FONDAMENTI DI
BANCHE DATI

A cura di Lucio SANSONE

INDICE DEI CONTENUTI

DESCRIZIONE DI UNA BANCA DATI (BASE DI DATI) RELAZIONALE

1. INTRODUZIONE
2. RELAZIONI E BASI DI DATI RELAZIONALI
3. CHIAVI E DIPENDENZE FUNZIONALI
4. DIPENDENZE FRA TABELLE
5. VINCOLI DI INTEGRITÀ
6. NORMALIZZAZIONE
7. DESCRIZIONE DI UNA BASE DI DATI IN MS – ACCESS

GESTIONE DI UNA BASE DI DATI RELAZIONALE

8. INTERROGAZIONE DELLA BASE DI DATI ED ALGEBRA RELAZIONALE
9. QUERY SU UNA TABELLA: PROIEZIONE E SELEZIONE
10. QUERY SU PIÙ TABELLE: JOIN
11. SINTASSI SEMPLIFICATA DELLA SELECT
12. OPERATORI DI AGGREGAZIONE E RAGGRUPPAMENTO
13. USO DELLE QUERY NIDIFICATE
14. INSERIMENTO, CANCELLAZIONE, AGGIORNAMENTO

PROGETTAZIONE DELLA BASE DATI

15. MODELLO ENTITÀ-ASSOCIAZIONE (E/R) E PROGETTO CONCETTUALE
16. PROGETTO LOGICO E DESCRIZIONE DEL DATABASE RELAZIONALE

DESCRIZIONE DI UNA BASE DI DATI RELAZIONALE

1. INTRODUZIONE
2. RELAZIONI E BASI DI DATI RELAZIONALI
3. CHIAVI E DIPENDENZE FUNZIONALI
4. DIPENDENZE FRA TABELLE
5. VINCOLI DI INTEGRITÀ
6. NORMALIZZAZIONE
7. DESCRIZIONE DI UNA BASE DI DATI IN MS – ACCESS

1. INTRODUZIONE

BANCHE DATI, DBMS E MODELLI DEI DATI

Una banca dati o *base di dati* (*Data Base, DB)* è un **sistema integrato di archivi** (file) opportunamente organizzati per consentirne una facile consultazione in **condivisione** da parte di altri programmi oppure da parte direttamente degli utenti finali.

Il **software di gestione** di tale "sistema integrato di archivi" è organizzato in un apposito *Sistema di Gestione di Basi di Dati* (Data Base Management System - DBMS).

Tra i **modelli classici** delle basi di dati (gerarchico, reticolare e relazionale), viene presentato unicamente il modello relazionale il quale può essere riguardato come quello meglio formalizzato e più facile da usare.

ARCHITETTURA DEI DBMS

L'architettura di un sistema di gestione di base di dati, così come per la maggioranza del software, è suddivisa essenzialmente in due livelli:

- *livello interno,* che riguarda la struttura fisica degli archivi, la loro memorizzazione, le tecniche interne per la ricerca dei dati, l'efficienza del sistema e così via;

- *livello esterno,* che riguarda la struttura logica dei dati così come viene vista dagli utenti.

La teoria generale delle basi di dati tratta il **livello esterno**.

DBMS E LINGUAGGI PER LA GESTIONE

Un DBMS è un sistema attraverso il quale l'utente esegue le seguenti operazioni:

- Definizione dei dati del database,
- Manipolazione dei dati,
- Interrogazioni al data base.

A tale scopo esistono linguaggi del DBMS strutturati in sotto linguaggi che possono essere così classificati:

- Data Definition Language - DDL, che si rivolge al progettista della base di dati

- Data Manipulation Language - DML, che si rivolge al gestore della base di dati (Data Base Administator) e all'utente che vuole trarre informazioni dalla base di dati.

IL LINGUAGGIO SQL

Linguaggi specifici (detti di IV generazione) sono in genere associati ai sistemi commerciali che gestiscono le basi di dati e si confondono con essi: il linguaggio di DB2, SYBASE, ORACLE, INFORMIX e così via.

Uno standard de facto è il sotto linguaggio SQL (Structured Query Language), comune a tutti i suddetti linguaggi, nato come linguaggio di interrogazione ma contenente altresì le due sezioni per la manipolazione e la definizione. Esso si presenta in tre distinte modalità applicative:

- modalità *interattiva:* l'utente fornisce una singola interrogazione ed ottiene in risposta il risultato a video (oppure su stampante); l'utente puo' poi proseguire con altre attività sul DB;

- modalità *batch* o programmata: viene scritto un programma contenente più manipolazioni e/o interrogazioni e quando il programma viene lanciato vengono eseguite in sequenza le interrogazioni richieste e visualizzati i risultati;

- modalità *incorporata (embedded)*: SQL opera come sezione di un linguaggio di programmazione di uso generale, ad esempio C.

IL LINGUAGGIO QBE

Altra tecnica di interfaccia fra DBMS e operatore umano è la classe di linguaggi cosiddetti QBE (Query By Example), che realizzano le operazioni sul data base a mezzo di strumenti grafici, menù, tabelle ed icone.

In realtà, la tecnica tipica e dei pacchetti specializzati (linguaggi di quarta generazione) consiste nell'interfacciare il DBMS con un sistema grafico e quindi nel tradurre in linguaggio SQL le operazioni richieste dall'utente in modalità grafica.

Alcuni pacchetti DBMS rendono evidente, a richiesta, il testo SQL associato ad una operazione programmata graficamente.

2. RELAZIONI E BASI DI DATI RELAZIONALI

DEFINIZIONE DI RELAZIONE

Il modello di base di dati relazionale trova i suoi fondamenti teorici nel concetto matematico di *relazione* o corrispondenza fra insiemi.

Dati n insiemi $D_1,..., D_n$ (non necessariamente distinti), dicesi relazione su di essi un sottoinsieme R del prodotto cartesiano fra questi:

$$R \subseteq D_1 \times \ldots \times D_n$$

dove:
- $D_1,..., D_n$ sono i *domini* delle relazione;
- n è detto *grado* della relazione.

DEFINIZIONE DI RELAZIONE (2)

Sul piano concreto, la relazione è un oggetto matematico che pone in corrispondenza fra loro alcuni dati. Ad esempio, costituiscono una relazione i dati relativi alle generalità di una classe di individui:

generalità $\subseteq$ **stringa** $\times$ **stringa** $\times$ **data** $\times$ **stringa**

Un esempio della relazione generalità:

(Gennaro, Esposito, 1/1/70, Napoli)
(Ambrogio, Rossi, 1/2/73, Milano)
(Romolo, Romano, 25/12/67, Roma)

ATTRIBUTI

Ciascuno degli n domini della relazione corrisponde ad un distinto attributo che lo identifica nella relazione ed il suo valore appartiene ad un determinato tipo.

Con l'introduzione esplicita del concetto di *attributo* si risolvono anche eventuali ambiguità relative al medesimo dominio che appaia due o più volte nella relazione, ma con significati diversi. Una relazione sia ad esempio:

paternità $\subseteq$ **nome individuo** $\times$ **nome padre** $\times$ **...**

Il primo ed il secondo attributo appartengono entrambi al dominio *stringa*, ma hanno significati diversi: per evitare equivoci occorre fornire i nomi:

- nell'ordine definito dal prodotto cartesiano
- affiancandoli con i loro attributi; per dire ad esempio che Aldo è il padre di Carlo o si dice (nome individuo=Carlo, nome padre=Aldo, etc.).

RAPPRESENTAZIONE TABELLARE DI UNA RELAZIONE

Una relazione è tipicamente rappresentata in una tabella, della quale ciascuna colonna rappresenta un attributo (detto anche *campo*) e ciascuna riga detta ***tupla*** una ennupla del prodotto cartesiano

Rappresentazione tabellare di una relazione

nome	data di nascita	residenza	...
n1	d1	r1	...
n2	d2	r2	...
...	...	...	...
nk	dk	rk	...

- lo schema della tabella, o *schema della relazione*, cioè la lista dei nomi degli attributi che la caratterizzano e del tipo cui ciascuno di essi appartiene;
- i contenuti della tabella, o *istanza di relazione*, cioè le *tuple* di valori che definiscono una particolare relazione fra tutte quelle possibili aventi il medesimo schema.

Una base di dati relazionale (B.D.R.) è un insieme di tabelle (con assegnato schema) istanziate.

UN ESEMPIO DI BASE DI DATI

DATA BASE: EDITRICE

AUTORI

Nome	Data nasc.	Indir.	C.F.
N1	D1	I1	F1
N2	D2	I2	F2
N3	D3	I3	F3
N4	D4	I4	F4
N5	D5	I5	F5
N6	D6	N6	F6

LIBRI

Codice Libro	Titolo	Costo	Genere	Contr.
C1	T1	LC1	G1	CT1
C2	T2	LC2	G2	CT2
C3	T3	LC3	G1	CT3
C4	T4	LC4	G3	CT4
C5	T5	LC5	G2	CT5

AUTORI - LIBRI

Id_autore	Cod. libro
F1	C1
F1	C2
F1	C4
F2	C2
F2	C3
F3	C4
F4	C4
F5	C5

GENERE

Genere	Colloc.	Resp.
G1	L1	R1
G2	L2	R2
G3	L3	R3
G4	L4	R4

SPECIFICHE DEL DATABASE EDITRICE

La base di dati è costituita dalle relazioni:

- AUTORI, che contiene i dati anagrafici degli autori;

- LIBRI, che memorizza i libri stampati, identificati da un *codice,* il loro *titolo*, il *costo* di copertina, il *genere* cui appartengono (libro giallo, avventure, saggistica,...) e il *contratto* stipulato con gli autori;

- AUTORI-LIBRI, che pone in corrispondenza ciascun autore con i libri che egli ha scritto:

- GENERE: si suppone che la casa editrice abbia organizzato la produzione in generi, che ogni genere abbia un *responsabile* e che associata alla casa editrice vi sia una biblioteca, ove i testi sono collocati per generi: il genere G1 nel reparto L1 e così via.

Inoltre:

- ciascun autore può scrivere uno o più libri;
- ciascun libro può essere scritto da uno o più autori;

VALORE NULLO

Dicesi *nullo* un valore di un attributo, indicato convenzionalmente con NULL, non specificato in tabella.

Il fatto che una tabella contenga valori nulli può dipendere dal fatto che:

- essi non sono noti all'atto del caricamento dell'archivio (ad esempio, potrebbe non essere ancora definito il responsabile di un genere)

- da esigenze strutturali della tabella: il valore di un attributo può essere parziale (ad. es. il possesso della patente può essere definito solo per alcuni cittadini).

3.CHIAVI E DIPENDENZE FUNZIONALI

CHIAVE

Un insieme di attributi (a valori non nulli) è detto ***chiave*** di una relazione se:

- identifica *univocamente* una ennupla ;
- *non è ridondante*, cioè non esiste un suo sottoinsieme con la medesima proprietà.

In altri termini, se un insieme di attributi è una chiave, allora ciascuno dei suoi valori nella tabella è unico (non vi sono omonimi). Si noti che:

- la chiave può essere costituita da un solo attributo o può essere una *chiave multipla* cioè costituita da plù attributi;
- una relazione può, in generale, possedere **più chiavi** distinte (singole o multiple).

CHIAVE PRIMARIA

Ogni tabella del data base è caratterizzata da una ed una sola *chiave primaria,* cioè da una chiave che è stata scelta dal progettista come caratteristica della tabella.

ESEMPIO: DATA BASE EDITRICE

Supponendo inesistenti omonimie fra Autori, si hanno le seguenti chiavi (indicate in grassetto corsivo):

AUTORI (***Nome***, Data_nascita, Indirizzo, ***CodFisc***)
(2 chiavi differenti a singolo attributo)

LIBRI (***CodiceLibro***, Titolo, Costo, Genere, Contratto)

AUTORI-LIBRI (***Id_autore, CodLibro***)
(1 chiave multipla)

GENERE (***Genere***, Collocazione, Responsabile)

Nel caso di AUTORI, la scelta della chiave primaria cade opportunamente su ***CodFisc****.* che certamente non comprende omonimi e meno si presta ad errori di digitazione incontrollabili .

ATTRIBUTI PRIMI E NON PRIMI

Un attributo dicesi **primo** se fa parte di almeno una chiave, **non primo** altrimenti.

Sono ad esempio primi tutti gli attributi sopra in grassetto. Gli attributi non primi sono tipicamente funzioni delle chiavi: infatti a ciascun valore della chiave (unico per definizione in tutta la tabella), corrisponde un (unico) valore dell'attributo non primo nella ennupla così identificata.

DIPENDENZA FUNZIONALE

Dicesi che l'attributo Y dipende funzionalmente da X (tipicamente, da una chiave), se ad ogni valore di X corrisponde – tramite f – uno ed un sol valore di Y cioè Y = f(X) e si scrive X -> Y – che si legge X determina Y.

Nel caso che vi siano più chiavi – ad esempio K1 e K2 - esisteranno dipendenze funzionali "biunivoche" fra queste, nell'esempio K1 -> K2 e K2 -> K1.

DIPENDENZE FUNZIONALI DEL DATA BASE EDITRICE

AUTORI:

Nome -> Data nascita
Nome -> Indirizzo
Nome -> ***CodFisc***
CodFisc -> Data_nascita
CodFisc -> Indirizzo
CodFisc -> ***Nome***

GENERE

Genere ->Collocazione
Genere ->Responsabile

LIBRI

Codice libro ->Genere
Codice libro ->Titolo;
Codice libro ->Costo;
Codice libro ->Contratto.

4. DIPENDENZE FRA TABELLE

Le diverse tabelle di un data base relazionale non sono ovviamente del tutto slegate fra loro ma, in quanto rappresentative di un'unica realtà, sono fra di loro connesse.

Nel Data Base **Editrice**, gli oggetti rappresentati dalle singole tabelle sono tutti appartenenti alla realtà di una casa editrice:

gli Autori (tabella AUTORI) hanno scritto dei Libri (tabella LIBRI), che sono di un determinato genere (tabella GENERE);

altre tabelle, poi, creano esplicitamente legami fra questi oggetti (tabella AUTORI-LIBRI).

In realtà, così come esistono dipendenze fra gli attributi (campi) di una tabella (ad esempio le dipendenze funzionali) , esistono altresì *dipendenze fra le tabelle.*

CHIAVE ESTERNA

Un attributo (o un insieme di attributi) **di una tabella T** – detta *referente* - **è chiave esterna con riferimento ad un'altra tabella T'** – detta *riferita* - **se si riferisce alla chiave primaria di T'.**

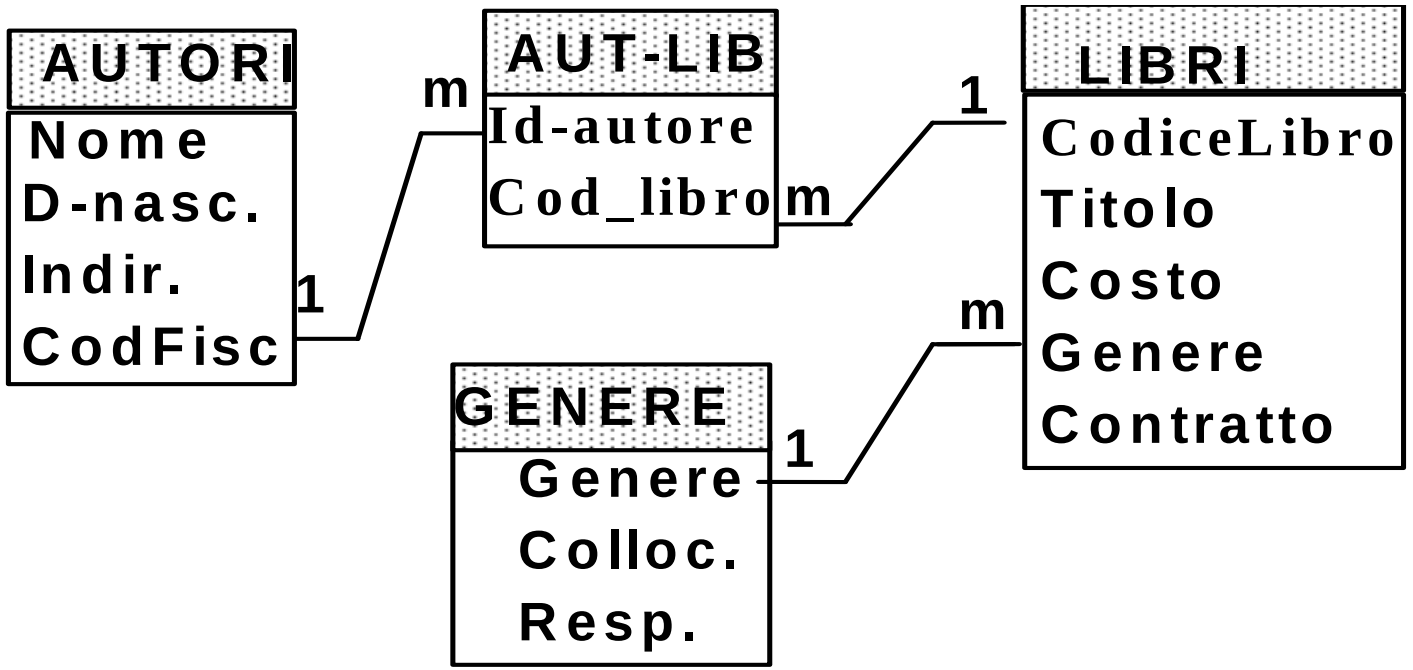
- Id_autore di AUTORI-LIBRI è chiave esterna, con riferimento alla tabella AUTORI, ove CodFisc (suo corrispondente) è chiave primaria;
- CodLibro di AUTORI-LIBRI è chiave esterna, con riferimento alla tabella LIBRI, ove Codice_libro è chiave primaria;
- Genere di LIBRI è chiave esterna di LIBRI con riferimento alla tabella GENERE, ove il campo Genere è chiave primaria.

Una *dipendenza fra due tabelle* TA, TB si stabilisce con riferimento a due campi a di TA e b di TB, che si corrispondono; in altri termini, rispetto a **due campi omogenei** destinati a contenere valori dello stesso dominio.

DATA BASE EDITRICE: DIPENDENZE TRA LE TABELLE

In figura sono mostrate le dipendenze fra le tabelle del database Editrice.

In esso esiste una dipendenza tra le tabelle AUTORI e AUTORI-LIBRI con riferimento ai campi CodFisc di AUTORI e Id_autore di AUTORI-LIBRI i quali si corrispondono, nel senso che il codice fiscale dell'autore, usato come chiave primaria in AUTORI, è un campo della tabella AUTORI-LIBRI.



MOLTEPLICITA' NELLE DIPENDENZE TRA TABELLE

Dipendenza uno ad uno

Fra le tabelle TA e TB esiste una dipendenza uno ad uno se per ciascun valore dell'attributo a in TA esiste al più un valore uguale di b in TB.

Nel database "Editrice" non esistono relazioni uno ad uno.

Dipendenza uno a molti

Fra le tabelle TA e TB esiste una dipendenza uno (TA) a molti (TB) se per ciascun valore dell'attributo a in TA possono esistere più valori uguali di b in TB.

Nel D.B. "Editrice" la dipendenza tra le tabelle AUTORI e AUTORI-LIBRI con riferimento ai campi CodFisc di AUTORI e Id_autore di AUTORI-LIBRI è una dipendenza uno a molti in quanto, potendo essere un Autore autore di più testi, per ciascun valore di CodFisc in AUTORI possono esistere più valori di Id_autore in AUTORI-LIBRI.

E' importante notare che Id_autore è chiave esterna in AUTORI-LIBRI, con riferimento alla relazione AUTORI in cui CodFisc è chiave primaria.

5. VINCOLI DI INTEGRITÀ

VINCOLI DI DOMINIO

I vincoli di dominio sono propri di tutte le applicazioni informatiche ed impongono l'assunzione di particolari classi di valori ai dati di ingresso.

I vincoli vanno programmati all'atto della stesura del programma o della definizione del database e vengono poi controllati in esercizio da parte del programma stesso oppure del software di base cui esso è affidato.

Possono essere espressi a mezzo di un predicato che coinvolge uno o più attributi fra di loro connessi con operatori logici, di relazione ed operatori speciali tipici dell' SQL.

VINCOLI DI DOMINIO (2)

Esempi di vincoli di dominio

- *Vincoli di appartenenza al tipo*
- *Controllo per fasce di valori*
- *Controllo per valori anomali*
- *Controlli per appartenenza a liste di valori*

I vincoli di dominio sono descritti nei database attraverso apposite clausole SQL e, nel caso che non vengano rispettati, sono associati ad un messaggio di errore definito dal progettista del database.

In MS-ACCESS, la medesima tabella che si usa per definire i campi di una relazione consente anche di definire i vincoli di dominio ed i messaggi di errore.

VINCOLI DI INTEGRITÀ: VINCOLI STRUTTURALI

Sono *i vincoli tipici di un database* che derivano dalla stessa teoria dei database, che non si ritrovano in altre applicazioni informatiche e contribuiscono a minimizzare le probabilità di malfunzionamento di un'applicazione. Scaturiscono dai concetti di chiave e di relazione.

- *Vincolo di chiave primaria*

  La chiave primaria deve essere **non nulla** (NOT NULL) per identificare univocamente la tupla. Se la chiave è costituita da un singolo attributo, si tratta di un vincolo di attributo, se è multipla di un vincolo di tabella.

- *Vincolo di unicità (unique)*

  Ad un singolo attributo oppure ad un insieme di attributi di una tabella può essere imposto il vincolo di presentare molteplicità al più pari ad 1 (si usa la clausola UNIQUE in SQL).

VINCOLI DI INTEGRITÀ: VINCOLI STRUTTURALI (2)

- *Vincolo di non nullità*

  Ad un singolo attributo oppure ad un insieme di attributi di una tabella può essere imposto il vincolo di non dovere assumere valore nullo (NOT NULL in SQL).
  Una chiave primaria possiede implicitamente entrambi i vincoli di unicità e di non nullità.

- *Vincolo di integrità referenziale*

  Se due tabelle sono legate da una dipendenza 1: m, un valore presente nel lato molti deve preesistere nel lato 1. Con riferimento ad esempio al database "Editrice", se si inserisce una tupla nella tabella AUTORI-LIBRI, è necessario che siano state già inserite (eventualmente nella stessa transazione) **l'autore** in tabella AUTORI e **il libro** in tabella LIBRI.
  Si tratta di un vincolo molto importante, che va verificato dal DBMS in corrispondenza di:
  - modifiche e di inserimento di tuple nella tabella referente
  - modifiche e cancellazioni di tuple nella tabella riferita.

NOTAZIONE RELAZIONALE CON RAPPRESENTAZIONE DEI VINCOLI DI INTEGRITA' REFERENZIALE

È possibile indicare nello schema del modello relazionale anche *i vincoli di integrità referenziale* usando la seguente sintassi per gli attributi delle tabelle che sono chiavi esterne:

<chiave esterna>: TABELLA_ESTERNA [(chiave primaria | unique)]

ove il costrutto tra parentesi [ ] deve obbligatoriamente comparire se il riferimento nella tabella referente è ad una chiave non primaria (unique) della tabella riferita.

Con riferimento al modello relazionale della base di dati editrice potremo scrivere:

- AUTORI (Nome, Data nascita, Indirizzo, **CodFisc**)
- LIBRI (**Codice_libro**, Titolo, Costo, Genere: GENERE, Contratto)
- AUTORI-LIBRI (**Id_autore** : AUTORI, **CodLibro**: LIBRI)
- GENERE (**Genere**, Collocazione, Responsabile)

6. NORMALIZZAZIONE

Su un piano intuitivo, una base di dati relazionale deve possedere un insieme di proprietà per essere ben gestita:

- deve rappresentare concetti diversi in tabelle diverse;
- non deve contenere inutili duplicazioni di dati;
- deve essere semplice da interpretare e da gestire;
- deve consentire agevolmente inserzioni, cancellazioni e aggiornamenti;

Questo si ottiene tramite un procedimento che si applica a ciascuna relazione che si chiama normalizzazione.

- Una relazione è in **1NF** (NF, Normal Form, Forma Normale) se tutti gli attributi non assumono valori molteplici o strutturati; ciascuna relazione *del modello relazionale è in 1NF* e pertanto ***ciascun attributo non primo dipende da ciascuna chiave***.

- Una relazione è in **2NF** se è in 1NF e ***ciascun attributo non primo dipende funzionalmente da tutta la chiave e non anche da parte di essa***.

- Una relazione è in **3NF** se è in 2NF e **ciascun attributo non primo *dipende in modo diretto dalla chiave*** (non transitivo, attraverso un altro attributo).

DATA BASE EDITRICE: UN CASO DI NON NORMALIZZAZIONE

- AUTORI (***Nome***, Data nascita, Indirizzo, ***C.F.***) (2 chiavi differenti a singolo attributo)
- LIBRI (***Codicelibro***, Titolo, Costo, Genere, Responsabile)
- AUTORI-LIBRI (***Id_autore, CodLibro,*** Contratto) (chiave multipla)
- GENERE (***Genere***, Collocazione)

AUTORI-LIBRI non è in seconda forma normale poiché
CodLibro-> Contratto , ma ***CodLibro*** è una parte della chiave di AUTORI-LIBRI

LIBRI non è in terza forma normale poiché:
Genere -> Responsabile, ma Responsabile dipende transitivamente dalla chiave:
Codicelibro -> Genere -> Responsabile

DATA BASE EDITRICE NORMALIZZATA

- AUTORI (***Nome***, Data nascita, Indirizzo, ***C.F.***) (2 chiavi differenti a singolo attributo)
- LIBRI (***Codicelibro***, Titolo, Costo, Genere, Contratto)
- AUTORI-LIBRI (***Id_autore, CodLibro***) (chiave multipla)
- GENERE (***Genere***, Collocazione, Responsabile)

7.DESCRIZIONE DI UNA BASE DI DATI IN MS ACCESS

La base di dati viene inizialmente presentata mediante un insieme di tabelle(schemi di relazione) istanziate accompagnate dai relativi vincoli.

Con riferimento al caso “Editrice” abbiamo:

- AUTORI (Nome, Data_nascita, Indirizzo, **CodFisc**)
- LIBRI (**Codice libro**, Titolo, Costo, Genere: GENERE, Contratto)
- AUTORI-LIBRI (**Id_autore** : AUTORI, **Cod_libro**: LIBRI)
- GENERE (**Genere**, Collocazione, Responsabile)

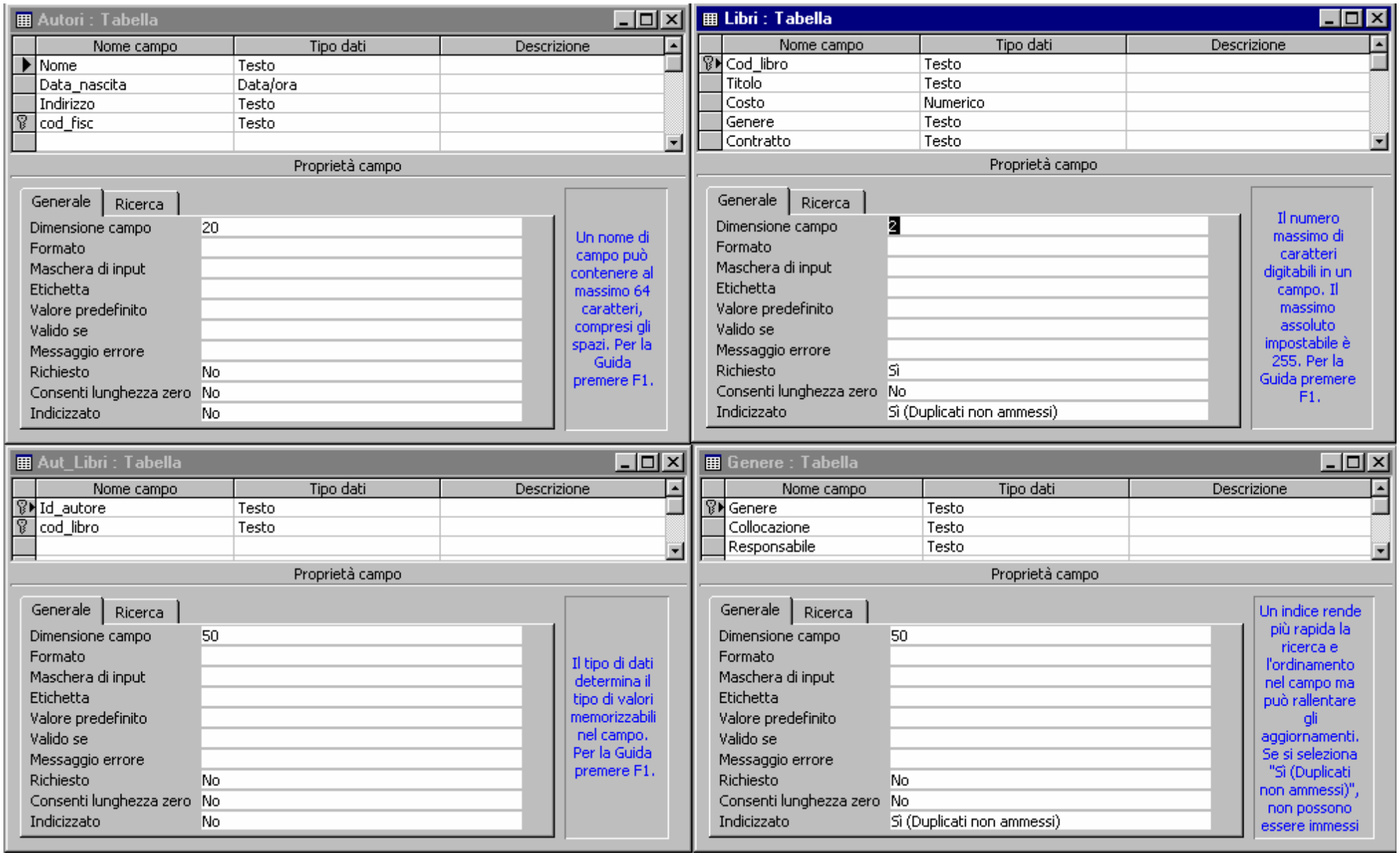
Ci proponiamo di descrivere graficamente la base dei dati di esempio mediante il DBMS MS – ACCESS. Occorrerà allo scopo:

1. *Definire delle tabelle relazionali includendo per ciascuna di esse i domini degli attributi con relativi vincoli ed il vincolo di chiave primaria*
2. *Definire i vincoli di integrità referenziale (dipendenze tra tabelle) e includerli nello schema della base di dati*
3. *Effettuare il caricamento iniziale della base di dati*

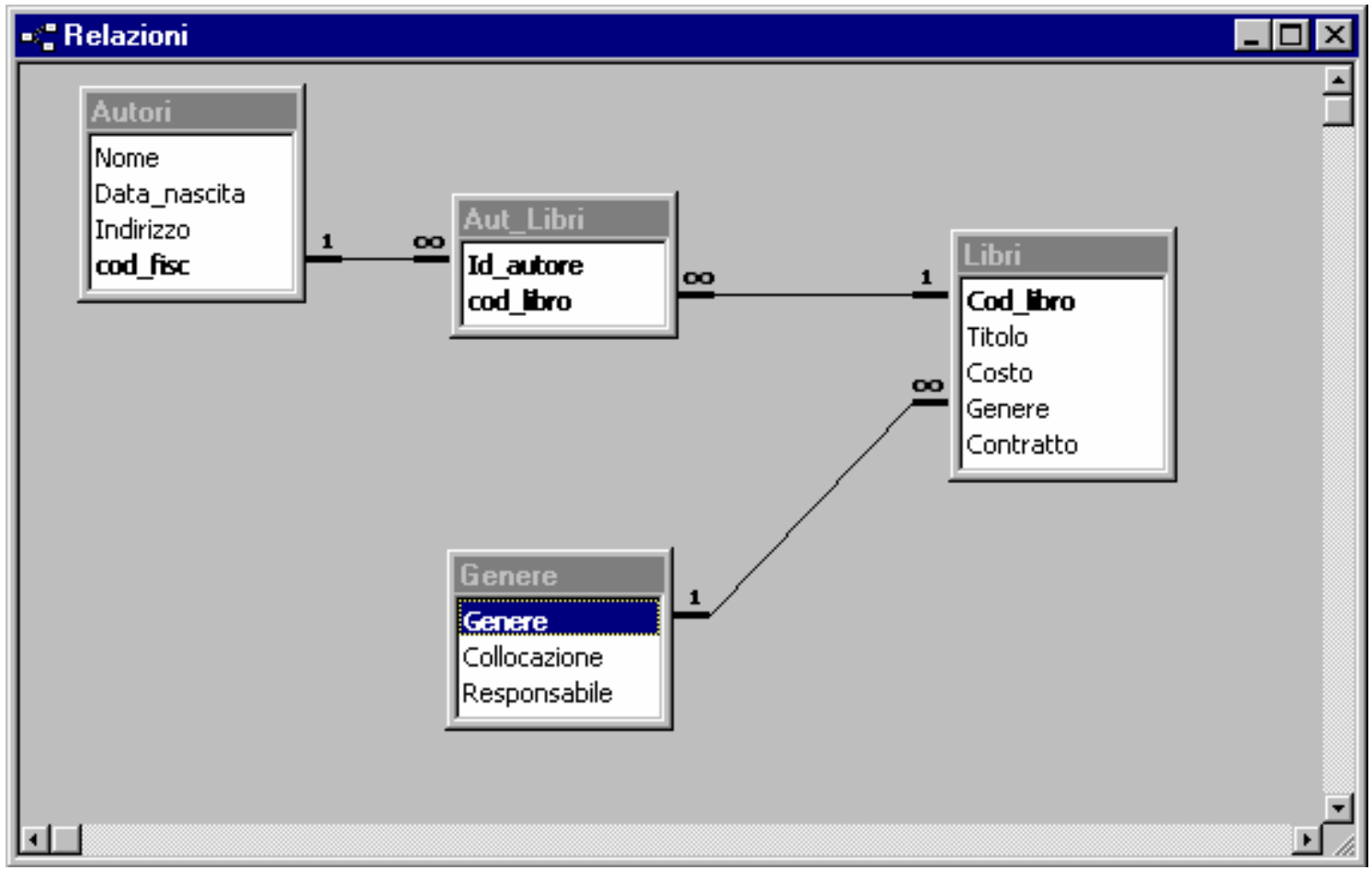
come mostrato nelle figure che seguono

A questo punto sarà possibile effettuare sulla base dei dati interrogazioni o manipolazioni – inserimento, cancellazione, aggiornamento – di tuple.

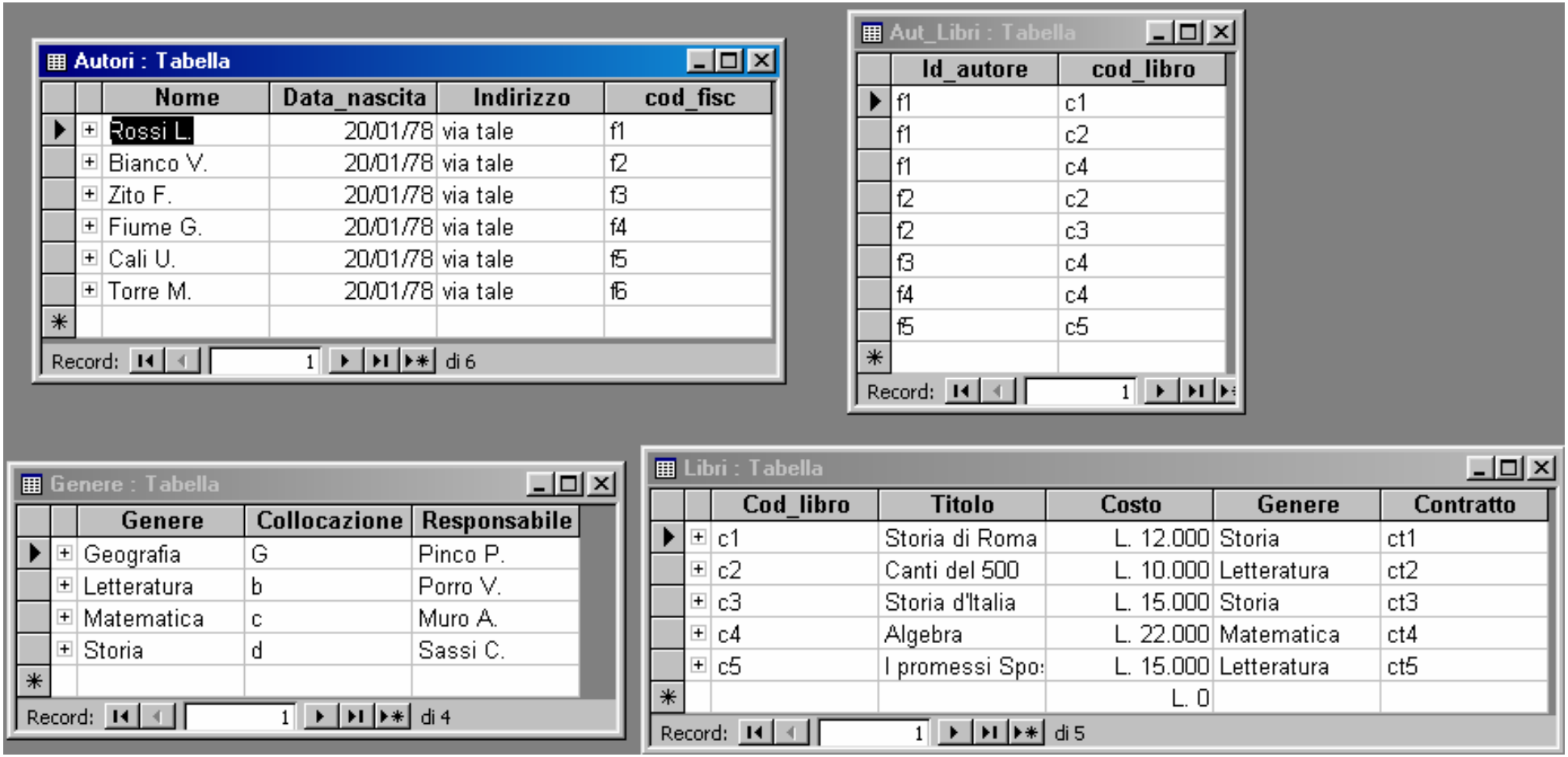
DEFINIZIONE DELLE TABELLE DELLA BASE DI DATI



SCHEMA RELAZIONALE CON I VINCOLI REFERENZIALI



POPOLAZIONE DELLA BASE DI DATI



Autori : Tabella

Nome	Data_nascita	Indirizzo	cod_fisc
Rossi L.	20/01/78	via tale	f1
Bianco V.	20/01/78	via tale	f2
Zito F.	20/01/78	via tale	f3
Fiume G.	20/01/78	via tale	f4
Cali U.	20/01/78	via tale	f5
Torre M.	20/01/78	via tale	f6

Record: 1 di 6

Aut_Libri : Tabella

Id_autore	cod_libro
f1	c1
f1	c2
f1	c4
f2	c2
f2	c3
f3	c4
f4	c4
f5	c5

Record: 1

Genere : Tabella

Genere	Collocazione	Responsabile
Geografia	G	Pinco P.
Letteratura	b	Porro V.
Matematica	c	Muro A.
Storia	d	Sassi C.

Record: 1 di 4

Libri : Tabella

Cod_libro	Titolo	Costo	Genere	Contratto
c1	Storia di Roma	L. 12.000	Storia	ct1
c2	Canti del 500	L. 10.000	Letteratura	ct2
c3	Storia d'Italia	L. 15.000	Storia	ct3
c4	Algebra	L. 22.000	Matematica	ct4
c5	I promessi Spo:	L. 15.000	Letteratura	ct5
		L. 0		

Record: 1 di 5

GESTIONE DI UNA BASE DI DATI RELAZIONALE

8. INTERROGAZIONE DEL DATABASE ED ALGEBRA RELAZIONALE
9. QUERY SU UNA TABELLA: PROIEZIONE E SELEZIONE
10. QUERY SU PIÙ TABELLE: JOIN
11. SINTASSI SEMPLIFICATA DELLA SELECT
12. OPERATORI DI AGGREGAZIONE E RAGGRUPPAMENTO
13. USO DELLE QUERY NIDIFICATE
14. INSERIMENTO, CANCELLAZIONE, AGGIORNAMENTO

8. INTERROGAZIONE DEL DATABASE ED ALGEBRA RELAZIONALE

INTERROGAZIONI

Le operazioni di interrogazione *non alterano lo stato del database*, nel senso che non inseriscono né aggiornano i valori dei dati, ma effettuano soltanto operazioni di lettura degli stessi.

Esse hanno l'obiettivo di estrarre informazioni utili e sintetiche da tutte quelle memorizzate. Ad esempio:

- in una B.D. *agenda:* il numero di telefono di una persona;
- in una B.D. *anagrafica:* tutti gli indirizzi di una determinata categoria di individui;
- nella B.D. *Editrice*: tutti i volumi di un determinato Autore, di un determinato genere e che costano meno di una data somma.
- in una B.D. *amministrativa*: tutte le fatture inevase oppure tutti gli ordini emessi per i quali la merce non sia ancora pervenuta.

Ogni interrogazione si riduce alla costruzione di una *nuova relazione* (anche una singola tupla o addirittura nessuna), tratta da quelle memorizzate.

ALGEBRA RELAZIONALE

La manipolazione dei dati di una base dati relazionale consiste dunque nel costruire o modificare tabelle (relazioni), a partire da quelle esistenti.

Per tale manipolazione si usa un metodo algebrico: viene definita una apposita *algebra relazionale,* con le proprie operazioni che agiscono su relazioni e generano nuove relazioni e l'interrogazione di un database consiste nell'invocare opportunamente tali operazioni;

Le operazioni dell'algebra relazionale sono:

- *operazioni insiemistiche,*
- selezione,
- proiezione,
- join.

Lasciando all'allievo la facile definizione delle operazioni insiemistiche (unione, intersezione, diferenza), faremo riferimento, in questa sede alle operazioni di *selezione*, *proiezione* e *join* che costituiscono altresì il **nucleo** dei linguaggi di interrogazione del database.

9.QUERY SU UNA TABELLA: PROIEZIONE E SELEZIONE

PROIEZIONE

L'operazione di proiezione:

$T(A) = PROJ_A(R)$

ove A è un insieme di attributi di R(X), $\mathbf{A \subseteq X}$, si applica alla relazione R(X) e genera la relazione T(A) contenente tutte le ennuple di R limitatamente ai soli attributi A. In altri termini, la proiezione *seleziona le colonne* di R definite dall'insieme di attributi A.

In SQL essa si esprime con la frase:

SELECT <elenco di colonne>
FROM <nome tabella>

L'operazione è definita in senso insiemistico e pertanto eventuali ennuple di T che risultassero uguali fra loro a seguito della proiezione sono considerate una sola volta. Queste peraltro vanno esplicitamente rimosse in SQL adoperando l'opzione DISTINCT:

SELECT DISTINCT <elenco di colonne>
FROM <nome tabella>

PROIEZIONE

Es.1

Con riferimento alla tabella *Libri*, la proiezione

PROJ (Cod_libro, Genere) (LIBRI)

è codificata come segue:

```
SELECT  Cod_libro, Genere
FROM LIBRI
```

e produce la relazione:

Cod_libro	Genere
C1	G1
C2	G2
C3	G1
C4	G3
C5	G2

PROIEZIONE (2)

PROIEZIONE SU TUTTI GLI ATTRIBUTI

Es. 2
La frase

```
SELECT *
FROM LIBRI
```

riproduce integralmente la tabella LIBRI.

PROIEZIONE CON RIGHE DISTINTE

Es. 3
La frase

```
SELECT  DISTINCT Genere
FROM LIBRI
```

produce l'elenco dei generi, ciascuno una sola volta.

SELEZIONE

L'operazione di selezione:

$T = SEL_F(R)$

ove F è un predicato (espressione logica), si applica ad una relazione R e genera una nuova relazione $T \in R$ costituita da tutte e sole le tuple per le quali è F=vero:

$$T = \{t \subseteq R / F = vero\}$$

In altri termini, l'operazione *seleziona le righe* della relazione R per le quali risulta vero il predicato F, che diremo condizione o criterio di selezione. In SQL la selezione si esprime con la frase:

SELECT *
FROM <nome tabella>
WHERE <predicato>

ESEMPI DI SELEZIONE

Es. 4

1.Selezione con operatori logici

La frase

```
SELECT *
FROM LIBRI
WHERE Costo > 12 000 AND costo < 20 000
```

seleziona, nella relazione LIBRI, quelli di costo compreso nell'intervallo.

Es. 5

2.Selezione con WHERE

La frase

```
SELECT *
FROM LIBRI
WHERE Genere = "Storia" AND Costo < 15 000
```

seleziona, in LIBRI, quelli di genere G1 e costo minore di 50 000.

ESEMPI DI SELEZIONE E PROIEZIONE

Es.6

È possibile con un'unica frase realizzare la proiezione su alcune colonne della tabella ottenuta con una selezione di righe.

3.Selezione e proiezione

La frase

SELECT Id_autore
FROM AUTORI_LIBRI
WHERE cod_libro=C2:

seleziona, in AUTORI_LIBRI, tutti i codici fiscali degli autori del libro C2.

ORDINAMENTO DI PROIEZIONI E SELEZIONI

È possibile ottenere le righe della tabella selezionata e/o proiettata in ordine rispetto ai valori di un attributo; a tale scopo in SQL si usa la clausola

ORDER BY <lista delle specifiche di ordinamento>
[ASC | DESC]

ove:

- la lista comprende il nome dell'attributo rispetto al quale si vuole ordinare e, se questo non è chiave, quello di ulteriori attributi da prendere in esame nell'ordine in caso di eguaglianza del primo;
- ASC significa ordine crescente, DESC decrescente e per default l'ordinamento è ascendente.

PROIEZIONE E SELEZIONE IN QBE

Es. 7

Le operazioni di proiezione e selezione su una tabella si esprimono in QBE con tecnica visuale, a mezzo di menu guidati ed apposite tabelle; ad esempio, la frase

SELECT Titolo, Costo
FROM LIBRI
WHERE Genere = "Storia" AND Costo < 23, 00
or Genere = "Letteratura" AND Costo > 22, 00
ORDER BY Costo

Si esprime selezionando la tabella nell'elenco di tutte le tabelle del DB che il DBMS propone, quindi riempendo un quadro proposto dallo stesso **:**

CAMPO	**Titolo**	**Costo**	**Genere**
ORDINAMENTO		Crescente	
MOSTRA	×	×	
CRITERI		<23,00	"Storia"
OPPURE		>22,00	"Letteratura"

(In *Access*: I nomi dei campi sono trascinati nel quadro mediante mouse prelevandoli da un menù suggerito, gli altri dati sono ottenuti mediante menù guidati).

10.QUERY SU PIÙ TABELLE: JOIN

L'operazione di join (detta anche di equi join) fra $R(X \cup Y)$ e $S(X \cup Z)$, aventi l'insieme X di attributi comuni, è una relazione $T(X \cup Y \cup Z)$:

$$T = R \text{ join}_X S$$

tale che per ogni coppia di ennuple di R ed S per le quali siano eguali gli attributi X, cioè sia:

$$t_R.X = t_S.X = t.X$$

viene generata una ennupla in T avente i valori di questi attributi X e di quelli Y e Z allineati con essi rispettivamente in R ed S:

$$t_R.Y, t.X, t_S.Z$$

JOIN IN SQL

In SQL, detti R ed S i nomi delle tabelle e X_R, X_S quelli del campo X rispettivamente in R, S, la notazione è:

SELECT <elenco colonne>
From R,S
WHERE R.X_R = S.X_S

Il join è un'operazione commutativa ed associativa e puo' essere esteso a più di due tabelle:

R join $_X$ (S join $_X$ P) = (R join $_X$ S) join $_X$ P

SQL prevede una seconda notazione, alla quale si ricorre tipicamente in caso di join fra tre o più tabelle, come sarà esemplificato in seguito:

SELECT <elenco colonne>
FROM R INNER JOIN S
ON R.X_R = S.X_S

In QBE il join viene programmato con tecniche visuali, indicando graficamente le tabelle da congiungere ed i campi rispetto ai quali esse debbono essere congiunte.

ESEMPI DI JOIN: DATABASE EDITRICE.

ES.8

join su due tabelle con clausola WHERE

Il join fra LIBRI e GENERE sull'attributo Genere produce una tabella simile a LIBRI di tab. 3.1, ma con in più gli attributi *collocazione* e *responsabile*; la proiezione di questa sugli attributi Titolo di LIBRI e Responsabile di Genere produce una tabella nella quale ad ogni titolo è affiancato il responsabile del genere corrispondente:

```
SELECT LIBRI.Titolo, GENERE.Responsabile
From LIBRI, GENERE
WHERE LIBRI.Genere= GENERE.Genere
```

Si noti che in questo esempio, come tipicamente avviene, il join si effettua con riferimento ad un attributo (Genere) che determina una relazione fra le due tabelle (chiave primaria in una tabella e chiave esterna nell'altra) e genera una tabella che esprime la relazione fra questo e tutti gli altri attributi delle due tabelle

Es. 9

Join su due tabelle con la clausola ON sul FROM

```
SELECT LIBRI.Titolo, GENERE.Responsabile
From LIBRI INNER JOIN GENERE ON Libri.Genere=Genere.Genere
Where Genere.Genere <>"Storia"
```

ESEMPI DI JOIN: DATABASE EDITRICE.

Es. 10 **join su due tabelle con clausola ON sul FROM**

La frase

```
SELECT DISTINCT AUTORI.nome, AUTORI_LIBRI.Cod_libro
FROM AUTORI INNER JOIN AUTORI_LIBRI ON AUTORI.C_F =
AUTORI_LIBRI.Cod_fisc
ORDER BY AUTORI.Nome
```

produce una tabella nella quale sono affiancati il nome dell'autore (tratto dalla tabella AUTORI) ed il corrispondente codice del libro (tratto da AUTORI_LIBRI).

Es 11**join su tre tabelle con clausola On su FROM**

Il join con le tre tabelle AUTORI, AUTORI_LIBRI E LIBRI fornisce la relazione in chiaro (non codificata) fra nomi degli autori e titoli dei libri: il join è effettuato fra LIBRI ed il risultato del join fra AUTORI e AUTORI_LIBRI, segnato in parentesi nelle righe ombreggiate.

```
SELECT DISTINCT
AUTORI.Nome, LIBRI.Titolo
FROM LIBRI INNER JOIN
(AUTORI INNER JOIN AUTORI_LIBRI ON AUTORI.C_F=
AUTORI_LIBRI.Cod_fisc)
ON LIBRI.Cod_libro = AUTORI_LIBRI.Cod_libro;
```

11- SINTASSI SEMPLIFICATA DELLA SELECT

```
select AttrExpr [ [as] Alias ] { , AttrExpr [ [as] Alias ] }
from Tabella [ [as] Alias ]  { , Tabella [ [as] Alias ] }
[where Condizione ]
```

- le tre parti vengono di solito chiamate
  - target list
  - clausola from
  - clausola where

  La sintassi con il join nel from è:

```
select AttrExpr [ [as] Alias ]  {,AttrExpr [ [as] Alias ] }
from Tabella [ [ as ] Alias ] {[ TipoJoin ] join
                    Tabella [ [ as ] Alias ] on CondDiJoin }, ...
[ where AltraCondizione ]
```

12- OPERATORI DI AGGREGAZIONE E RAGGRUPPAMENTO

UN ALTRO ESEMPIO: DATA BASE TURISTICO

E' assegnata la seguente base di dati relativa alle informazioni sui ristoranti di una città offerte da un ente turistico:

ZONE

Zona	**NomeZona**
C	Centro
N	Nord
O	Ovest

RISTORANTI

Cod	**Nome**	**Indirizzo**	**Tipo**	**Zona**
342	Da Piero	Via Larga 32	R	C
421	Buona Cucina	Vico Corto 1	R	C
425	Paris	Via Lunga 4	I	N
655	Canton	Via Breve 2	C	O

CUCINE

Tipo	**DescrTipo**
R	Regionale
I	Internazionale
C	Cinese

CONVENZIONI

Cod	**CodCarta**
342	V
342	A
421	A
425	D
425	A
655	V

CARTEDICREDITO

CodCarta	**Carta**
V	Visa
A	Amex
D	Diners

SCHEMA RELAZIONALE

Lo schema relazionale, usando la sintassi già descritta, è riportato di seguito:

ZONE (Zona, NomeZona)

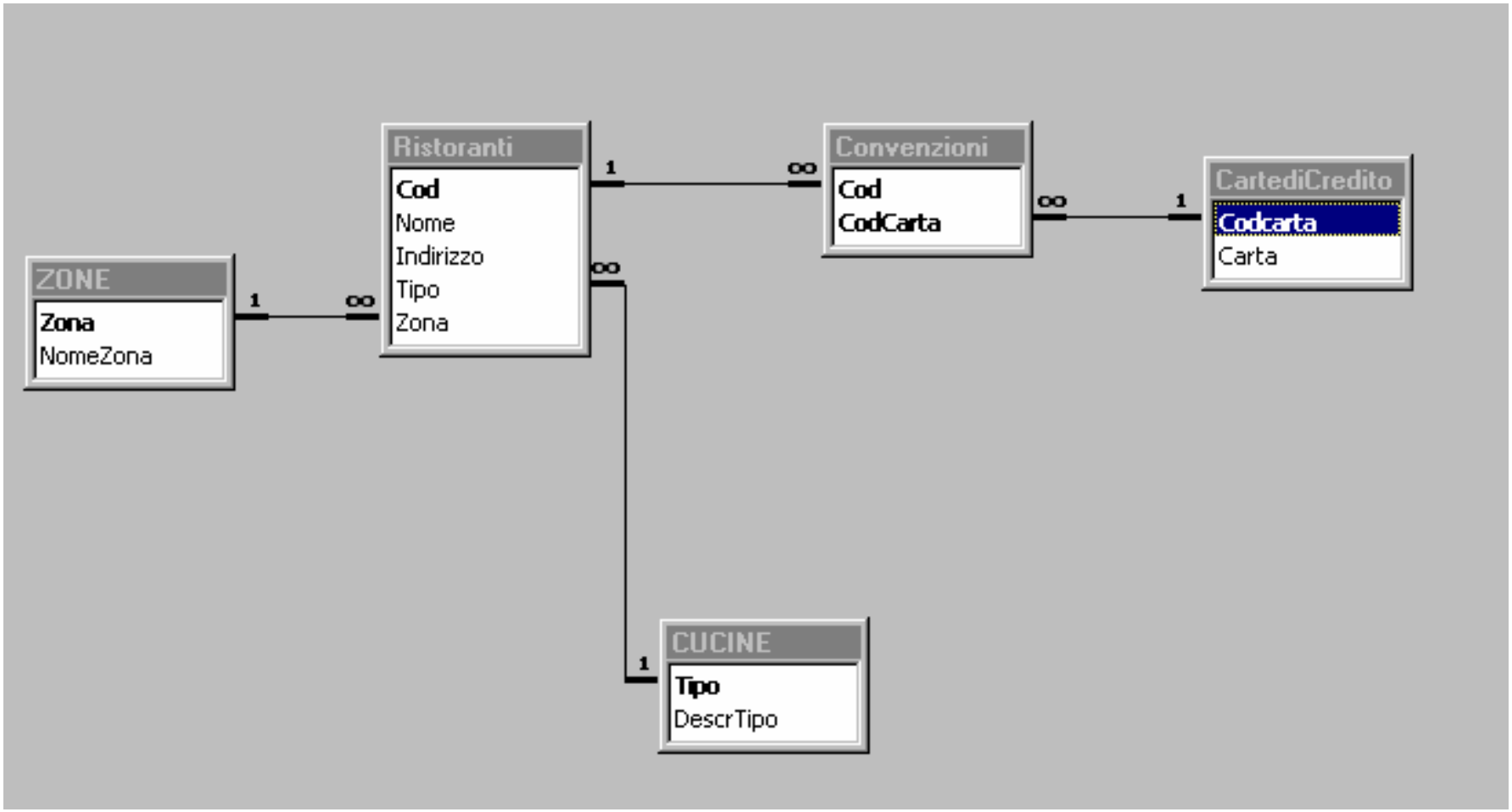
RISTORANTI(Cod, Nome, Indirizzo, Tipo: CUCINE, Zona: ZONE)

CUCINE(Tipo, DescrTipo)

CONVENZIONI(Cod: RISTORANTI, CodCarta: CARTEDICREDITO)

CARTEDICREDITO (CodCarta, Carta)

I CAMMINI DI JOIN



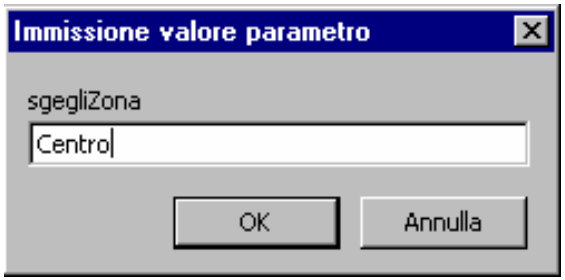
USO DEGLI OPERATORI DI AGGREGAZIONE: COUNT

SINTASSI

```
count ( < * | [ distinct | all ] ListaAttributi >)
```

Es.12 **Conteggio dei ristoranti per zona (con scelta della zona tramite il parametro scegliZona fornito da input)**

SELECT DISTINCT count(RISTORANTI.Nome) **AS** QuantiNeSono
FROM ZONE **INNER JOIN** RISTORANTI **ON** ZONE.Zona = RISTORANTI.Zona
WHERE ZONE.NomeZona=*sgegliZona*;



QuantiNeSono
2

OPERATORE DI AGGREGAZIONE COUNT

- l'operatore aggregato (`count`) viene applicato al risultato dell'interrogazione:

  SELECT DISTINCT *
  FROM ZONE **INNER JOIN** RISTORANTI **ON** ZONE.Zona = RISTORANTI.Zona
  WHERE ZONE.NomeZona=*sgegliZona*;

ALTRI OPERATORI AGGREGATI SONO:

Somma, media, massimo, minimo

con la seguente sintassi

< `sum` | `max` | `min` | `avg` > ([`distinct` | `all`] AttrEspr)

USO DEGLI OPERATORI DI RAGGRUPPAMENTO

Es.13 Per ogni zona contare il numero di ristoranti presenti

```
SELECT Zona AS LeZone, count(Cod) AS QuantiNeSono
FROM RISTORANTI
GROUP BY Zona;
```

La semantica dell' interrogazione con operatori aggregati e raggruppamenti è la seguente:

1- interrogazione senza **group by** e senza operatori aggregati (count nel nostro caso)

```
SELECT Zona,Cod
FROM RISTORANTI
```

Zona	Cod
C	342
N	425
C	421
O	655

2- poi si raggruppa

Zona	Cod
C	342
C	421
N	425
O	655

3- e si applica l'operatore aggregato ad ogni gruppo

Le Zone	QuantiNeSono
C	2
N	1
O	1

USO DEL GROUP BY CON HAVING :

Es. 14 Trovare i nomi di zona con numero di ristoranti maggiori di quello fornito da input tramite il parametro *numris*.

```
SELECT Zone.NomeZona
FROM ZONE  INNER JOIN RISTORANTI ON ZONE.Zona = RISTORANTI.Zona
GROUP BY ZONE.NomeZona
HAVING count( RISTORANTI.Nome)>numRis;
```

NomeZona
Centro
Ovest

13- USO DELLE QUERY NIDIFICATE

LA SINTASSI (SEMPLIFICATA) DELLE QUERY NIDIFICATE

SELECT <Lista Attributi o espressioni>
FROM <Lista Tabelle eventualmente con join>
WHERE <condizione con nidificazione> ;

dove:

<condizione con nidificazione> ::=

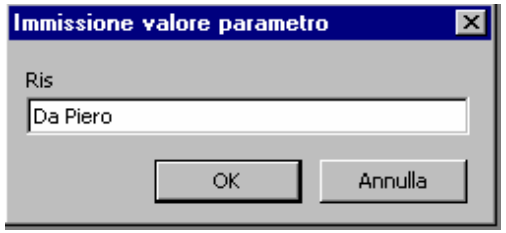
<Scalare> < in | not in | = | <> > (
<SelectAttributoSingolo>
)

dove:

- <scalare> è una costante o un attributo semplice
- <SelectAttributosingolo> è una interrogazione nidificata in cui dopo l'operatore select deve seguire un attributo singolo il cui tipo deve essere congruente con quello dello scalare.
  Il risultato dell'interrogazione è una lista (anche vuota) i cui valori vengono comparati - mediante l'operatore in (not in, =, <>) - con lo scalare per definire il risultato della condizione.

Es. 15 **Trovare i nomi e gli indirizzi dei ristoranti che si trovano nella stessa zona di un assegnato ristorante (fornito da input) con l'uso di una query nidificata.**

```
SELECT Nome, Indirizzo
FROM Ristoranti
WHERE Nome <>Ris AND Indirizzo<>Ind AND Zona IN
                                    ( SELECT Zona
                                      FROM Ristoranti
                                      WHERE Nome=Ris and Indirizzo=Ind);
```



Nome	Indirizzo
Buona Cucina	Vico Corto 1

UN ALTRO ESEMPIO DI QUERY NIDIFICATA

Es 16 **Trovare i ristoranti che non sono convenzionati con la Carta di Credito Diners.**

```
SELECT Nome
FROM RISTORANTI
WHERE Nome NOT IN (
                    SELECT Nome
                    FROM RISTORANTI INNER JOIN
                        (CARTEDICREDITO INNER JOIN CONVENZIONI
                         ON  CARTEDICREDITO.CodCarta
                                              = CONVENZIONI.CodCarta)
                         ON    RISTORANTI.Cod = CONVENZIONI.Cod
                    WHERE CARTEDICREDITO.Carta='Diners '
                   );
```

Nome
Da Piero
Buona Cucina
Canton

14. INSERIMENTO, CANCELLAZIONE, AGGIORNAMENTO

INSERIMENTO

Per inserimento si intende l'accodamento ad una tabella di nuove righe; esso si puo' ottenere mediante tre tecniche distinte:

- digitazione di dati in forma interattiva;

- inserimento programmato di valori definiti attraverso una frase SQL;

  INSERT
  INTO <nome tabella> [<elenco di colonne>]
  VALUES <elenco di costanti>

ove l'elenco di colonne puo' mancare, ed allora si intende che vanno inserite tutte le colonne e le costanti sono i valori da inserire.

- inserimento in una tabella dei risultati di una query.

  INSERT
  INTO <nome tabella> [<elenco di colonne>]
  SELECT .....FROM .....WHERE

e consiste nell'inserire in tabella le righe ottenute dall'operazione di SELECT (proiezione, selezione o join).

CANCELLAZIONE

Per cancellazione si intende l'eliminazione da una tabella di alcune righe. In linguaggio visuale viene indicato con l'aiuto dei menù, in SQL si esprime come

DELETE
FROM <nome tabella>
[WHERE <predicato>]

e comanda la cancellazione da <nome tabella> di tutte le righe per le quali il predicato è vero (se la clausola WHERE manca, vengono cancellate tutte le righe).

È da notare che la cancellazione di un rigo da una tabella oppure della tabella intera difficilmente produce la cancellazione fisica dell'oggetto, ma soltanto una *cancellazione logica,* ad esempio mediante apposizione di un segnale di spunta (flag). Ciò per evitare le pesanti operazioni di riorganizzazione fisica degli archivi conseguenti l'eliminazione fisica, che avvengono invece di tanto in tanto. Ne segue che è anche possibile in qualche DBMS il ripristino di un dato precedentemente cancellato (operazione detta di *undelete*).

AGGIORNAMENTO

Per aggiornamento si intende la modifica di valori di una tabella e, così come per le altre operazioni, l'aggiornamento puo' avvenire in forma interattiva o programmata; quest'ultima si ottiene in SQL con una frase del tipo:

UPDATE <nome tabella>
SET <nome campo> = <espressione>
[, <nome campo> = <espressione>] . . .
[WHERE <predicato>]

e, per le righe per le quali il predicato è vero, assegna ai campi elencati il corrispondente valore derivante dall'espressione calcolata;
quest'ultima puo' essere

- una costante (si pensi ad esempio all'aggiornamento dell'aliquota IVA derivante da una nuova legge)
- calcolata in funzione del campo stesso (si pensi ad un aumento percentuale del costo di una categoria di prodotti)
- funzione qualsiasi anche di altri campi.

PROGETTAZIONE DI UNA BASE DI DATI RELAZIONALE

15. MODELLO ENTITÀ-ASSOCIAZIONI (E/R) E PROGETTO CONCETTUALE
16. PROGETTO LOGICO E DESCRIZIONE DELLO SCHEMA DEL DATABASE TRAMITE SQL

15. MODELLO ENTITÀ-ASSOCIAZIONI E PROGETTO CONCETTUALE:

In un approccio più generale, si puo' sviluppare un primo modello dei dati detto *concettuale* ed un secondo, detto *logico, c*he è appunto quello della BDR fin qui visto; a questo poi, come accennato, segue il modello *fisico* o interno.

Come modello concettuale dei dati si usa tecnicamente il cosiddetto modello *Entità-Relazioni (ER- Entity Relationship)*, secondo il quale la realtà è costituita da entità e da relazioni fra queste, rappresentate in un grafo.

Il modello ER (Entity Relationship), è in generale un modello di rappresentazione dei dati indipendente dalla eventuale sua trasformazione in una BDR e viene adoperato nelle fasi preliminari della progettazione, in collaborazione fra l'esperto del problema da modellare e l'esperto della progettazione della base dei dati.

ENTITA’

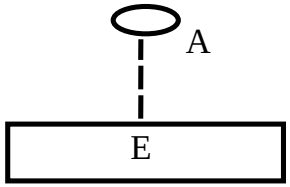
Per *entità* si intende in generale un qualsiasi elemento della realtà che può essere chiaramente individuato: un libro, un autore, un impiegato, una fattura, un ordine etc. Un insieme di entità (anche l'insieme è detto impropriamente *Entità*) è il concetto astratto associato all'insieme dei libri, degli autori e così via.

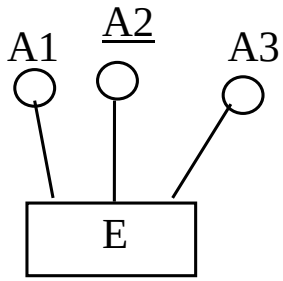
Nel *modello logico* l'insieme di entità diventa la tabella, la singola entità una sua riga. Un'entità è caratterizzata dal suo dominio (non rappresentato in figura) e da un insieme di attributi (l'equivalente delle colonne del modello logico), ad esempio per un libro il titolo, il costo, il genere e così via. Fra gli attributi si possono riconoscere i cosiddetti “identificatori” con gli stessi criteri adoperati per identificare le chiavi nel modello logico.

Con riferimento all'esempio della casa editrice si potrebbero ad esempio individuare come insiemi di entità i libri, gli autori, i generi dei libri (*e non ciò che corrisponde alla tabella AUTORI-LIBRI*), ma con una visione diversa della realtà si potrebbe identificare un numero maggiore o minore di entità.

ENTITA’ E ATTRIBUTI

Il modello E-R, in quanto modello concettuale, prende inizialmente in esame anche attributi con molteplicità M:N, cioè attributi multivalore. Tali attributi, in una successiva fase di progetto logico, vengono opportunamente trasformati in altre entità.





E = Entità

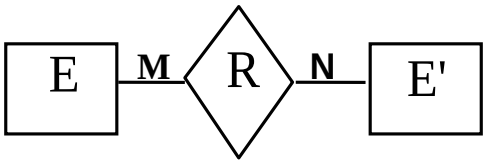
$e \in E$

A = Attributo di E

Entità con più attributi
A2 = identificatore

ASSOCIAZIONI E RAPPORTO DI CARDINALITA'

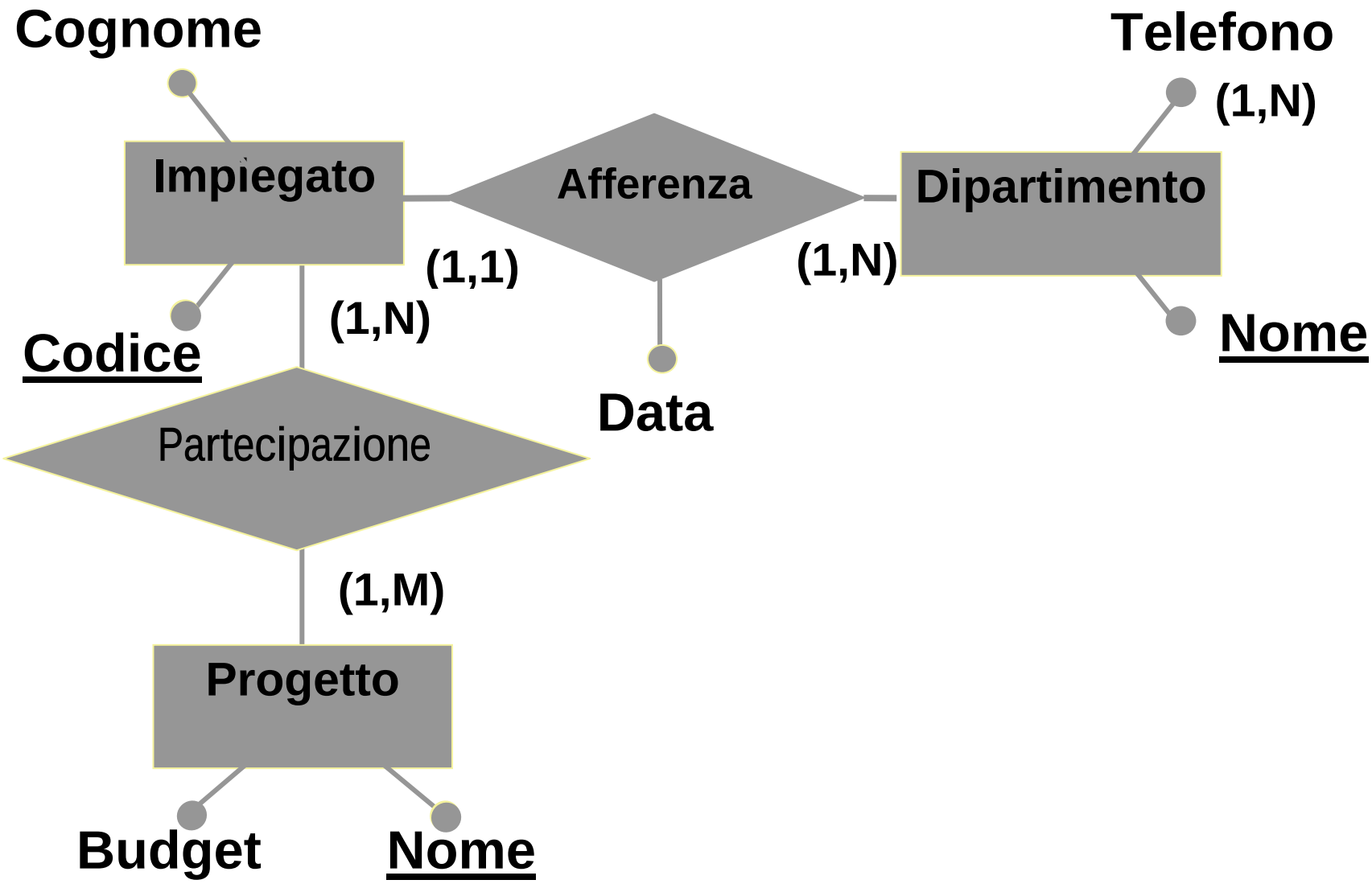
Il collegamento fra le entità avviene attraverso associazioni, nel senso matematico del termine; sul piano semantico, la relazione si può esprimere attraverso un gruppo verbale: un autore SCRIVE un libro, un libro APPARTIENE ad un genere, un operaio LAVORA su un turno; graficamente la relazione si esprime come in figura :



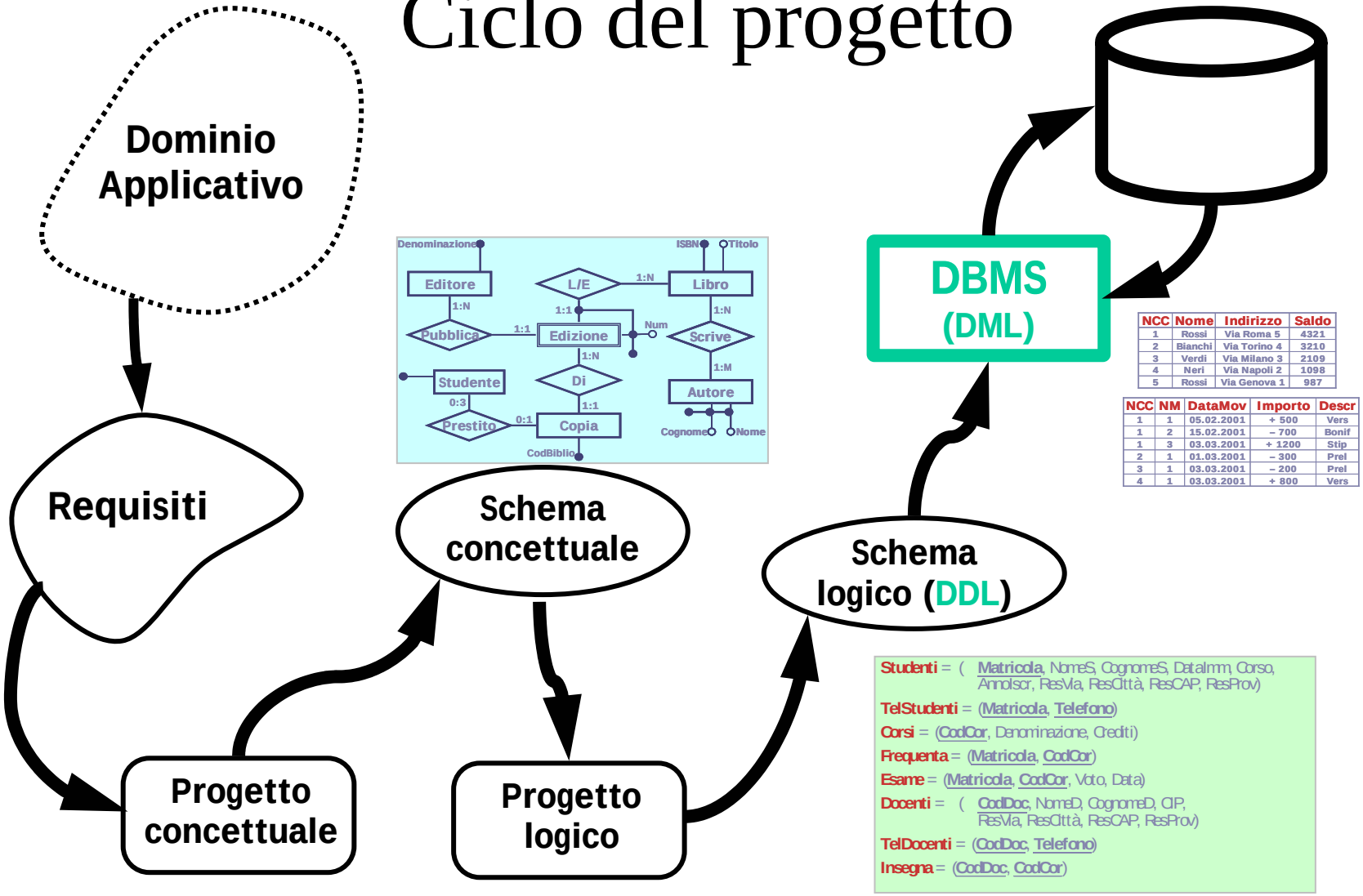
La associazione R ha in generale un rapporto di *cardinalità* M:N, dove M e N sono i numeri massimi di occorrenze di associazione a cui può partecipare un occorrenza di E e E' rispettivamente; in particolare, se M=1 o N=1 si ottiene un rapporto di cardinalità 1:N oppure M:1 infine se M=1 e N=1 si ottiene un rapporto di cardinalità 1:1. Si possono infine definire oltre che quelle massime le cardinalità minime - 0 o 1 - di ogni occorrenza di entità in una associazione R.

La associazione, infine, può essere caratterizzata, al pari dell'entità, da *attributi*, rappresentati come quelli delle entità.

UN ESEMPIO DI SCHEMA ENTITA' RELAZIONE



Ciclo del progetto



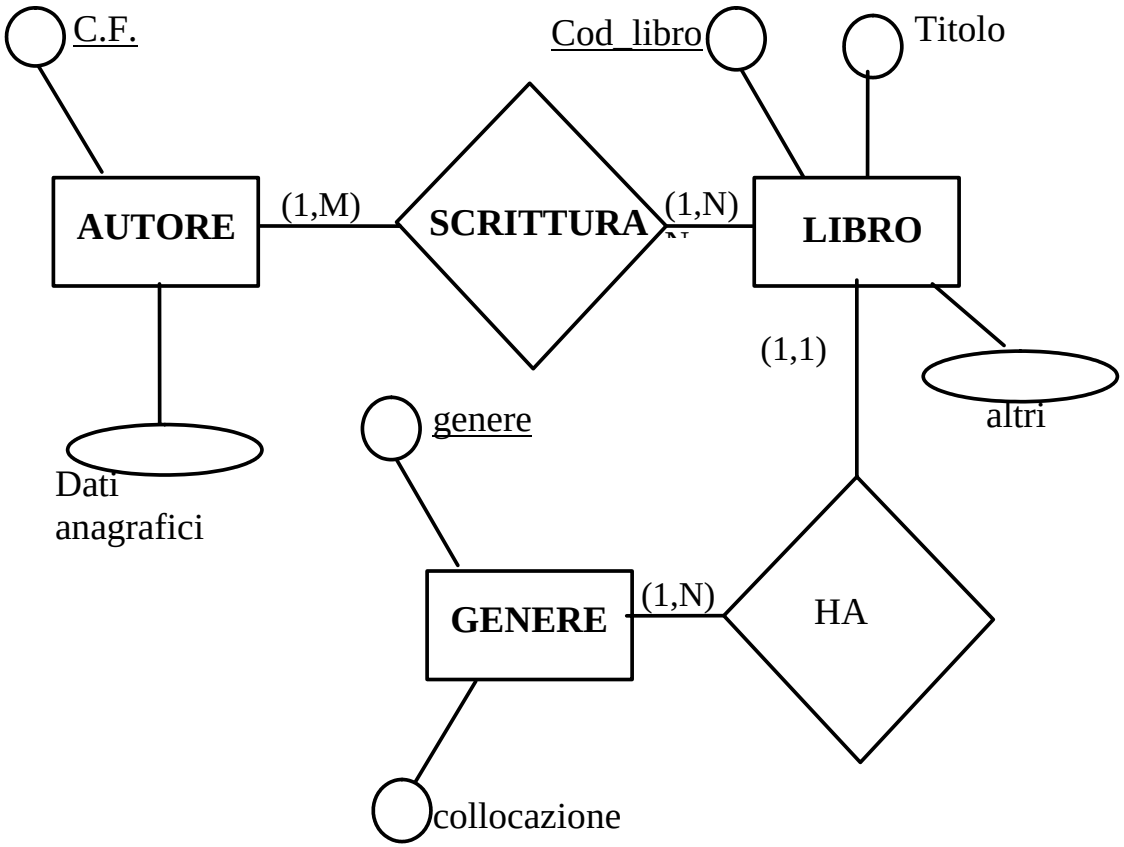
REQUISITI BASE DATI CASA EDITRICE

Le specifiche relative a una base dati di una casa editrice di libri che ne gestisce la vendita sono le seguenti:

Si vuole rappresentare una base di dati per una casa editrice che gestisce i libri scritti dai suoi clienti autori.
Per gli autori, identificati dal codice fiscale, descrivere il cognome e nome, la data di nascita, e l'indirizzo.
Per i libri rappresentiamo il codice, il titolo, il costo e il codice del contratto.
Supponiamo che un autore possa scrivere più libri e che un libro si riferisca a più autori.
Ciascun libro ha inoltre un genere (storia, letteratura, rosa,…)**;** per ciascun genere rappresentiamo il nome del genere, il responsabile ed la collocazione in libreria.

- Definire lo schema entità/relazione.
- Definire lo schema relazionale.
- Impostare il problema in Access (tabelle e "relazioni").
- Popolare la base dei dati.
- Usando QBE/SQL scrivere le interrogazioni relative alle seguenti operazioni:
    - ……

SCHEMA ER PER LA BASE DATI: EDITRICE



REQUISITI BASE DATI INFORMAZIONI TURISTICHE

Le specifiche relative a una base dati di un ente turistico che fornisce le informazioni sui ristoranti di una città sono le seguenti:

Si vuole rappresentare una base di dati di un ente turistico che fornisce informazioni sui **ristoranti** delle varie **zone** di una città fornendo i tipi di **cucina** offerti e le **carte di credito** convenzionate.

Per **i ristoranti**, identificati da un codice, descrivere il nome e l'indirizzo.

Per **le zone** rappresentiamo il codice della zona ed il relativo nome.

Ciascun ristorante è convenzionato con almeno una **carta di credito** di cui occorre descrivere il codice della carta e il nome della carta.

Per ciascuna **cucina** rappresentare il tipo e la relativa descrizione

1) Definire lo schema entità/relazione.

2) Definire lo schema relazionale

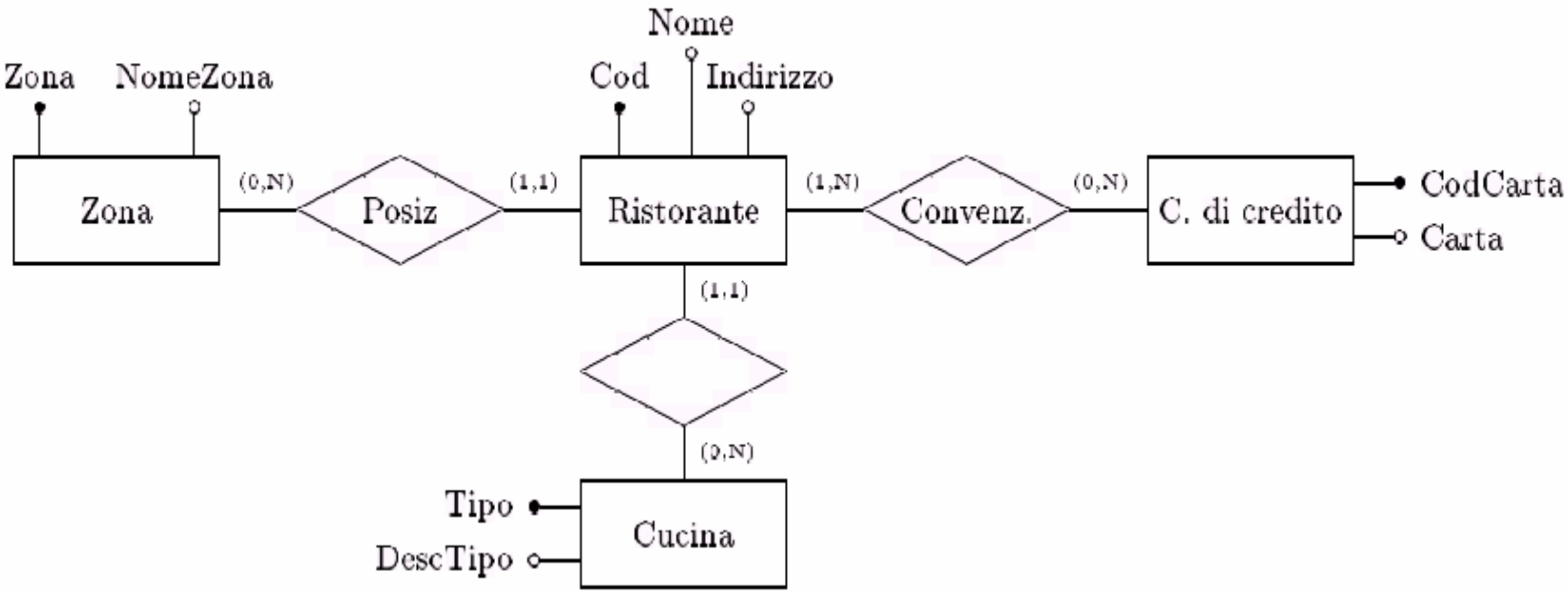
3) Impostare il problema in Access (tabelle e "relazioni").

4) Popolare la base dei dati.

5) Usando QBE/SQL scrivere le interrogazioni relative alle seguenti operazioni:

- ……

SCHEMA E/R PER LA BASE DATI INFORMAZIONI TURISTICHE



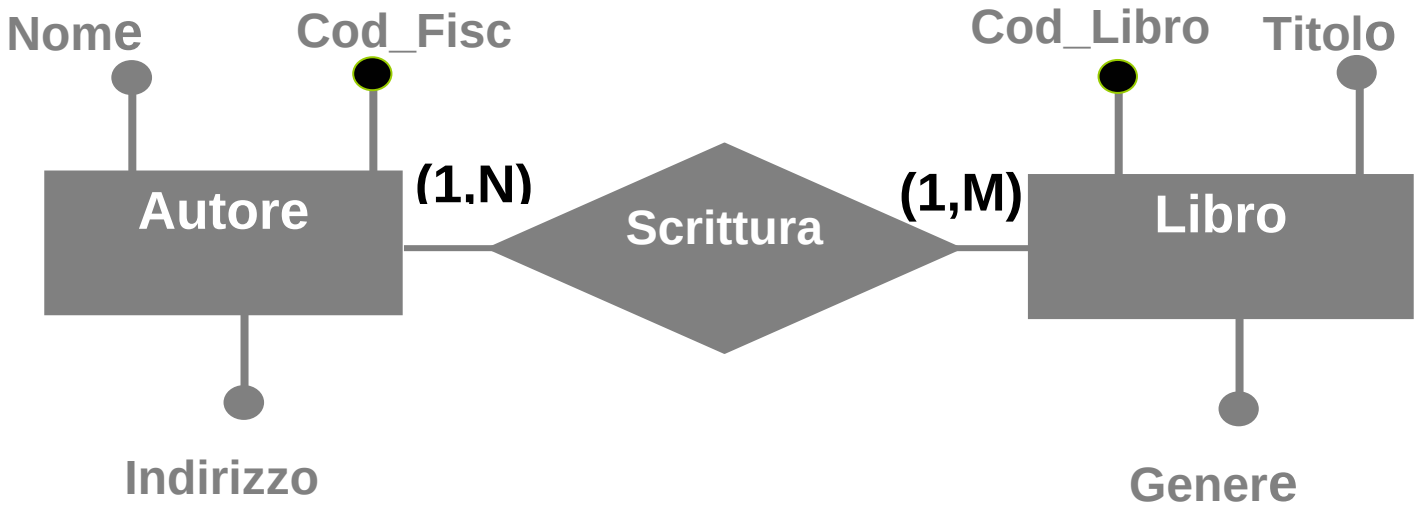
16. PROGETTO LOGICO: DEFINIZIONE DATABASE RELAZIONALE

INTRODUZIONE

In fase di progetto, lo schema concettuale del database (formalizzato con uno schema ER) viene trasformato nello schema logico seguendo regole di traduzione per la risoluzione delle relazioni concettuali.

Con riferimento ad esempi di immediata interpretazione, nel seguito si riportano i fondamenti della traduzione di uno schema E-R in uno relazionale.

RELAZIONI MOLTI A MOLTI

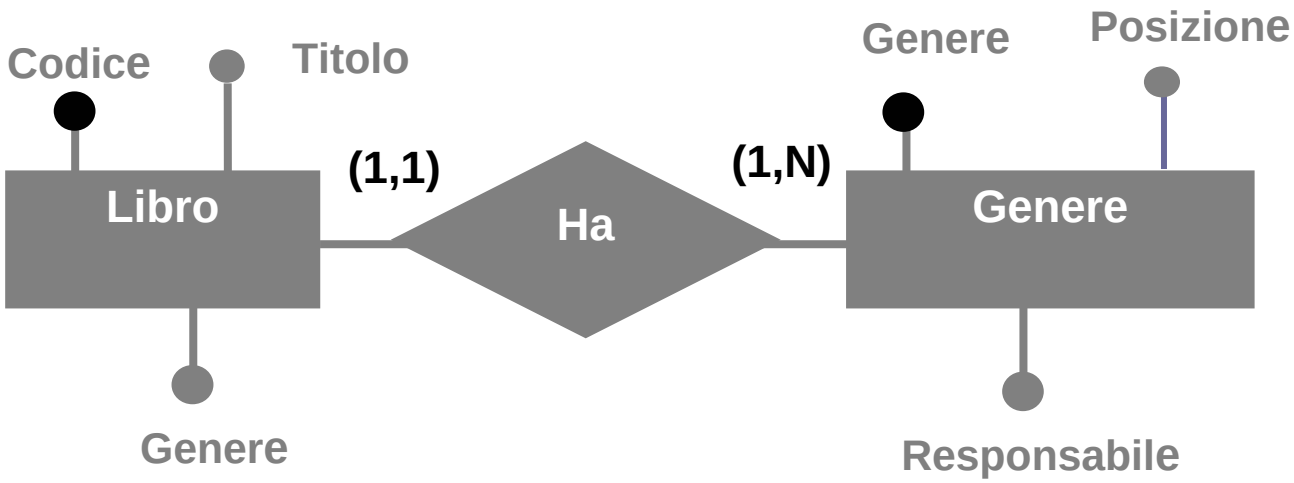


AUTORE(<u>Cod_Fisc</u>, Nome, Indirizzo)

SCRITTURA(<u>Cod_Fisc</u> : AUTORE, <u>Cod_Libro</u> : LIBRO)

LIBRO(<u>Cod_Libro</u>, Titolo, Genere, …)

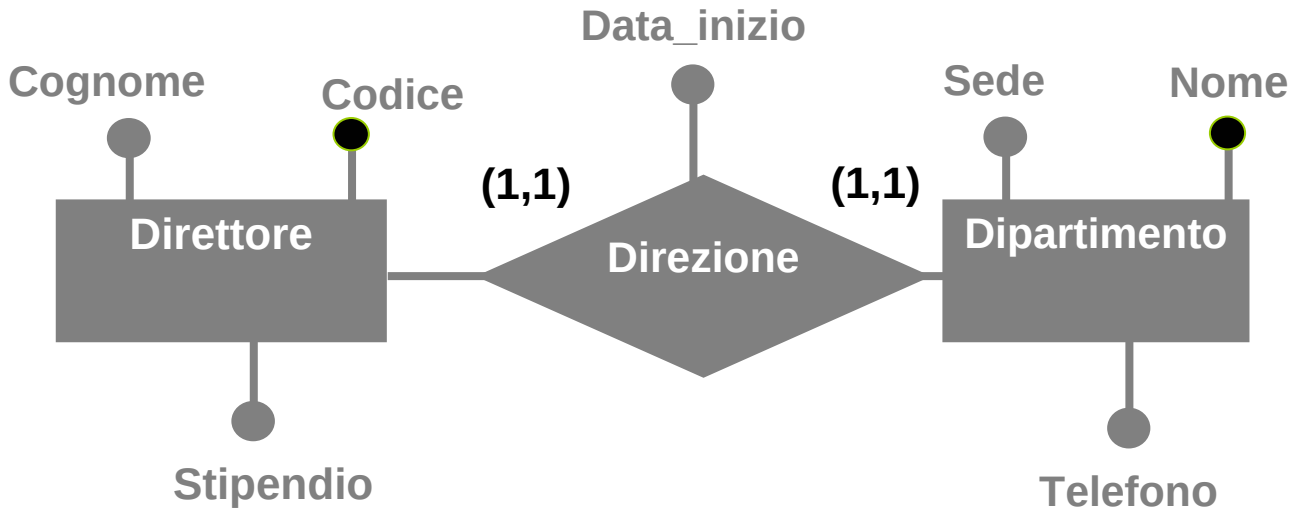
RELAZIONI UNO A MOLTI



LIBRO(<u>Codice</u>, Titolo, Genere: GENERE, …)

GENERE(<u>Genere</u>, Posizione, Responsabile)

RELAZIONI UNO A UNO



DIRETTORE (Codice, Cognome, Stipendio)

DIPARTIMENTO (Nome, Sede, Telefono,
Codice:DIRETTORE, Data_inizio)

OUTPUT PROGETTO LOGICO: SCHEMA RELAZIONALE DATA BASE EDITRICE

AUTORI (Nome, Data_nascita, Indirizzo, **cod_fisc**)
AUT-LIBRI (**Id_autore**:AUTORI, **cod_libro**:LIBRI)
LIBRI(**Cod_libro**, Titolo, Costo, Genere:GENERE, Contratto)
GENERE (**Genere**, Collocazione, Responsabile)

Per impostare lo schema del database relazionale occorre poi – come già visto:

- definire le tabelle;
- definire i campi di ciascuna tabella;
- definire il tipo di ciascun campo;
- definire i vincoli di integrità dei singoli campi;
- definire la chiave primaria di ciascuna tabella;
- definire le chiavi esterne delle tabelle;
- definire gli attributi eventualmente associati ad indice

La definizione delle chiavi esterne e di quelle primarie equivale alla definizione delle relazioni fra le tabelle.

I TIPI DEI DATI

- *Tipo numerico esatto*, con vari possibili intervalli di definizione (intero corto o lungo); comprende anche i dati rappresentati in valuta (Lire, dollari, sterline,...); sono anche diffuse notazioni (ereditate dal COBOL) nelle quali si specifica il numero di cifre e il fattore di scala; ad esempio:

  Nome INTEGER
  Nome SMALLINT
  Nome NUMERIC
  Nome DECIMAL (6,2)

- *Tipo numerico approssimato*, con varie precisioni (singola e doppia); ad esempio:

  Nome REAL
  Nome FLOAT
  Nome DOUBLE PRECISION

- *Tipo stringa di caratteri e di bit* a lunghezza fissa o variabile; ad esempio

  Nome CHAR (20)
  Nome VARCHAR (20)
  Nome BIT (16)

- *Tipo "temporale"*,per rappresentare date dell'anno e ora della giornata; ad esempio:

  Nome DATE *// anno, mese, giorno*
  Nome TIME *// ora, minuti, secondi*

DEFINIZIONE TABELLE

In SQL la definizione di tabella e delle sue proprietà (*schema*) avviene con la frase CREATE TABLE che qui si esemplifica, sempre con riferimento alla base di dati EDITRICE:

ES.17

```
CREATE TABLE AUTORI
 ( Nome varchar(20) ,
   Data_nascita date,
   Indirizzo varchar(50),
   cod_fisc varchar(15) CONSTRAINT Pk primary key
);
CREATE TABLE AUT_LIBRI
( Id_autore varchar(50) CONSTRAINT Fk2 REFERENCES AUTORI(cod_fisc),
  cod_libro varchar(50) CONSTRAINT Fk3 REFERENCES LIBRI(Cod_libro),
 CONSTRAINT Pk primary key( Id_autore, cod_libro)
);
CREATE TABLE LIBRI
 ( Cod_Libro varchar(2) CONSTRAINT Pk primary key,
   Titolo       varchar(50),
   Costo        integer,
   Genere      varchar(15) CONSTRAINT Fk1 REFERENCES GENERE(Genere),
   Contratto  varchar(3)
);
CREATE TABLE GENERE
( Genere varchar(50) CONSTRAINT Pk primary key,
  Collocazione varchar (50),
  Responsabile varchar (50)
);
```

DEFINIZIONE TABELLE (2)

Si riporta infine la definizione delle tabelle per il caso informazioni turistiche:

Es.18

```
CREATE TABLE CARTEDICREDITO
( Codcarta char(1) CONSTRAINT PK primary key,
  Carta varchar(50)
);
CREATE TABLE CONVENZIONI
(   Cod   integer  CONSTRAINT FK3  REFERENCES Ristoranti(Cod),
    CodCarta  char(1) CONSTRAINT FK4  REFERENCES CartediCredito(CodCarta),
    CONSTRAINT PK primary key  (Cod,CodCarta)
);
CREATE TABLE CUCINE
( Tipo char(1) CONSTRAINT PK primary key ,
  DescrTipo varchar(50)
);
CREATE TABLE RISTORANTI
(  Cod   integer  CONSTRAINT PK  primary key,
    Nome varchar(50),
    Indirizzo varchar(50),
    Tipo  char(1)  CONSTRAINT FK2 REFERENCES CUCINE(Tipo),
    Zona char(1) CONSTRAINT FK1  REFERENCES ZONE(Zona)
);
CREATE TABLE ZONE
( Zona char(1)  CONSTRAINT PK primary key ,
    NomeZona varchar(50)
);
```