OPEN DATA E DBMS

LINGUAGGIO SQL
CREAZIONE E MODIFICA DI SCHEMI

SISTEMI INFORMATIVI E DBMS

CORSO DI LAUREA MAGISTRALE IN
MANAGEMENT E MONITORAGGIO DEL TURISMO SOSTENIBILE

PROF. ANDREA PINNA

AA 2019/2020

SQL INTRODUZIONE

Il linguaggio SQL è il linguaggio per le basi di dati relazionali più diffuso.

Permette di:

- Definire gli schemi relazionali e i vincoli di integrità (Data Definition Language)
- Interrogare e aggiornare i dati delle istanze della bade di dati (Data Manipulation Language)

SQL INTRODUZIONE

Il linguaggio SQL ha origine nel 1974, quando venne proposto il linguaggio SEQUEL.

Il nome deriva dall'acronimo di Structured Query Language (liguaggio strutturato per le interrogazioni).
I DBMS relazionali incorporano un interprete per una specifica versione (dialetto) del linguaggio SQL.

SQL INTRODUZIONE

In questo corso useremo il DBMS MySQL che utilizza la sua versione del linguaggio SQL

Scaricare il pacchetto per il proprio SO da questo link ed installarlo sulla propria macchina.

`https://dev.mysql.com/downloads/mysql/`

SQL INTRODUZIONE

Le istruzioni SQL vengono inviate all'interprete del DBMS e si traducono in operazioni sui dati del database.

Il tool Workbench di MySQL consente di scrivere codice in linguaggio SQL e di vedere l'effetto delle istruzioni.

In molti linguaggi di programmazione come Python è possibile incorporare codice SQL sotto forma di stringa e trasmetterlo al DBMS tramite i connettori, sia per creare o modificare i dati , sia per estrarre i dati del DBSM.

CREAZIONE DATABASE

La creazione di nuovi database e la definizione di schemi riguarda la parte DDL del linguaggio.
In generale, un DBMS consente di gestire più di un database. Possiamo creare un database con l'istruzione:

```
CREATE DATABASE [nomeDB]
```

Per convenzione (ma non è obbligatorio) le parole del linguaggio vanno scritte maiuscole.

CREAZIONE DATABASE

Esempio: creiamo il database “balneari”

```
CREATE DATABASE balneari
```

Possiamo specificare che vogliamo creare il database soltanto se non è già stato creato. Per farlo scriviamo l’espressione condizionale `IF NOT EXISTS`

```
CREATE DATABASE IF NOT EXISTS balneari
```

CREAZIONE DATABASE

Con la parola USE posso indicare al DBMS quale dei database utilizzare.

```
USE balneari
```

Dopo questa istruzione è possibile interagire con il database “balneari”

CREAZIONE TABELLE

Il modello relazionale si basa sulla definizione degli schemi delle relazioni (le tabelle con i cari campi). Per creare una relazione si usa il costrutto CREATE TABLE


```
CREATE TABLE [nome tabella](
   definizione attributi e vincoli
)
```

La definizione della tabella comprende la definizione di tutti gli attributi e vincoli.

CREAZIONE TABELLE

La definizione degli attributi prevede la sintassi:

```
nomeAttributo DOMINIO VINCOLI
```

Esempio: creo una tabella con tre attributi di tre diversi domini ed esamino il risultato con DESCRIBE.

```
CREATE TABLE localita(
   codice CHAR(6),
   nominativo VARCHAR(20) NOT NULL,
   indirizzo  VARCHAR(100) NOT NULL
);
DESCRIBE localita;
```

virgola

punto e virgola

CREAZIONE TABELLE: DOMINI

SQL include diversi domini "elementari":

Numeri (interi e reali)

Stringe di testo (lunghezza fissa o variabile),

Data, ora, intervalli temporali,

Booleani,

Dati in formato binario (utili per inserire immagini)

CREAZIONE TABELLE: DOMINI

```
CREATE TABLE localita(
    codice CHAR(6),
    nominativo VARCHAR(20) NOT NULL,
    indirizzo  VARCHAR(100) NOT NULL,
    PRIMARY KEY (codice)
)
```

CHAR: l'attributo è una stringa di testo di 6 caratteri. Il numero tra parentesi indica il numero massimo di caratteri ammessi.

VARCHAR: l'attributo viene registrato con un numero variabile di caratteri. Il numero tra parentesi è il numero massimo di caratteri ammessi.

CREAZIONE TABELLE: VINCOLI

SQL permette di implementare vincoli sui dati presenti nella realtà che vogliamo rappresentare.

In alcune versioni di SQL si può utilizzare l'istruzione CREATE DOMAIN per creare domini personalizzati.

Su MySQL non è possibile creare domini personalizzati per cui i vincoli vanno specificati nella definizione dell'attributo.

CREAZIONE TABELLE: VINCOLI

Esempio: creiamo la tabella voti che contiene un solo attributo "voto." Il voto è un numero intero e non può essere nullo e il voto deve essere compreso tra 18 e 30.
Uso la parola chiave NOT NULL e la parola CHECK

```
CREATE TABLE voti(
   voto INT NOT NULL CHECK(voto >= 18 AND voto <=30)
    )
```

CREAZIONE TABELLE: VINCOLI

```
CREATE TABLE localita(
   codice CHAR(6)
   nominativo VARCHAR(20) NOT NULL UNIQUE,
   indirizzo  VARCHAR(100) NOT NULL
   PRIMARY KEY (codice)
)
```

NOT NULL: impone di dare un valore all'attributo.

UNIQUE: non possono esserci due tuple con la stesso valore di questo attributo,

PRIMARY KEY: impone che l'attributo (o l'insieme di attributi) è chiave primaria .

Questi vincoli sono detti vincoli intrarelazionali

CREAZIONE TABELLE: VINCOLI

Si può impostare un valore di default per un attributo. Se non viene inserito alcun valore per quell'attributo, questo prende il valore di default. Si usa la parola DEFAULT seguita dal valore.

Esempio: "salvo diversamente indicato, le località hanno ricevuto zero ispezioni."

```
CREATE TABLE localita(
    codice CHAR(6)
    nominativo VARCHAR(20) NOT NULL UNIQUE,
    indirizzo  VARCHAR(100) NOT NULL
    numeroIspezioni INT DEFAULT 0
    PRIMARY KEY (codice)
)
```

CREAZIONE TABELLE: VINCOLI

Si. possono esprimere vincoli di unicità su coppie o insiemi di valori. Si utilizza la parola UNIQUE seguita da una lista di attributi.

Esempio: due giocatori non possono avere contemporaneamente lo stesso nickname e la stessa email.

```
CREATE TABLE giocatori(
   codice CHAR(6),
   nickname VARCHAR(20) NOT NULL,
   email  VARCHAR(100) NOT NULL,
   punteggio INT DEFAULT 0,
   UNIQUE(nickname,email)
)
```

CREAZIONE TABELLE: VINCOLI

In SQL tutte le tabelle sono relazioni del modello relazionale.. Abbiamo visto che con il modello Entity-Relationship si possono rappresentare in maniera distinta le entità e le associazioni.

Una tabella SQL rappresenta una entità se la tabella non contiene vincoli di integrità referenziale ne vincoli di chiave esterna..

Con i vincoli di chiave esterna implementiamo una associazione.

Con i vincoli di integrità referenziale implementiamo le entità deboli.

CREAZIONE TABELLE: VINCOLI

Per riferire l'attributo ad un'altra tabella si utilizza REFERENCES. Esempio di sintassi:

```
CREATE TABLE Infrazioni(
  codice CHAR(6) NOT NULL
  data DATE NOT NULL,
  vigile INTEGER NOT NULL REFERENCES Vigili(Matricola),
  PRIMARY KEY (codice),
 )
```

Tabella alla quale si riferiscono i valori

Attributo della tabella esterna

Questa è una entità debole.

CREAZIONE TABELLE: VINCOLI

Per creare una associazione utilizzo le chiavi esterne.

Esempio di sintassi:

```
CREATE TABLE esami(
  codiceStudente INT NOT NULL,
  codiceCorso INT NOT NULL,
  voto INT NOT NULL CHECK(voto >= 18 AND voto <=30),
  dataEsame DATE NOT NULL,
  FOREIGN KEY (codiceStudente) REFERENCES studenti (matricola),
  FOREIGN KEY (codiceCorso) REFERENCES corsi (codice)
 )
```

Attributo di questa tabella

Tabella alla quale si riferiscono i valori

Attributo della tabella esterna

MODIFICA DELLE TABELLE

Una volta creato uno schema (una tabella) possiamo eliminarlo utilizzando l'istruzione DROP TABLE

Esempio: utilizzo il database registroComunale ed elimino la tabella Infrazioni.

```
/*Elimino la tabella*/
```

Questo è un commento nel codice

```
// altro commento
USE registroComunale;
DROP TABLE Infrazioni
```

NOTA: in un programma SQL le istruzioni si separano con il simbolo "punto e virgola".

MODIFICA DELLE TABELLE

Per modificare la struttura di uno schema già creato (gli attributi, i vincoli, ecc) utilizziamo le parole ALTER TABLE seguite dall'operazione richiesta.

Rinominare una tabella:

```
ALTER TABLE nome_tabella RENAME TO nuovo_nome_tabella;
```

Aggiungere un attributo

```
ALTER TABLE studenti ADD COLUMN email VARCHAR(60);
```

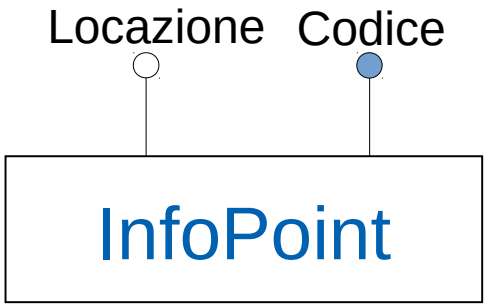
Modificare un attributo

```
ALTER TABLE studenti CHANGE email e-mail VARCHAR(30);
```

DAL DIAGRAMMA E-R A SQL

Il diagramma E-R ci permette di rappresentare il mondo reale tramite un costrutto grafico.

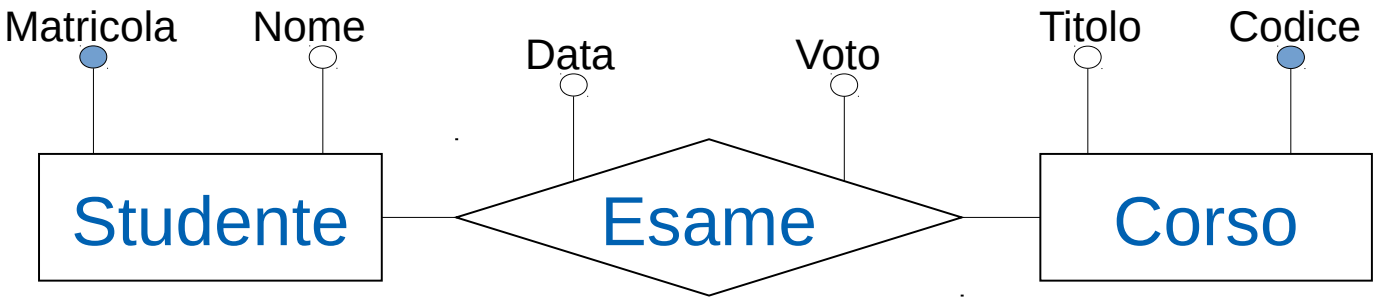
È possibile tradurre gli elementi del diagramma E-R in codice. Come passaggio intermedio è possibile trascrivere il diagramma nel suo schema relazionale in modo da esplicitare i vincoli (unique, not null ecc).




```
InfoPoint(Codice, Locazione)
    Locazione UNIQUE
```

DAL DIAGRAMMA E-R A SQL

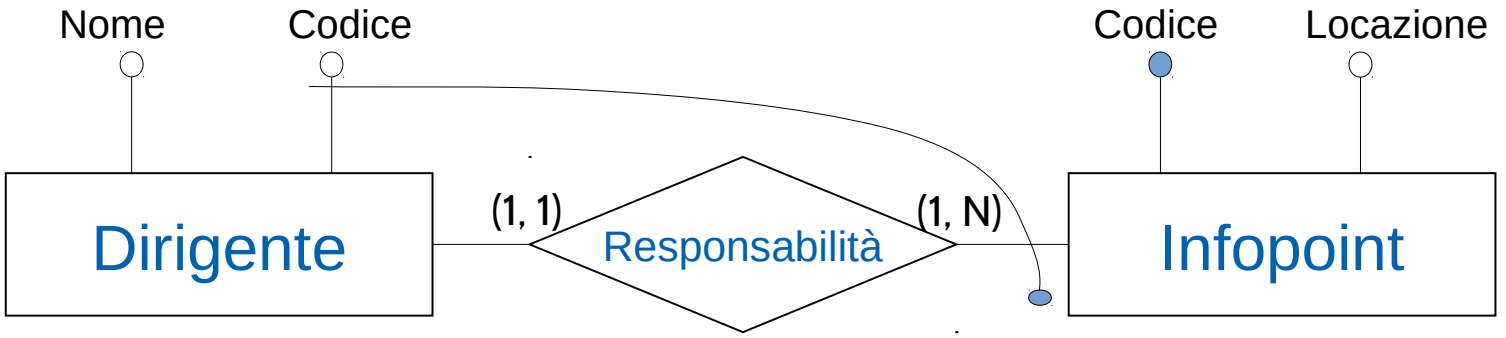
Lo schema relazionale permette di rappresentare i vincoli di chiave primaria (sottolineatura del nome dell'attributo) e i vincoli di chiave esterna (FK)




```
Studente(Matricola, Nome)
Corso(Codice, Titolo)
Esame(Matricola,CodiceEsame, Data, Voto )
    FK: Matricola REFERENCES Studente
    FK: CodiceEsame REFERENCES Corso
```

DAL DIAGRAMMA E-R A SQL

In presenza di entità deboli, l'associazione non viene rappresentata come relazione dello schema relazionale.




```
InfoPoint(Codice, Locazione)
    Locazione UNIQUE
Dirigente(Codice, CodiceInfoPoing, Nome)
    Nome NOT NULL
    FK: CodiceInfoPoint REFERENCES InfoPoint
```

DAL DIAGRAMMA E-R A SQL

Possiamo tradurre lo schema relazionale nel codice SQL necessario per creare la tabella.

```
Dirigente(Codice, CodiceInfoPoing, Nome)
    Nome NOT NULL
    FK: CodiceInfoPoint REFERENCES InfoPoint
```

```
CREATE TABLE Dirigente(
Codice CHAR(4) NOT NULL,
CodiceInfoPoint CHAR(6) NOT NULL,
Nome VARCHAR(30) NOT NULL,
PRIMARY KEY(Codice),
FOREIGN KEY (CodiceInfoPoint) REFERENCES InfoPoint (Codice)
);
```

INSERIMENTO DEI DATI

Il linguaggio SQL permette di aggiungere e modificare le ennuple in una tabella e di interrogare i dati inseriti (DML).

Per inserire una ennupla:

```
INSERT INTO nome_tabella (elenco attributi) VALUES
(elenco valori);
```

Esempio:

```
INSERT INTO giocatori(codice, nickname,  email) VALUES
('A123B7','Th3 M4st3r','ciccio99@supermail.em');
```

INSERIMENTO DEI DATI

La lista degli attributi è opzionale.

Se non viene esplicitata si fa riferimento a tutti gli attributi della relazione, secondo l'ordine con cui sono stati definiti.
Se viene inserita, la lista degli attributi e la lista dei valori devono essere coerenti (si rispetta l'ordine).

Se la lista di attributi non contiene tutti gli attributi della relazione viene inserito l'eventuale valore di default.
Altrimenti il valore nullo (se ammesso)

Se i dati non rispettano i vincoli, l'istruzione non viene eseguita

ELIMINAZIONE DEI DATI

Per eliminare una ennupla da una tabella si utilizza DELETE FROM e la parola WHERE per specificare cosa eliminare.

```
DELETE FROM nome_tabella WHERE [condizione]
```

Esempi:

```
DELETE FROM localita WHERE codice = 'LB1215';
```

```
DELETE FROM giocatori WHERE punteggio <= 10;
```

Su MySQL quest'ultima operazione viene interpretata come non sicura. Sono ammesse condizioni che valutano soltanto la chiave primaria.

ELIMINAZIONE DEI DATI

Esempio. Se codice è chiave primaria di giocatori:

```
DELETE FROM giocatori WHERE codice IN (
SELECT codice FROM giocatori WHERE punteggio <= 10
);
```

Vederemo più avanti il significato di SELECT.

MODIFICA DEI DATI

Per modificare i valori di una tabella si può utilizzare le parole UPDATE e SET. Sintassi:

```
UPDATE nomeTabella SET attributo = espressione WHERE
condizione
```

Esempio: aggiungi 1 al punteggio del giocatore 'A1234B'

```
UPDATE giocatori SET punteggio = punteggio + 1 WHERE
codice = 'A1234B'
```