Matricola FROM docenti WHERE Cognome = 'Mosby')
4. UPDATE studenti SET Tutor = '030405’

View

1. Creare una vista che mostri solo i nomi e i cognomi dei docenti che sono stati assunti dopo il 2013
2. Creare una vista che mostri solo i nomi e i codici dei dipartimenti che hanno più costi che finanziamenti
3. Creare una vista che mostri il codice del dipartimento e la media del costo di ogni ora di lezione (Rapporto Stipendio/OreLezione)
4. Creare una vista che mostri solo i dipartimenti in cui la somma degli stipendi dei docenti supera la meta del finanziamento totale

Viste Es1

```
CREATE VIEW DocentiNeoAssunti (Nome, Cognome) AS
SELECT Nome, Cognome
FROM docenti
WHERE DataAssunzione > '2013-01-01‘
```

Viste Es2

```
CREATE VIEW DipartimentiRosso (Codice, Nome) AS
SELECT CodiceDipartimento, NomeDipartimento
FROM dipartimenti
WHERE CostiTotaliAnnui > FinanziamentoTotaleAnnuo
```

Viste Es3

```
CREATE VIEW CostoOra (CodiceDipartimento, CostoMedio) AS
SELECT CodiceDipartimento,
        AVG(StipendioAnnuo / OreLezioneAnnue)
FROM docenti GROUP BY CodiceDipartimento
```

Viste Es4

```
CREATE VIEW DipartimentiAltoCostoDocenti AS
SELECT * FROM dipartimenti AS d
WHERE FinanziamentoTotaleAnnuo/2 <
        (SELECT SUM(StipendioAnnuo)
         FROM docenti WHERE CodiceDipartimento =
        d.CodiceDipartimento);
```