

In merito al rapporto fra informazione e conoscenza, D. Bogliolo in *Knowledge vs Information*¹ delinea un percorso che parte da Karl M. Wiig, attraversa Charles H. Green ed arriva a Gene Bellinger. Quest'ultimo completando Green, che aveva ripreso i concetti espressi da Wiig, sostanzialmente propone una struttura piramidale che vede alla base i dati, quindi le informazioni, ancora sopra la conoscenza, e quindi la saggezza:

Dato	rozzo, non agito dal ricevente
Informazione	i dati elaborati cognitivamente, cioè trasformati in un qualche schema concettuale
Conoscenza	informazione applicata, come un senso comune[...], che "sa" quando e come usarla
Saggezza	verità eterna distillata dalla conoscenza

Limitando la nostra attenzione ai primi tre livelli, possiamo intendere:

- per dato, la rappresentazione codificata di un frammento di realtà osservata;
 - es. 36, 38, 39
- per informazione, il significato associato al dato in un certo contesto;
 - es. 39 è la misura della temperatura di Mario alle ore 10.00,
 - es. 38 è la misura della temperatura di Mario alle ore 16.00,
 - es. 36 è la misura della temperatura di Mario alle ore 20.00;
- per conoscenza, informazione che si può estrarre mediante ragionamento sulle informazioni;
 - es. analizzando la serie di informazioni precedenti possiamo dire che Mario non ha più febbre.

Quindi la conoscenza deriva da una gestione dell'informazione secondo regole, scopi e contesto di riferimento.

Controllando la temperatura di Mario lungo una settimana si potrà ricavare una conoscenza sull'andamento della stessa nei diversi giorni e dire, eventualmente, che oscilla a giorni alterni fra valori alti e valori bassi. Ed ancora, la conoscenza dell'andamento della temperatura di Mario può divenire, insieme all'andamento delle temperature di altri individui, un insieme di informazioni da cui ricavare conoscenza sull'andamento di una malattia in un determinato gruppo.

Quindi le informazioni possono *fornire* conoscenza *diversa* in base alla contestualizzazione che viene scelta. Questa scelta è determinata da ciò di cui noi abbiamo bisogno, da ciò che vogliamo *estrarre* dall'insieme delle informazioni.

Possiamo dire che queste forniscono un insieme di stati di cose (di descrizioni) di una porzione di realtà che intendiamo analizzare. Quali stati di cose scegliere dalla porzione della realtà e come relazionare i loro individui e le loro proprietà determina la conoscenza che noi possiamo *estrarre*.

La conoscenza di una porzione di realtà può essere, quindi, rappresentata individuando rappresentazioni di insiemi di individui, con le relative proprietà, e relazioni che intercorrono fra essi.

Un modello² per la rappresentazione della conoscenza di una porzione della realtà è il modello Entity-Relationship³ (E-R; in italiano Entità-Relazioni).

Esso viene utilizzato nella modellizzazione di domini di conoscenza e, spesso, nella rappresentazione concettuale di una porzione di realtà nella prima fase della progettazione di una base di dati⁴.

¹ Bogliolo D., 2000, *Knowledge vs Information*, <http://www.aidainformazioni.it/pub/kmappunti12000.html>, verificato il 20 Aprile 2009

² Albano A., Ghelli G., Orsini R., in *Fondamenti di basi di dati* (2005), affermano che “Un modello astratto è la rappresentazione formale di idee e conoscenze relative a un fenomeno.
Aspetti di un modello:

- il modello è la rappresentazione di certi fatti;
- la rappresentazione è data con un linguaggio formale;

il modello è il risultato di un processo di interpretazione, guidato dalle idee e conoscenze possedute dal soggetto che interpreta.

³ E' stato ideato da Peter Chen nel 1976.

⁴ Verrà illustrata subito sotto.

Progettazione di base di dati

Viene illustrato il procedimento per la progettazione di una base di dati; gli intenti sono quelli di:

1. analizzare come sia possibile rappresentare conoscenza attraverso un modello (Entity-Relationship);
2. familiarizzare con la gestione delle informazioni in quanto componente significativa nei sistemi informativi aziendali e nei sistemi per la gestione della conoscenza esplicita e formale.

Una base di dati è un insieme di dati organizzati secondo uno schema prefissato; lo schema è il risultato di una progettazione che si sviluppa attraverso tre fasi:

1° fase: *progettazione concettuale*,

2° fase: *progettazione logica*,

3° fase: *progettazione fisica*.

Ciascuna fase si sviluppa secondo dei modelli scelti e restituisce uno schema:

progettazione concettuale → *schema concettuale*,

progettazione logica → *schema logico*,

progettazione fisica → *schema fisico*.

Nella prima fase, della Progettazione concettuale, si realizza una rappresentazione della realtà, che è la nostra conoscenza della realtà, attraverso un modello concettuale. Utilizzando il modello Entità-Relazioni si rappresentano in uno schema/diagramma i concetti, le loro proprietà e le relazioni fra essi (i concetti si chiamano entità e le associazioni fra essi si chiamano relazioni).

Nella seconda fase si sceglie il modello di database che si utilizzerà e, in base a questa scelta, si individueranno le strutture per le entità e le modalità per realizzare le relazioni fra esse.

Nella terza fase si delinea la struttura delle entità affinché sia possibile creare la versione digitale.

Progettazione concettuale

Nella prima fase, quella della Progettazione concettuale, vengono individuate le classi della realtà che vogliamo analizzare e le relazioni fra esse; entrambe vengono generate attraverso un processo di astrazione.

“Una astrazione è un procedimento concettuale per mezzo del quale creiamo un concetto più generale (e in tal senso più astratto) a partire da concetti più elementari (e in tal senso più concreti), rimuovendo dalla descrizione del concetto gli aspetti di dettaglio e particolari e mettendo in evidenza gli aspetti comuni e generali.” (Batini C., Grosso R.)

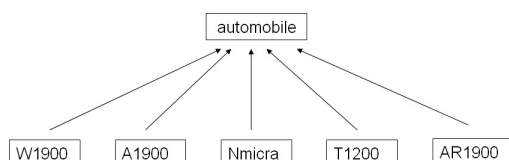
Diverse sono i possibili processi di astrazione; qui prendiamo in considerazione quelli in base a:

- *classificazione*: identifica classi di oggetti del mondo reale aventi proprietà comuni;
- *aggregazione*: definisce un nuovo concetto a partire da concetti componenti;
- *generalizzazione*: definisce una classe astraendo dalle differenze esistenti tra due o più classi.

Astrazione di classificazione

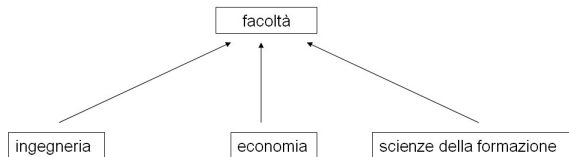
Definizione di una classe a partire da un insieme di oggetti, della porzione di realtà di interesse, aventi proprietà comuni. Gli oggetti che definiscono la classe sono chiamati istanze della classe.

Esempio 1.



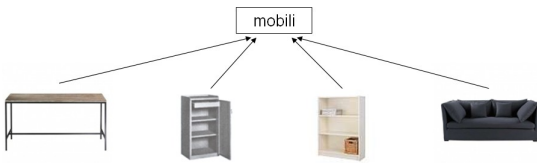
W1900 è una istanza
della classe *automobile*.

Esempio 2.



Ingegneria è una istanza della classe *facoltà*.

Esempio 3.



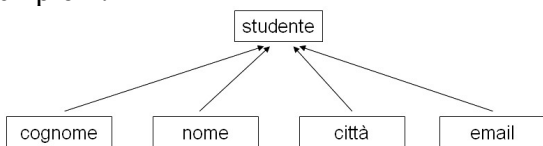
Il mobile *tavolo* è una istanza della classe *mobili*.

Un oggetto può essere membro di più classi; non è univocamente determinato, ma dipende dal punto di vista che si sceglie per analizzare la porzione di realtà che si vuole modellare.

Astrazione di *aggregazione*

A partire da un insieme di classi della realtà osservata, si crea una nuova classe di cui le classi di partenza sono parti componenti.

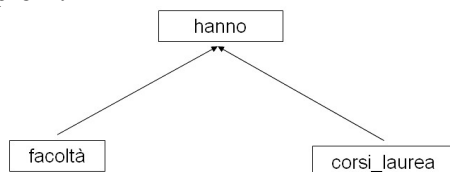
Esempio 1.



Le classi *cognome*, *nome*, *città*, *email* sono componenti di *studente*.

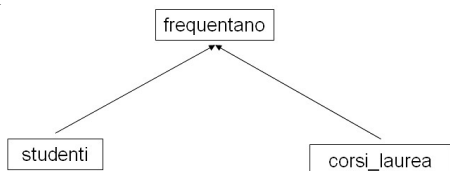
Un caso particolare si ha quando le classi che vengono aggregate sono già delle classi esistenti nello diagramma che si sta realizzando; in questo caso abbiamo una astrazione di *associazione* che rappresenta una *relazione* fra classi.

Esempio 2.



La relazione *hanno* esprime che le *facoltà* hanno dei *corsi di laurea*.

Esempio 3.

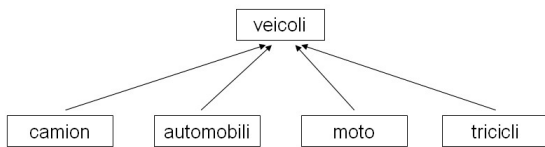


La relazione *frequentano* esprime che gli studenti frequentano *corsi di laurea*.

Astrazione di *generalizzazione*

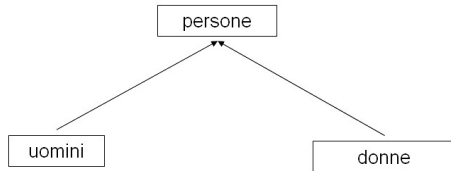
Si definisce una classe, chiamata *superclasse* a partire da più classi, chiamate *sottoclassi*; le istanze delle sottoclassi sono anche istanze della superclasse.

Esempio 1.



Le istanze di *moto* sono un sottoinsieme delle istanze di *veicoli*.

Esempio 2.



Le istanze di *moto* sono un sottoinsieme delle istanze di *veicoli*.

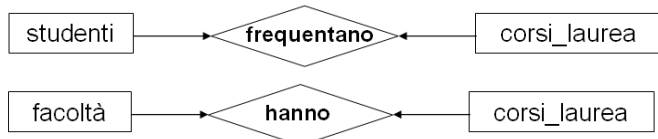
Ciò che caratterizza un veicolo caratterizza anche un camion, una moto, un'automobile, un triciclo e così ciò che caratterizza una persona, caratterizza anche un uomo ed una donna.

Tuttavia, rispetto alle loro superclassi, gli uomini e le donne hanno caratteristiche proprie particolari e così pure i camion, le automobili, le moto, i tricicli.

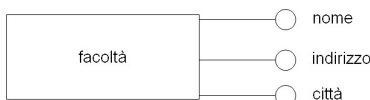
Nella progettazione concettuale, attraverso processi di astrazione si individuano le classi⁵ del dominio preso in considerazione e le relazioni⁶ che le legano. Nello schema concettuale le classi vengono così rappresentate:



e così le relazioni fra classi:



Per ciascuna classe (entità) e/o relazione vengono indicati gli attributi, che descrivono proprietà elementari di tali entità o relazioni, che siano ovviamente significative per l'applicazione. Ad esempio:



Nome, *indirizzo*, *città* sono attributi per l'entità *facoltà*; un attributo associa un valore del proprio dominio ad una istanza⁷ dell'entità (classe).

Domini possibili sono:

- interi, reali, booleani, caratteri,
- intervalli di interi e di caratteri,
- stringhe di caratteri,
- domini definiti dall'utente.

Fra le entità esistono delle relazioni; esse possono essere:

- 1:1 → ad un elemento in una entità corrisponde un solo elemento nell'altra entità e viceversa;
- 1:n → ad un elemento della prima entità corrispondono n elementi nella seconda e ad un elemento nella seconda corrisponde un elemento nella prima;
- n:m → ad un elemento della prima entità corrispondono m elementi nella seconda e ad un elemento nella seconda corrispondono n elementi nella prima.

⁵ Attraverso processi di astrazione di *classificazione* o di *aggregazione*.

⁶ Attraverso processi di aggregazione fra entità.

⁷ Gli oggetti che definiscono la classe sono chiamati istanze della classe.

Leggere, per approfondire, *Vincoli fra relazioni* in Approfondimenti 1.

Come esempio di progettazione concettuale, facciamo riferimento al problema di voler realizzare una base di dati per la gestione degli studenti iscritti ad una Università, rappresentando gli studenti, i Corsi di Laurea, le Facoltà e le relazioni fra queste entità.

Progettazione concettuale

Si parte da una analisi della situazione reale; da questa si deve ottenere uno schema, chiamato schema/diagramma *Entità-Relazioni* (*Entity-Relationships*), che rappresenta le principali entità nel dominio del problema, gli attributi delle entità e le relazioni tra le entità.

Le entità rappresentano classi di oggetti; in questo caso possono essere: *studenti*, *corsi_laurea*, *facoltà*.

Le entità hanno degli attributi che le definiscono:

- *studenti*: matricola, cognome, nome, indirizzo;
- *facoltà*: codice_facoltà, nome, indirizzo;
- *corsi_laurea*: codice_corso_laurea, nome, indirizzo.

Le entità sono rappresentate come mostrato in Figura 1.

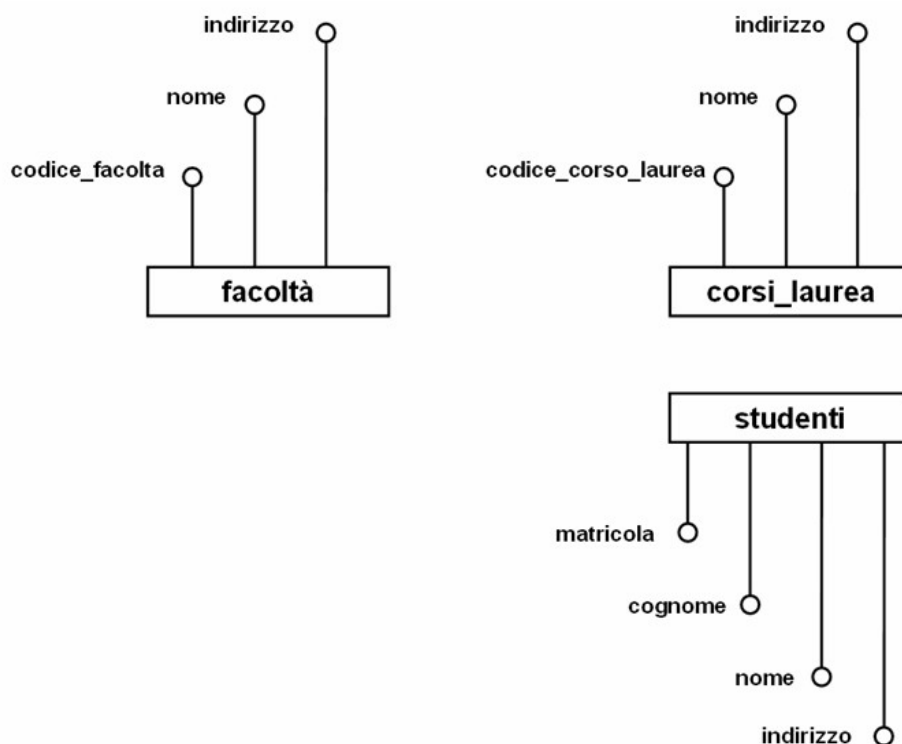


Figura 1-Entità e attributi

Fra le entità esistono delle relazioni; nel nostro esempio:

- fra *facoltà* e *corsi_laurea* esiste una relazione 1:n, ad una facoltà corrispondono più corsi di laurea; il viceversa è 1:1, un corso di laurea è di una facoltà;
- fra *corsi_laurea* e *studenti* esiste una relazione 1:n, ad un corso di laurea corrispondono più studenti; il viceversa è 1:1, uno studente è di un corso di laurea.

Lo schema *E-R* risultante, con entità, attributi, relazioni, è presentato in Figura 2.

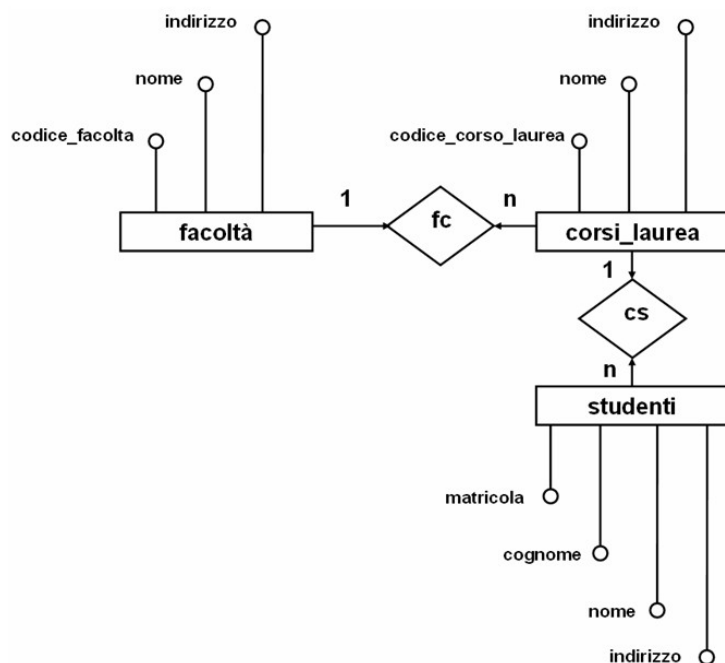


Figura 2-Schema *Entità Relazioni* (E-R)

Se prendiamo in considerazione anche le materie che gli studenti debbono studiare, abbiamo una nuova entità *materie* con attributi *cod_mater* e *denominazione_mat*.

Fra *studenti* e *materie* esiste una relazione n:m, uno studente può studiare più materie ed una materia può essere studiata da più studenti, comunque può esistere una materia che non è studiata da alcun studente e così che uno studente può non studiare alcuna materia.

La relazione n:m, fra *studenti* e *materie* è rappresentata in Figura 3.

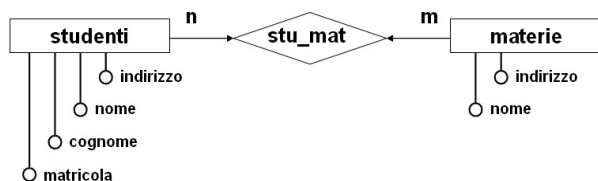


Figura 3-Relazione n:m fra *studenti* e *materie*

Il diagramma E-R finale, prendendo in considerazione anche l'entità *materie*, è rappresentato in Figura 4.

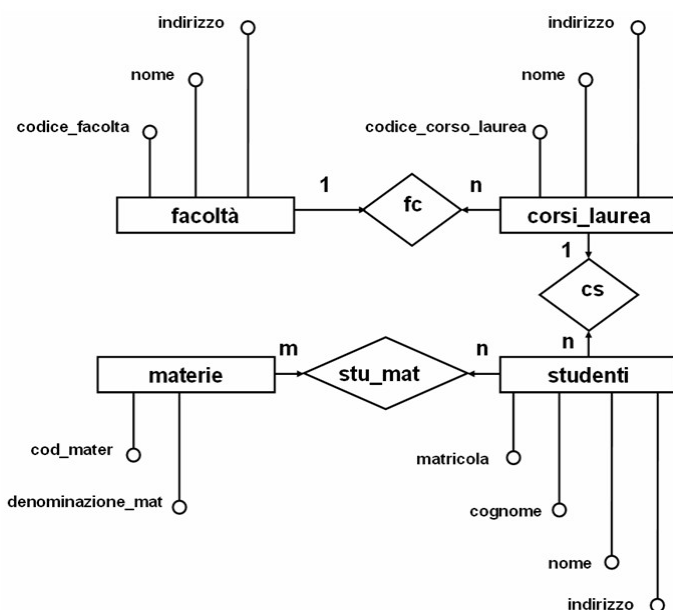


Figura 4-Schema E-R, risultato della progettazione concettuale

Progettazione logica

Nella seconda fase della progettazione (*progettazione logica*), si sceglie il modello di database che si vuole utilizzare per rappresentare la nostra realtà; è possibile scegliere fra database *gerarchico*, *reticolare* *relazionale*. Il modello che qui prendiamo in considerazione è quello relazionale. Nella progettazione logica, occorre trasformare le entità e le relazioni del modello concettuale secondo le regole del modello di database scelto. In quello relazionale le *entità* sono rappresentate da tabelle. Le tabelle sono chiamate *relazioni*⁸. Ciascuna relazione rappresenta una entità definita nello schema concettuale. Una tabella/relazione è formata da righe, chiamate *n-ple*. Gli attributi delle entità diventano le colonne della tabella. Per ciascuna tabella/relazione occorre scegliere un attributo, i cui valori identificano univocamente le n-ple. Tale attributo viene chiamato *chiave* e i suoi valori debbono essere tutti diversi fra loro. Nella progettazione logica, per evitare confusioni, le *relazioni* fra tabelle/relazioni è preferibile chiamarle *associazioni*.

Nel nostro esempio, pensando di aver inserito dei dati, potremmo avere le seguenti relazioni.

facoltà

#codice_facoltà	nome	indirizzo
10	Scienze della Formazione	Via ..
20	Giurisprudenza	Via ..

Il campo codice_facoltà rappresenta la chiave. Per indicare che un campo rappresenta la chiave della relazione, si fa precedere il suo nome da un simbolo, in questo caso un #.

corsi_laurea

#codice corso laurea	nome	indirizzo
10	Consulente lavoro e impresa	Via ..
20	Form. e gest. risorse umane	Via ...
30	Format.e-learning e multimed.	
40	Scienze della Formazione Primaria	

studenti (per comodità è stato inserito un solo attributo nella relazione *studenti*)

# matricola	studente
10	Bianchi
20	Rossi
30	Neri

materie

# cod mater	denominazione mat
1	material

⁸ Le tabelle sono chiamate *relazioni*, in quanto, dalla matematica, vengono viste come sottoinsiemi del prodotto cartesiano fra insiemi. L'argomento viene trattato in Approfondimenti 2.

2	materia2
3	materia3
4	materia4
5	materia5

Osservando le varie tabelle notiamo che le associazioni fra le relazioni non sono state esplicitate; non possiamo capire, cioè,

- che il Corso di Laurea ‘Scienze della Formazione Primaria’ sia della Facoltà ‘Scienze della Formazione’;
- che la Facoltà di ‘Scienze della Formazione’ abbia i Corsi di Laurea ‘Form. e gest. risorse umane’, ‘Scienze della Formazione Primaria’, ‘Format.e-learning e multimed.’;
- che il Corso di Laurea ‘Consulente lavoro e impresa’ sia della Facoltà di Giurisprudenza,
- che lo studente Bianchi può studiare ‘materia1’, ‘materia3’, ‘materia4’;

Dobbiamo impostare un ‘meccanismo’ che ci permetta di esplicitare le associazioni fra relazioni. Ciascun tipo di database ha proprie modalità; in questo contesto analizziamo quelle relative a database relazionali.

Una relazione **uno-a-molti** fra due relazioni, ad esempio:

facolta

#codice_facolta	nome	indirizzo
10	Scienze della Formazione	
20	Giurisprudenza	

corsi_laurea

#codice corso laurea	nome	indirizzo
10	Consulente lavoro e impresa	
20	Form. e gest. risorse umane	
30	Format.e-learning e multimed.	
40	Scienze della Formazione Primaria	

viene realizzata introducendo, come attributo, nella relazione lato n , la chiave della relazione lato 1. Quindi ogni n -pla della relazione n avrà i suoi campi ed in più un campo, chiamato *chiave esterna*, il cui valore esprime l’associazione con il record corrispondente dell’altra relazione.

Allora la relazione ‘corsi_laurea’ avrà la seguente struttura:

corsi_laurea

#codice corso laurea	nome	indirizzo	codice_facolta
10	Consulente lavoro e impresa		20
20	Form. e gest. risorse umane		10
30	Format.e-learning e multimed.		10
40	Scienze della Formazione Primaria		10

Dalla relazione precedente si può notare che:

- il Corso di Laurea ‘Consulente lavoro e impresa’ è della Facoltà che ha codice_facoltà uguale a 20, cioè ‘Giurisprudenza’;
- il Corso di Laurea ‘Scienze della Formazione Primaria’ è della Facoltà che ha codice_facoltà uguale a 10, cioè ‘Scienze della Formazione’;
- la facoltà con codice_facoltà uguale a 10 ha i Corsi di Laurea seguenti: ‘Form. e gest. risorse umane’, ‘Scienze della Formazione Primaria’, ‘Format.e-learning e multimed.’.

Una relazione **molti-a-molti**, fra due tabelle, ad esempio:

studenti (per comodità è stato inserito un solo attributo nella relazione *studenti*)

# matricola	studente
10	Bianchi
20	Rossi
30	Neri

materie

# cod mater	denominazione mat
1	materia1
2	materia2
3	materia3
4	materia4
5	materia5

viene realizzata **introducendo un’apposita tabella**, i cui record contengono le chiavi dei record che si corrispondono nelle due tabelle (lo studente X studia la materia A, la materia B, ...):

stu_mat

#matricola	#cod_mater
10	1
20	3
20	5
10	4

Lo studente con matricola 10, studia le materie con *cod_mater* 1 e 4; lo studente con matricola 20, quelle con *cod_mater* 3 e 5. Questa tabella ha una chiave composta formata dai campi matricola e cod_mater.

Infine una relazione **uno-a-uno** fra due tabelle, viene realizzata introducendo, come attributo, in una qualsiasi delle due tabelle la chiave dell’altra. Ad esempio fra le due seguenti tabelle:

nazioni

#cod_naz	nazione
10	Italia
20	Francia
30	Inghilterra

capitali

#cod_cap	capitale
1	Parigi
2	Londra
3	Roma

esiste una associazione 1:1, che può essere esplicitata nel seguente modo:

capitali

#cod_cap	capitale	cod_naz
1	Parigi	20
2	Londra	30
3	Roma	10

Nella progettazione logica, comunque, non occorre trasformare le entità in tabelle scrivendole così come abbiamo appena fatto. Occorre distinguere fra estensione e schema di una relazione; l'estensione è data dall'insieme di n-ple (Figura 5), lo schema è dato dal nome della relazione e dai suoi attributi (Figura 6).

Facoltà

# codice_facoltà	nome	indirizzo
10	Scienze della Formazione	
20	Giurisprudenza	

Figura 5-Estensione di una relazione

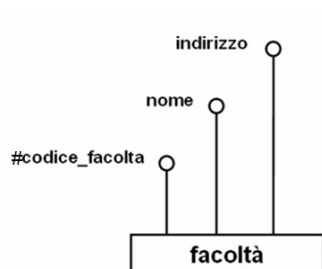


Figura 6-Schema di una relazione

Tutto il percorso, relativo alla progettazione logica, fino ad ora sviluppato, è un itinerario didattico per far capire come le entità e le relazioni definite nel modello concettuale si rappresentano in uno schema logico.

Una volta capito il tutto, quando si progetta si procede, in modo più snello, nel seguente modo:

1. analisi della realtà e realizzazione del diagramma relativo alla progettazione concettuale,
2. realizzazione del **diagramma** relativo alla progettazione logica, esplicitando in modo opportuno le associazioni fra le relazioni (tabelle).

Quest'ultimo diagramma è rappresentato in Figura 8.

Mettendo a confronto il diagramma per progettazione concettuale di Figura 7, con il diagramma per progettazione logica di Figura 8, si può notare che, nel secondo, vengono riportate le chiavi esterne (campi sottolineati) che esplicitano le seguenti associazioni:

- fra *facoltà* e *corsi_laurea*, relazione 1:n;
- fra *corsi_laurea* e *studenti*, relazione 1:n;
- fra *studenti* e *materie*, relazione n:m.

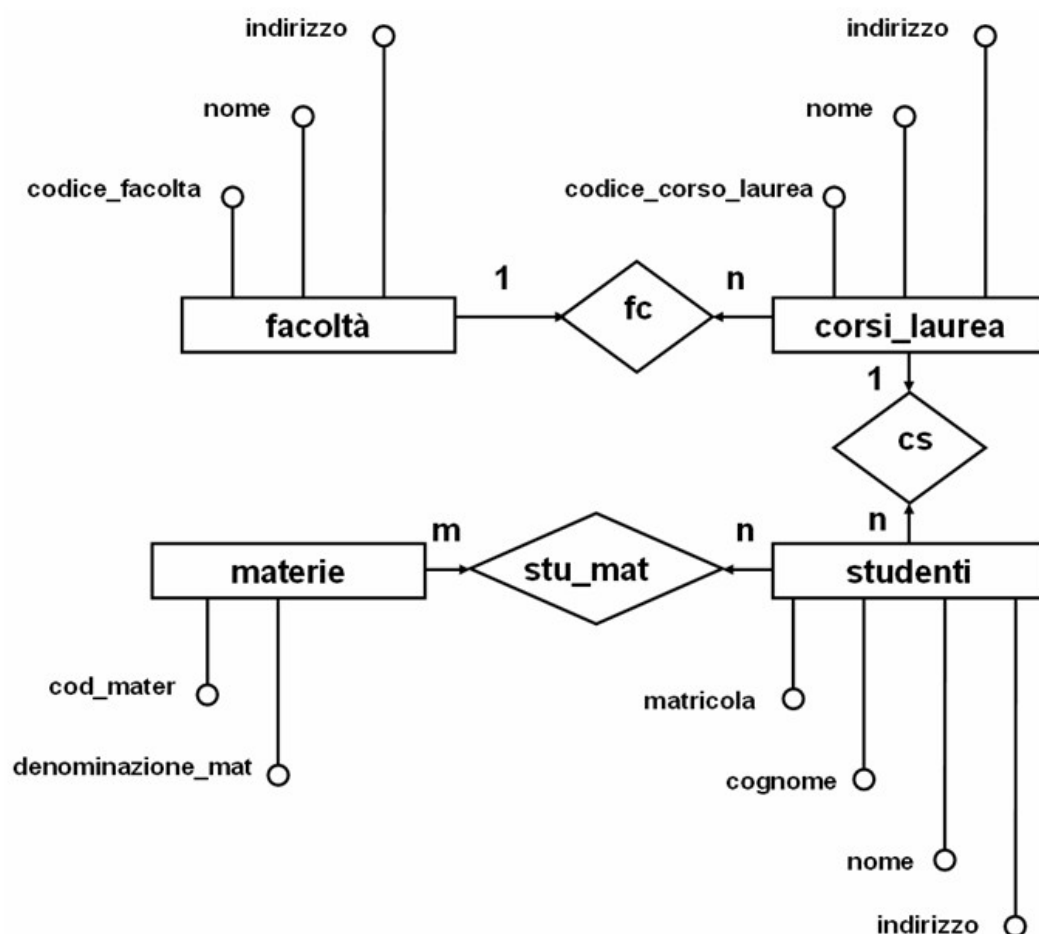
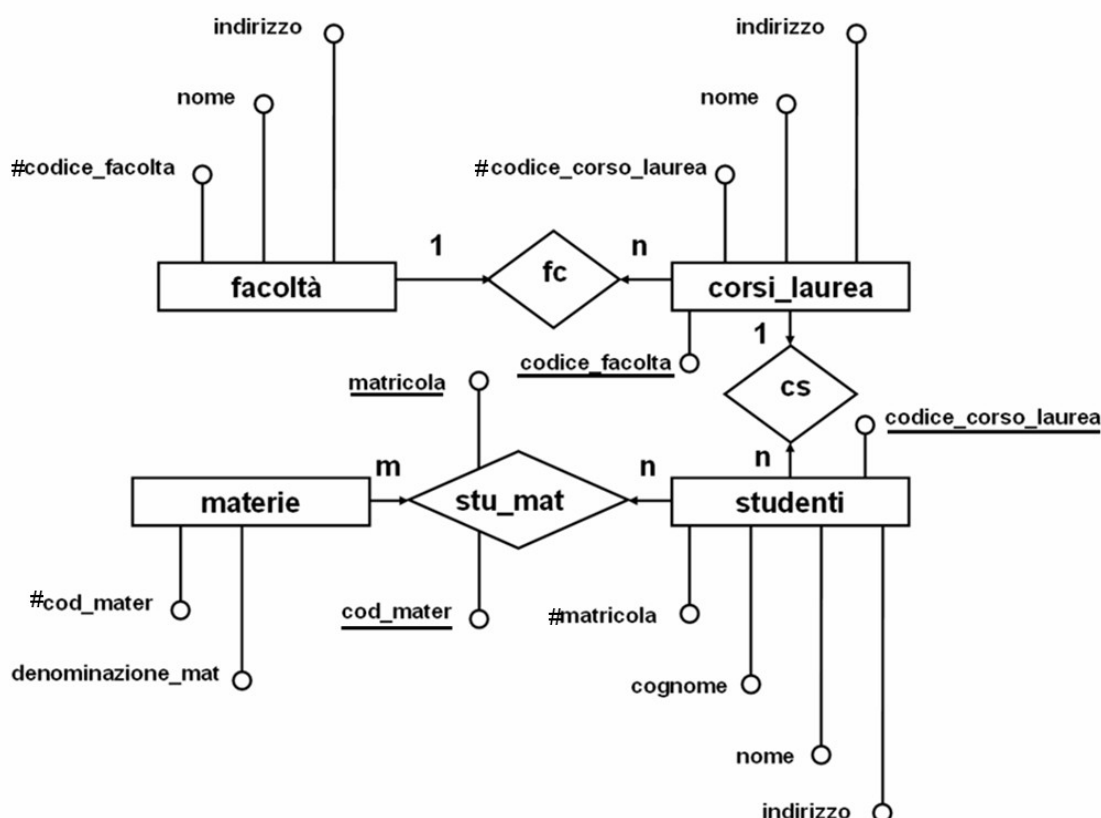


Figura 7-
Schema della
progettazione
concettuale

Figura 8-
Schema della
progettazione
logica



Come si può notare dalla Figura 8, le relazioni (*facoltà*, *corsi_laurea*, *studenti*, *materie*) sono rappresentate attraverso *schemi* che sono identici alle rappresentazioni delle entità nello schema concettuale.

Le relazioni, nel modello relazionale, debbono rispettare delle regole che tendono ad evitare problemi legati all'aggiornamento dei dati:

- in modifica
- in inserimento
- in cancellazione
- come informazioni ridondanti.

Queste si chiamano regole di *normalizzazione*; vengono spiegate in dettaglio in Approfondimenti 3.

Progettazione fisica

Nella progettazione fisica vengono definite le strutture di memorizzazione dei dati; nel nostro caso:

facoltà

```
(
    codice_facoltà: testo(20 ch9),
    nome: testo(100 ch),
    indirizzo: testo(255 ch)
)
```

corsi_laurea

```
(
    codice_corso_laurea: testo(20 ch),
    nome: testo(100 ch),

```

⁹ *ch* sta per carattere, *intero* per numero intero; possono esistere anche altri tipi di dati, qui non riportati.

```

        indirizzo: testo(255 ch),
        codice_facolta: testo(20 ch)
    )
    studenti
    (
        matricola: testo(10 ch),
        cognome: testo(80 ch),
        nome: testo(60 ch),
        indirizzo: testo(255 ch),
        codice_corso_laurea: testo(20 ch)
    )
    materie
    (
        cod_mater: intero,
        denominazione_mat: testo(200 ch)
    )
    stu_mat
    (
        matricola: testo(10 ch),
        cod_mat: intero
    )

```

La progettazione fisica rappresenta l'ultimo passaggio prima della costruzione del database sull'elaboratore. Occorre quindi specificare delle caratteristiche significative come ad esempio il tipo del dato e la sua grandezza.

La costruzione sull'elaboratore avviene utilizzando un DBMS¹⁰ (*Data Base Management System*).

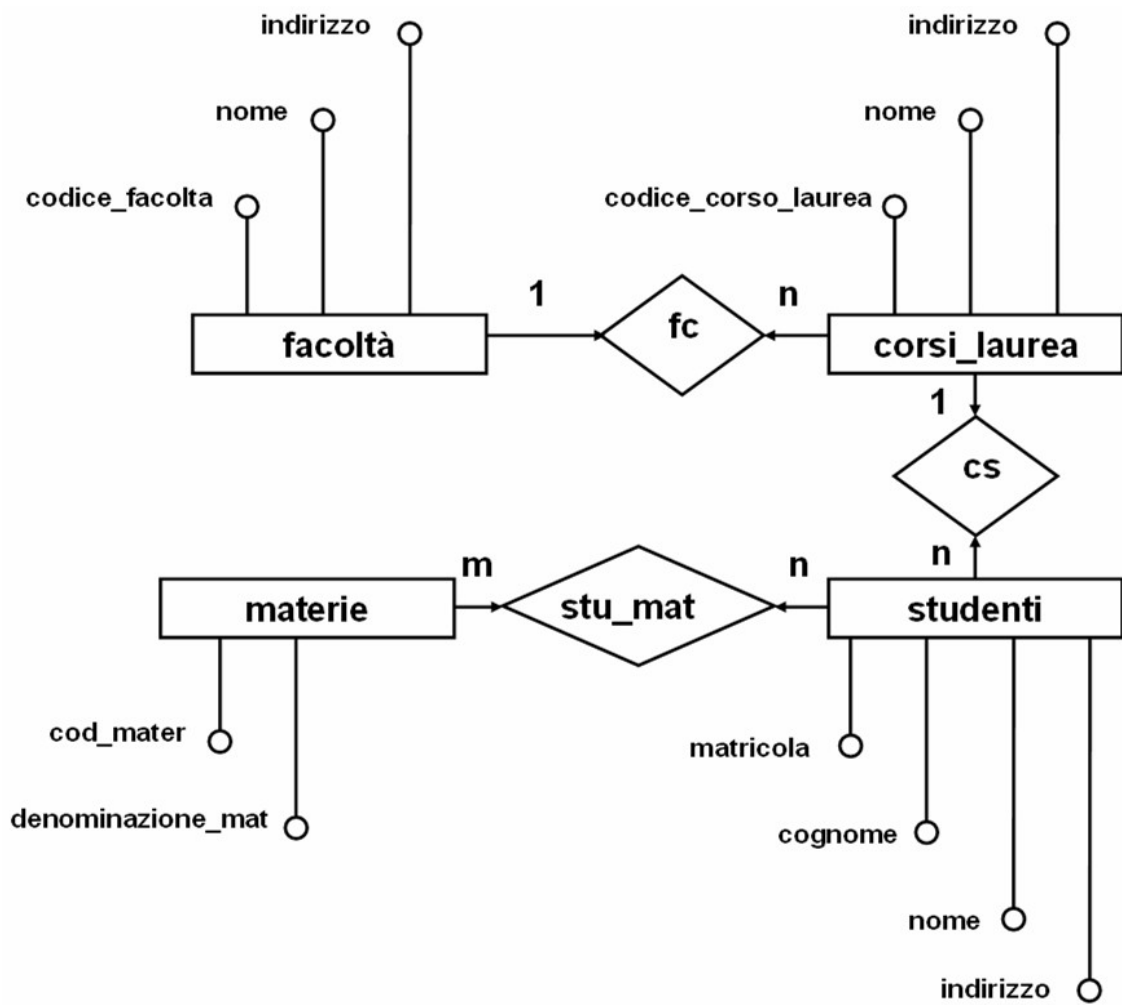
Si tenga presente che quando il database viene creato su calcolatore occorre:

- creare delle tabelle che rappresentano le relazioni,
- nella creazione delle tabelle occorre specificare l'elenco degli attributi e, per ciascun attributo,
- quale tipo dato potrà contenere (insieme di caratteri, numero intero o con la virgola, valore logico, data, ...).

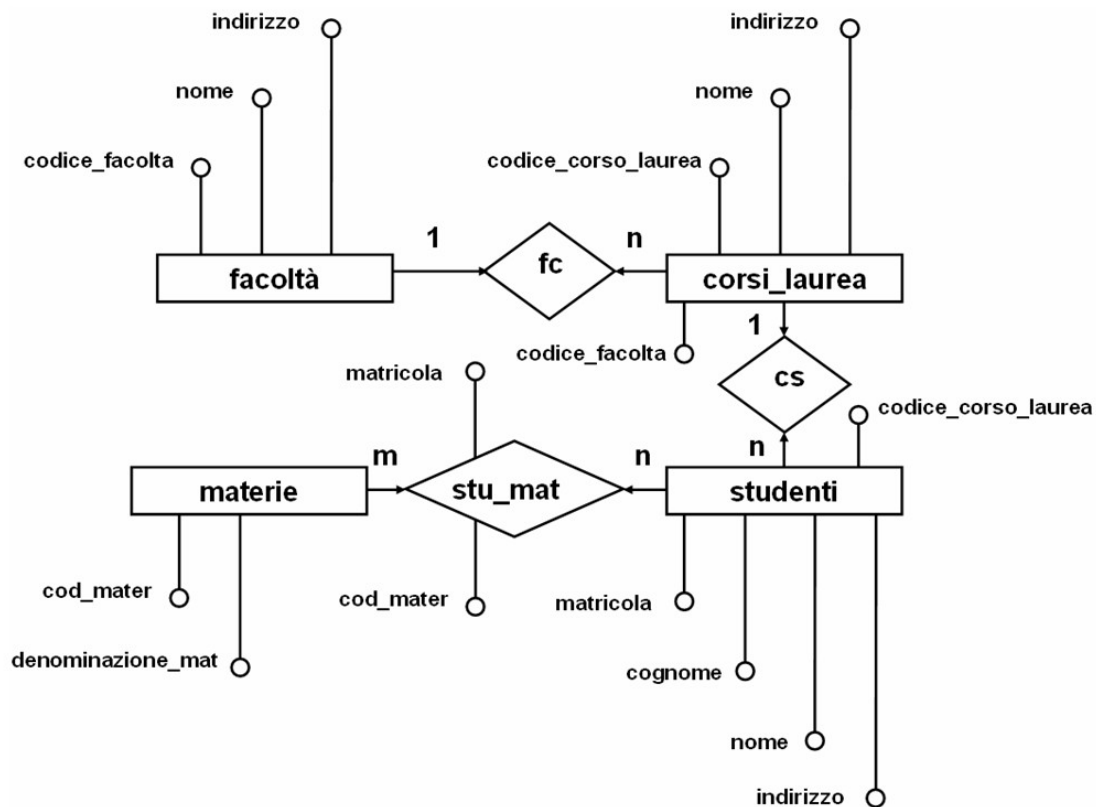
Ecco un prospetto riassuntivo sulle fasi della progettazione e sui relativi modelli.

1° fase: *progettazione concettuale*; restituisce lo schema concettuale.

¹⁰ Un DBMS è un software per la gestione di un database; Microsoft Access è un DBMS.



2° fase: *progettazione logica*; restituisce lo schema logico.



3° fase: *progettazione fisica*; restituisce modello fisico.

facoltà(codice_facoltà: testo(20 ch), nome: testo(100 ch), indirizzo: testo(255 ch))

corsi_laurea(codice_corso_laurea: testo(20 ch), nome: testo(100 ch), indirizzo: testo(255 ch), codice_facoltà: testo(20 ch))

studenti(matricola: testo(10 ch), cognome: testo(80 ch), nome: testo(60 ch), indirizzo: testo(255 ch), codice_corso_laurea: testo(20 ch))

materie(cod_mater: intero, descrizione_mat: testo(200 ch))

stu_mat(matricola: testo(10 ch), cod_mat: intero).

Esempio

Progettare una base di dati per la gestione di una biblioteca.

Le caratteristiche sono le seguenti:

- i testi sono distribuiti in vari reparti;
- in ciascun reparto ci sono degli armadi ed i testi sono depositati negli armadi;
- ciascun testo può essere scritto da più autori e ciascun autore può scrivere più testi;
- un testo è edito da una casa editrice e ciascuna casa editrice può pubblicare più testi;
- ciascun testo appartiene a più generi ed un genere è di riferimento per più testi;
- alla biblioteca sono iscritti soci;
- ciascun socio può prelevare più libri ed un libro può essere prelevato da più soci (nel tempo).

Si vuole organizzare una base di dati che organizzi le informazioni in modo che si possa rispondere alle seguenti richieste:

- dato un testo si vuole conoscere l'autore/gli autori, la casa editrice, il genere;
- dato un autore o un genere o un editore, si vuole conoscere il testo/i testi relativi
-
- quali sono i testi presi in prestito da un certo socio.

Progettazione concettuale

Analizzando la situazione si può capire che le entità che interessano sono: *reparti*, *armadi*, *libri*, *autori*, *editori*, *generi*, *soci*.

Le associazioni che possiamo individuare sono:

- fra reparti e armadi, 1:n; un reparto può avere più armadi, un armadio può stare in un solo reparto;
- fra armadi e libri, 1:n; in un armadio ci possono essere più libri, un libro può stare in un solo armadio;
- fra libri e editori, n:1; un editore può pubblicare più libri, un libro da un solo editore¹¹;
- fra libri e autori, m:n; un libro può essere scritto da più autori e un autore può scrivere più libri;
- fra libri e generi, m:n; un libro può essere riferito a più generi (di guerra, giallo, d'amore, ..) ed un genere può essere riferito a più libri;
- fra libri e soci, m:n; un socio può prelevare più libri e un libro può essere prelevato da più soci¹².

Una proposta per il diagramma E-R è visualizzata in Figura 1.

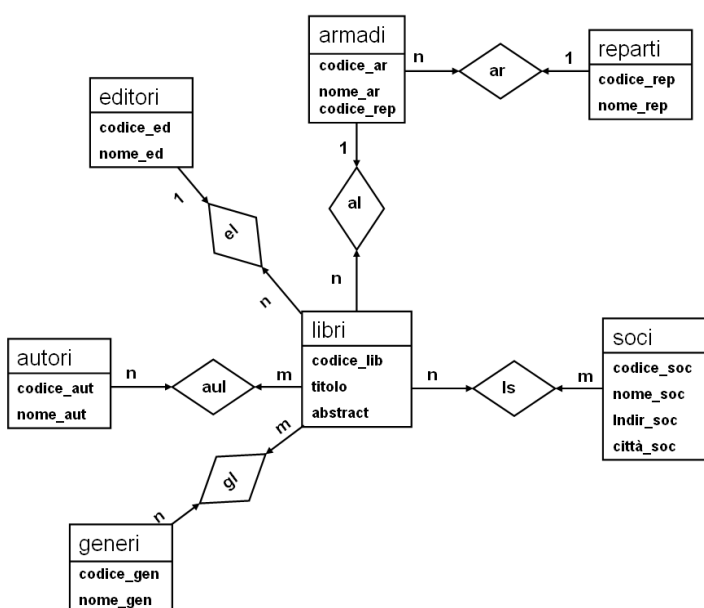


Figura 1-Schema concettuale E-R.

Ciascuna entità è stata rappresentata scrivendo in un rettangolo il nome dell'entità, in testa, e gli attributi dopo una linea di separazione. E' una notazione che ricorda i diagrammi UML (*Unified Modeling Language*), utilizzati nella modellazione di applicazioni nell'ambito della programmazione object oriented. In questo caso abbiamo utilizzato questa notazione per una maggiore semplicità nella creazione digitale dello schema.

¹¹ Non prendiamo in considerazione l'eventualità che un libro possa essere pubblicato da più autori in tempi diversi.

¹² Nel tempo.

Progettazione logica

Nella progettazione logica dobbiamo scegliere il tipo di database e, conseguentemente, esplicitare le relazioni dichiarate nel diagramma E-R.

Scegliamo il database relazionale e, in questo, caso sappiamo che:

- le entità, del modello concettuale, diventano tabelle, chiamate *relazioni* (v. in Approfondimenti 2);
- le relazioni fra entità si realizzano secondo le seguenti modalità:
 - una relazione **uno a uno** fra due relazioni, viene realizzata introducendo, come attributo, in una qualsiasi delle due relazioni la chiave dell'altra;
 - una relazione **1 ad n** viene realizzata introducendo, come attributo, nella relazione lato n, la chiave della relazione lato 1. Quindi ogni n-pla della relazione *n* avrà i suoi campi e in più un campo, chiamato *chiave esterna*, il cui valore esprime l'associazione con il record corrispondente dell'altra relazione;
 - una relazione **n ad m** viene realizzata **introducendo un'apposita tabella**, le cui righe contengono le chiavi che si corrispondono nelle due relazioni.

Lo schema relativo alla progettazione logica è rappresentato in Figura 2.

Le diverse associazioni sono state così esplicitate:

associazioni 1:n:

- fra *reparti* e *armadi* è stata realizzata aggiungendo alla relazione *armadi* la chiave esterna *codice_rep*;
- fra *armadi* e *libri* è stata realizzata aggiungendo alla relazione *libri* la chiave esterna *codice_ar*;
- fra *editori* e *libri* è stata realizzata aggiungendo alla relazione *libri* la chiave esterna *codice_ed*.

associazioni n:m:

- fra *autori* e *libri* è stata realizzata creando la nuova relazione *aul* con attributi le chiavi delle relazioni *autori* e *libri*;
- fra *generi* e *libri* è stata realizzata creando la nuova relazione *gl* con attributi le chiavi delle relazioni *generi* e *libri*;
- fra *libri* e *soci* è stata realizzata creando la nuova relazione *ls* con attributi le chiavi delle relazioni *libri* e *soci*.

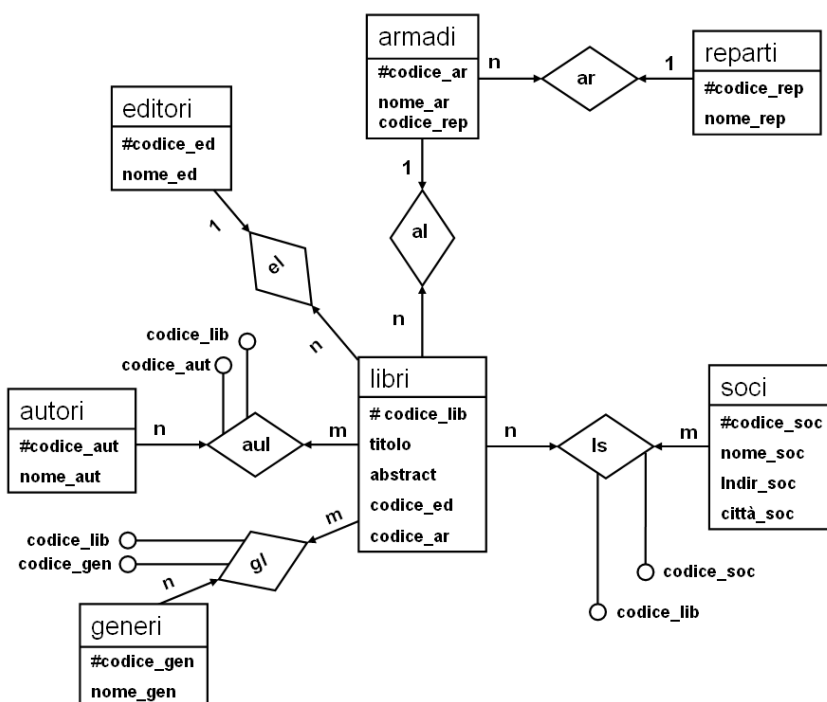


Figura 2-Schema logico

Analizziamo in particolare la relazione *ls* fra *libri* e *soci*, supponendo di avere nelle due relazioni i seguenti valori:

libri

#codice_lib	titolo	abstract	codice_ed	codice_ar
10	Il lago ..	L'autore	10	a
20	La vita .		20	a
30	Ieri ...		10	b

soci

#codice_soc	nome_soc	indir_soc	citta_soc
1	Marconi Franco		
2	Calogeri Anna		
3	Ferrari Augusto		
4	Tesei Gianna		
5	Bernini Claudio		

La relazione *ls* potrebbe contenere le seguenti informazioni

ls

codice_soc	codice_lib
1	10
3	20
5	20
4	10
3	10
1	20
2	30

La relazione *ls*, in effetti, rappresenta i libri che i vari soci hanno prelevato nel tempo; tuttavia le informazioni in essa contenute sono parziali, infatti non sono sufficienti, ad esempio, a far capire quando il socio ha prelevato il libro e se il socio ha restituito il libro. Per soddisfare queste richieste basta aggiungere alla relazione altri due attributi: uno per inserire la data di prelievo e l'altro per inserire un valore di controllo che faccia capire se il libro è stato restituito oppure no. La relazione *ls* diventa:

ls

codice_soc	codice_lib	data_prelievo	controllo
1	10	01/02/2009	si
3	20	10/02/2009	si
5	30	10/02/2009	si
4	10	20/03/2009	si
3	30	25/03/2009	si
1	20	27/03/2009	no
2	30	20/04/2009	no

La relazione *ls*, così strutturata, non permette di sapere quando il libro è stato restituito; allora la relazione *ls* potrebbe contenere anche un attributo per registrare la data di restituzione.

ls

codice_soc	codice_lib	data_prelievo	data_restituzione	controllo
1	10	01/02/2009	15/02/2009	si
3	20	10/02/2009	25/02/2009	si
5	30	10/02/2009	27/02/2009	si
4	10	20/03/2009	15/04/2009	si
3	30	25/03/2009	10/04/2009	si
1	20	27/03/2009		no
2	30	20/04/2009		no

Come si può notare, nelle ultime due righe i valori per l'attributo *data_restituzione* è nullo, in quanto i libri non sono ancora stati restituiti.

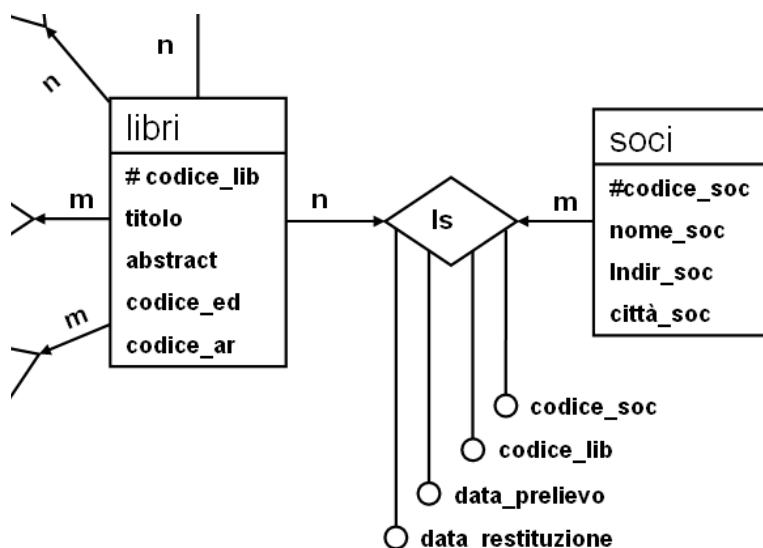
Potremmo ottimizzare la struttura della relazione *ls*, eliminando l'attributo *controllo*, in quanto la restituzione può essere verificata controllando la presenza o meno di un valore per la data di restituzione.

ls

codice_soc	codice_lib	data_prelievo	data_restituzione
1	10	01/02/2009	15/02/2009
3	20	10/02/2009	25/02/2009
5	30	10/02/2009	27/02/2009
4	10	20/03/2009	15/04/2009
3	30	25/03/2009	10/04/2009
1	20	27/03/2009	
2	30	20/04/2009	

L'associazione fra *libri* e *soci*, nel diagramma relativo alla progettazione logica è rappresentato in figura 3.

Figura 3-Associazione fra *libri* e *soci*



Progettazione fisica

Nella progettazione fisica vengono definite le strutture di memorizzazione dei dati; nel nostro caso:

libri

```
(  
    codice_lib: intero,  
    titolo: testo(... ch),  
    abstract: testo(... ch),  
    codice_ed: intero,  
    codice_ar: intero
```

```
)
```

soci

```
(  
    codice_soc: intero,  
    nome_soc: testo(... ch),  
    indir_soc: testo(... ch),  
    città_soc: testo(... ch)
```

```
)
```

autori

```
(  
    codice_aut: intero,  
    nome_aut: testo(... ch)
```

```
)
```

editori

```
(  
    codice_ed: intero,  
    nome_ed: testo(... ch)
```

```
)
```

generi

```
(  
    codice_gen: intero,  
    nome_gen: testo( ...ch)
```

```
)
```

armadi

```
(  
    codice_ar: intero,  
    nome_ar: testo( ... ch),  
    codice_rep: intero
```

```
)
```

reparti

```
(  
    codice_rep: intero,  
    nome_rep: testo(... ch)
```

```
)
```

aul

```
(  
    codice_aut: intero,  
    codice_lib: intero
```

```
)
```

gl

```

(
  codice_gen: intero,
  codice_lib: intero
)
ls
(
  codice_soc: intero,
  codice_lib: intero,
  data_prelievo: testo( ... ch),
  data_restituzione: testo( ... ch)
)

```

Notare: le relazioni (tabelle) che esprimono le associazioni n:m fra relazioni, in questo caso *aul*, *gl*, *ls*, vanno dichiarate nella progettazione fisica e vanno create quando si realizza il database sul computer utilizzando il DBMS a disposizione.

Interrogare una base di dati

I *DBMS* permettono d'immagazzinare dei dati in forma organizzata, ottenendo delle informazioni che possono facilmente essere rintracciate a fronte di richieste da parte di utenti. Un *DBMS*, per un database di tipo relazionale, è essenzialmente costituito da tre insiemi di comandi: *DDL* (*Data Definition Language*), *DML* (*Data Manipulation Language*), *DCL* (*Data Control Language*).

Il *DDL* permette la creazione, la modifica, l'eliminazione di un database e dei suoi oggetti.

Il *DML* permette l'inserimento, la modifica, l'eliminazione, la ricerca di dati all'interno delle tabelle di un database.

Il *DCL* permette di gestire gli utenti e i permessi.

Come esempio vediamo alcuni comandi del secondo insieme: *selezione*, *proiezione*, *join*.

La *selezione*, su una relazione, restituisce un insieme di righe che è la totalità o un sottoinsieme delle righe della relazione che si interroga. L'insieme di righe restituito è determinato da un criterio di selezione. Il criterio di selezione è una espressione logica che viene valutata (*vero/falso*) su ciascuna riga: le righe per cui l'espressione vale *vero* sono selezionate e fanno parte del risultato, le altre, per cui l'espressione vale *falso* sono scartate. Se si vuole che l'insieme sia l'intera relazione si omette il criterio.

Esprimiamo i comandi in linguaggio naturale ricordando tuttavia che, dovendo interagire concretamente con un *dbms* relazionale, i comandi vanno scritti utilizzando il linguaggio *SQL* (*Structured Query Language*).

Esempio 1

ricercare nella seguente relazione *studenti_con_esami*, gli studenti con media maggiore di 24.

Studenti con esami				
matricola	cognome	nome	esami_sost	media
100	Bianchi	Nicola	10	25
200	Verdi	Franca	14	21
300	Rossi	Enrica	10	28
400	Neri	Mario	18	29
500	Viola	Andrea	5	24

Il comando da impostare è il seguente:

seleziona matricola, cognome, nome, esami_sost, media
da studenti_con_esami
dove media>24

e la tabella che verrà restituita sarà:

matricola	cognome	nome	esami_sost	media
100	Bianchi	Nicola	10	25
300	Rossi	Enrica	10	28
400	Neri	Mario	18	29

Esempio 2

Visualizzare tutti gli studenti della relazione *studenti_con_esami*.

Il comando da impostare è il seguente:

seleziona matricola, cognome, nome, esami_sost, media
da studenti_con_esami

e la tabella restituita sarà l'intera relazione *studenti_con_esami*.

Nel primo caso è stato impostato il criterio di selezione ('dove media>24'); nel secondo non è stato impostato.

La struttura del comando 'seleziona' è la seguente:

seleziona <elenco campi da visualizzare>

da <relazione presa in considerazione>

dove <criterio di selezione>

Il criterio di selezione può essere omesso o può anche essere formato da più condizioni; ad es:

Esempio 3

Visualizzare tutti gli studenti della relazione *studenti_con_esami* con media > 21 e < 28.

Il comando da impostare è il seguente:

seleziona matricola, cognome, nome, esami_sost, media

da studenti_con_esami

dove media > 21

e media < 28

e è l'operatore logico AND; restituisce il valore *vero* solo quando tutte le condizioni sono vere; attraverso la *query* precedente vengono analizzate tutte le righe della relazione ed estratte solo quelle per le quali le due condizioni sono verificate; la tabella restituita sarà:

matricola	cognome	nome	esami_sost	media
100	Bianchi	Nicola	10	25
500	Viola	Andrea	5	24

Esempio 4

Visualizzare tutti gli studenti della relazione *studenti_con_esami* con media = 25 o media = 28

Il comando da impostare è il seguente:

seleziona matricola, cognome, nome, esami_sost, media

da studenti_con_esami

dove media = 25

o media = 28

o è l'operatore logico OR; restituisce il valore vero quando almeno una delle condizioni è *vera*; attraverso la *query* precedente vengono analizzate tutte le righe della relazione ed estratte quelle per le quali almeno una delle due condizioni è verificata; la tabella restituita sarà:

matricola	cognome	nome	esami_sost	media
100	Bianchi	Nicola	10	25
300	Rossi	Enrica	10	28

L'operazione di *proiezione* consente di effettuare la scelta di particolari attributi di una relazione. Mentre la *selezione* elimina, se si applica un criterio di selezione, delle righe dalla relazione indicata, la *proiezione* elimina delle colonne. Contrariamente al caso della *selezione* il criterio di eliminazione non dipende da un'espressione da valutare, è sufficiente specificare direttamente l'insieme degli attributi che vanno selezionati.

Esempio 5

visualizzare gli attributi *cognome*, *nome*, *media* dalla relazione *studenti_con_esami*

Il comando da impostare è il seguente:

seleziona cognome, nome, media
da studenti_con_esami

e la tabella che verrà restituita sarà:

cognome	nome	media
Bianchi	Nicola	25
Verdi	Franca	21
Rossi	Enrica	28
Neri	Mario	29
Viola	Andrea	24

Il *join* è una operazione fra due o più tabelle e rappresenta un sottoinsieme del prodotto cartesiano fra le due o più tabelle. Il prodotto cartesiano crea una relazione avente per righe tutte le possibili combinazioni ottenibili combinando ciascuna riga del primo operando con tutte le righe del secondo.

tabella r			tabella s	
cr1	cr2	cr3	cs1	cs2
a	a1	a2	x	x1
b	b1	b2	y	y1
c	c1	c2	a	aa1
			c	cc1

Tabella T =(R×S): prodotto cartesiano fra R e S:

cr1	cr2	cr3	cs1	cs2
a	a1	a2	x	x1
a	a1	a2	y	y1
a	a1	a2	a	aa1
a	a1	a2	c	cc1
b	b1	b2	x	x1
b	b1	b2	y	y1
b	b1	b2	a	aa1
b	b1	b2	c	cc1
c	c1	c2	x	x1
c	c1	c2	y	y1
c	c1	c2	a	aa1
c	c1	c2	c	cc1

Il prodotto cartesiano raramente è utile, perché generalmente si vogliono ottenere solo le combinazioni di righe per le quali vale una certa proprietà; ciò è possibile attraverso il *join*.

Esistono diversi tipi di *join*; analizziamo il *natural join*: si prendono in considerazione, nella tabella del prodotto cartesiano, solo le righe per le quali i valori di due attributi sono uguali. Supponendo che gli attributi che debbono verificare l'uguaglianza siano *cr1* e *cs1*, avremo:

cr1	cr2	cr3	cs1	cs2
a	a1	a2	x	x1
a	a1	a2	y	y1
a	a1	a2	a	aa1
a	a1	a2	c	cc1
b	b1	b2	x	x1
b	b1	b2	y	y1
b	b1	b2	a	aa1
b	b1	b2	c	cc1
c	c1	c2	x	x1
c	c1	c2	y	y1
c	c1	c2	a	aa1
c	c1	c2	c	cc1

e quindi la tabella risultato del *join* fra le due tabelle sarà:

cr1	cr2	cr3	cs1	cs2
a	a1	a2	a	aa1
c	c1	c2	c	cc1

Il comando da impostare è il seguente:

seleziona cr1, cr2, cr3, cs1, cs2

da R, S

dove cr1=cs1

Il comando *Join* viene utilizzato quando si ha la necessità di gestire dati che appartengono a tabelle diverse che sono in relazione fra loro. In questo caso occorre inserire, nella clausola *dove*, l'elenco delle tabelle da prendere in considerazione e nel criterio la condizione (o le condizioni se le tabelle sono più di 2) di *join*.

Esempio 6

Visualizzare l'elenco delle facoltà con i propri corsi di laurea.

Le tabelle interessate sono *facoltà* e *corsi_laurea* e supponiamo che contengano i seguenti dati:

facoltà		
codice_f	nome	indirizzo
sdf	Scienze della Formazione	
Let_fil	Lettere e Filosofia	

corsi_laurea			
codice corso laurea	nome	indirizzo	codice f
sfp	Scienze della Formazione Primaria		sdf
let	Lettere		let_fil

Il comando da impostare è il seguente:

seleziona facoltà.nome, corsi_laurea.nome, corsi_laurea.indirizzo¹³

da facoltà, corsi_laurea

dove facoltà.codice_facoltà=corsi_laurea.codice_f

¹³ Quando nomi di campi, appartenenti a relazioni diverse, sono uguali, occorre qualificarli utilizzando il nome della relazione di appartenenza, secondo la notazione [nome relazione].[nome campo].

e la tabella risultato sarà:

facolta.nome	corsi_laurea.nome	corsi_laurea.indirizzo
Scienze della Formazione	Scienze della Formazione Primaria	
Lettere e Filosofia	Lettere	

Approfondimenti 1. Vincoli fra relazioni.

In ogni diagramma E-R sono presenti dei vincoli sulle relazioni¹⁴ che esprimono quante volte una istanza di una entità può essere legata ad istanze dell'altra entità e viceversa. I vincoli vengono impostati nella progettazione concettuale e dipendono dal punto di vista che il progettista ha sulla realtà che deve modellare¹⁵.

Esistono diverse impostazioni nella dichiarazione dei vincoli; in questo contesto facciamo riferimento a quella di Albano A, Ghelli G., Orsini R., riportata in *Fondamenti di basi di dati* (2005). Gli autori affermano che una relazione è caratterizzata dai vincoli di *univocità* e *totalità*.

Vincolo di univocità

Una relazione “ $R(X, Y)$ è *univoca* rispetto ad X se per ogni elemento x di X esiste al più un elemento di Y che è associato ad x; se non vale questo vincolo, l'associazione è *multivalore* rispetto ad X.” (ibidem). Se indichiamo con $R(X, Y)$ una generica relazione fra l'entità X e l'entità Y, diciamo che:

- $R(X, Y)$ è (1:N) se essa è **multivalore** su X ed **univoca** su Y e si dice che ha cardinalità 1:N;



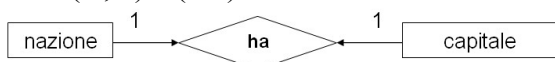
- $R(X, Y)$ è (N:1) se essa è **univoca** su X e **multivalore** su Y e si dice che ha cardinalità N:1;



- $R(X, Y)$ è (N:M) se essa è **multivalore** su X e **multivalore** su Y e si dice che ha cardinalità N:M;



- $R(X, Y)$ è (1:1): se essa è **univoca** su X e **univoca** su Y e si dice che ha cardinalità 1:1;



Vincolo di totalità

Una relazione “ $R(X, Y)$ è *totale* [...] su X se per ogni elemento x di X **esiste almeno un elemento** di Y che è associato ad x; se non vale questo vincolo, l'associazione è *parziale* rispetto ad X.” (ibidem)

“Ad esempio, $Insegna(Professori, Corsi)$ è totale su Corsi in quanto non può esistere un corso del piano di studi senza il corrispondente docente che lo tiene.” (ibidem)

Esempi

Relazioni fra Persone e Città:

“ $NataA(Persone, Città)$ ha cardinalità (N:1), totale su Persone e parziale su Città;

$HaVisitato(Persone, Città)$ ha cardinalità (N:M), parziale su Persone e Città;

$ÈSindacoDi(Persone, Città)$ ha cardinalità (1:1), parziale su Persone e Città;

$ÈVissutaA(Persone, Città)$ ha cardinalità (N:M), parziale su Persone e Città.” (ibidem)

¹⁴ In questo contesto non vengono presi in considerazione i vincoli sugli attributi.

¹⁵ Con ciò si vuole affermare che fra due entità possono esistere vincoli diversi in situazioni diverse, e ciò dipende dalla rappresentazione della realtà in quel momento scelta.

Approfondimenti 2. Tabelle e relazioni

In matematica.

A: {x, y, z} A è un insieme formato da x, y, z;

B: {10, 20, 30} B è un insieme formato da 10, 20, 30.

Il prodotto cartesiano $A \times B$ è dato dalle coppie ordinate¹⁶ che si possono ottenere dagli elementi di A e B: (x, 10), (x, 20), (x, 30), (y, 10), (y, 20), (y, 30), (z, 10), (z, 20), (z, 30).

Nelle coppie dovrà verificarsi sempre: il primo elemento appartenente al primo insieme, il secondo appartenente al secondo. Ad esempio, la coppia (10, x) non appartiene al prodotto cartesiano.

Il prodotto cartesiano può essere esteso a n insiemi e in questo caso si considerano n -uple ordinate.

A: {x, y, z} A è un insieme formato da x, y, z;

B: {10, 20, 30} B è un insieme formato da 10, 20, 30.

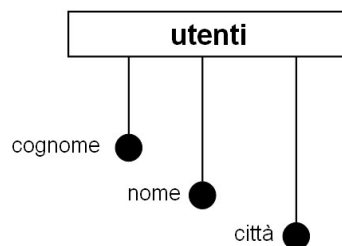
C: {palla, pelle, pollo} C è un insieme formato da palla, pelle, pollo.

Il prodotto cartesiano $A \times B \times C$ sarà: (x, 10, palla), (x, 10, pelle), (x, 10, pollo), (x, 20, palla), (x, 20, pelle), (x, 20, pollo), (x, 30, palla), (x, 30, pelle), (x, 30, pollo), (y, 10, palla), (y, 10, pelle), (y, 10, pollo), (y, 20, palla), (y, 20, pelle), (y, 20, pollo), (y, 30, palla), (y, 30, pelle), (y, 30, pollo), (z, 10, palla), (z, 10, pelle), (z, 10, pollo), (z, 20, palla), (z, 20, pelle), (z, 20, pollo), (z, 30, palla), (z, 30, pelle), (z, 30, pollo).

Nelle triple dovrà essere sempre: il primo elemento appartenente al primo insieme, il secondo appartenente al secondo, il terzo appartenente al terzo. Ad esempio, la tripla (pollo, 20, y) non appartiene al prodotto cartesiano.

In un database.

Facciamo riferimento ad una entità, ad esempio la seguente:



che, in un database relazionale, rappresenta una relazione con tre colonne e che pensiamo riempita nel seguente modo:

cognome	nome	città
Bianchi	Mario	Roma
Neri	Franca	Milano
Verdi	Enrica	Ancona
Rossi	Andrea	Napoli

¹⁶ Per coppia ordinata si intende che il primo elemento della coppia appartiene al primo insieme ed il secondo elemento appartiene alla seconda.

L'attributo *cognome* è un insieme formato da {Bianchi, Neri, Verdi, Rossi}, l'attributo *nome* è un insieme formato da {Mario, Franca, Enrica, Andrea} ed infine *città* sarà formato da {Roma, Milano, Ancona, Napoli}.

Il prodotto cartesiano *cognome* x *nome* x *città* sarà: {**Bianchi, Mario, Roma**}, {Bianchi, Mario, Milano}, {Bianchi, Mario, Ancona}, {Bianchi, Mario, Napoli}, {Bianchi, Franca, Roma}, {Bianchi, Franca, Milano}, {Bianchi, Franca, Ancona}, {Bianchi, Franca, Napoli}, ..., {Neri, Mario, Roma}, {Neri, Mario, Milano}, {Neri, Mario, Ancona}, {Neri, Mario, Napoli}, {Neri, Franca, Roma}, {**Neri, Franca, Milano**}, {Neri, Franca, Ancona}, {Neri, Franca, Napoli}, ...

Proseguendo nel prodotto cartesiano si avrà un insieme di n-ple (triple), tuttavia fra queste solo quattro saranno quelle da prendere in considerazione: {**Bianchi, Mario, Roma**}, {**Neri, Franca, Milano**}, {**Verdi, Enrica, Ancona**}, {**Rossi, Andrea, Napoli**}. Solo queste quattro n-ple rappresentano la nostra relazione.

Viene preso, cioè, un sottoinsieme del prodotto cartesiano; un sottoinsieme del prodotto cartesiano si chiama *relazione*. Quindi in un database relazionale con il termine *relazione* si intende indicare una **tabella**.

Approfondimenti 3. Normalizzazione

La normalizzazione è una tecnica di controllo usata, nei database relazionali, per verificare la correttezza delle strutture delle relazioni. E' solitamente utilizzata nel modello relazionale, ma ha validità anche in altri contesti, ad esempio nel modello E-R.

E' di particolare utilità quando si sviluppa una progettazione direttamente dal modello relazionale senza passare preventivamente per una analisi E-R (progettazione concettuale).

In genere, comunque, se nel modello E-R (concettuale) si individuano bene le classi, poi potrebbe essere non necessario operare con una normalizzazione nel modello logico relazionale.

Alla base della normalizzazione ci sono le forme normali; ciascuna di queste detta delle regole che le relazioni debbono rispettare per essere in quella forma.

Originariamente Codd¹⁷ definì la *prima*, la *seconda* e la *terza forma normale* (abbreviate in *1NF*, *2NF* e *3NF*). Boyce e Codd formularono quindi la *forma normale di Boyce e Codd (BCNF)*; una relazione in *BCNF* è anche in *3NF* ma non viceversa.

In seguito furono definite la *quarta forma normale (4NF)*, la *quinta forma normale (5NF, chiamata anche projection-join normal form (PJ/NF))* e la *Domain Key Normal Form (DKNF)*.

Le forme normali che qui presentiamo sono la *1NF*, la *2NF* e la *3NF*

Prima forma normale (1FN)

Gli attributi delle relazioni debbono essere informazioni elementari (atomiche).

Esempio 1

venditori

nome	indirizzo	
	via	città
Bianchi	Garibaldi	Milano
Neri	Cavour	ancona

La relazione va riscritta nel seguente modo:

nome	via	città
Bianchi	Garibaldi	Milano
Neri	Cavour	ancona

Esempio 2

vendite

venditore	ordini		
	auto	auto	auto
Bianchi	Fiat500	Alfa190	W1900
Neri	Audi6		NissanM

La relazione va riscritta nel seguente modo:

¹⁷ http://it.wikipedia.org/wiki/Edgar_F._Codd

nome	auto
Bianchi	Fiat500
Bianchi	Alfa190
Bianchi	W1900
Neri	Audi6
Neri	NissanM

Seconda forma normale (2FN)

Esempio 1

Rappresentiamo in una relazione che contenga l'elenco dei dottori di un laboratorio di analisi con il loro indirizzo, il numero delle analisi effettuate da ciascun dottore, i costi di ciascuna analisi

analisi

#dottore	# esame	indirizzo_dottore	città_dottore	prezzo_esame	numero_esami
Bianchi	sangue	Via milano, 3	Bologna	20	10
Neri	urine	Via peri, 100	Napoli	25	15
Rossini	rx torace	Via covour, 5	Firenze	30	5
Bianchi	urine	Via milano, 3	Genova	25	25
Rossini	sangue	Via covour, 5	Palermo	20	5

La chiave è composta dagli attributi *dottore*, *esame*.

Si hanno le possibili seguenti anomalie:

- si potrebbe aggiornare l'indirizzo di un dottore solamente in una n-pla e non su tutte (Bianchi nella prima n-pla e non nella quarta);
- un dottore che non ha ancora effettuato analisi non risulta nell'elenco e non è possibile conoscerne l'indirizzo;
- non si conosce quali sono le possibili analisi che si possono effettuare e non è possibile conoscerne il costo finché non se ne effettua almeno una di ciascun tipo;
- se si inserisce una nuova analisi, occorre inserire anche il dottore (i dottori) che può (che possono) eseguire quella analisi;
- se si inserisce un nuovo dottore, occorre inserire anche l'analisi (le analisi) che può eseguire;
- le informazioni per i dottori sono ripetute più volte e così quelle degli esami.

Queste anomalie dipendono dal fatto che esistono degli attributi che dipendono solo da una parte della chiave; in questo caso:

prezzo_esame dipende da *esame*

indirizzo_dottore dipende da *dottore*.

Le anomalie si risolvono normalizzando la relazione secondo la seconda forma normale:

una relazione è in seconda forma normale se e solo se soddisfa la prima forma normale e non esistono nella relazione attributi che dipendono solo da una parte della chiave.

La normalizzazione si realizza nel seguente modo:

dottori

# dottore	indirizzo dottore	città dottore
Bianchi	Via milano, 3	Bologna
Neri	Via peri, 100	Napoli
Rossini	Via covour, 5	Firenze

esami

# esame	prezzo esame
sangue	20
urine	25
rx torace	30

analisi

# dottore	# esame	numero esami
Bianchi	sangue	10
Neri	urine	15
Rossini	rx torace	5
Bianchi	urine	25
Rossini	sangue	5

Ossia, per normalizzare una relazione occorre eliminare le dipendenze fra attributi della relazione e sottoinsiemi della chiave creando con questi attributi nuove tabelle.

Nel nostro caso abbiamo:

- estratto la dipendenza $esame \leftarrow prezzo_esame$ ed abbiamo creato la nuova tabella *esami*;
- estratto la dipendenza $dottore \leftarrow indirizzo_dottore$ ed abbiamo creato la nuova tabella *dottori*.

Esempio 2.

La relazione

vendite(#*codice_cliente*, #*codice_prodotto*, *quantità_venduta*, *prezzo_prodotto*)¹⁸

indica l'elenco degli articoli venduti da una azienda ai propri clienti. La chiave è data da *codice_cliente* e *codice_prodotto*, in quanto il solo *codice_cliente* o il solo *codice_prodotto* non individuano in modo univoco una vendita. Infatti uno stesso prodotto può essere stato venduto a diversi clienti ed uno stesso cliente può aver acquistato più prodotti.

La relazione non è in seconda forma normale in quanto l'attributo *prezzo_prodotto* dipende solo dall'attributo *codice_prodotto*, che è un sottoinsieme della chiave.

La relazione *vendite* va normalizzata generando le due seguenti relazioni:

vendite(#*codice_cliente*, # *codice_prodotto*, *quantità_venduta*)
prodotti(#*codice_prodotto*, *prezzo_prodotto*).

¹⁸ E' possibile indicare le relazioni anche attraverso il formalismo testuale indicato.

Terza forma normale (3FN)

Nelle relazioni normalizzate in seconda forma possono esistere degli attributi che non dipendono dalla chiave o da suoi sottoinsiemi, ma dipendono da altri attributi non chiave che a loro volta dipendono dalla chiave.

La relazione

prodotti(#*codice_prodotto*, *descrizione_prodotto*, *quantità*, *codice_fornitore*, *nome_fornitore*, *indirizzo_fornitore*, *città_fornitore*)

indica l'elenco degli articoli disponibili in un magazzino di una certa azienda e la chiave è *codice_prodotto*. Gli attributi *nome_fornitore*, *indirizzo_fornitore*, *città_fornitore* dipendono da *codice_fornitore* ma non dal campo chiave, quindi questa relazione non è in terza forma normale in quanto degli attributi non chiave dipendono da altri attributi non chiave.

Una relazione è in terza forma normale se, oltre ad essere in seconda¹⁹, ciascun suo attributo, che non fa parte della chiave, non è dipendente da qualche altro attributo non chiave.

Possiamo normalizzare la relazione eliminando le dipendenze fra attributi non chiave della relazione ed altri attributi non chiave creando nuove relazioni.

Nel nostro caso:

prodotti(#*codice_prodotto*, *descrizione_prodotto*, *quantità*, *codice_fornitore*)

fornitori(#*codice_fornitore*, *nome_fornitore*, *indirizzo_fornitore*, *città_fornitore*)

¹⁹ Ed essendo in seconda forma normale è anche in prima forma normale.