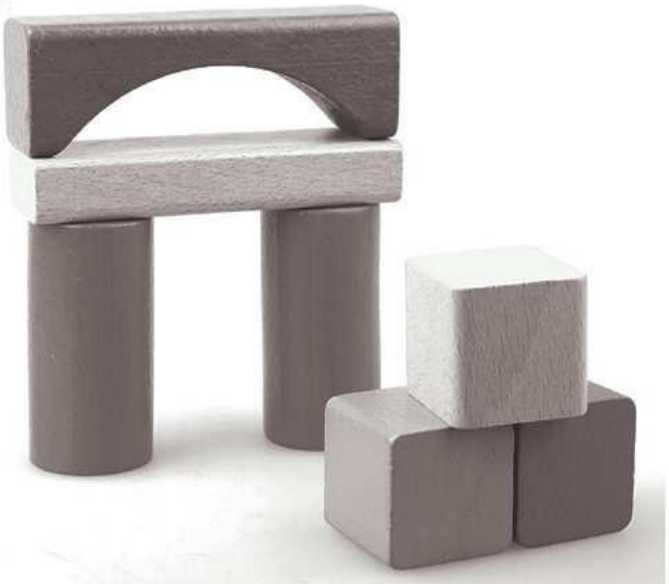
3

Modello relazionale

ESERCIZI COMMENTATI
Segui la risoluzione passo passo

LABORATORI CASE STUDIES
Approfondisci con i casi pratici



PREREQUISITI

- Conoscere le caratteristiche del modello E/R.
- Conoscere il concetto di vincolo.

PER COMINCIARE

1. **Che cosa indica il termine entità?**
2. **Che cos'è la cardinalità di una relazione?**
3. **Che cos'è una vista?**
4. **Una base di dati è:**
   - A un insieme di programmi
   - B un insieme di dati strutturati
   - C un insieme di routine per la gestione dei file
   - D un insieme di interfacce grafiche
5. **Il modello E/R descrive:**
   - A la singola vista
   - B lo schema concettuale
   - C lo schema logico
   - D lo schema fisico

CONOSCENZE

- Conoscere le caratteristiche del modello relazionale.
- Conoscere i principali tipi di operatori relazionali.
- Conoscere le operazioni dell'algebra relazionale.
- Conoscere il processo di normalizzazione e le principali forme normali.
- Conoscere i principali vincoli di integrità.

ABILITÀ

- Saper passare dal modello E/R al modello relazionale.
- Saper operare con i principali operatori relazionali.
- Saper normalizzare una relazione.
- Saper impostare dei vincoli su una relazione.

COMPETENZE

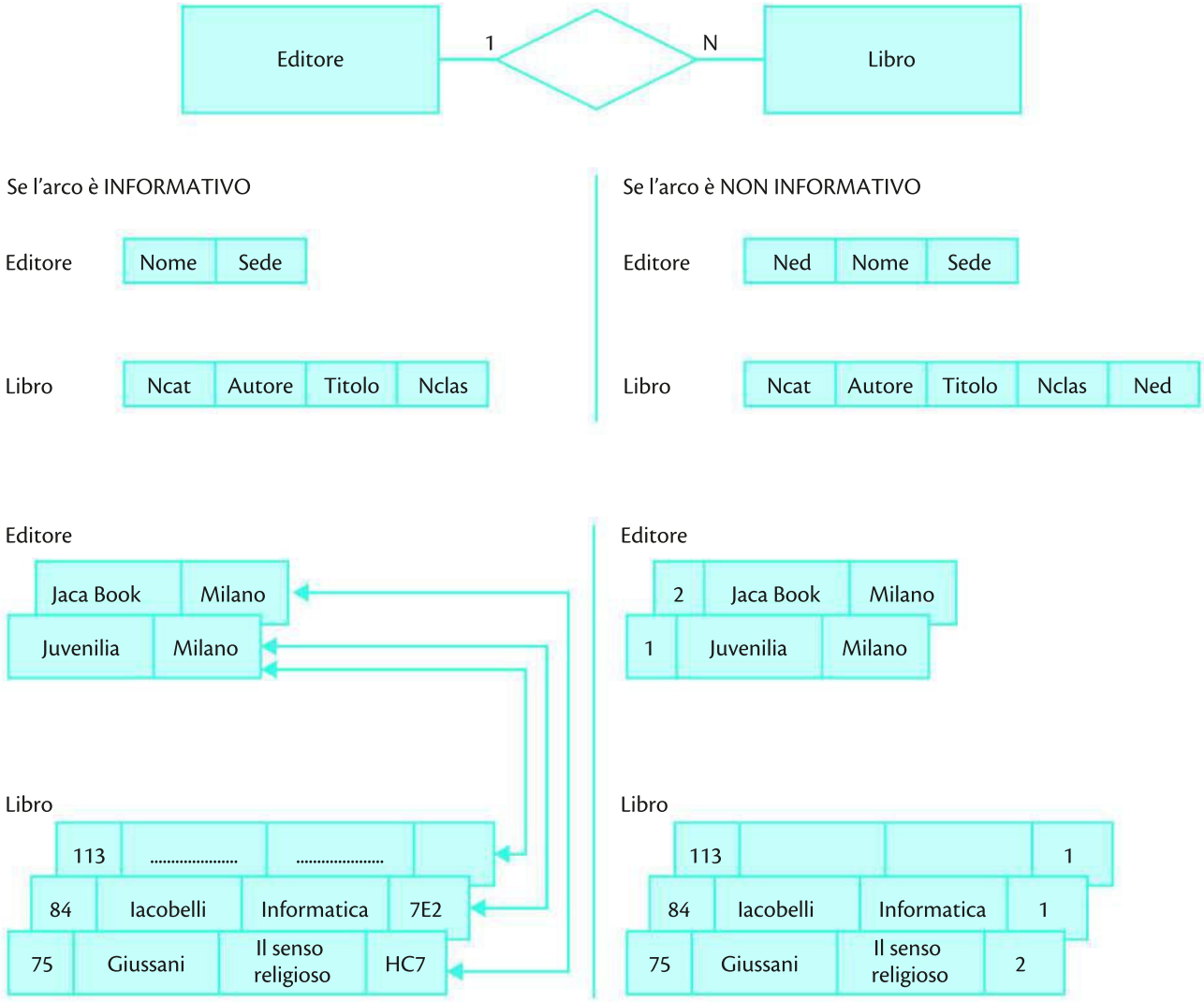
- Utilizzare le strategie del pensiero razionale negli aspetti dialettici e algoritmici per affrontare situazioni problematiche elaborando opportune soluzioni.
- Scegliere dispositivi e strumenti in base alle loro caratteristiche funzionali.
- Gestire progetti secondo le procedure e gli standard previsti dai sistemi aziendali di gestione della qualità e della sicurezza.
- Redigere relazioni tecniche e documentare le attività individuali e di gruppo relative a situazioni professionali.

1. I MODELLI LOGICI

Descriviamo ora i possibili modelli logici, cioè quei modelli che permettono una reale implementazione in uno specifico DBMS. Tutta la gestione delle basi di dati dipende da come questi dati sono rappresentati: gli stessi linguaggi dipendono in modo strettissimo dallo schema su cui devono operare.

I modelli dei dati utilizzati per descrivere lo schema possono essere classificati in base al modo in cui vengono rappresentate e trattate le associazioni, in particolare nel modo diverso in cui vengono gestiti gli archi delle associazioni nella rappresentazione grafica dello schema. L'arco si dice informativo se senza la sua presenza fisica non è possibile collegare le due entità. Si parla invece di archi non informativi se il collegamento è di tipo logico, e quindi anche senza la presenza fisica degli archi è possibile collegare le due entità (tramite l'uguaglianza di due attributi). Si può capire meglio la differenza esistente tra i due modelli analizzando l'esempio della fig. 1.

fig. 1 Esempio di arco informativo/non informativo



Nel caso di archi informativi, nel tracciato record dell'archivio Libro non è presente alcun campo che permetta di risalire ai dati dell'Editore. I collegamenti sono realizzati per mezzo dei puntatori gestiti dal DBMS indipendentemente dai campi dei record; l'assenza del collegamento fisico non ci permette quindi di conoscere l'editore di un libro.
Nel caso dell'arco non informativo, il campo Ned presente nel tracciato record dell'archivio Libro determina la casa editrice da cui il libro è pubblicato. L'arco è non informativo poiché tutte le informazioni necessarie per collegare i due archivi sono già contenute nei record stessi e quindi non è necessario utilizzare puntatori fisici.
I modelli che utilizzano archi informativi sono detti **modelli a grafo**, e possono essere a loro volta suddivisi in modello gerarchico e in modello reticolare. L'uso di archi non informativi è caratteristico invece del **modello relazionale.** Un discorso a parte va invece fatto per il **modello Object-Oriented.**

Modelli a grafo

Nei modelli a grafo le relazioni sono implementate tramite puntatori (indirizzi fisici) che indicano quale istanza di entità è connessa a quella considerata. Nella relazione è individuato sempre un proprietario (padre, owner) e un posseduto (figlio, member). Se consideriamo la relazione Editore_Libro, l'entità Editore è l'entità padre, Libro è l'entità figlio. Tramite i collegamenti realizzati con i puntatori è possibile conoscere quali sono i libri pubblicati da un determinato editore. Nella fig. 2 si vede che un puntatore, nell'occorrenza (istanza) Mondadori, collega questa con il suo primo libro (libro1). Ciascun libro, figlio della relazione, contiene il puntatore al fratello: libro1 contiene il puntatore a libro2, libro2 a libro3 ecc. Scorrendo sequenzialmente la catena dei fratelli si conoscono tutti i libri della casa editrice. Al momento di inserire un nuovo libro ci si deve prima posizionare sul padre Editore (inserendolo se non c'è ancora), inserire il libro desiderato e infine collegare questa istanza con i suoi fratelli o con il padre, se non ci sono fratelli.

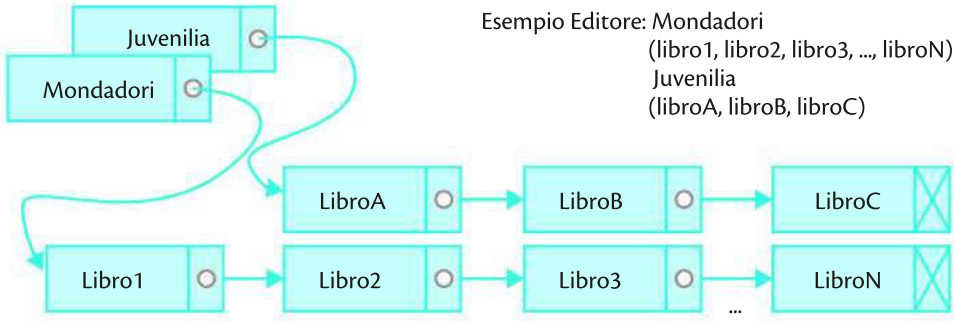


fig. 2 Modalità di memorizzazione dei dati nel modello grafico

Modello gerarchico

Lo schema è rappresentato da una struttura ad albero, cioè da un grafo privo di cicli. L'albero, detto **albero di definizione**, ha un'organizzazione tipicamente gerarchica: ciascun nodo rappresenta una classe di entità e ciascun arco una relazione che può essere di tipo uno a molti (1:N) o uno a uno (1:1). Nella fig. 3 è illustrato un esempio di schema gerarchico.

Ogni nodo ha un solo genitore, mentre un padre può avere più figli. Ogni entità figlio può essere collegata a elementi di un'altra entità figlio solo passando dall'entità superiore. Dall'esempio della fig. 3 si può vedere che è possibile passare dall'entità Uffici all'entità Laboratori solo attraverso l'entità Edificio. Dato che fra le due entità è possibile definire una sola relazione, non è necessario dare i nomi alla relazione, ma è sufficiente specificare chi è il padre e chi è il figlio.

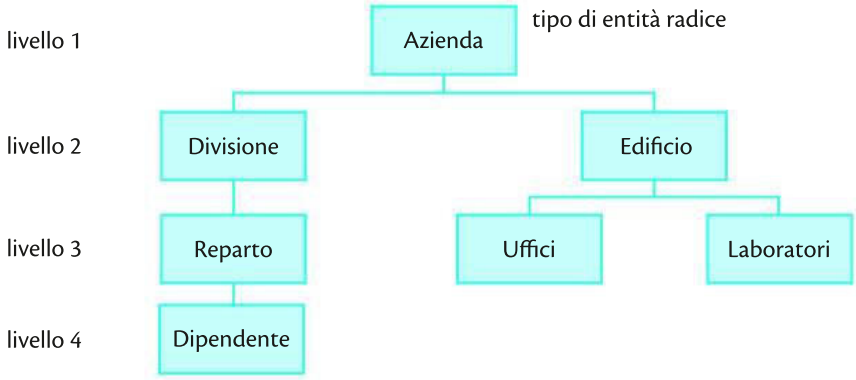


fig. 3 Modello gerarchico

LO SAI CHE

Storicamente i modelli gerarchici sono i primi tipi di DBMS immessi sul mercato, nati per implementare in modo rapido ed efficiente situazioni aventi una reale natura gerarchica. Risultano invece lenti e macchinosi per problemi di natura diversa.

Modello reticolare

Non è richiesto che lo schema sia un albero, ma ogni tipo di entità può avere nello schema un numero qualsiasi di tipi di entità connessi a esso (lo schema è un reticolo). È possibile l'esistenza di più di una relazione tra due entità ed è per questo che le relazioni devono avere un nome. Sono permesse anche relazioni di tipo N:N e relazioni unarie (archi che partono e arrivano nello stesso nodo). Nello schema sono descritti i record-type (le entità), i data-items (gli attributi) e i set (le associazioni tra entità) in cui c'è una gerarchia tra un owner e un member. È tramite il set che si effettua la navigazione, perché un record-type può appartenere a più set. Nella fig. 4 viene illustrato un esempio di schema reticolare.

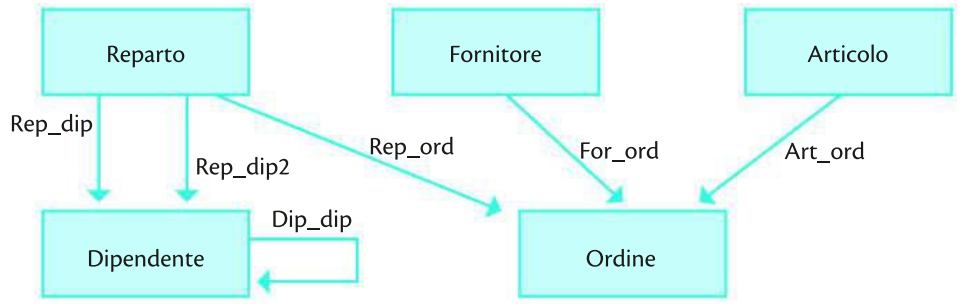


fig. 4 Modello reticolare

Modello relazionale

Nel modello relazionale i dati vengono rappresentati in formato tabellare. Le tabelle sono formate da colonne per gli attributi e da righe per i record (fig. 5).

Il legame esistente tra due tabelle è di tipo logico ed è realizzato tramite un attributo in comune, che nell'esempio della fig. 5 è rappresentato dal campo CodEd della tabella Editori e dal campo CodEd della tabella Libri. Il modello relazionale sarà trattato più approfonditamente nelle prossime lezioni.

fig. 5 Organizzazione dei dati in forma tabellare

Libri

Ncat	autore	titolo	Nclas	CodEd
75	Manzoni	I promessi sposi	7	1
90	Leopardi	I canti	89	2
123	Pellico	Le mie prigioni	H3	1

Editori

CodEd	nome	sede
1	Mondadori	Milano
2	Juvenilia	Milano

Modello Object-Oriented

Recentemente si stanno diffondendo i sistemi basati sul modello **Object-Oriented.** Tale modello utilizza oggetti indipendenti che permettono di modellizzare in modo naturale la realtà, e non un modello della realtà. Si parla di "natural modelling" (fig. 6).

fig. 6 Natural modelling

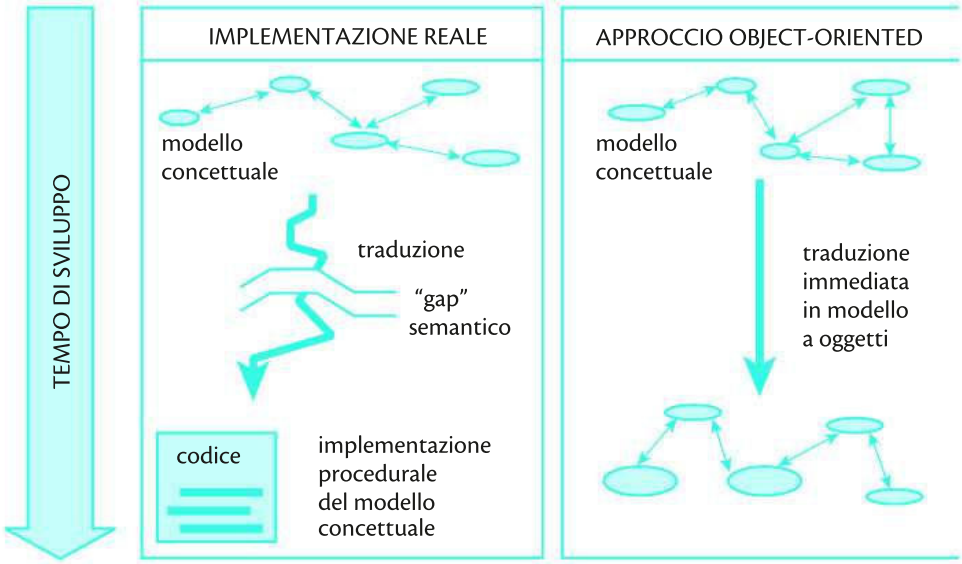


fig. 7 Oggetto

Il concetto base di questa nuova tecnologia è l'inscindibilità della struttura dei dati e delle operazioni che li manipolano: un oggetto è formato da un insieme di dati, detti proprietà, e da un insieme di operazioni, dette metodi (fig. 7). Le proprietà (o attributi) sono dati privati di un oggetto e non possono essere raggiunte esplicitamente dall'esterno se non applicando le operazioni (o metodi) associate.

Encapsulation

Il principio della **encapsulation** afferma che l'unico modo per accedere ai dati dell'oggetto è l'invocazione dei metodi: non è quindi possibile usare la struttura in modo differente da quello per cui è stata prevista. L'encapsulation può essere vista come un'estensione del concetto di modularità. Dal momento che le interazioni tra gli oggetti possono avvenire solamente tramite i metodi, risulta facilitata la costruzione incrementale di sistemi di dimensioni ragguardevoli, nonché la localizzazione di eventuali malfunzionamenti.

Ereditarietà

Un'altra caratteristica importante che differenzia i sistemi Object-Oriented da quelli tradizionali è l'ereditarietà. L'ereditarietà (*inheritance*) consente la definizione di un nuovo oggetto mediante il raffinamento: è cioè possibile creare proprietà e metodi del nuovo oggetto modificando opportunamente quelli dell'oggetto preesistente e aggiungendo ulteriori proprietà (e/o metodi) caratterizzanti. Per esempio, nell'ipotesi di possedere la definizione del tipo "persona", possiamo definire l'oggetto "impiegato" semplicemente affermando che un impiegato è una persona (ed eredita quindi tutte le proprietà e i metodi del tipo persona) e specificando solamente ciò che caratterizza gli impiegati rispetto alle persone (per esempio lo stipendio e le operazioni di assunzione, pensionamento e licenziamento). Con l'ereditarietà diventa dunque possibile riutilizzare il software prodotto senza duplicazioni e senza conoscere i dettagli di implementazione, con evidenti vantaggi in termini di produttività.

Modelli NoREL

I modelli visti fino ad ora si basano su una strutturazione rigida dei contenuti. Questo porta, prima di inserire e utilizzare i dati, a dover progettare la struttura del database, e successivamente a inserire i dati secondo le regole fornite nel database stesso. Questa è una delle caratteristiche base che hanno portato alla nascita dei database. Oggi la quantità di dati a disposizione è incrementata in modo esponenziale, e si è passati da sistemi con pochi dati mirati utilizzati da specialisti, a montagne di dati facilmente reperibili e creabili da tutti. La strutturazione dei dati diventa quindi in alcuni casi un ostacolo allo loro fruibilità, per cui sono nati database **NoREL** o **NoSQL** in cui la strutturazione è l'elemento assente, e proprio tale assenza è uno degli aspetti che maggiormente ne hanno permesso il successo. Le informazioni non trovano più posto in record e archivi, ma in oggetti completamente diversi e non necessariamente strutturati, come ad esempio documenti archiviati in collezioni.

FISSA LE CONOSCENZE

1. Come sono classificati i modelli di DBMS?
2. Come si realizza un arco informativo?
3. Come si realizza un arco non informativo?
4. Quali sono le caratteristiche del modello Object-Oriented?

2. IL MODELLO RELAZIONALE

Le tabelle

Il modello relazionale è stato proposto nel 1970 dal matematico inglese Edgar Codd e grazie alla sua coerenza e usabilità è diventato dagli anni Ottanta del secolo scorso quello più utilizzato per la produzione di DBMS.
La struttura fondamentale del modello relazionale è una tabella bidimensionale costituita da righe e colonne, detta **relazione.**
Le relazioni rappresentano le **entità** che si ritengono interessanti per il database: gli **attributi** delle entità rappresentano i campi delle relazioni, cioè le **colonne** delle tabelle; per ennuple (N-ple) o tuple si intendono le occorrenze delle relazioni, cioè le **righe**, o **record**, delle tabelle.
Il modello relazionale si basa sulla seguente definizione matematica di relazione:

> dati due insiemi D_1 e D_2, non necessariamente disgiunti, una relazione R definita su D_1 e D_2 è un qualunque sottoinsieme del prodotto cartesiano $D_1 \times D_2$.

ESEMPIO

Siano D_1 = {Giovanni, Mario, Anna} e D_2 = {10, 25, 18}.
Il prodotto cartesiano $D_1 \times D_2$ è formato dalle coppie {(Giovanni,10), (Giovanni, 25), (Giovanni, 18), (Mario, 10), (Mario, 25), (Mario, 18), (Anna, 10), (Anna, 25), (Anna, 18)}.
Una relazione è un sottoinsieme del prodotto cartesiano e pertanto, dal prodotto cartesiano appena calcolato, estraiamo solo le coppie per noi significative, quelle che rappresentano, per esempio, l'età delle persone.
Definiamo quindi la relazione Persone, formata dalle coppie {(Giovanni, 10), (Mario, 25), (Anna, 18)}, che si rappresenta in tabella come è illustrato nella fig. 8.

PERSONE

Giovanni	10
Mario	25
Anna	18

fig. 8 Rappresentazione tabellare della relazione Persone

Potrebbe nascere confusione tra relazione intesa come "insieme di dati nel modello relazionale" e relazione intesa come "associazione tra entità". **In questa Unità utilizzeremo il termine tabella invece del termine relazione, ricordando che sono sinonimi**, così come lo sono i termini "campo" e "attributo", e i termini "record", "occorrenza" ed "ennupla".

Nella fig. 9 sono mostrati altri esempi di tabelle relazionali.

Amici

cognome	indirizzo	città	n. telefono
Rossi	Via Roma 4	Torino	011 3572943
Bianchi	Via Torino 12	Milano	02 4372049
Verdi	Via Genova 12	Torino	011 6993524

Località

città	regione
Torino	Piemonte
Milano	Lombardia
Roma	Lazio

fig. 9 Tabelle relazionali

L'identificazione dei record

Per identificare i record all'interno della tabella si fa uso di **campi chiave** o di identificatori.

Un **identificatore** è un insieme di uno o più attributi che identifica univocamente una data occorrenza.

Al posto di identificatore si parla a volte di **chiave candidata**: in una tabella, possono esistere più chiavi candidate (per esempio, nella tabella Persona possono esistere due identificatori, il Codice fiscale e l'insieme di Nome, Cognome e Data_Nascita).
Si osservi che almeno una chiave candidata esiste sempre, poiché al limite può essere costituita dall'insieme di tutti gli attributi (non possono infatti esistere record uguali, cioè le righe della tabella devono essere tutte diverse).
Tra tutte le chiavi candidate viene scelta una **chiave primaria** per la memorizzazione e per effettuare i collegamenti con le altre relazioni. Normalmente ne viene scelta una tra quelle che hanno il minor numero di attributi.
Gli attributi della chiave primaria hanno la caratteristica di non potere mai assumere il valore **null** (che significa un valore non determinato), perché in tal caso non permetterebbero più di identificare una particolare tupla in una relazione.

Una **chiave esterna** è un attributo (o un insieme di attributi) che non è chiave nella tabella, ma lo è in un'altra e serve per realizzare il collegamento logico tra le tabelle.

Il dominio dei campi

Due tabelle possono essere *messe in relazione* solo attraverso l'uguaglianza dei valori di un certo campo. Due valori sono *confrontabili* solo se appartengono allo stesso dominio.

> Per **dominio** si intende l'insieme dei valori su cui è definito un certo attributo, cioè l'insieme di tutti i valori elementari che un attributo può assumere (per esempio, il campo Età potrà essere definito sul dominio degli interi che va da 0 a 150).

Il concetto di dominio è fondamentale: gli attributi di diverse relazioni definite sullo stesso dominio permettono di realizzare i collegamenti tra le diverse tabelle.
In particolare questo è possibile tramite l'uguaglianza tra chiave primaria di una tabella e chiave esterna di un'altra. Nell'esempio della fig. 10 si vede come la tabella Località sia collegata alla tabella Amici, poiché in entrambe esiste l'attributo Città, il quale è chiave primaria nella tabella Località e chiave esterna nella tabella Amici.

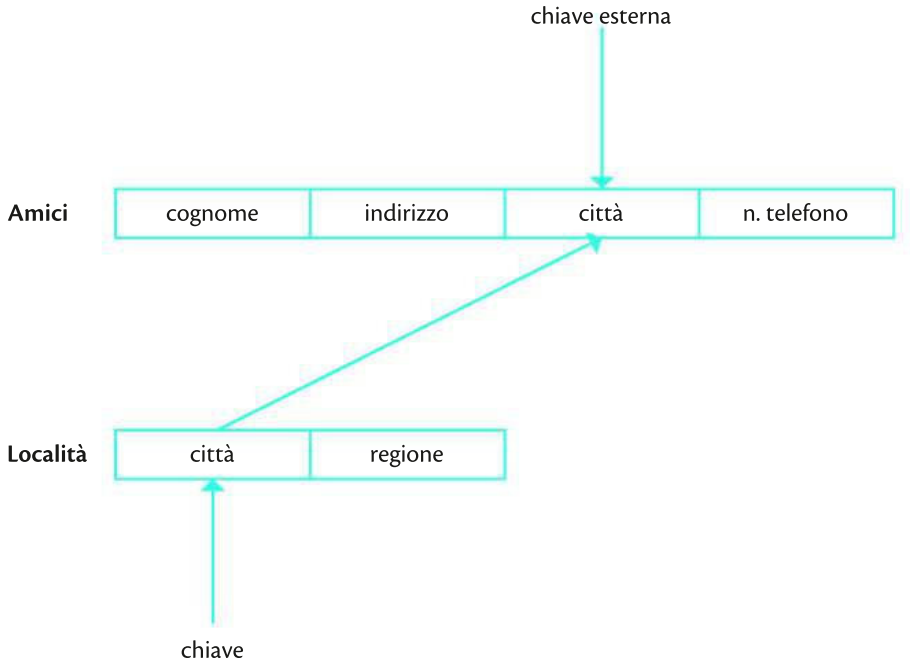


fig. 10 Chiave primaria e chiave esterna

Metriche

Altre caratteristiche risultano utili per definire le **metriche**, ossia le *misure quantitative*, sulle relazioni presenti in un database.

> Per **grado** di una relazione si intende il numero di campi delle tabelle, mentre per **cardinalità** si intende il numero delle occorrenze presenti in un dato istante.

Nello schema della fig. 11 sono riassunti e illustrati i concetti appena espressi sull'esempio della tabella Amici.

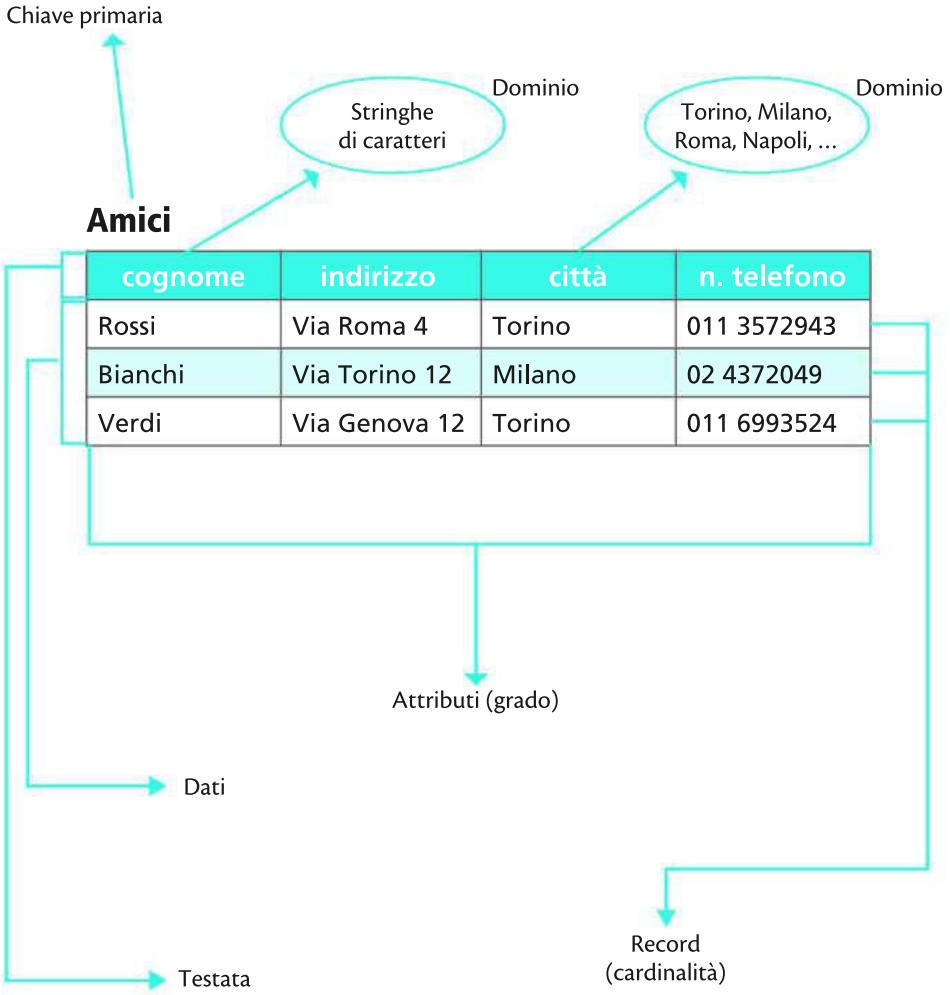


fig. 11 Caratteristiche delle tabelle

Altre caratteristiche

Vi è una serie di caratteristiche che permettono di definire un database relazionale. Queste sono riassunte nelle **12 regole di Codd** che riportiamo nell'approfondimento. Si parla di dodici regole, ma in realtà sono tredici, in quanto si parte dalla regola 0.
Le regole sono molto restrittive e attualmente nessun sistema le rispetta veramente tutte.
Evidenziamo, a partire dalle 12 regole di Codd, alcuni punti che riassumono le caratteristiche che devono avere le relazioni nel modello relazionale.

- Non è definito alcun ordinamento fra le N-ple, quindi due tabelle con le stesse righe, anche in ordine diverso, rappresentano la stessa relazione (fig. 12).

Auto

targa	colore	modello
AB067BX	rosso	Fiat 600
CD832AT	verde	Ford Fiesta
LD325CZ	bianco	Fiat Panda

fig. 12 Esempio di tabelle coincidenti

Automobile

targa	colore	modello
CD832AT	verde	Ford Fiesta
AB067BX	rosso	Fiat 600
LD325CZ	bianco	Fiat Panda

- Le N-ple di una relazione devono essere distinte l'una dall'altra; quindi una tabella rappresenta una relazione solo se le sue righe sono tutte diverse fra loro (fig. 13).

Studente

cognome	età	città
Rossi	18	Torino
Bianchi	17	Roma
Verdi	18	Torino
Rossi	17	Roma
Rossi	18	Torino

N_ple uguali

fig. 13 Esempio di N-uple non distinte

- È possibile che alcune informazioni non siano disponibili per tutte le N-ple della relazione: possono essere ignote o non definite. Questi attributi assumono un valore speciale, denominato valore nullo (**null**); tale valore non fa parte di alcun dominio e rappresenta sia un valore ignoto, sia un valore non definito.
  Inoltre, si fa uso del valore null per un attributo quando questo non è significativo per gli altri attributi oppure è un valore che verrà inserito in un secondo momento.
  È possibile che alcune colonne di una tabella non possano assumere valore null. Per esempio, nella relazione **Studente (Matricola, Cognome, Data_Nascita, Telefono, Anno_Laurea)** il telefono può essere ignoto e per uno studente non ancora laureato l'anno di laurea non è ancora definito, ma la matricola non potrà mai essere ignota (fig. 14).

Studente

matricola	cognome	data_nascita	telefono	anno_laurea
74325	Rossi	22/06/1994	011 3786354	null
45861	Bianchi	10/08/1988	null	2013
69437	Verdi	09/03/1990	null	null

fig. 14 Campi con valore ignoto o non definito

- In una relazione esiste sempre una chiave candidata, poiché al limite può essere costituita dall'insieme di tutti gli attributi.
- Tra tutte le chiavi candidate (gli identificatori) viene scelta una chiave primaria per la memorizzazione e per effettuare i collegamenti con

le altre relazioni.
Normalmente ne viene scelta una tra quelle con il minor numero di attributi.

- Gli attributi della chiave primaria non possono mai assumere il valore null (che significa un valore non determinato), perché in tal caso non permetterebbero più di identificare una particolare N-pla in una relazione.
- Una chiave esterna è un attributo (o un insieme di attributi) che non è chiave nella tabella, ma lo è in un'altra e serve per realizzare il collegamento logico tra le entità.
- Due tabelle possono essere messe in relazione solo attraverso l'uguaglianza dei valori di un certo campo.
- Due valori sono confrontabili solo se appartengono allo stesso dominio.

APPROFONDIMENTO

Le 12 regole di CODD

- **Regola 0**: il sistema deve potersi definire come *relazionale*, *base di dati* e *sistema di gestione*.
  Affinché un sistema possa definirsi sistema relazionale per la gestione di basi di dati (RDBMS), tale sistema deve usare le proprie funzionalità *relazionali* (e solo quelle) per gestire la base di dati.

- **Regola 1**: l'informazione deve essere rappresentata sotto forma di tabelle.
  Le informazioni nel database devono essere rappresentate in maniera *univoca*, e precisamente attraverso valori in colonne che costituiscano, nel loro insieme, righe di tabelle.

- **Regola 2**: la regola dell'accesso *garantito* o delle *chiavi primarie*.
  Tutti i dati devono essere accessibili senza ambiguità. Ogni singolo valore scalare nel database dev'essere logicamente indirizzabile specificando il nome della tabella che lo contiene, il nome della colonna in cui si trova e il valore della chiave primaria della riga in cui si trova.

- **Regola 3**: trattamento sistematico del valore NULL.
  Il DBMS deve consentire all'utente di lasciare un campo vuoto, o con valore NULL. In particolare, deve gestire la rappresentazione di *informazioni mancanti* e quella di *informazioni inadatte* in maniera predeterminata, distinta da ogni valore consentito (per esempio, "diverso da zero o qualunque altro numero" per valori numerici), e indipendente dal tipo di dato. Queste rappresentazioni devono inoltre essere gestite dal DBMS sempre nello stesso modo.

- **Regola 4**: dizionario del modello relazionale.
  La descrizione della struttura del database e degli oggetti che lo compongono deve avvenire ad un *livello logico*, tramite un dizionario di metadati, e questo dizionario deve essere accessibile agli utenti del database con le stesse modalità e lo stesso linguaggio di interrogazione utilizzato per accedere ai dati.

- **Regola 5**: accessibilità dei dati.
  Tutti i contenuti del database devono essere accessibili attraverso almeno un linguaggio relazionale (come ad esempio l'SQL) che abbia le seguenti caratteristiche:
  - abbia una *sintassi lineare* (ovvero le cui istruzioni possono essere semanticamente interpretate con una semplice lettura da sinistra verso destra)

- possa essere utilizzato sia in forma interattiva che dall'interno di applicazioni
- supporti operazioni di definizione e di manipolazione dei dati, le regole di sicurezza e i vincoli di integrità del database.

- **Regola 6**: aggiornamento delle viste di dati.
  In un database relazionale si possono creare viste che forniscono l'accesso a specifici subset di informazioni. Laddove il contenuto in termini di dati di queste viste è concettualmente modificabile, lo deve anche essere nella pratica.

- **Regola 7**: manipolazione dei dati ad alto livello.
  Da un database possiamo reperire informazioni multiple costituite da set di dati provenienti da più righe e/o più tabelle. Gli stessi set di dati, piuttosto che le singole informazioni, devono anche poter essere inseriti, aggiornati e cancellati.

- **Regola 8**: indipendenza dalla rappresentazione fisica.
  La struttura logica di un database deve essere indipendente dalle strutture di memorizzazione fisica: modifiche al piano fisico (come i dati vengono memorizzati, su quali unità, con quale organizzazione, ecc.) non devono richiedere un cambiamento alle modalità di accesso al database.

- **Regola 9**: indipendenza dalla rappresentazione logica.
  Le modifiche al livello logico (tabelle, colonne, righe, chiavi primarie, ...) non devono richiedere cambiamenti non giustificati alle applicazioni che utilizzano il database.

- **Regola 10**: i vincoli logici sui dati devono essere memorizzati nel database.
  I vincoli di integrità propri delle entità e delle relazioni, le regole di sicurezza e le restrizioni di accesso devono essere definiti nel dizionario del database e sono quindi separati dalle applicazioni che lo utilizzano. Deve quindi essere possibile modificare tali vincoli senza inutilmente interessare le applicazioni esistenti.

- **Regola 11**: indipendenza di localizzazione.
  La distribuzione di porzioni del database su una o più allocazioni fisiche o geografiche deve essere invisibile agli utenti del sistema. Le applicazioni esistenti devono continuare ad operare con successo quando i dati esistenti vengono ridistribuiti in modo diverso.

- **Regola 12**: regola di non sovversione.
  Gli strumenti di accesso ai dati non devono poter annullare le restrizioni del database, per esempio aggirando i vincoli di integrità, le relazioni o le regole di sicurezza.

FISSA LE CONOSCENZE

1. Che cosa è una tabella?
2. Da dove deriva il termine modello relazionale?
3. Come si definisce una chiave primaria?
4. Che cosa indica il dominio di un attributo?
5. Che cosa indica il grado di una tabella? E la cardinalità?

3. RISTRUTTURAZIONE DELLO SCHEMA E/R

Ristrutturare lo schema

Prima di procedere alla traduzione dello schema E/R in uno schema logico relazionale è necessario effettuare una sua ristrutturazione con lo scopo di semplificarne la traduzione e ottimizzare le prestazioni. Uno schema E/R ristrutturato risulta "meno concettuale", perché tiene già conto del modello logico adottato (e quindi dei dettagli implementativi): questo ci permetterà di ottimizzare le prestazioni. Nell'attività di ristrutturazione è necessario quindi semplificare lo schema, eliminando le gerarchie e gli attributi multipli, ed effettuare un partizionamento o accorpamento di entità e relazioni.

Eliminazione delle gerarchie

Nello schema relazionale le entità e le associazioni sono direttamente rappresentabili, non è così per le gerarchie. È quindi necessario eliminarle, sostituendole con entità e associazioni. Consideriamo la gerarchia presente nella fig. 15; per eliminarla senza perdere informazioni possiamo adottare metodi diversi.

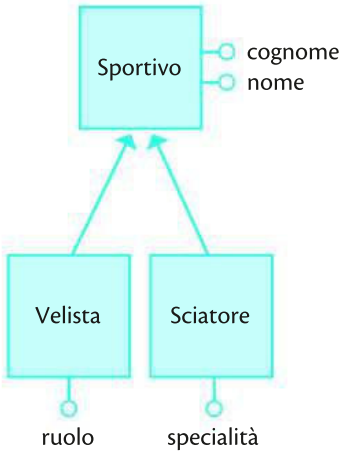
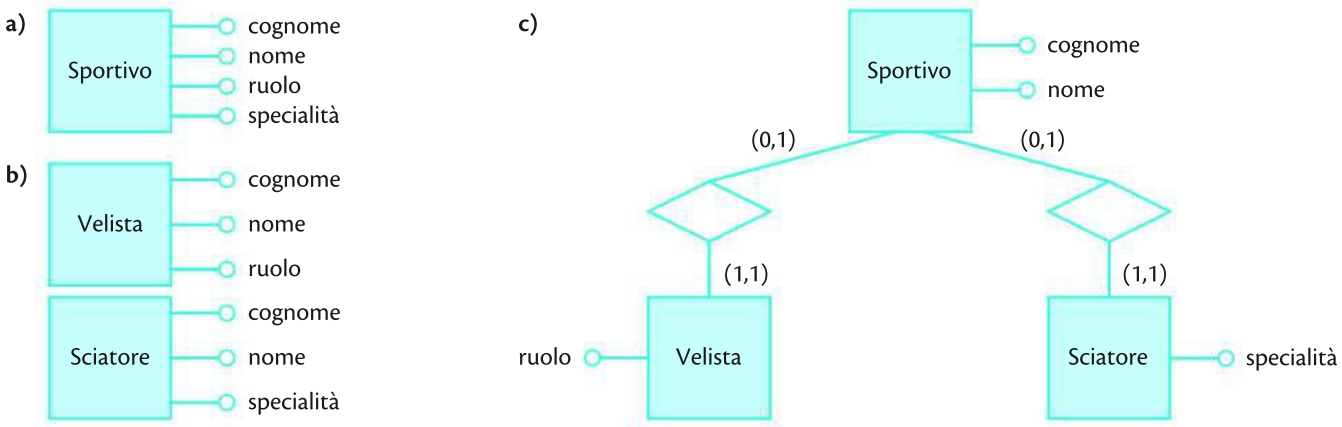


fig. 15 Gerarchia

1. Accorpamento delle figlie nel padre: gli attributi delle entità figlie vengono fatte migrare nell'entità padre e diventano attributi opzionali. Nell'esempio di fig. 16a, ruolo è un attributo opzionale che avrà un valore nel caso di un velista, mentre risulterà null nel caso di uno sciatore. Per specialità sarà l'opposto.
2. Accorpamento del padre nelle figlie: nelle entità figlie vengono riportati gli attributi presenti nell'entità padre introducendo *ridondanza* di dati. Nel caso della fig. 16b vengono create le entità Velista e Sciatore, che avranno entrambe, oltre ai propri attributi specifici, anche quelli generali (cognome e nome).
3. Sostituzione della gerarchia con associazioni (fig. 16c). In questo caso le entità "Velista" e "Sciatore" risultano **Entità Deboli**, sono collegate ad una entità "Sportivo" (Forte) da associazioni e vengono da questa identificate esternamente.

Sono anche possibili soluzioni "ibride" in cui vengono create solo alcune nuove entità.

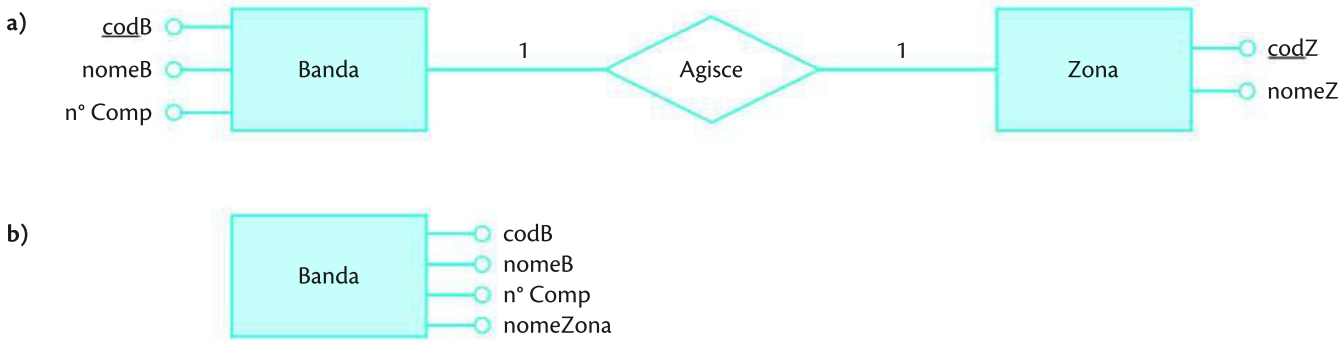
fig. 16 Eliminazioni gerarchie



Partizionamento o accorpamento di entità e relazioni

In alcuni casi, prendendo in considerazione le richieste fatte alla base di dati dalle applicazioni, è conveniente accorpare delle entità che contengono informazioni che vengono richieste. Consideriamo l'esempio della fig. 17a: l'entità Zona ha solo gli attributi nome e codice zona (codZ) ed è collegata con un'associazione 1:1 con Banda; possiamo quindi accorparla nell'entità Banda, aggiungendo l'attributo nomeZona (fig. 17b).

fig. 17 Accorpamento di entità



Partizionamento verticale

Supponiamo di avere la relazione impiegato, come è mostrata nella fig. 18; se le nostre applicazioni accedono separatamente ai dati anagrafici o a quelli lavorativi di un impiegato, conviene separarli, perché se i record sono più piccoli in una sola lettura dal disco (operazione costosa) riesco a portare più record in memoria (fig. 19).

fig. 18 Entità impiegato

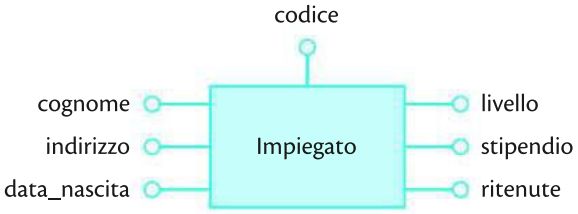
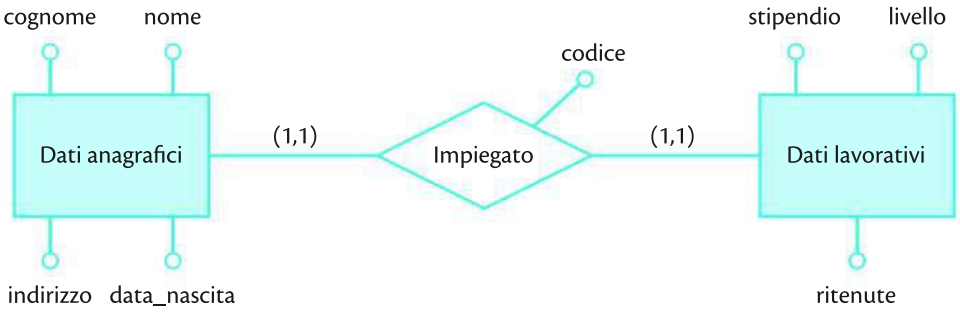


fig. 19 Partizionamento verticale



Partizionamento orizzontale

Supponiamo di avere lo schema rappresentato nella fig. 20 e che la nostra applicazione acceda separatamente alla composizione attuale della squadra rispetto a quelle degli anni passati. Separando i due tipi di collegamento, lo schema risulta più chiaro e le operazioni più efficienti (fig. 21).

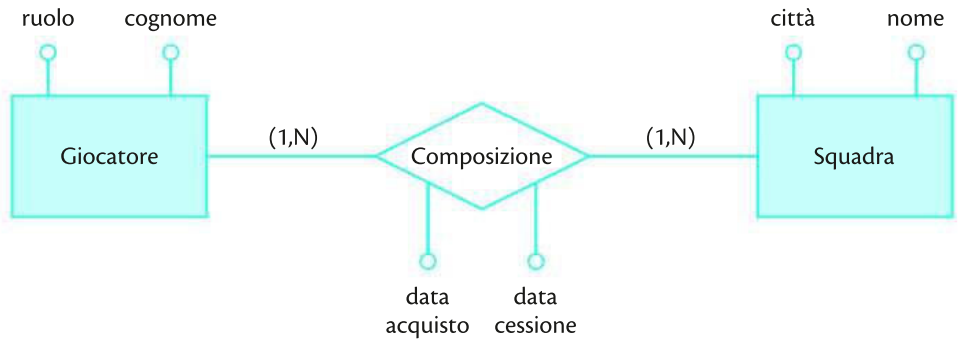


fig. 20 Schema giocatore-squadra

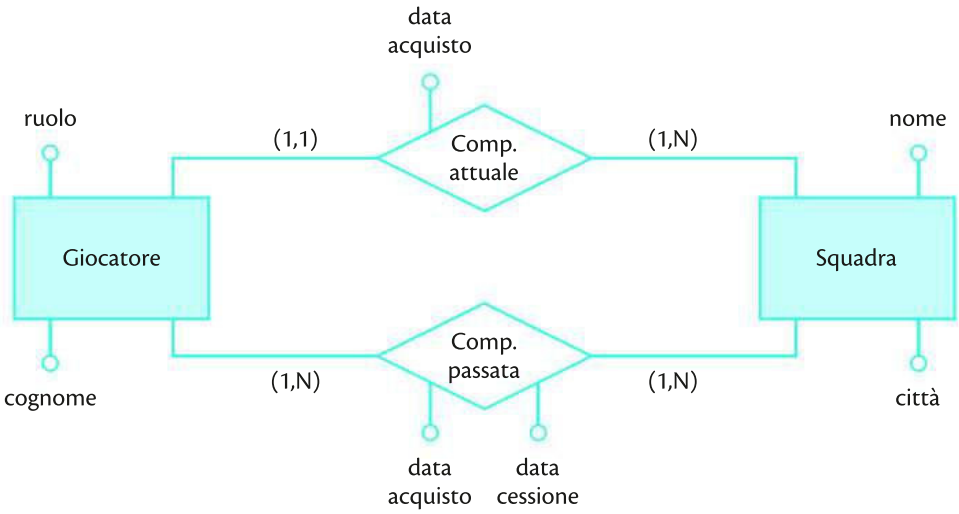


fig. 21 Partizionamento orizzontale

Eliminazione degli attributi multipli

Per eliminare gli attributi multipli che non sono rappresentabili nel modello relazionale, si crea una nuova entità con il nome dell'attributo e si inserisce una nuova relazione. Nella fig. 22a è mostrata l'entità libro con l'attributo multiplo autore; nella fig. 22b l'attributo multiplo è diventato un'entità collegata a libro da un'associazione 1:N.

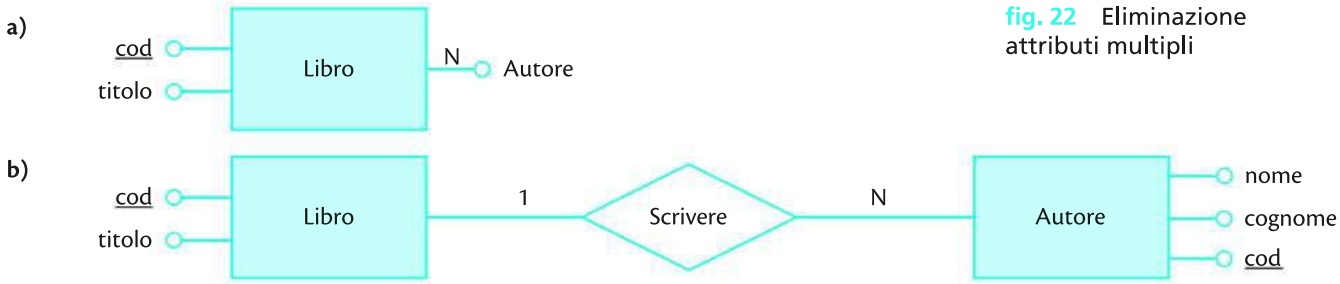


fig. 22 Eliminazione attributi multipli

FISSA LE CONOSCENZE

1. Perché è necessario ristrutturare lo schema E/R prima della sua traduzione nello schema relazionale?
2. Come vengono ristrutturate le gerarchie?
3. Come si eliminano gli attributi multipli?

4. TRADUZIONE NEL MODELLO LOGICO

La rappresentazione delle entità

Ogni entità dello schema E/R viene rappresentata nel modello relazionale con una tabella: ogni istanza di entità rappresenta un record della tabella e ogni attributo diventa un campo del record. La chiave dell'entità diventa la chiave d'accesso per la tabella. All'entità Prodotto, rappresentata nella fig. 23, corrisponde la tabella relazionale nella fig. 24.

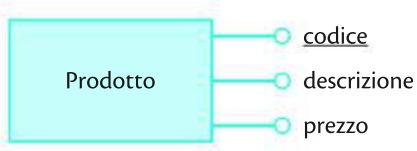


fig. 23 Entità Prodotto

Prodotto

codice	descrizione	prezzo

fig. 24 Tabella relazionale

Per descrivere l'entità in forma più concisa, si può scriverne il nome seguito dal nome dei suoi attributi racchiusi tra parentesi; l'attributo chiave è sottolineato. L'entità Prodotto viene in questo caso rappresentata così:

Prodotto (Codice, Descrizione, Prezzo)

La rappresentazione dettagliata dell'entità spesso avviene mediante una tabella descrittiva, nella quale sono evidenziati gli attributi e le loro caratteristiche (nome, descrizione, tipo, dimensione), come è illustrato nella fig. 25.

Prodotto

nome campo	descrizione	tipo	lunghezza	chiave
Codice	codice del prodotto	stringa	3 byte	primaria
Descrizione	descrizione del prodotto	stringa	30 byte	
Prezzo	prezzo del prodotto	numerico	6 byte	

fig. 25 Descrizione della tabella Prodotto

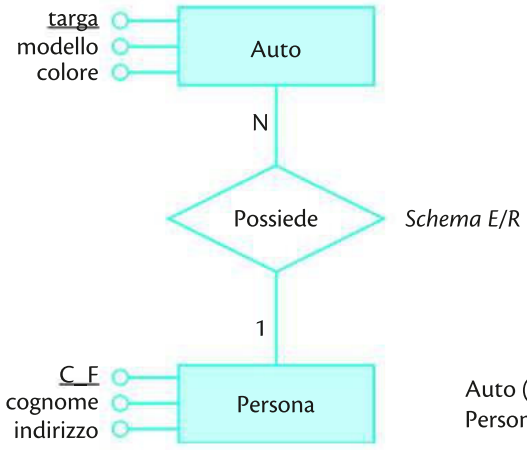
La rappresentazione delle associazioni

La traduzione dallo schema concettuale a quello logico delle associazioni (si dice anche *risolvere l'associazione*) risulta essere più complessa rispetto a quella delle entità. Per rappresentare le associazioni si possono effettuare scelte diverse, in base al tipo di associazione e alla destinazione d'uso.

Uno a molti

Nelle associazioni uno a molti la chiave primaria della tabella di partenza dell'associazione (quella con caratteristica 1) diventa chiave esterna (*foreign key*) dell'entità di arrivo associata, cioè l'attributo che è identificatore univoco diventa un campo nella tabella dell'entità di arrivo. Nell'esempio della fig. 26, poiché una persona può possedere più auto, ma un'auto può essere posseduta da una sola persona, si farà migrare la chiave di Persona (in questo caso il codice fiscale) nella tabella Auto.

Questa diventerà chiave esterna, permettendo così il collegamento logico tra Auto e Persona.



Schema E/R

Auto (targa, modello, colore, C_F)
Persona (C_F, cognome, indirizzo)

fig. 26 Soluzione delle associazioni 1:N

Dal modello concettuale vengono derivate le tabelle che rappresentano le entità e l'associazione uno a molti viene tradotta aggiungendo agli attributi dell'entità "a molti" la chiave dell'entità "a uno".

Uno a uno

Le associazioni uno a uno si risolvono facendo migrare la chiave di una delle due tabelle create nell'altra, in base alle modalità con cui si pensa verranno utilizzate.

ESEMPIO

Le entità *Persona* e *Documento identità* sono collegate tra loro (fig. 27). La loro traduzione porterà a creare due tabelle. Nella traduzione dell'associazione, la scelta potrà essere (fig. 28):

a. di far migrare il numero di documento nella tabella *Persona*, se le richieste di informazioni sul numero di documento sono subordinate a quelle sulla persona (cioè prima si accede alla tabella *Persona* e poi alla tabella *Documento identità*, partendo dal suo numero di documento);
b. di far migrare il codice fiscale nella tabella *Documento identità* se, viceversa, l'accesso ai dati di una persona avviene sempre dopo l'accesso ai dati del documento;
c. di far migrare la chiave di ciascuna tabella nell'altra, se invece non vi sono prevalenze nell'accesso.

fig. 27 Relazione

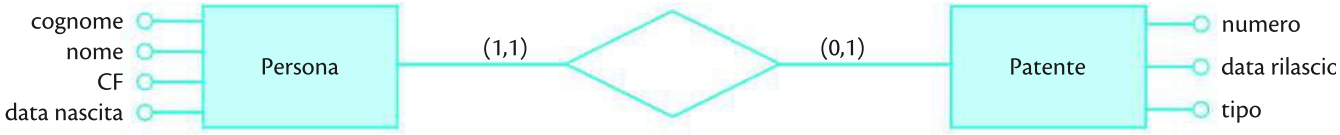
A) Documento identità (n., data rilascio, data scadenza)
Persona (CF, cognome, nome, n. documento)

B) Persona (CF, cognome, nome)
Documento identità (n., data rilascio, data scadenza, CF_Persona)

C) Persona (CF, cognome, nome, n. documento)
Documento identità (n., data rilascio, data scadenza, CF_Persona)

fig. 28 Possibili traduzioni del modello E/R

Nel caso in cui l'associazione 1:1 sia di tipo *opzionale*, come nella fig. 29, la traduzione ottimale è quella di far migrare la chiave primaria dall'entità obbligatoria a quella opzionale (da persona a patente), in modo da non avere attributi con valore nullo.

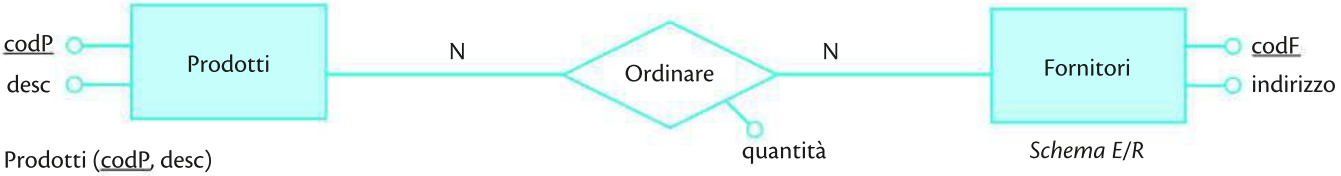


Persona (CF, cognome, nome, data nascita)
Patente (numero, data rilascio, tipo, CFP)

fig. 29 Traduzione associazione 1:1 opzionale

Molti a molti

Le associazioni molti a molti (N:N) si risolvono creando una nuova tabella (in aggiunta alle tabelle derivate dalle due entità), in cui vengono inserite le chiavi delle due entità e gli eventuali attributi dell'associazione (fig. 30).



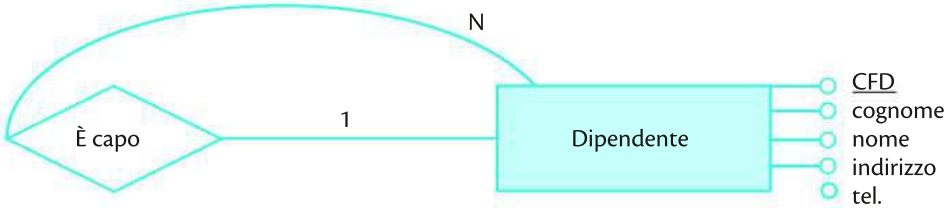
Prodotti (codP, desc)
Fornitori (codF, indirizzo)
Ordinare (codP, codF, quantità)

fig. 30 Schema e soluzione delle associazioni N:N

Nell'esempio riportato, poiché un fornitore può fornire più prodotti e un prodotto può essere fornito da più fornitori, si crea una nuova tabella (Ordinare) composta dalle chiavi primarie delle tabelle di partenza (CodF e CodP) e dall'attributo quantità.
Si nota come né CodF né CodP possano essere chiave primaria della relazione Ordinare, perché non individuerebbero in modo univoco una riga della tabella. In questo caso la chiave primaria è data dalla concatenazione delle due chiavi esterne. Ogni record della tabella Ordinare indica quindi la quantità di un certo prodotto venduta da un determinato fornitore.

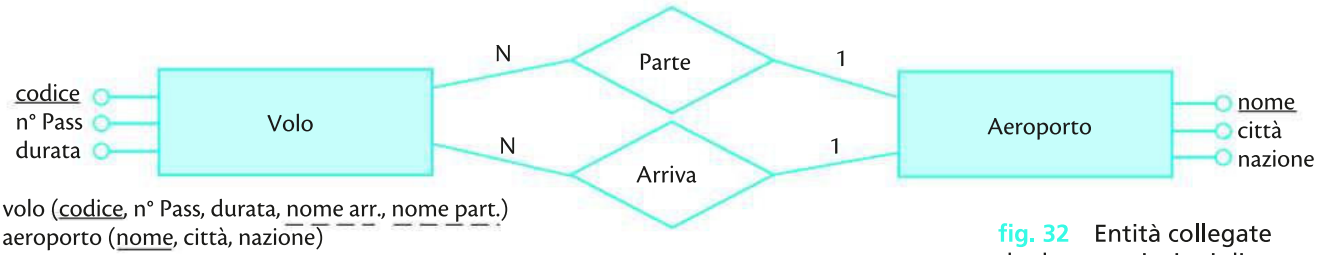
Dal modello concettuale vengono derivate le tabelle che rappresentano le entità e l'associazione molti a molti viene tradotta introducendo una terza tabella contenente le chiavi delle due entità e gli eventuali attributi dell'associazione.

Le regole di traduzione enunciate vengono utilizzate anche per i casi particolari, come le associazioni ricorsive ed entità collegate da associazioni diverse. Esempi di tali traduzioni sono riportati in fig. 31 e fig. 32.



dipendente (CFD, cognome, nome, indirizzo, tel., CFCAPO)

fig. 31 Traduzione di associazioni ricorsive



volo (codice, n° Pass, durata, nome arr., nome part.)
aeroporto (nome, città, nazione)

fig. 32 Entità collegate da due associazioni diverse

LABORATORIO

IL PROBLEMA Definire lo schema relazionale derivante dallo schema E/R riportato nella figura.

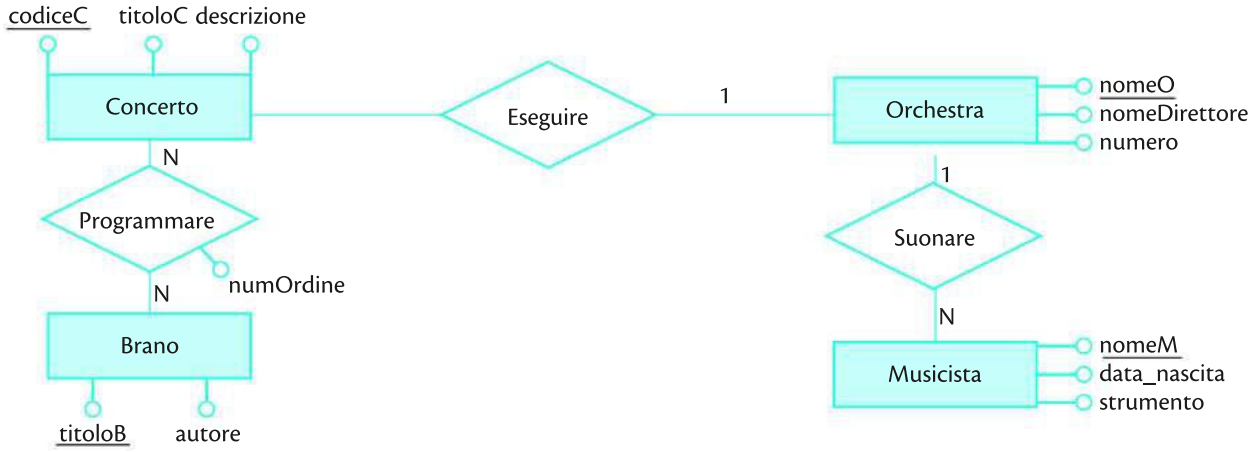


fig. 33 Schema E/R

L'ANALISI Lo schema da tradurre non ha bisogno di essere ristrutturato; applicando le regole di derivazione, otteniamo lo schema logico relazionale.

LA RISOLUZIONE

Si possono riassumere le informazioni presenti nello schema E/R con le seguenti tabelle.
Le entità presenti: Concerto, Orchestra, Musicista e Brano diventano altrettante tabelle contenenti gli stessi attributi delle entità.
Per quanto riguarda le relazioni, tra l'entità Orchestra e l'entità Concerto esiste la relazione uno a molti, che viene tradotta nel modello logico relazionale, inserendo la chiave primaria della tabella Orchestra (NomeO) nella tabella Concerto.
Tra l'entità Orchestra e l'entità Musicista esiste la relazione uno a molti, che viene tradotta inserendo la chiave primaria della tabella Orchestra (NomeO) nella tabella Musicista.
Tra l'entità Concerto e l'entità Brano esiste la relazione molti a molti, che viene tradotta inserendo una nuova tabella (Programmare) contenente le chiavi delle due entità a essa relazionate (CodiceC, TitoloB) e l'attributo della relazione NumOrdine.
Lo schema logico relazionale risulta pertanto:
Concerto (CodiceC, TitoloC, Descrizione, *NomeO*)
Orchestra (NomeO, Nome Direttore, Numero)
Musicista (NomeM, Data Nascita, Strumento, *NomeO*)
Brano (TitoloB, Autore)
Programmare (*CodiceC*, *TitoloB*, NumOrdine)

Più dettagliatamente avrai:

CONCERTO

nome campo	descrizione	tipo	lunghezza	chiave
CodiceC	codice del concerto	stringa	4 byte	primaria
Titolo	titolo del concerto	stringa	20 byte	
Descrizione	descrizione del concerto	stringa	50 byte	
NomeO	nome dell'orchestra	stringa	30 byte	esterna

BRANO

nome campo	descrizione	tipo	lunghezza	chiave
TitoloB	titolo del brano	stringa	50 byte	primaria
Autore	autore del brano	stringa	20 byte	

PROGRAMMARE

nome campo	descrizione	tipo	lunghezza	chiave
TitoloB	titolo del brano	stringa	50 byte	esterna
CodiceC	codice del concerto	stringa	4 byte	esterna
NumOrdine	numero d'ordine di esecuzione del brano all'interno del concerto	intero	2 byte	

ORCHESTRA

nome campo	descrizione	tipo	lunghezza	chiave
NomeO	nome dell'orchestra	stringa	30 byte	primaria
NomeDirettore	nome del direttore dell'orchestra	stringa	30 byte	
Numero	numero di elementi dell'orchestra	intero	2 byte	

MUSICISTA

nome campo	descrizione	tipo	lunghezza	chiave
NomeM	nome del musicista	stringa	30 byte	primaria
DataNascita	data di nascita del musicista	data	8 byte	
Strumento	strumento suonato dal musicista	stringa	20 byte	
NomeO	nome dell'orchestra	stringa	30 byte	esterna

RICAPITOLIAMO

PROGETTARE UN DATABASE

La progettazione delle basi dati si articola nelle seguenti fasi operative, illustrate nella fig. 34:

- progettazione concettuale;

- progettazione logica;
- progettazione fisica.

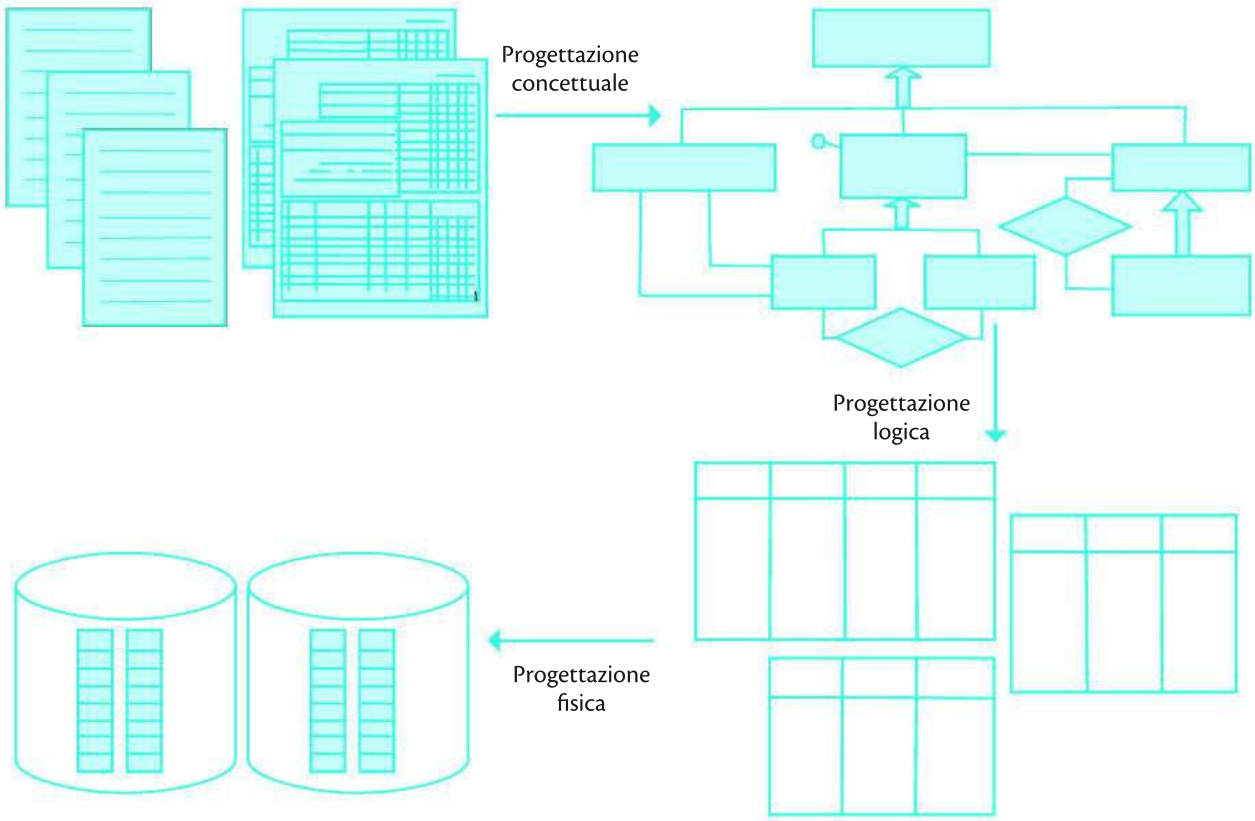


fig. 34 Fasi della progettazione

La **progettazione concettuale** ha l'obiettivo di tradurre i requisiti espressi dal cliente in una descrizione formale delle informazioni necessarie al sistema. Tale descrizione è detta **schema concettuale**, in quanto:

- è indipendente dalle caratteristiche di ogni specifico DBMS;
- la sua strutturazione dipende esclusivamente dai legami di significato che esistono tra le varie informazioni in esso contenute, non da criteri di efficienza.

La **progettazione logica** ha l'obiettivo di trasformare lo **schema concettuale** in uno schema logico conforme alle strutture proprie del DBMS scelto per la realizzazione (per esempio: schema logico gerarchico, relazionale, ecc.).

La **progettazione fisica** ha l'obiettivo di completare lo schema logico con i parametri fisici di memorizzazione e di ricerca dei dati (organizzazione dei file e degli indici), tenendo conto delle caratteristiche del DBMS e dell'ambiente hardware e software in cui la base dati sarà realizzata.

FISSA LE CONOSCENZE

1. Come si risolve la relazione 1:1 nel modello logico?
2. Come si risolve la relazione 1:N nel modello logico?
3. Come si risolve la relazione N:N nel modello logico?

5. OPERAZIONI SULLE TABELLE RELAZIONALI

Come si è già detto, le operazioni da compiere sulle tabelle sono basate sull'**algebra relazionale.** Le operazioni di reperimento sono orientate agli insiemi di record (*set-oriented*): infatti a ogni richiesta viene fornito l'**insieme delle occorrenze che soddisfano certe condizioni** e non solo il primo record.
Dal punto di vista logico, il risultato di ogni operazione è una **nuova tabella** (fig. 35).

fig. 35 Operatore di algebra relazionale

Alcune di queste operazioni si basano sui concetti dell'insiemistica (unione, intersezione, differenza), mentre altre sono più specifiche. Alcune operano su una sola tabella di partenza (selezione, proiezione), mentre altre utilizzano più tabelle (prodotto cartesiano, congiunzione).

Operatori insiemistici

Unione: può avvenire tra due *tabelle compatibili* (cioè con lo stesso numero di campi, e questi devono avere lo stesso dominio). Il risultato dell'unione è l'insieme di tutte le occorrenze delle due tabelle esclusi i duplicati (fig. 36).

fig. 36 Esempio per l'operatore di unione

R1

Cod_F	Nome_F	città
F1	Bianchi	Torino
F2	Rossi	Milano

R2

Cod_F	Nome_F	città
F4	Gialli	Torino
F1	Bianchi	Torino
F6	Blu	Genova

Unione

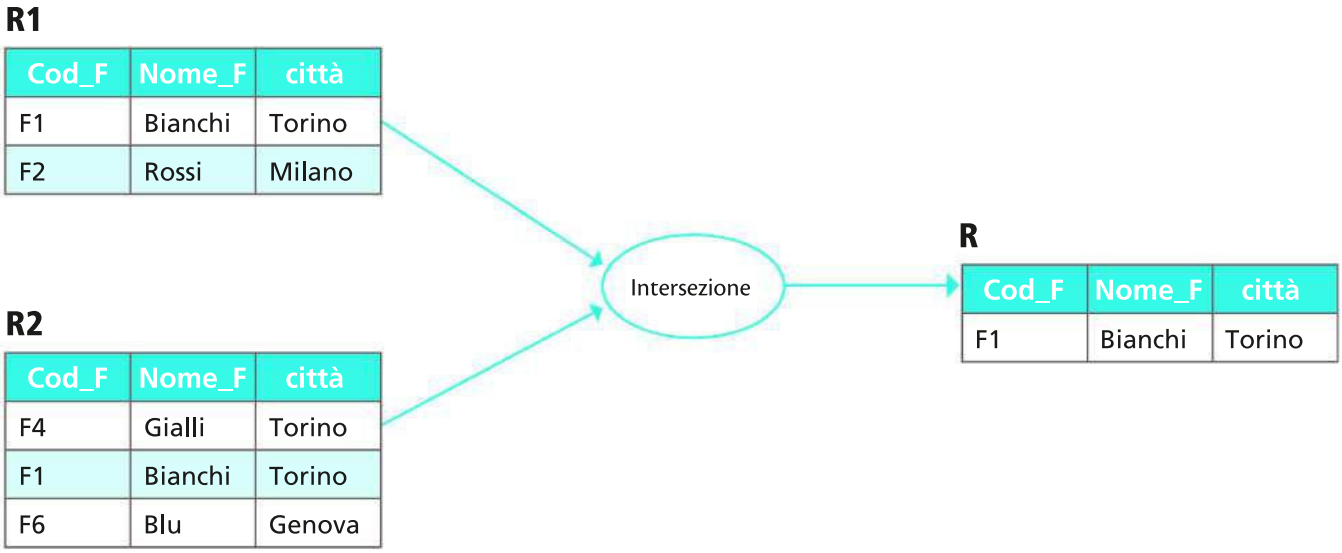
R

Cod_F	Nome_F	città
F1	Bianchi	Torino
F2	Rossi	Milano
F4	Gialli	Torino
F6	Blu	Genova

I duplicati vengono eliminati (F1)

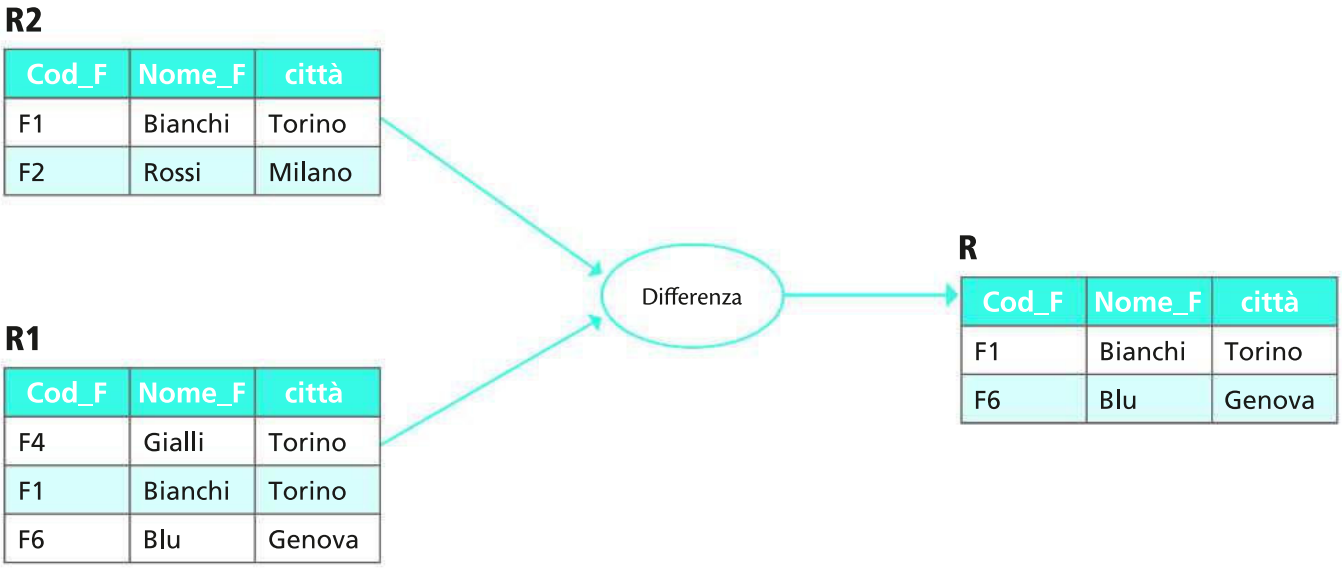
Intersezione: anche in questo caso l'operazione può essere effettuata solo su tabelle compatibili. Il risultato dell'intersezione è una tabella che contiene solo le occorrenze che sono presenti in entrambe le tabelle (fig. 37).

fig. 37 Esempio per l'operatore di intersezione



Differenza: anche in questo caso l'operazione può essere effettuata solo su tabelle compatibili. Il risultato della differenza è una tabella che contiene le occorrenze che sono presenti solo nella prima tabella, ma non nella seconda (fig. 38).

fig. 38 Esempi per gli operatori di differenza



Prodotto cartesiano (*natural join*): applicato a due tabelle restituisce una nuova tabella le cui righe sono ottenute concatenando ogni riga della prima tabella con tutte quelle della seconda tabella (fig. 39). In genere è un insieme molto ampio e generalmente non tutte le combinazioni sono significative.

Il risultato del prodotto cartesiano è una tabella che ha P + Q colonne (P indica il numero di colonne della prima tabella, Q quello della seconda) e M * N righe (M indica il numero di righe della prima tabella, N quello della seconda).

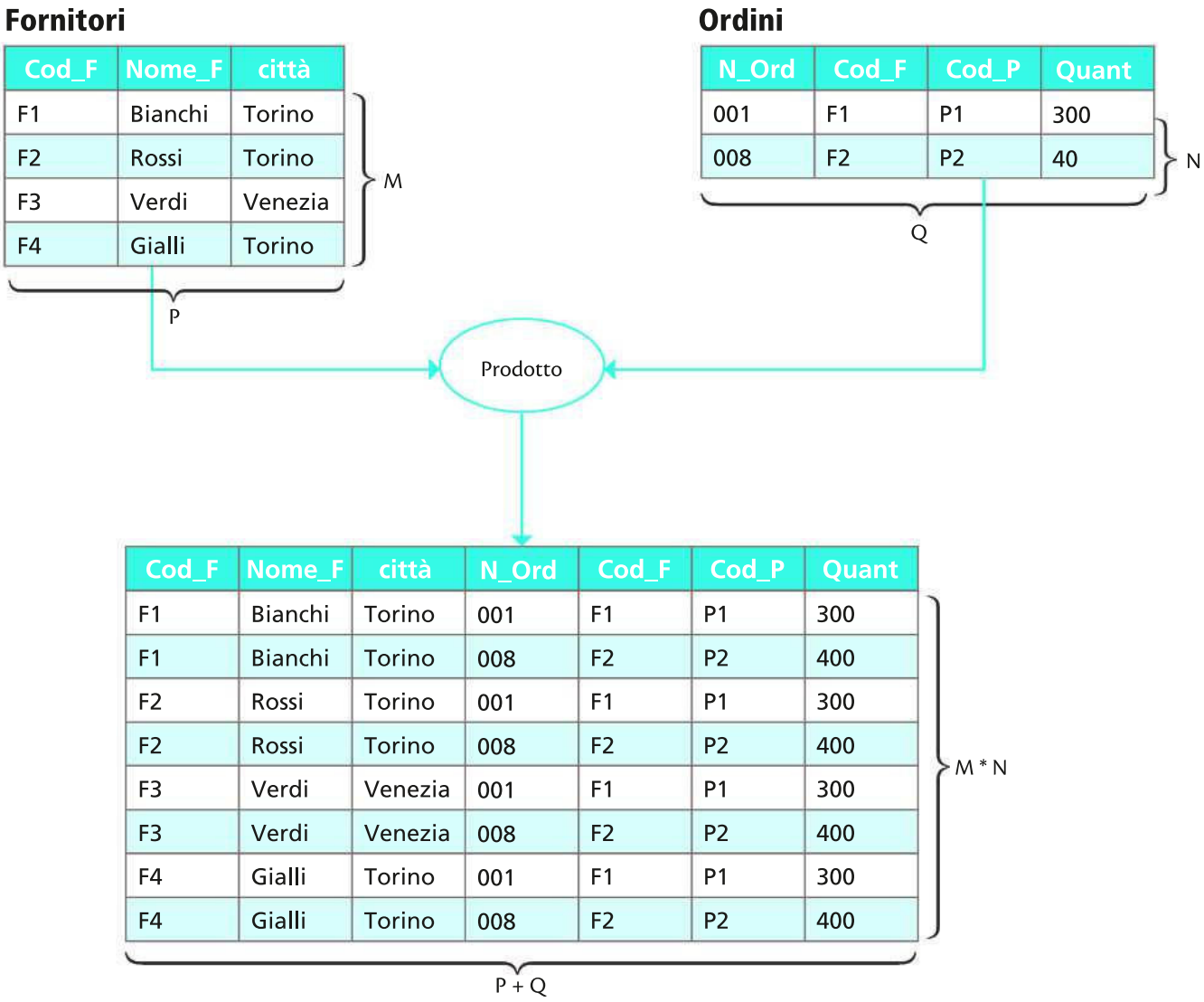


fig. 39 Esempio di prodotto cartesiano

Gli operatori relazionali agiscono su una relazione per ottenere una nuova relazione, estraendo da una tabella una sottotabella, oppure combinando tra loro due o più tabelle e generando così nuove relazioni. Nel primo caso si parla di **operatori unari** (*selezione* e *proiezione*), mentre nel secondo si ha generalmente un **operatore binario** (*congiunzione*).

Selezione (*select*): applicata a una tabella, fornisce come risultato l'insieme delle occorrenze che soddisfano la condizione specificata. Il risultato è una nuova tabella, che contiene tutte le colonne della tabella di partenza (stesso grado), ma ha cardinalità (numero di righe) minore o al limite uguale (fig. 40).

fig. 40 Selezione

LABORATORIO

IL PROBLEMA Si abbia il seguente schema logico relazionale:
Fornitori (Cod_F, Nome_F, città, anno_nascita).
La tabella contiene i seguenti dati.

FORNITORI

Cod_F	Nome_F	città	anno_nascita
F1	Bianchi	Torino	1971
F2	Rossi	Torino	1973
F3	Verdi	Genova	1981
F4	Gialli	Torino	1983
F5	Marrone	Napoli	1980
F6	Blu	Genova	1974

Fornire:
- i dati dei fornitori residenti a "Torino";
- i dati dei fornitori che abitano a Genova e che sono nati dopo il 1980.

L'ANALISI

Come si vede, la tabella Fornitori ha cardinalità 6 e grado 4.

- Se vogliamo ottenere i dati dei fornitori residenti a Torino, possiamo utilizzare l'operatore di selezione in questo modo:
  Selezione su Fornitori dove città = "Torino"
  Il risultato di questa operazione è una nuova tabella, che ha cardinalità 3 e grado 4.

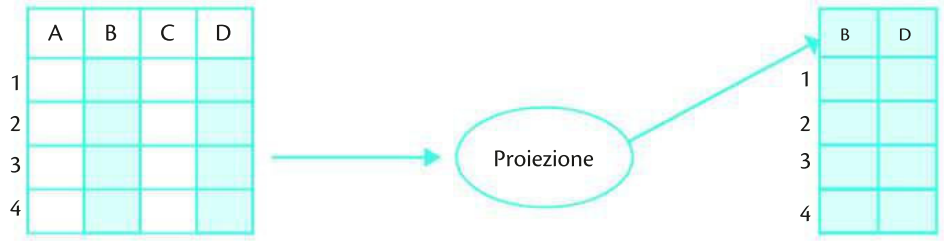
Cod_F	Nome_F	città	anno_nascita
F1	Bianchi	Torino	1971
F2	Rossi	Torino	1973
F4	Gialli	Torino	1983

- Se invece vogliamo i dati dei fornitori che abitano a Genova e che sono nati dopo il 1980, possiamo impostare l'operatore logico AND nella condizione di ricerca.
  Avremo pertanto:
  Selezione su Fornitori dove città = "Genova"and anno_nascita > 1980
  che produce il risultato seguente.

Cod_F	Nome_F	città	anno_nascita
F3	Verdi	Genova	1981

Proiezione (*project*): applicata a una tabella, fornisce come risultato una nuova tabella, che contiene tutte le righe della tabella di partenza (stessa cardinalità), ma con le sole colonne indicate (grado inferiore) (fig. 41).

fig. 41 Proiezione



LABORATORIO

IL PROBLEMA Dallo schema logico relazionale e la tabella precedenti, ottenere:

- il codice di tutti i fornitori;
- il codice e il nome dei fornitori.

L'ANALISI

- Se vogliamo ottenere il codice di tutti i fornitori contenuti nella tabella Fornitori dell'esempio precedente, possiamo impostare l'operatore di proiezione nel seguente modo:
  Proiezione su Fornitori di Cod_F
  Il risultato di questa operazione è una nuova tabella, che ha cardinalità 6 e grado 1.

Cod_F
F1
F2
F3
F4
F5
F6

- Volendo invece ottenere il codice e il nome dei fornitori, possiamo impostare l'interrogazione per mezzo della seguente proiezione:
  Proiezione su Fornitori di Cod_F, Nome_F
  La nuova relazione ottenuta è espressa dalla nuova tabella.

Cod_F	Nome_F
F1	Bianchi
F2	Rossi
F3	Verdi
F4	Gialli
F5	Marrone
F6	Blu

Congiunzione (*join*): si effettua tra due tabelle che hanno un attributo in comune (cioè definito sullo stesso dominio). È un sottoinsieme del prodotto cartesiano a cui è stata applicata una clausola di selezione sull'uguaglianza dell'attributo comune. Il risultato di una join è una tabella che ha grado P + Q – 1: P indica il grado della prima tabella, Q quello della seconda, mentre l'attributo comune usato per la join compare una sola volta.
La congiunzione qui descritta è detta *equi join*, poiché è stata formulata una condizione di uguaglianza sull'elemento comune. Con la equi join (che viene anche chiamata *inner join*) nella tabella risultante vengono inserite le righe delle due tabelle che trovano corrispondenza.

LABORATORIO

IL PROBLEMA Consideriamo le tabelle Proprietari e Auto collegate logicamente dall'attributo Cod_P (codice del proprietario).

PROPRIETARI

Cod_P	cognome
P45	Rossi
M98	Bianchi
D33	Verdi

AUTO

targa	Cod_P
AY888BD	P45
BH754AJ	D33
DK712DW	M98
BZ876AA	D33

Ottenere l'elenco dei proprietari e delle targhe delle auto da loro possedute.

L'ANALISI Per ottenere l'elenco dei proprietari e le targhe delle auto da loro possedute, possiamo impostare l'operazione join secondo la seguente sintassi:
Giunzione su Proprietari, Auto dove Proprietari.Cod_P = Auto.Cod_P
Il risultato di questa operazione è una nuova tabella, che ha cardinalità 3 e grado 4.

Cod_F	cognome	targa
P45	Rossi	AY888BD
M98	Bianchi	DK712DW
D33	Verdi	BH754AJ
D33	Verdi	BZ876AA

Oltre alla equi join è possibile avere altri tipi di join.

- *Outer join*: è un'operazione di congiunzione tra tabelle (fig. 42), in cui nella tabella risultante compaiono non solo le righe che trovano corrispondenza, ma tutte le righe delle tabelle di partenza (fig. 42a). Le tuple che non hanno controparte vengono completate con valori Null.
- *Left join*: nella tabella risultante compaiono tutte le righe della prima tabella e quelle della seconda a esse correlate (fig. 42b).

- *Right join*: nella tabella risultante compaiono tutte le righe della seconda tabella e quelle della prima a esse correlate (fig. 42c).

Parti

Cod_F	Sede_Disp
P1	Torino
P7	Genova

Fornitori

Cod_F	città
F1	Torino
F2	Milano
F4	Torino

Cod_F	città	Cod_P
F1	Torino	P1
F2	Milano	
F4	Torino	
	Genova	P7

a. Outer join

Cod_F	città	Cod_P
F1	Torino	P1
F2	Milano	
F4	Torino	

b. Left join

Cod_F	città	Cod_P
F1	Torino	P1
	Genova	P7
F4	Torino	P1

c. Right join

fig. 42 Operatore join ed esempi di outer, left e right join.

Se dalle relazioni della fig. 43 sorgesse la necessità di conoscere quali studenti non hanno ancora superato esami, la equi join tra Studenti ed Esami_Sostenuti non sarebbe sufficiente, perché le matricole degli studenti che non hanno superato esami non compaiono fra le righe della tabella Esami_Sostenuti e quindi neppure nel risultato della join.

Studenti

matricola	cognome	nome
A1	Rossi	Mario
A2	Verdi	Michela
A3	Bianchi	Giovanni
A4	Neri	Giuseppe

Esami

Cod_Esame	materia
ES001	analisi
ES002	fisica
ES003	informatica
ES004	geometria

Studenti

matricola	cognome	voto
ES001	A1	25
ES001	A2	30
ES002	A1	30
ES002	A4	18
ES003	A1	30

fig. 43 Risultato della left join

Utilizzando la left join fra le tabelle Studenti ed Esami_Sostenuti nella tabella risultante si hanno, oltre alle righe che trovano corrispondenza, anche i record di Studenti che non trovano corrispondenza con Esami_Sostenuti. Nella fig. 44 è riportato il risultato di questa operazione.

fig. 44 Esempi di schema

matricola	cognome	nome	Cod_Esame	voto
A1	Rossi	Mario	ES001	25
A1	Rossi	Mario	ES002	30
A1	Rossi	Mario	ES003	30
A2	Verdi	Michela	ES001	30
A4	Neri	Giuseppe	ES002	18
A3	Banchi	Giovanni		

Un altro particolare tipo di join è la self join, che effettua la join di una tabella su se stessa.
Esaminando la tabella della fig. 45 si vogliono conoscere le coppie delle persone sposate.

Cod_Fisc	cognome	nome	Cf_Coniuge
CF1	Rossi	Mario	CF3
CF2	Bruni	Riccardo	CF5
CF3	Riva	Maria	CF1
CF5	Benna	Luisa	CF2
CF7	Verdi	Giovanni	

fig. 45 Tabella di esempio

Per risolvere questo problema bisogna congiungere la tabella Persona con se stessa, perché il codice fiscale del coniuge (Cf_Coniuge) è anche il codice fiscale (Cod_Fisc) di una persona.
Per non creare ambiguità, l'operazione join è effettuata sulla tabella Persona e sulla tabella T1, che è semplicemente un altro modo di chiamare la tabella Persona (è un alias).
Nella fig. 46 è rappresentata la tabella risultante della self join.

Cod_Fisc	cognome	nome	Cf_Coniuge	cognome	nome	Cf_Coniuge
CF1	Rossi	Mario	CF3	Riva	Maria	CF1
CF2	Bruni	Riccardo	CF5	Benna	Luisa	CF2
CF3	Riva	Maria	CF1	Rossi	Mario	CF3
CF5	Benna	Luisa	CF2	Bruni	Riccardo	CF5

fig. 46 Tabella di self join

Le operazioni descritte, a seconda dei linguaggi e dei sistemi, vengono tradotte in modo differente. In alcuni casi è necessario un comando per ogni operazione, mentre in altri è possibile effettuare più operazioni con un'unica istruzione o combinando comandi elementari.
Oltre a queste operazioni in genere vengono fornite anche altre funzionalità, come operazioni di somma, media e così via.

FISSA LE CONOSCENZE

1. Quali sono i principali operatori insiemistici?
2. Che cosa restituisce l'operatore di selezione?
3. Che cosa restituisce l'operatore di proiezione?
4. Che cosa restituisce l'operatore join?
5. Qual è la differenza tra left join e right join?

6. ALGEBRA RELAZIONALE

L'algebra relazionale è un linguaggio di interrogazione. Ciascun operatore applicato a una o più tabelle produce come risultato un'altra tabella relazionale.
Nell'algebra relazionale, oltre agli operatori insiemistici, vengono implementati gli operatori relazionali nel seguente modo.

Selezione
L'operazione di selezione si rappresenta con la lettera greca σ (sigma), indicando a pedice la condizione di selezione. L'espressione $\sigma_p T$ indica la selezione delle righe della tabella T che rispecchiano il predicato P.

Esempio: $\sigma_{età<18}$ **Studente**

indica la selezione di tutti gli studenti minorenni.

Proiezione
L'operazione di proiezione si rappresenta con la lettera greca π (pi), indicando a pedice la lista degli attributi selezionati. L'espressione $\pi_L T$ indica la proiezione delle colonne della tabella T specificate nella lista L.

Esempio: $\pi_{nome,cognome}$ **Studente**

indica la proiezione dei campi nome e cognome della tabella Studente.

Congiunzione
L'operazione di congiunzione si rappresenta con il simbolo ⋈. L'espressione $\mathbf{T1}_{PK} \bowtie \mathbf{T2}_{FK}$ indica la congiunzione della tabella T1 con la tabella T2 sugli attributi PK di T1 e FK di T2.

Esempio: $\mathbf{Proprietario}_{CF} \bowtie \mathbf{Auto}_{CFP}$

indica la congiunzione della tabella proprietario su CF e la tabella Auto su CFP.

LABORATORIO

IL PROBLEMA Considera il seguente database relazionale di una banca utilizzato per la gestione dei prestiti e dei mutui dei clienti:
cliente (cf, nome, cognome, numconto)
mutuo (idmutuo, importo, data, n.rate, cfc)

Utilizzando l'algebra relazionale, risolvi le seguenti query.

1. Idmutuo e importo dei mutui contratti dal cliente con codice fiscale "BRCLST66A19F432V".

2. Elenco dei clienti (codice fiscale) che hanno contratto un mutuo con importo superiore a 100.000 euro.
3. Nome e cognome dei clienti che hanno contratto un mutuo in data 25/05/2019.
4. Numero di rate e importo dei mutui stipulati in data 11/09/2019.
5. Elenco dei conti dei clienti che hanno stipulato un mutuo nel 2018.
6. Elenco dei clienti che hanno stipulato mutui con un numero di rate superiore a 10.

L'ANALISI

1. Si tratta di una proiezione di idmutuo e importo sulla tabella risultante da una selezione dei mutui contratti dal dato cliente:
   $\pi_{\text{idmutuo,importo}} (\sigma_{\text{cfc="BRCLST66A19F432V"}} \textbf{mutuo})$

2. Dobbiamo fare una selezione dei mutui con importo superiore a 100.000 euro e visualizzare attraverso una proiezione il loro codice fiscale:
   $\pi_{\text{cfc}} (\sigma_{\text{importo}>100.000})$

3. In questa richiesta è necessario fare una congiunzione tra le due tabelle, perché ci vengono chiesti il nome e cognome del cliente:
   $\pi_{\text{nome,cognome}} (\sigma_{\text{data='25/05/2019'}} (\textbf{cliente}_{\text{cf}} \bowtie \textbf{mutuo}_{\text{cfc}}))$

4. Dobbiamo fare una proiezione su n.rate e importo sulla tabella risultante da una selezione di tutti i mutui stipulati nella data richiesta:
   $\pi_{\text{n.rate,importo}} (\sigma_{\text{data='11/09/2019'}} \textbf{mutuo})$

5. Per esaudire questa richiesta è necessario fare una congiunzione tra le due tabelle, dal risultato estrarre i valori la cui data è compresa tra l'1 gennaio e il 31 dicembre del 2018 e poi visualizzare solo il numero di conto.
   $\pi_{\text{numconto}} (\sigma_{\text{data}>=\text{'01/01/2018' and data} <=\text{'31/12/2018'}} (\textbf{cliente}_{\text{cf}} \bowtie \textbf{mutuo}_{\text{cfc}}))$

6. È necessario fare una congiunzione tra le due tabelle, dal risultato estrarre i mutui con un numero di rate superiori a 10 e poi visualizzare solo il nome e cognome dei clienti.
   $\pi_{\text{nome,cognome}} (\sigma_{\text{n.rate}>10}(\textbf{cliente}_{\text{cf}} \bowtie \textbf{mutuo}_{\text{cfc}}))$

FISSA LE CONOSCENZE

1. Che cosa è l'algebra relazionale?
2. Quali sono gli operatori insiemistici?
3. Quando può avvenire l'unione di due tabelle?
4. Quali righe contiene la differenza di due tabelle?
5. Che cosa contiene il prodotto cartesiano di due tabelle?

7. NORMALIZZAZIONE

Il concetto di normalizzazione

Lo schema, definito dalla progettazione concettuale, deve essere tradotto e ottimizzato, cioè depurato da gran parte delle *anomalie* di gestione che si possono verificare.
Si distinguono tre tipi di anomalia:

- *anomalia di inserimento*: se nell'inserire un nuovo record in una tabella si è costretti a inserire informazioni già presenti nel DB;
- *anomalia di cancellazione*: se nel cancellare un record si è costretti a cancellare informazioni che possono essere ancora utili nel DB;
- *anomalia di aggiornamento*: se dovendo aggiornare un record si è costretti ad aggiornarne molti altri.

Per chiarire meglio questi concetti, consideriamo le anomalie che possono verificarsi aggiornando la seguente tabella:

codProdotto	prezzo	fornitore	indirizzo	città	quantità fornita
Quaderni	1,30	Rossi	via Roma 10	Genova	100
Pennarelli	3,10	Rossi	via Roma 10	Genova	50
Matite	1,10	Verdi	via Genova 8	Torino	10.000
Penne a sfera	1,20	Verdi	via Genova 8	Torino	10.000
Colori a olio	5,50	Bruni	via Milano 10	Pisa	10

- Anomalia di inserimento: per inserire un nuovo ordine bisogna inserire nuovamente i dati anagrafici del fornitore (indirizzo, città) e del prodotto (prezzo).
- Anomalia di cancellazione: cancellando il record relativo all'ordine del prodotto Colori a olio si cancellano anche le informazioni relative al fornitore Bruni.
- Anomalia di aggiornamento: aggiornando l'indirizzo del fornitore Rossi, bisogna aggiornare due tuple.

La **normalizzazione** è l'insieme di criteri di progettazione di un database relazionale diretto a prevenire l'insorgere di tali anomalie.

> La **normalizzazione** è il procedimento che trasforma successivamente le relazioni di partenza, suddividendole in altre più piccole aventi lo stesso contenuto di informazione.

Sono stati definiti diversi gradi di normalizzazione (secondo alcuni testi sono 4, secondo altri 5) a cui si fanno corrispondere le forme normali. In una trattazione rigorosa, una tabella non può essere considerata relazionale se non è normalizzata, cioè almeno in prima forma normale.
Nel corso del tempo però ci si è resi conto che la rigorosità e la purezza

matematica di una relazione non corrispondono necessariamente a una base di dati efficiente e si sono quindi sviluppati altri metodi di progettazione.
Di seguito, completiamo la descrizione del modello relazionale, riportando le definizioni delle prime tre forme normali.

La prima forma normale

Una relazione si dice **normalizzata** o in prima **forma normale** (**1FN**), se tutti i suoi attributi hanno un dominio semplice. Non sono ammessi gruppi e ripetizioni.

La 1FN implica che ogni informazione deve essere "atomica", cioè un campo deve contenere una e una sola informazione. I campi devono contenere sempre un tipo di dato "semplice" (stringa, numero, binario ecc.) e mai aggregazioni (insieme, vettore).
La tabella Dipendenti nella fig. 47 non è normalizzata, perché il campo Figli non è elementare e, inoltre, vi sono dei gruppi ripetuti (i figli di Rossi e di Bianchi). Affinché la tabella sia nella prima forma normale, è necessario trasformarla nella tabella Figli dipendenti.

Dipendenti

nome dip.	figli	
	nome	età
Rossi	Ugo	10
	Andrea	7
Bianchi	Gianni	6
	Maria	3
	Luca	1
Verdi	Pia	8

Figli dipendenti

dipen.	figlio	età
Rossi	Ugo	10
Rossi	Andrea	7
Bianchi	Gianni	6
Bianchi	Maria	3
Bianchi	Luca	1
Verdi	Pia	8

fig. 47 Tabella di figli dipendenti

Dipendenze funzionali

Per formalizzare i problemi visti, si introduce un nuovo tipo di vincolo, la **dipendenza funzionale**.

Consideriamo una relazione R nella quale siano definiti almeno due attributi X e Y. In R vale la **dipendenza funzionale di Y da X** (e si indica con $X \rightarrow Y$) se per ogni coppia di tuple t1 e t2 di R con gli stessi valori su X, t1 e t2 hanno gli stessi valori anche su Y. Si dice anche che **X determina funzionalmente Y** o ancora che **X è un determinante per Y**.

Chiariamo meglio questo concetto con un esempio: consideriamo la relazione Studente e stabiliamo che nella relazione la chiave primaria sia Matricola: Studente (Matricola, Nome, Telefono, Corso, Voto).

In essa si possono evidenziare le seguenti dipendenze funzionali:

- Matricola → Nome
- Matricola → Telefono
- Matricola, Corso → Voto

Nome ha una dipendenza funzionale da Matricola, perché a ogni Nome corrisponde una Matricola. Telefono dipende funzionalmente da Matricola, perché a ogni Telefono corrisponde una Matricola. Voto dipende funzionalmente da Matricola, Corso, perché a ogni Voto corrisponde l'insieme di attributi Matricola, Corso.

La seconda forma normale

La seconda forma normale si applica alle tabelle che hanno la chiave primaria composta da più attributi: è richiesto che tutti gli attributi di una riga dipendano dall'intera chiave primaria e non solo da una parte di essa.

Una relazione si dice in **seconda forma normale (2FN)** se è in 1FN e tutti i suoi attributi che non appartengono alla chiave dipendono funzionalmente e completamente dall'intera chiave; non possono esistere attributi che dipendono solamente da una parte della chiave.

Per esempio nella relazione Campionati, che contiene i dati dei giocatori che hanno segnato reti nei campionati, l'attributo Reti, che indica il numero di reti segnate da un giocatore in un campionato, dipende dalla chiave composta Nome-Anno, mentre l'attributo Luogo, che indica il luogo di nascita del giocatore, è dipendente solo dall'attributo Nome, che è un sottoinsieme della chiave.

Campionati

nome	luogo	anno	reti
Rossi Mario	Firenze	1998/1999	3
Rossi Mario	Firenze	1999/2000	4
Conti Bruno	Roma	1998/1999	6

La relazione non è in 2FN e presenta le seguenti anomalie:

- *anomalia di inserimento*: non è possibile inserire un nuovo giocatore sino a quando non ha segnato reti in un campionato;
- *anomalia di cancellazione*: cancellando la terza riga della tabella, si perdono le informazioni del giocatore Conti Bruno;
- *anomalia di aggiornamento*: aggiornando il luogo di nascita del giocatore Rossi, bisogna aggiornare due tuple.

Affinché la relazione Campionati sia in 2FN, bisogna eliminare dalla tabella Campionati l'attributo Luogo e aggiungere la nuova tabella Giocatori, che contiene i campi Nome e Luogo.

Giocatori

nome	luogo
Rossi Mario	Firenze
Conti Bruno	Roma

Campionati

nome	reti	anno
Rossi Mario	3	1998/1999
Rossi Mario	4	1999/2000
Conti Bruno	6	1998/1999

La terza forma normale

In una tabella in terza forma normale tutte le dipendenze tra colonne devono essere basate sulla chiave primaria; vengono eliminate le dipendenze funzionali transitive di un attributo non chiave da un altro attributo anch'esso non chiave.

Una relazione si dice in **terza forma normale** (**3FN**) se è in 2FN e tutti i suoi attributi che non appartengono alla chiave dipendono direttamente dalla chiave; non possono esistere attributi non chiave che dipendono funzionalmente da altri attributi non chiave.

Esaminiamo la relazione Fattura.

Fattura

numero	cliente	ragione_sociale	importo
001	Rossi	001-344	700
002	Rossi	001-344	600
003	Verdi	022-455	550
004	Rossi	001-344	1800

In questa relazione, che ha come chiave il campo Numero, è presente il campo Ragione_Sociale, che non dipende dalla chiave primaria, bensì da un altro campo, Cliente, che non è chiave: la tabella non è in terza forma normale, data la presenza della dipendenza transitiva:

Ragione_Sociale → Cliente →Numero

La relazione Fattura presenta le seguenti anomalie:

- *anomalia di inserimento*: non è possibile inserire la ragione sociale relativa a un cliente sino a che quest'ultimo non abbia emesso una fattura;
- *anomalia di cancellazione*: cancellando la terza riga della tabella, si perde la ragione sociale del cliente Verdi;
- *anomalia di aggiornamento*: se varia la ragione sociale di Rossi, occorre aggiornare tre tuple.

Anche la terza forma normale può essere ottenuta con un procedimento di scomposizione, separando dalla relazione di partenza il sottoinsieme

di attributi dipendente transitivamente dalla chiave primaria.
Per rendere la relazione Fattura in terza forma normale, bisogna eliminare dalla tabella Fattura l'attributo Ragione_Sociale e aggiungere la nuova tabella Cliente, che contiene i campi Cliente e Ragione_Sociale.

Fattura

numero	cliente	importo
001	Rossi	700
002	Rossi	600
003	Verdi	550
004	Rossi	1800

Cliente

cliente	ragione_sociale
Rossi	001-344
Verdi	022-455

La terza forma normale può essere espressa secondo la formulazione di Boyce-Codd.

Si dice che una relazione è in **BCNF** (**Forma normale di Boyce-Codd** o **Boyce-Normal Form**) se è in prima forma normale (1FN) e ogni determinante è una chiave candidata, cioè ogni attributo dal quale dipendono altri attributi può essere una chiave.

LABORATORIO

IL PROBLEMA Dato il seguente database relazionale, modificarlo in modo che le tabelle siano tutte in 3FN.
Libri (codiceISBN, titolo, autore, telefono_autore, prezzo, codice-editore)
Editore (codice, nome, indirizzo, telefono)

L'ANALISI La tabella Libri non è in 3FN, perché l'attributo telefono_autore dipende da autore, che a sua volta dipende da codiceISBN (chiave primaria). Per renderla in 3FN, bisogna eliminare la dipendenza transitiva, costruendo una nuova tabella Autore.
La tabella Editore è invece in 3FN. Lo schema del database corretto sarà il seguente:

Libri (codice ISBN,titolo, cod-autore, prezzo, codice-editore)
Autore (codice, telefono)
Editore (codice, nome, indirizzo, telefono)

LABORATORIO

IL PROBLEMA Dato il seguente database relazionale, modificarlo in modo che le tabelle siano tutte in 3FN.

Ordine (codiceO, quantità, prezzo, fornitore, telefono_fornitore)
Magazzino (NumeroM, azienda, località, capacità_massima, indirizzo_azienda, telefono_azienda)

L'ANALISI La tabella Ordine è in 2FN, ma non è in 3FN, perché c'è una dipendenza transitiva (telefono_fornitore dipende da fornitore e non da codiceO), per cui deve essere divisa in due tabelle diverse:

Ordine (codiceO, quantità, prezzo, fornitore)
Fornitore (fornitore, telefono_fornitore)

La tabella Magazzino non è in 3FN, perché non è in 2FN. Infatti, gli attributi indirizzo_azienda e telefono_azienda dipendono solo da una parte della chiave; per renderla in 2FN, è necessario dividerla in due tabelle diverse:

Magazzino (NumeroM, azienda, località, capacità_massima)
Azienda (azienda, indirizzo, telefono)

Il procedimento termina, perché sia la tabella Magazzino che Azienda sono in 3FN.
Lo schema finale in 3FN è il seguente:

Ordine (codiceO, quantità, prezzo, fornitore)
Fornitore (fornitore, telefono_fornitore)
Magazzino (NumeroM, azienda, località, capacità_massima)
Azienda (azienda, indirizzo, telefono)

FISSA LE CONOSCENZE

1. Quando una tabella si dice normalizzata?
2. Quando una relazione è in seconda forma normale?
3. Quando una tabella è in terza forma normale?
4. Che cos'è la dipendenza funzionale?

8. VINCOLI DI INTEGRITÀ REFERENZIALE

Nella Lezione 7 dell'Unità 1 abbiamo trattato le problematiche della sicurezza nelle basi di dati. Ora approfondiamo il tema dei vincoli di integrità con alcuni esempi.
Ricordiamo che un vincolo è una proprietà che deve essere soddisfatta da tutte le istanze corrette della base di dati e il DBMS stabilisce delle regole da rispettare per soddisfare tali vincoli. Queste regole possono essere semplici **vincoli intra-relazionali**, se sono definiti sugli attributi di una sola relazione (vincoli di unicità, vincoli di dominio e di tupla), limitando per esempio il dominio di esistenza di un campo, oppure i più sofisticati **vincoli inter-relazionali**, se sono definiti su più relazioni contemporaneamente, tenendo conto di interrelazioni fra i campi, anche di archivi diversi (*integrità referenziale*). I **vincoli di integrità** vengono definiti dall'amministratore della banca dati, che li codifica in uno Schema fornito a tutti i software applicativi che accedono alla base di dati.
Abbiamo visto nelle lezioni precedenti esempi di vincoli intra-relazionali. Soffermiamoci ora sui **vincoli di integrità referenziale**, analizzando alcuni esempi. Una tupla in Esami per essere valida deve contenere un numero di matricola che esista anche in Studenti. Lo studente di matricola 995566, presente in Esami, non è presente in Studenti; quindi è una tupla non valida.

Studenti

matricola	cognome	nome	nascita
254782	Rossi	Giacomo	21/10/1994
564387	Lorenzi	Paolo	20/09/1990
345218	Bianchi	Franca	10/12/1989
123459	Bruni	Chiara	17/07/1991
789657	Verdi	Luigi	20/06/1990

Esami

studente	voto	lode	corso
254782	30	X	A04
995566	28	–	A10
345218	25	–	A16
123459	30	X	A34
789657	30	–	A06

Nella fig. 48 è mostrato un altro esempio di relazioni diverse correlate attraverso valori comuni di uno o più attributi. I valori assunti da un

attributo nella relazione referenziante devono esistere effettivamente come valori di un attributo nell'istanza della relazione referenziata.

codice	articolo	provincia	numero	agente
1123	567	RIM	2F8534	55-89
1256	125	RIM	6D9909	12-88
1433	322	FI	D45668	82-33
2300	219	NA	F81642	23-43

provincia	numero	proprietario
RIM	2F8534	Rossi Mario
RIM	6D9909	Broni Aldo
FI	D45668	Sineri Luigi

agente	nome
55-89	Mirco Sergio
12-88	Luni Piero
82-33	Ronchi Aldo

fig. 48 Relazioni collegate attraverso le chiavi esterne

I vincoli di integrità referenziale vietano le operazioni che potrebbero rendere inconsistente la base di dati; in particolare sono rifiutati alcuni inserimenti e cancellazioni.

Per esempio, facendo riferimento alle tabelle Corsi e Docenti, si può notare che l'attributo MatrDocente nella relazione Corsi fa riferimento a matricola nella relazione Docenti. Perché sia rispettata l'integrità referenziale, è necessario che i valori assunti dall'attributo MatrDocente nella relazione Corsi esistano come valori dell'attributo matricola nella relazione Docenti.

fig. 49 Relazioni Corsi-Docenti

Corsi

codice	nome	MatrDocente
M2170	Fondamamenti di informatica	D101
M4880	Sistemi di elaborazione	D102
F0410	Basi di dati	D321

Docenti

matricola	nome	telefono
D101	Verdi	123456
D102	Bianchi	636363
D321	Neri	414243

Se si impone un vincolo di integrità referenziale non è possibile effettuare un'operazione di inserimento di un Corso tenuto da un docente che non sia già presente nella tabella Docenti; inoltre non è possibile cancellare un docente se prima non sono stati cancellati tutti i corsi tenuti da quel docente.

FISSA LE CONOSCENZE

1. Quali sono i vincoli intra-relazionali?
2. Quando si impongono dei vincoli di integrità referenziale?
3. Quali sono le operazioni vietate tra due tabelle con vincoli di integrità referenziale?

DIDATTICA INCLUSIVA

RIPASSIAMO INSIEME

I modelli logici

Esistono due tipi diversi di DBMS: il **modello a grafo**, basato su puntatori utilizzati come riferimenti espliciti fra record di tipi diversi, e il **modello relazionale**. Ad oggi si stanno sviluppando anche i modelli Object-Oriented e i modelli NoRel, utilizzati soprattutto per gestire dati non strutturati.

Il modello relazionale

Nel modello relazionale, un database è un **insieme di tabelle**. Una tabella relazionale è formata da righe (tuple), che rappresentano le istanze degli oggetti, e da colonne, che rappresentano le proprietà degli oggetti. Le colonne assumono valori appartenenti a un dato dominio e possono contenere al massimo un valore. Solitamente vi è una colonna (o un insieme di colonne), detta **chiave primaria**, che identifica univocamente una riga della tabella.

Ristrutturazione dello schema E/R

Lo schema concettuale va **semplificato** per permettere una più agevole traduzione nel modello relazionale e una gestione più efficiente dei dati. In particolare vanno eliminate le gerarchie e gli attributi multipli, e si possono accorpare due tabelle o dividere una in due per ottimizzare gli accessi ai dati.

Traduzione nello schema logico

Per ottenere uno **schema logico relazionale** dallo schema concettuale E/R, le entità del diagramma E/R diventano tabelle e gli attributi i campi. La relazione di cardinalità 1:1 può essere realizzata introducendo in una delle due tabelle la chiave dell'altra. Nella relazione 1:N il legame fra le due tabelle viene creato aggiungendo agli attributi dell'entità "a molti" la chiave dell'entità "a uno". Per risolvere invece la relazione M:N, si deve introdurre una nuova tabella contenente le chiavi delle due entità e gli eventuali attributi dell'associazione.

Operazioni nelle tabelle relazionali

Gli **operatori di base insiemistici** sono: unione, intersezione, differenza e prodotto cartesiano.
Gli **operatori di base relazionali** sono: selezione, che seleziona da una tabella le tuple che hanno una determinata caratteristica; proiezione, che prende solo le colonne richieste all'interno di ogni tupla della tabella; congiunzione, che seleziona i dati comuni da due o più tabelle tramite un operatore di confronto.

Algebra relazionale

L'algebra relazionale è un **linguaggio di interrogazione** ed è definita da operatori di base. L'applicazione di ognuno di questi operatori produce una relazione.

La normalizzazione

La normalizzazione di uno schema è un procedimento che ha come scopo quello di raffinare lo schema, eliminando eventuali **errori nell'organizzazione dei dati**.
Esistono vari livelli di normalizzazione (forme normali). Una tabella è in 1FN quando tutti i suoi attributi sono definiti su domini elementari; è in 2FN quando tutti i suoi attributi dipendono dall'intera chiave; è in 3FN quando non vi è dipendenza transitiva.

Vincoli di integrità

I vincoli di integrità sono vincoli che devono essere rispettati per garantire la **consistenza** dei dati all'interno delle basi di dati relazionali. Esistono vincoli intra-relazionali (di tupla, di dominio) e inter-relazionali (integrità referenziale).

MAPPA CONCETTUALE

MAPPA MODIFICABILE

DIDATTICA INCLUSIVA