**Lezione 5**

**SQL PER CREARE LE TABELLE E ALTRE ISTRUZIONI DDL**

In Questa lezione iniziamo a studiare il linguaggio che serve a gestire i database: l’**SQL**.

Anche se ACCESS lo genera automaticamente è bene conoscere le istruzioni più importanti.
Per prima cosa bisogna creare le tabelle di un database con l’SQL visto come DDL che crea la struttura o definizione dei dati.
Partiamo da un nuovo esempio: un database che gestisce registi che realizzano film appartenenti a case cinematografiche.

Il database si chiamerà “CINEMA.ACCDB” le tabelle: **regista, film, produzione.**Una possibile soluzione potrebbe essere quella sotto raffigurata da cui si possono vedere i campi e le associazioni.(un regista dirige più film, ogni casa cinematografica produce più film).



Per scrivere il codice SQL andare nell’editor di ACCESS.
Dal menù CREA cliccare su struttura query



Chiudere la finestra “mostra tabella”



E cliccare su sql:



**Un esempio di una tabella studenti contenente i principali tipi di campo:**

**CREATE TABLE Studenti
(
    IdStudente COUNTER PRIMARY KEY,
    Nominativo TEXT(60) NOT NULL,
    NatoIl DATETIME,
    Ripetente BIT,
    Eta BYTE,
    AnnoDiIscrizione SMALLINT,
    RedditoDiRiferimentoInLire LONG,
    ConsiderazioniSulloStudente LONGTEXT,
    Foto LONGBINARY,
    RedditoDiRiferimento DOUBLE,
    MediaVoti SINGLE,
    ImportoTassePagato CURRENCY
   )**



Per riprendere il nostro esempio il codice sql sarà:

**SQL PER CREARE TABELLA REGISTA**

create table regista (

cod\_reg char(4) primary key not null,

nome char(30),

nazione char(20),

vivente bit )

Dopo avere scritto il codice bisogna lanciare dal pulsante **esegui** l’esecuzione del codice stesso come da figura sotto:



**SQL PER CREARE TABELLA PRODUZIONE**

create table produzione (

cod\_casa\_cinematografica char(3) primary key not null,

denominazione char(40),

nazione char(30) )



**SQL PER CREARE TABELLA FILM**

create table film (

cod\_film char(4) primary key not null,

titolo char(30),

genere char(20),

incassi\_annui single,

cod\_reg char(4),

cod\_casa\_cinematografica char(3) ,

 foreign key (cod\_reg) references regista(cod\_reg),

 foreign key (cod\_casa\_cinematografica) references produzione(cod\_casa\_cinematografica) );



Questa tabella contiene due chiavi esterne (***cod\_reg*** e ***cod\_casa\_cinematografica)*** e il legame tramite SQL e stabilito nelle ultime due righe di codice da foreign key(chiave esterna).
per esempio **foreign key (cod\_reg) references regista(cod\_reg)** vuol dire
 il campo **cod\_reg** della tabella **film** è chiave esterna e si riferisce al campo **cod\_reg** (chiave primaria) della tabella **regista.**

Lo stesso vale per l’ultima riga di codice.
il campo **cod\_casa\_cinematografica** della tabella **film** è chiave esterna e si riferisce al campo **cod\_casa\_cinematografica** (chiave primaria) della tabella **produzione**.

Se aggiungiamo le clausole **ON DELETE CASCADE** e **ON UPDATE CASCADE** in fondo all’istruzione che diventerà:
create table film (

cod\_film char(4) primary key not null,

titolo char(30),

genere char(20),

incassi\_annui single,

cod\_reg char(4),

cod\_casa\_cinematografica char(3) ,

foreign key (cod\_reg) references regista(cod\_reg),

foreign key (cod\_casa\_cinematografica) references produzione(cod\_casa\_cinematografica)

**on delete cascade
on update cascade);**

vorrà dire che (**on delete cascade**) eliminerà i record correlati come chiave esterna se cancellassimo una chiave primaria e (**on update cascade**) aggiorna i record correlati se modificassimo una chiave primaria. La mancanza di tali opzioni corrisponde a **On delete no action** e **on update no action** per default. In questo caso è impedita la cancellazione della chiave primaria.