

Indice corso Basi di dati

Cos'è un data base	2
Definizione di data base	2
Definizione di DBMS	2
Differenze tra DBMS e file system	3
Componenti di un DBMS	3
Livelli di astrazione di un DBMS (introduzione)	5
Le funzioni del DBMS	5
Livelli di astrazione di un DBMS	6
Livello fisico	7
Livello concettuale	7
Livello esterno	8
Perché utilizzare un DBMS	9
Modelli di dati	10
Il modello entità-relazione	11
Modelli logici	14
Modello gerarchico	14
Modello reticolare	15
Il modello relazionale	16
Su cosa si basa il modello relazionale	16
Definizione di tupla	17
Valori nulli	17
Vincoli di integrità	18
Concetto di chiave	18
Vincolo di integrità referenziale	19
Tipi di associazioni tra tabelle	19
NORMALIZZAZIONE	20
Definizione di ridondanza	20
Dipendenza funzionale	20
Dipendenza funzionale transitiva	20
Prima forma normale	21
Seconda forma normale	21
Terza forma normale	22
INTRODUZIONE ARCHITETTURA CLIENT SERVER	23
INTRODUZIONE ARCHITETTURA DISTRIBUITA	24
Note legali	25

BASI DI DATI

Cos'è un data base

Per comprendere appieno cos'è un data base e quali sono i vantaggi legati al suo impiego, è necessario definire in modo esatto e preciso cosa si intende per :

- Data base
- DBMS (Data Base Management System).

Definizione di data base:

Un *Data Base* può essere definito come un insieme di informazioni strettamente correlate e memorizzate su un supporto di memoria di massa, costituenti un tutt'uno, che possono essere manipolate da più programmi applicativi.

Il data base, pertanto, è costituito da dati.

Finora abbiamo concepito i dati come file, record, campi, registrazioni su disco, ecc. Adesso interpretiamo questi concetti in modo differente:

Se prima il file era visto come l'unità fondamentale dell'elaborazione dei dati, adesso questo tipo di raggruppamento perde importanza a favore della base di dati, ovvero della **collezione globale** dei dati inerenti un certo soggetto.

Esempio:

Si pensi ad un'agenzia bancaria. L'insieme dei dati relativi ai clienti, alle transazioni, ai conti, ai dipendenti, ecc. costituiscono la base dati. Ciò significa che anche se le entità dei dati sono differenti tra loro (dati che riguardano i dipendenti sono di altro genere, ad esempio, dai dati che riguardano i clienti o i conti bancari) essi comunque appartengono alla medesima realtà, ovvero alla realtà di quella particolare agenzia bancaria.

Definizione di DBMS (Data Base Management System):

Un DBMS è un sistema software per la gestione di basi dati; esso si occupa dell'aggiornamento, della manutenzione e della consultazione di un insieme di registrazioni contenute in un supporto di memoria di massa.

Il DBMS, pertanto, è un insieme di programmi, che sono rivolti alla gestione di dati memorizzati in archivi. Ovviamente, tra Data Base e DBMS esiste una forte iterazione per cui spesso si tende a confonderli, ma sono comunque due cose ben diverse e distinte.

Esempio:

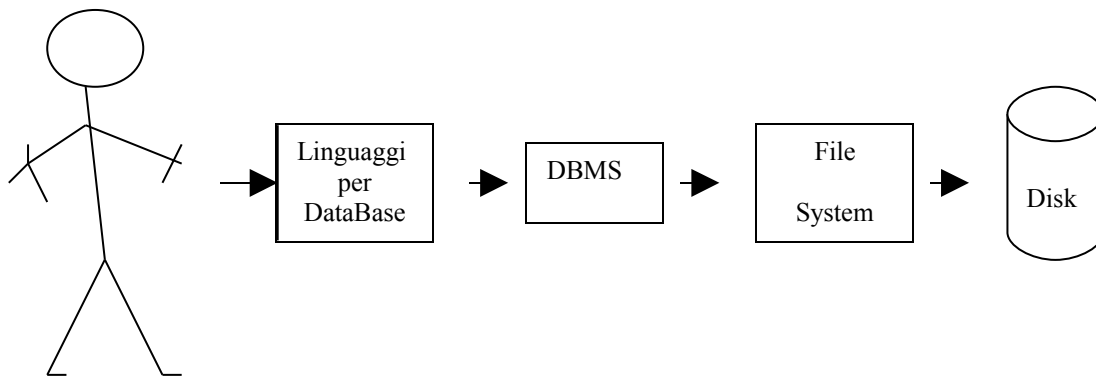
Si pensi ad un bambino che sta giocando con delle conchiglie disponendole in un certo ordine, le conchiglie (entità che vengono manipolate) rappresentano il data base, il bambino (colui che opera) rappresenta il DBMS.

Differenze tra DBMS e file system:

Il file system è un nucleo, costituito da programmi, presente in ogni sistema operativo. La sua funzione è quella di gestire le varie operazioni sui file; questa gestione non è visibile all'utente, infatti il file system opera direttamente al servizio di altri programmi o utility.

Il DBMS è per l'appunto un software che poggiando sul sistema operativo utilizza il file system di quest'ultimo.

In questo caso dunque, la maggiore 'distanza' del DBMS dall'hardware, rispetto al file system, permette un grado di iterazione maggiore. Questo significa che l'utente (programmatore, amministratore di sistema, ecc.) non dovrà più avere a che fare con record e file, bensì con entità astratte che rappresentano la realtà.



Componenti di un DBMS:

È evidente che un DBMS è un insieme di programmi atti ad effettuare interventi su un data base.

Vediamo quali sono le modalità in cui un DBMS può ricevere comandi:

- Direttamente dall'utente in modo interattivo, tramite particolari comandi appartenenti ai linguaggi 'accettati' da quel particolare DBMS.
- Attraverso un programma scritto completamente in uno o più di quei linguaggi di cui parlavamo sopra.
- Tramite un programma scritto in un linguaggio algoritmico tradizionale (COBOL, Pascal, C, ecc.) che ingloba alcuni comandi appartenenti ai linguaggi 'accettati' dal DBMS.

Vediamo ora come possono essere raggruppati questi linguaggi in base alle loro funzioni:

- DDL (*Data Description Language*), tramite i quali si definiscono le strutture del data base. Si dice cioè come dovrà essere la base di dati.
- DML (*Data Manipulation Language*), che servono per impartire comandi di elaborazione dei dati.

- QL (*Query Language*) o linguaggi di interrogazione, che presentano un natura interattiva.

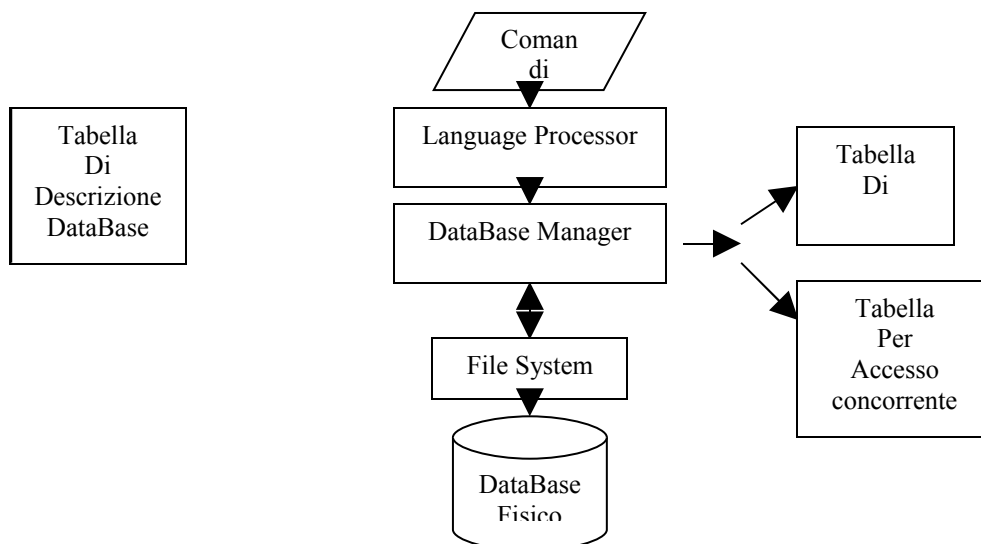
Così, la descrizione delle strutture avverrà tramite il DDL, i programmi di elaborazione saranno scritti in DML e le interrogazioni saranno effettuate con un QL.

Se si desidera fare uso di un linguaggio algoritmico tradizionale, è possibile ricorrere all'*incapsulamento*, ovvero all'inserimento di blocchi di istruzioni scritte in uno dei linguaggi che il DBMS può leggere.

Non sempre la distinzione tra i diversi tipi di linguaggi (DDL, DML, QL) è netta e chiara: vi possono essere infatti linguaggi che comprendono addirittura tutti e tre i tipi di comandi.

Componenti funzionali di un DBMS:

- *Tabella di descrizione data base*, è una tabella in cui è descritto il modello e le caratteristiche del data base.
- *Tabella delle autorizzazioni*, è una tabella in cui sono presenti le informazioni dei vari utenti riguardo ai loro permessi di accesso ai vari dati.
- *Tabella per accesso concorrente*, ha la funzione di permettere la gestione del traffico di più richieste operati contemporaneamente sui medesimi dati.
- *Language processor*, i comandi vengono ricevuti da questa unità, che ha il compito di metterli in relazione con le specifiche di definizione del modello (contenute nella tabella di descrizione del data base) e trasmetterli al Data Base Manager
- *Data Base Manager*, ha il ruolo di ricevere i comandi espressi a livello concettuale cioè operanti sul modello astratto dei dati e tradurli in comandi a livello fisico trasmettendoli al file system del sistema operativo non prima di aver fatto gli opportuni controlli sulla *tabella di autorizzazione* e sulla *tabella di accesso concorrente*.



Livelli di astrazione di un DBMS (introduzione):

Abbiamo accennato al fatto che il DBMS ha, tra le sue funzioni, quella di mantenere un modello astratto dei dati che consenta all'utente di concepire le informazioni secondo strutture riferite direttamente alle entità reali che i dati sono preposti a rappresentare.

Abbiamo inoltre visto che spetta al DBMS, tramite il Data Base Manager, tradurre le operazioni ordinate dall'utente sulle strutture astratte in operazioni sui file.

Abbiamo visto che l'attività del DBMS consiste nel ricevere comandi, metterli in relazione con il modello dei dati, verificare l'eseguibilità e tradurli in operazioni sugli archivi.

Nell'esecuzione di tali operazioni, il DBMS, così come ogni altro programma applicativo, dipende dal sistema operativo: per l'utilizzo concorrente delle risorse, per l'input/output a video o su stampa, per gli interventi in memoria di massa e così via.

Le funzioni del DBMS:

È sufficiente pensare al significato dell'acronimo DBMS (Data Base Management System ovvero Sistema gestore della base dei dati) per capire che la principale funzione di un DBMS è quella di governare ogni operazione di accesso al Data Base, sia per quanto riguarda aggiornamenti o ricerche effettuati sugli archivi, sia per le richieste da parte di più applicazioni contemporaneamente.

La caratteristica principale del DBMS consiste nel mantenimento di un **modello di dati**, ovvero di un'astrazione logica tramite la quale l'utente può vedere i **dati** non in termini di record, file, campi, bensì in termini di **unità informative**, direttamente riferibili agli oggetti della realtà descritti dall'informazione stessa.

Esempio:

Prendiamo in considerazione l'entità *dipendenti* di una ditta; è possibile mettere in relazione (il concetto relazione nasce dalla matematica, ed in particolare dalla teoria degli insiemi) i dati anagrafici ad essi relativi e disegnare la seguente:

DIPENDENTI (Cognome, Nome, Data_nascita)

Questa relazione può corrispondere ad un file, ma ciò non è necessario: non è detto vi sia corrispondenza tra *relazioni*, ovvero tra entità costituite da informazioni unitarie, e *registrazioni*, ovvero record di dati memorizzati su disco.

Tramite il modello dei dati, è possibile progettare e gestire un data base in maniera completamente svincolata:

- Dalle applicazioni
- Dall'organizzazione fisica dei dati

Esempio:

Supponiamo di volere stampare l'elenco dei dipendenti della ditta che hanno uno stipendio base superiore ad un certo ammontare.

Operando tramite il DBMS è possibile mettere in relazione tra loro i dati anagrafici dei dipendenti (in questo caso il nominativo) con dati di tipo amministrativo (la paga base), registrati dal punto di vista fisico, in due archivi differenti. L'utente, tramite un'astrazione dei dati stessi, può combinare i valori ed operare solo sulla porzione di quelli che gli interessano; nel caso in esame opererà su una relazione relativa ai dipendenti con paga superiore all'ammontare indicato, effettuandone la stampa.

Il linguaggio di interrogazione, o *query language*, fornisce inoltre il grosso vantaggio di permettere una consultazione interattiva della base di dati, senza dover redigere un programma apposito per ciascuna esigenza informativa. All'utente basta impostare il comando adeguato per ottenere la risposta cercata.

Esempio:

Utilizzando la programmazione tradizionale, ciascuna richiesta informativa necessita di un programma: i dipendenti con un certo numero di anni di anzianità, la lista di coloro che hanno figli, l'elenco dei lavoratori in ferie in un determinato periodo, ecc. È necessario impostare varie routine con diverso accesso ai dati, che verificano condizioni ogni volta differenti.

Tramite il query language, invece, ciascuna esigenza informativa è soddisfatta tramite un'interrogazione: un solo comando va a sostituire un intero programma di centinaia di righe.

Accanto a tali funzioni, il DBMS ne espleta anche altre relative per lo più alla *sicurezza* (non permettere all'accesso ai dati a persone non autorizzate) e *all'integrità* dei dati (garantire e permettere delle modifiche e aggiornamenti che non stravolgano lo schema logico della base di dati).

Vediamo quali sono queste altre funzioni:

- *La gestione delle transazioni* (transazione: è un'azione che opera sui dati del database: li legge, li scrive, ecc.), per fornire un accesso corretto e concorrente al data base da parte di molti utenti contemporaneamente.
- *L'accesso controllato*, per limitare l'accesso ai dati nei riguardi degli utenti non autorizzati e controllare la validità dei dati immessi.
- *La capacità di recupero*, cioè la possibilità di ripristino a seguito di guasti del sistema senza perdere i dati.

Livelli di astrazione di un DBMS:

Abbiamo accennato al fatto che il DBMS ha tra le sue funzioni, quella di mantenere un modello astratto dei dati che consenta all'utente di concepire le informazioni secondo strutture riferite direttamente alle entità reali che i dati sono preposti a rappresentare.

Abbiamo inoltre visto che spetta sempre al DBMS, per mezzo del Data Base Manager, tradurre le operazioni ordinate dall'utente sulle strutture astratte in operazioni sui file.

Vediamo adesso quali sono i livelli di astrazione consentiti.

Livello fisico:

Il livello fisico del data base è rappresentato dalle strutture di memoria di massa usate per conservare i dati e per accedervi in modo rapido ed efficiente.

È necessario distinguere tra:

1. I dati veri e propri
 - Le strutture che li contengono e che ci permettono di accedere ai medesimi

Il livello fisico è, per l'utente, del tutto trasparente: egli, infatti, non si preoccupa affatto di come i dati vengano registrati sui supporti: tale funzione è compito esclusivo del DBMS. L'utente si occuperà principalmente del cosa vi è registrato: quali sono i dati e in quale relazione si trovano tra loro.

Essendo la gestione dei file realizzata dal sistema, l'utente non deve interessarsi della forma assunta dai dati in memoria di massa; non gli resta dunque che occuparsi del valore informativo dei dati.

Livello concettuale:

Il livello concettuale rappresenta la struttura globale del data base, relativa a tutte le informazioni in esso presenti, rappresentate in modo organizzato tramite un modello astratto.

La descrizione è realizzata tramite il Data Description Language (DDL), ovvero il linguaggio di descrizione dei dati, che fornisce la descrizione del modello astratto del data base.

Il modello è composto da *entità*, a ciascuna entità saranno riferite informazioni, rappresentate da *attributi* inerenti all'entità medesima. Si crea quindi un rapporto biunivoco tra un'entità del mondo reale, che costituisce l'oggetto di riferimento, e un insieme di dati omogenei e coerenti all'interno del data base. Questi dati riporteranno le caratteristiche dell'oggetto a cui si riferiscono, realizzando per l'appunto un processo di *astrazione* dal livello reale al livello concettuale.

Un'entità è costituita da ciascun *oggetto* al quale si vogliono riferire informazioni: sono entità gli studenti di una scuola, i dipendenti di un'azienda, i correntisti di un istituto di credito, i libri di una biblioteca. A ciascuna entità fanno riferimento dati ben precisi: nome, data di nascita, reparto, classe, titolo, autore, ecc. Gli attributi sono collegati alle entità, costituendo un oggetto di potenziale informativo maggiore rispetto ad una collezione di informazioni non correlate.

Esempio:

Prendiamo in esame l'entità STUDENTI

STUDENTI (nome, classe, sezione)

Il dominio di <nome> sarà costituito da una stringa di caratteri, il dominio <classe> da un intero compreso tra 1 e 5, il dominio di <sezione> da un carattere alfabetico limitato alle sole sezioni presenti nella scuola.

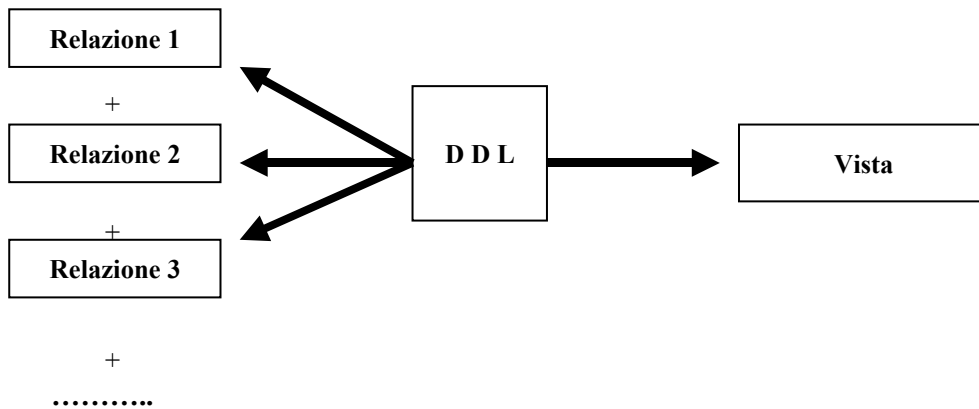
Livello esterno:

Una volta definito lo schema concettuale, è possibile limitare l'accesso ai dati ad alcuni utenti. Quindi sono realizzabili accessi e navigazioni personalizzate. Ciò è reso possibile dalle funzioni del DBMS relative al *livello* esterno che danno la sensazione a ciascuno di essere l'unico utilizzatore dei dati che vede. Accessi e navigazioni si attuano attraverso schemi detti *viste logiche* dei dati.

Definizione di vista:

Una **vista** è l'**astrazione** di una parte del data base concettuale che coinvolge i dati dell'istanza del data base limitatamente alla **porzione interessata**.

Attraverso il DDL, l'utente può descrivere viste logiche dei dati che abbiano tra loro informazioni appartenenti a entità diverse, creando così *nuove entità informative*



Si può inoltre limitare l'accesso in *orizzontale*, cioè a una sola parte degli attributi di una entità, oppure in *verticale*, condizionando il dominio di uno o più attributi. In questo modo, si verrà a disporre di una nuova configurazione della base dei dati, rispondente in modo completo alle esigenze di quel utente.

Esempio:

Tornando alla entità di cui sopra:

STUDENTI (nome, classe, sezione)

Qualora ci interessi una stampa dei nomi, dei soli studenti che frequentano la classe quinta, è possibile descrivere una vista logica dei dati limitando il dominio di <classe> al valore 5 e gli attributi a <nome>. Se invece si desidera il listato dei soli alunni della sezione E, si limiterà il dominio di <sezione> a E appunto.

In sostanza, la vista logica è un modo di vedere, accedere e navigare sui dati che sono stati descritti in precedenza dallo schema concettuale e tramite questo registrati nell'istanza del data base a livello fisico.

A questo punto siamo in grado di riassumere quanto detto per i tre livelli:

- Il *livello fisico* riguarda l'effettiva memorizzazione dei dati, organizzati in file, record e strutture di accesso.
- Il *livello concettuale* riguarda la struttura logica assunta dai dati registrati, quindi il loro schema astratto.
- Il *livello esterno* si riferisce al modo in cui ciascun utente può vedere gli stessi dati, che mantengono l'organizzazione fisica e concettuale precedentemente descritte, ma vengono messe a disposizione secondo il formato richiesto.

Il concetto fondamentale da comprendere è la differenza tra i *dati* e lo *schema*; solo i file contengono effettivamente raggruppamenti di byte, mentre gli schemi, le entità e le viste non fanno altro che descrivere il formato con cui possono essere di volta in volta manipolati. Tale realtà, tuttavia, è completamente trasparente all'utente, che opera tramite proprie viste logiche come se si trattasse di una base dati fisica vera e propria.

Perché utilizzare un DBMS

Il largo successo ottenuto nel corso degli ultimi quindici anni dai sistemi di gestione delle basi di dati lascia intendere che i vantaggi legati al loro utilizzo siano numerosi. I DBMS sono sistemi di utilità che si occupano di una vasta gamma di operazioni relative alla gestione dei dati non volatili del sistema di elaborazione; in particolare si tratta di operazioni che:

- *Sostituiscono il programmatore*: in questo caso, i vantaggi sono legati al fatto che il sistema esegue funzioni che in precedenza erano a carico dei programmatori. Si pensi alla possibilità di consultare la base di dati senza necessità di redigere un apposito programma, ma attraverso un semplice comando di interrogazione, utilizzando il query language; in questo caso l'accesso ai dati è realizzato tramite routine del DBMS.
- *Arricchiscono la gestione precedente*, ovvero permettono l'esecuzione di nuove funzioni, in basi di dati già esistenti, aumentandone quindi il potenziale informativo senza che sia necessario aumentare la quantità di dati raccolti e memorizzati.

In una gestione tradizionale è conveniente scrivere un programma di interrogazione solo quando questa esigenza interessa più utenti e si verifica con una certa frequenza. Con i data base invece ciascun utente può ottenere informazioni di grande valore semplicemente manipolando i dati secondo i criteri che desidera.

In sostanza, con il DBMS si realizza una *gestione centralizzata e controllata* della base dati i cui vantaggi principali possono essere così sintetizzati:

- *Riduzione della ridondanza dei dati*, dove con ridondanza si intende la possibilità che i dati si presentino ripetutamente nella base.
- *Eliminazione dell'incongruenza*.
- *Condivisione* dei dati da parte di tutte le applicazioni che ne facciano richiesta.
- *Sicurezza e riservatezza* delle informazioni, per ridurre il rischio di distruzione di dati e di accessi non autorizzati.
- *Ottimizzazione della struttura* della base dati, che ne facilita l'accesso e la manutenzione e ne garantisce una crescita ordinata.
- *Indipendenza dei dati dalle applicazioni*.

Modelli di dati

Come si è visto, uno degli scopi del DBMS è il mantenimento di un modello astratto dei dati in grado di rappresentare la realtà cui questi dati si riferiscono e di mantenere l'indipendenza tra l'aspetto formale della base dati e la sua effettiva implementazione fisica.

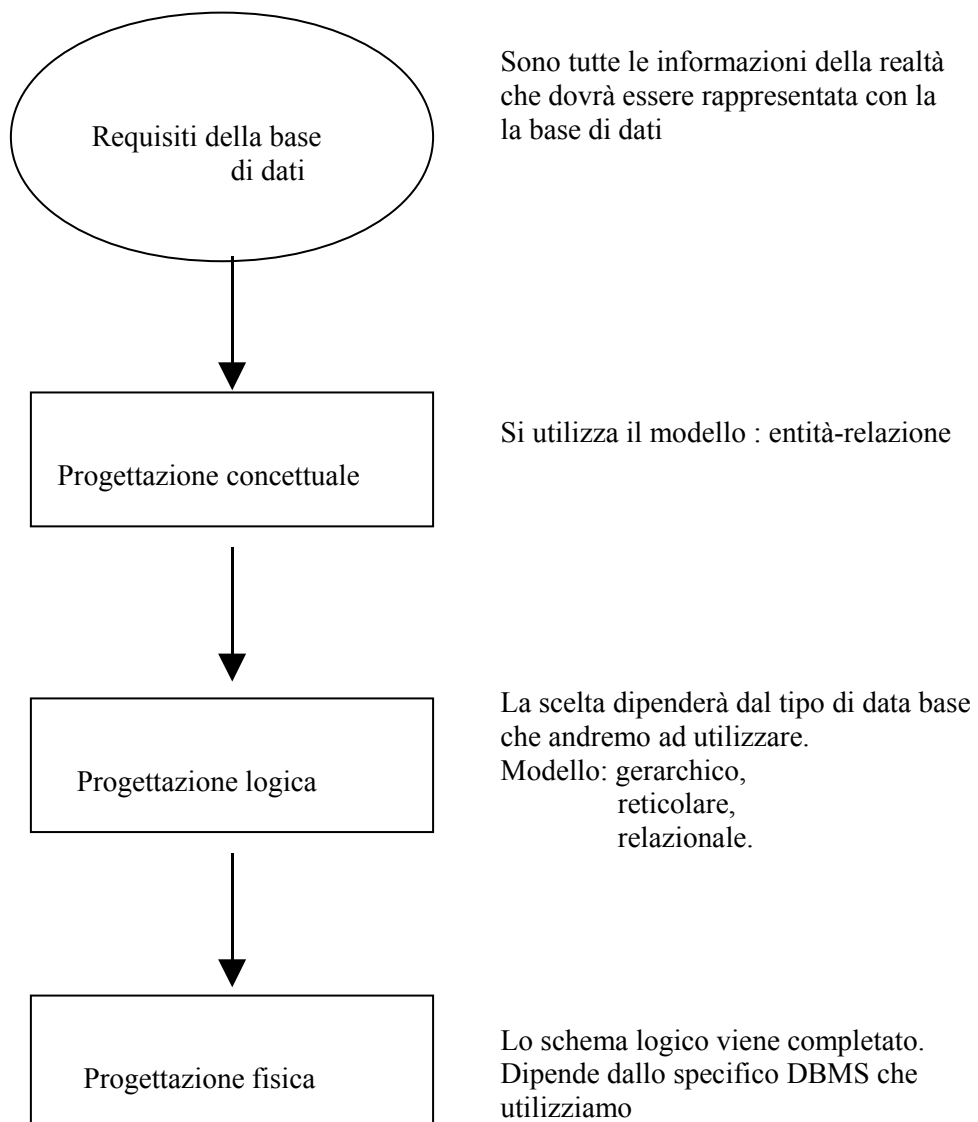
Vediamo ora di comprendere più a fondo cosa significa *modello* e quali sono gli elementi che ne fanno parte.

Definizione di modello:

Un modello è una rappresentazione della realtà, descritta mediante un apposito formalismo.

Una volta che si è avuta percezione della realtà, si procede alla sua descrizione, omettendo quei particolari che non interessano al fine del modello stesso e utilizzando invece quelle strutture che il modello mette a disposizione.

I vari tipi di modelli servono per semplificare la progettazione del data base, vediamo quali sono e quando si usano nelle varie fasi:


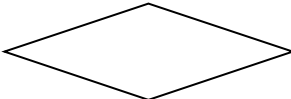
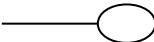
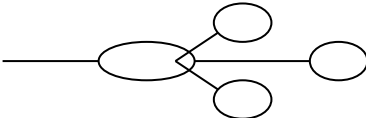
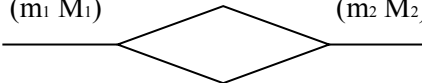
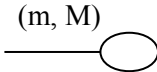
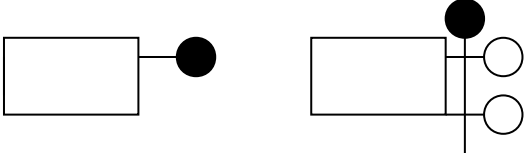
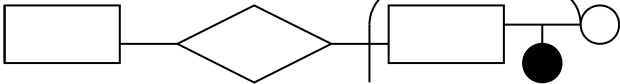




Dunque utilizzeremo in genere solo due tipi di modello nella progettazione: il modello entità-relazione che è un modello concettuale e quindi utilizzabile per qualsiasi tipo di data base e uno dei tre modelli logici scelto in base al tipo di data base che si utilizzerà.

Il modello entità-relazione:

Il modello entità-relazione E-R è un modello concettuale di dati e fornisce una serie di strutture, dette *costrutti*, atte a descrivere la realtà di interesse in una maniera facile da comprendere e che prescinde dai criteri di organizzazione dei dati negli elaboratori.

Nella figura sottostante vengono elencati tutti i costrutti che il modello E-R mette a disposizione

Costrutti	Rappresentazione grafica
Entità	
Relazione	
Attributo semplice	
Attributo composto	
Cardinalità di relazione	
Cardinalità di attributo	
Identificatore interno	
Identificatore esterno	
Generalizzazione	
Sottoinsieme	

Entità:

È un oggetto esistente nel mondo reale che si vuole rappresentare nel modello concettuale. *Libro* e *Utente* sono esempi di entità di una applicazione per la gestione di una biblioteca. Il Sig. Mario Rossi (in carne ed ossa) è un esempio di una **occorrenza** dell'entità *Utente*.



Relazioni:

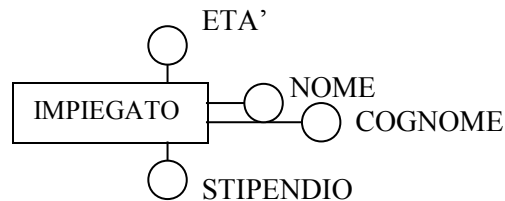
Rappresentano legami logici, significativi per l'applicazione, tra due o più entità. Assegnamento è un esempio di relazione che può sussistere tra le entità Impiegato e Incarico.



Attributo:

Descrivono proprietà elementari di entità o relazioni di interesse ai fini dell'applicazione. Per esempio, Cognome, Stipendio, Età sono possibili attributi dell'entità Impiegato.

Un attributo associa a ciascuna occorrenza di entità o di relazione un valore appartenente a un insieme, detto **dominio** dell'attributo. Per esempio, l'attributo Età dell'entità Impiegato può avere un dominio di interi compreso tra 18 e 65.



Cardinalità delle relazioni:

Vengono specificate per ciascuna entità che partecipa a una relazione e descrivono il numero minimo e massimo di occorrenze di relazione a cui le occorrenze delle entità coinvolte possono partecipare.

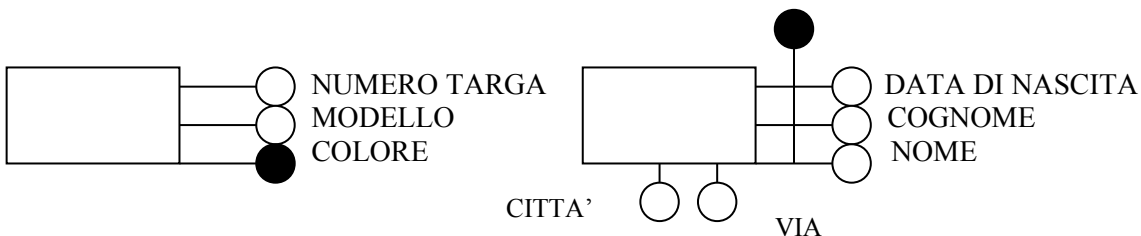
Cardinalità degli attributi:

Possono essere specificate per gli attributi di entità o relazioni e descrivono il numero minimo o massimo di valori dell'attributo associati ad ogni occorrenza di entità o relazione.

Nella maggior parte dei casi, la cardinalità di un attributo è pari a (1,1) e viene omessa.

Identificatori delle entità:

Permettono di identificare in maniera univoca le occorrenze delle entità. In molti casi, uno o più attributi di entità sono sufficienti a individuare un identificatore: si parla in questo caso di un identificatore interno (detto anche chiave). Per esempio, un identificatore interno per l'entità Automobile con attributi Modello, Targa e colore è l'attributo Targa. Alla stessa maniera, un identificatore interno per l'entità Persona con attributi Nome, Cognome, Indirizzo e data di Nascita può essere l'insieme degli attributi Nome, Cognome e Data di Nascita, avendo assunto che nella nostra applicazione non esistono due persone aventi, contemporaneamente, lo stesso nome, lo stesso cognome e la stessa data di nascita.

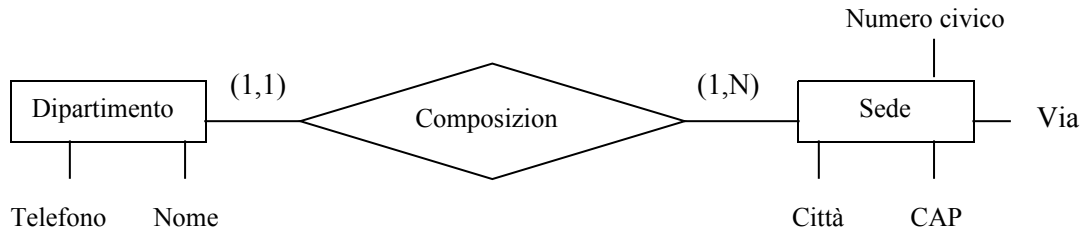


Identificatore esterno:

Dove l'identificazione di una entità è ottenuta utilizzando altre entità.

È il caso in cui ad esempio l'entità Dipartimento utilizza come identificativo oltre al suo attributo Nome anche l'attributo Città dell'entità Sede.

Per ogni città esiste un solo dipartimento; un dipartimento fa capo a più sedi; possono esistere dipartimento con nome uguale.

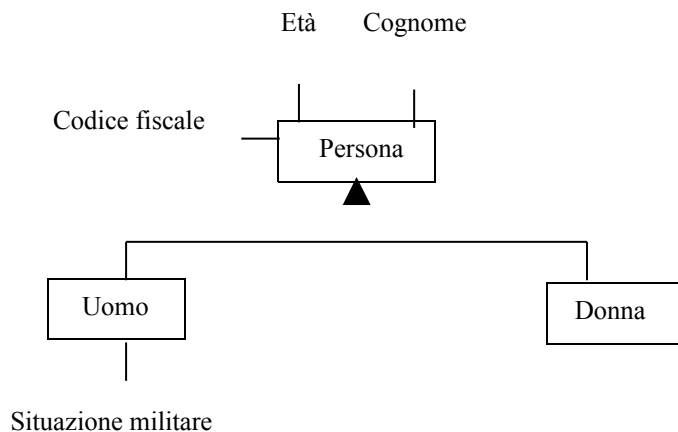


Generalizzazione:

Rappresenta legami logici tra una entità E , detta entità *padre*, e una o più entità E_1, \dots, E_n , dette entità *figlie*.

Si dice in questo caso che E è *generalizzazione* di E_1, \dots, E_n e che le entità E_1, \dots, E_n sono *specializzazioni* dell'entità E .

Per esempio, l'entità Persona è una generalizzazione delle entità Uomo e Donna e le entità Uomo e Donna sono specializzazioni dell'entità Persona.



Abbiamo visto come il modello E-R è il modello concettuale per eccellenza: ciò significa che attraverso il simbolismo che gli è proprio, è possibile rappresentare la struttura di qualunque base di dati espressa sotto forma di entità, relazioni, attributi.

Tutto ciò lo rende adatto a descrivere la realtà indipendentemente dall'ambiente nel quale il Data Base verrà poi effettivamente realizzato. Pertanto, il modello E-R presenta un livello di astrazione che:

- da un lato, si rivela indispensabile per adeguarsi alla molteplice realtà, tramite l'uso di un formalismo di valenza universale;
- dall'altro, richiede di un processo di esplicitazione per passare dal modello concettuale dei dati, ovvero quello realizzato tramite le strutture che gli sono proprie, al *modello logico*, che tiene conto invece dell'ambiente operativo nel quale il data base verrà implementato.

Modelli logici

Il modello logico discende dal modello concettuale e disegna un'architettura che tiene conto delle strutture proprie di quel particolare tipo di data base. Ciò significa che è possibile realizzare diversi tipi di data base a partire da uno stesso modello concettuale.

Nel corso degli anni si sono sviluppati vari tipi di modelli logici, tutti possono essere ricondotti a tre categorie principali: *gerarchici*, *reticolari*, e *relazionali*.

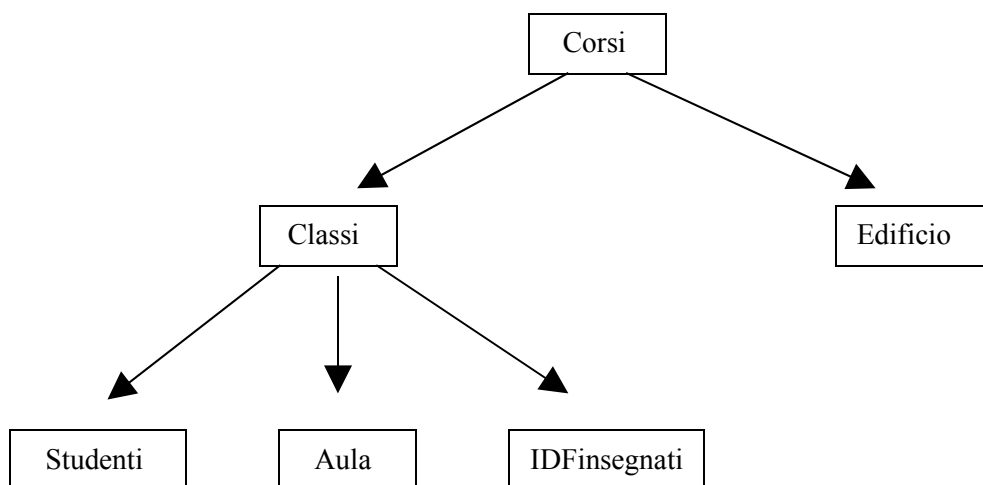
Modello gerarchico:

Volendo tracciare un percorso storico che attraverso l'evoluzione subita dai DBMS nel corso degli anni, è necessario iniziare la trattazione a partire dal *modello gerarchico*. Si può fissare la data di nascita di questo modello alla fine degli anni '60, quando IBM sviluppa e introduce sul mercato IMS, il primo data base gerarchico, ma anche il primo DBMS in assoluto.

Definizione:

Un **data base** gerarchico è un insieme di archivi. Gli archivi sono composti da record chiamati **segmenti**. I segmenti sono in rapporto gerarchico tra loro attraverso legami di tipo **padre-figlio**.

La struttura ad albero che caratterizza il modello gerarchico si basa sulla possibilità di individuare un *segmento principale*, il padre o la *radice*, dal quale dipendono *n segmenti figli*, che a loro volta si trasformano in padri per altri figli e così via. A questi, in virtù della totale dipendenza dal padre, è possibile fare riferimento solo attraverso il passaggio dal nodo principale. Non è possibile dal figlio risalire al padre.



Questa architettura mal si adatta ad una gestione moderna e dinamica delle basi di dati.

Modello reticolare:

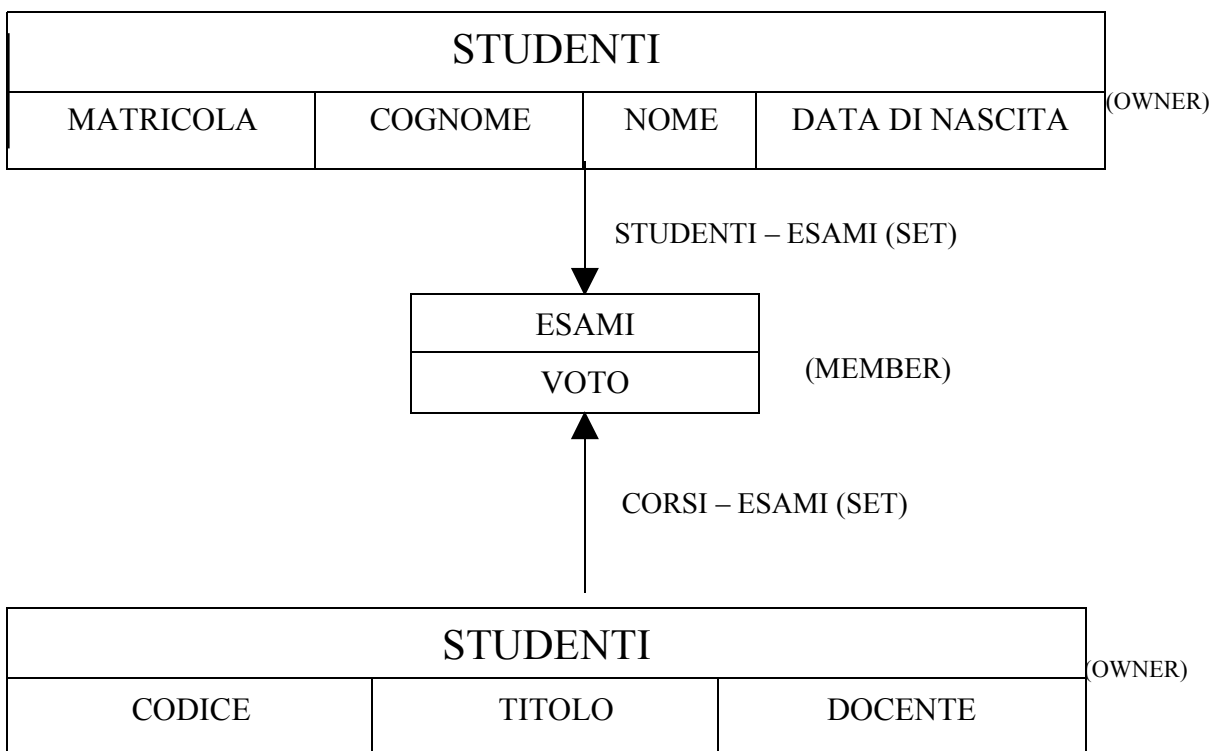
Il modello gerarchico rappresenta una prima soluzione al problema della gestione di grosse moli di dati ma la sua intrinseca rigidità ne limita la potenzialità; per questo, nasce il *modello reticolare* che dotato di maggiore flessibilità, può adattarsi a situazioni più complesse.

Il modello reticolare può essere visto come un'estensione del modello gerarchico, al quale siano apportati importanti miglioramenti :

In una struttura gerarchica un segmento figlio può avere solo un segmento padre; non è così nel modello reticolare: ogni record può avere un numero qualsiasi di record subordinati e di record precedenti e le correlazioni vengono espresse attraverso record particolari, chiamati *record di collegamento* (member), che formano delle catene tra le varie parti del sistema.

La struttura:

le strutture utilizzate nel modello reticolare sono due, il *record* (si pensi ai comuni file) e il *set*, che permette di correlare i record, per mezzo di catene di puntatori. Dunque una base di dati reticolare è definita con riferimento ad uno schema, che contiene tipi record collegati fra loro da tipo set.



Il modello relazionale

Il modello relazionale, fu proposto per la prima volta, nel 1970. Fin da allora ha avuto un crescente successo, dovuto principalmente alla sua semplicità e alla sua flessibilità.

Su cosa si basa il modello relazionale:

Il modello *relazionale* si basa su due concetti *relazione* e *tabella*. La nozione di relazione proviene dalla teoria degli insiemi, mentre il concetto di tabella è semplice ed intuitivo.

Dati due insiemi D1 e D2 si chiama *prodotto cartesiano* di D1 e D2, l'insieme delle coppie ordinate $(v1, v2)$, tali che $v1$ è un elemento di D1 e $v2$ è un elemento di D2.

Una *relazione matematica* sugli insiemi D1 e D2 è un sottoinsieme del prodotto cartesiano di D1 x D2.

Esempio:

Dati gli insiemi $A=\{1, 2, 4\}$ e $B=\{a, b\}$

moltiplicando $A \times B$

si ottiene il prodotto cartesiano $\{(1, a), (1, b), (2, a), (2, b), (4, a), (4, b)\}$

una possibile relazione matematica su A e B è costituita dall'insieme di coppie $\{(1, a), (1, b), (4, b)\}$

1	a
1	a
a	b

Le relazioni possono esse rappresentate graficamente, in maniera utilmente espressiva, sotto forma tabellare.

Abbiamo visto dunque il concetto di relazione e la rappresentazione tabellare, facciamo ora qualche esempio:

JUVENTUS	LAZIO	3	1
ROMA	MILAN	2	0
FIorentINA	NAPOLI	0	1

La relazione della pagina precedente è un sottoinsieme del prodotto cartesiano :

Stringa x Stringa x Intero x Intero

L'ordine con cui compaiono i dati all'interno delle righe è fondamentale, per il concetto stesso di relazione. Infatti ci accorgiamo che cambiando l'ordine di alcune colonne i risultati delle partite vengono stravolti.

A tutto ciò possiamo ovviare utilizzando gli **attributi** come intestazioni per le colonne.

Squadra di casa	Squadra ospitata	Reti casa	Reti ospitata
Juventus	Lazio	3	1
Roma	Milan	2	0
Fiorentina	Napoli	0	1

Modificando la definizione di relazione con l'introduzione degli attributi, possiamo vedere che l'ordinamento delle colonne risulta irrilevante.

Definizione di tupla:

Una tupla su un insieme di attributi X (x rappresenta l'insieme di attributi della relazione) è una funzione t che associa a ciascun attributo A di X un valore del dominio $DOM(A)$

Valori nulli:

La struttura del modello relazionale è molto semplice e potente. Al tempo stesso, essa impone però un certo grado di rigidità. In ogni relazione possiamo rappresentare solo tuple corrispondenti allo schema della relazione stessa. In effetti in molti casi, i dati disponibili possono non corrispondere esattamente al formato previsto. Ad esempio:

Persone(Cognome, Nome, Indirizzo, Telefono)

Il valore dell'attributo Telefono potrebbe non essere disponibile per tutte le tuple. Vale la pena notare che non sarebbe corretto utilizzare un valore del dominio per rappresentare l'assenza di informazione, in quanto in tal modo si potrebbe ingenerare confusione.

Per rappresentare in modo semplice la non disponibilità di valori, viene assunto un particolare valore per quel attributo detto *valore nullo*.

In genere quando si definisce una relazione bisogna specificare quali saranno gli attributi in cui è possibile inserire il valore nullo.

Vincoli di integrità:

In una base di dati, è opportuno evitare il più possibile che vengano inseriti dati privi di senso o sbagliati. Per limitare che ciò accada, durante la creazione del data base vengono stabiliti dei vincoli.

Un **vincolo** è un predicato che associa ad ogni istanza il valore vero o falso. Se il predicato assume il valore vero diciamo che l'istanza soddisfa il vincolo. In generale, ad uno schema di base di dati associamo un insieme di vincoli e consideriamo corrette o lecite le istanze che soddisfano tutti i vincoli.

Esempi di vincoli di integrità:

Non permettere l'inserimento del codice di avviamento postale usando lettere invece di cifre numeriche o non accettare che vengano inseriti più di 5 cifre.

Non consentire la cancellazione di un capovendite senza che i suoi subalterni (venditori) siano gerarchizzati ad altro capovendita o cancellati precedentemente.

Concetto di chiave:

Si definisce **chiave candidata** l'attributo o l'insieme di attributi che permettono di individuare univocamente la tupla all'interno della relazione.

Si definisce **chiave primaria** la chiave con il minor numero di attributi.

Si definisce **chiave esterna** l'attributo, o l'insieme di attributi, che può essere usata come chiave primaria per un'altra relazione.

AUTO

Targa	Proprietario	Indirizzo
RM 1A2396	Verdi Piero	Via Tigli
FI 1A2300	Verdi Piero	Via Tigli
TV 2F4560	Bianchi Antonio	Via Tigli

Possono essere considerate chiavi candidate tutti e tre gli attributi insieme della tabella Auto, o gli attributi Targa e Proprietario insieme. Non possono essere considerati chiave candidata gli attributi Proprietario e Indirizzo, sia da soli che insieme. È decisamente chiave primaria l'attributo Targa. L'attributo Proprietario comparando come chiave primaria nella relazione Proprietari Auto (Persona) è chiave esterna.

PROPRIETARI AUTO

Persona	Indirizzo
Verdi Piero	Via Tigli
Bianchi Antonio	Via Tigli

Vincolo di integrità referenziale:

Si ha un vincolo di integrità referenziale quando uno o più attributi di una relazione $R1$ compaiano come valori della chiave primaria in un'altra relazione $R2$

Esempio:

CANI

ID_Cane	Nome	Razza	Proprietario
56	Fido	Setter	134
93	Birillo	Meticcio	356
21	Diana	Cocker	356

PROPRIETARI

ID_Proprietari	Nome	Cognome	Via	Città
356	Carlo	Rossi		
134	Mario	Bianchi		
200	Gianni	Verdi		

Nelle due tabelle abbiamo rispettato il vincolo di integrità in quanto le tutte le tuple sono correlate tra di loro. Tutti i cani hanno un proprietario.

Tipi di associazioni tra tabelle:

Bisogna subito specificare che il termine *relazione* nel modello relazionale assume due significati diversi: il termine relazione che abbiamo considerato fino ad ora nasce dalla teoria degli insiemi e sta ad indicare una tabella; il termine relazione che andremo ora a considerare sta per sinonimo di *associazione* e sta a indicare un rapporto di corrispondenza tra informazioni di due tabelle. Andiamo a vedere che tipi di relazioni (associazioni) possiamo avere:

Relazioni uno a molti:

Una relazione uno a molti è il tipo più comune di relazione. In una relazione uno a molti un record della tabella A può avere molti record corrispondenti nella tabella B, ma un record della tabella B non ha più di un record corrispondente nella tabella A.

NORMALIZZAZIONE

Per eliminare definitivamente la ridondanza dalle relazioni e quindi dal data base, si ricorre ad un processo di normalizzazione: si tratta di un procedimento di tipo graduale, che realizza un'ottimizzazione progressiva a partire da relazioni non normalizzate fino a raggiungere un certo livello di normalizzazione.

Prima di entrare in merito alla normalizzazione è bene introdurre alcuni particolari concetti:

Definizione di ridondanza:

Si ha ridondanza dei dati ogni volta che vengono memorizzati inutilmente dei dati ripetuti.

Esempio:

Si supponga di disporre di una tabella in cui vengono registrati tutti gli acquisti di merci effettuati da un'impresa mercantile.

Acquisti(Cod_fornitore, Nome_fornitore, Codice_fiscale, Merce, Quantità)

In un caso come questo i dati di Fornitore e Codice_fiscale vengono ripetuti inutilmente ogni volta che compare la registrazione di un acquisto. Si può ovviare ad una tale problema separando le informazioni ridondanti della tabella acquisti:

Acquisti(Merce, Cod_fornitore, Quantità)

Fornitori (Cod_fornitore, Nome_fornitore, Codice_fiscale)

In questo modo se gli acquisti effettuati da un certo fornitore compaiono mille volte, non dovremo per mille volte ripetere inutilmente l'informazione Nome_fornitore, Codice_fiscale.

Dipendenza funzionale:

Si ha dipendenza funzionale tra un attributo x e una chiave se i valori di x dipendono dai valori assunti dalla chiave, ovvero se si viene a determinare una relazione biunivoca tra i valori di x e i valori della chiave.

Esempio:

Stabiliamo che nella relazione di esempio abbiamo come chiave primaria Nome e Cognome.

ANAGRAFICA (Nome, Cognome, Indirizzo)

È evidente che Indirizzo è dipendente funzionalmente dalla chiave, ciò significa che per ogni persona corrisponde un indirizzo.

Dipendenza funzionale transitiva:

Si verifica una dipendenza funzionale transitiva quando un attributo J appartenente alla relazione r dipende da un attributo K della relazione che non è chiave candidata e che a sua volta dipende da un attributo A , che è chiave candidata o primaria.

Data la relazione: $r(A, B, K, J)$ con A chiave primaria o candidata e avendo che: (il simbolo ' \rightarrow ' significa "determina il valore di"): $A \rightarrow K$ e $K \rightarrow J$ si verifica una dipendenza transitiva tra A e J, in quanto J dipende da A tramite K.

Esempio:

Nella seguente relazione possiamo renderci conto come il valore di Collocazione dipenda da Genere e come il valore di Genere dipenda da Identificativo_videocassetta:

VIDEOCASSETTE (Identificativo_videocassetta, Genere, Collocazione)

Identificativo_videocassetta è la chiave primaria, essa identificando la videocassetta determina il Genere; le videocassette vengono collocate raggruppandole per genere.

Prima forma normale:

Una relazione si dice in prima forma normale (1NF) se e solo se tutti i suoi attributi sono valori atomici.

Ciò implica che né gli attributi, né i valori da questi assunti possono essere scomposti ulteriormente.

La relazione che segue non è in prima forma normale:

NOME	INDIRIZZO
Rossi & C. s.p.a.	Via Verdi, 5 Roma
Martini s.n.c.	Via Centrale, 4 Milano
Paoli s.a.s.	P.za Libertà, 12 Pisa
.....

in quanto l'attributo INDIRIZZO non è atomico: può essere suddiviso in Indirizzo e Città. La relazione si normalizza trasformandola in:

NOME	INDIRIZZO	CITTA
Rossi & C. s.p.a.	Via Verdi, 5	Roma
Martini s.n.c.	Via Centrale, 4	Milano
Paoli s.a.s.	P.za Libertà, 12	Pisa
.....

Seconda forma normale:

Una relazione è in seconda forma normale se e solo se soddisfa la 1NF e inoltre ciascun attributo che non fa parte della chiave è indipendente funzionalmente da una qualunque chiave candidata.

Per normalizzare la relazione è necessario estrarre la dipendenza funzionale riscontrata tra un sottoinsieme della chiave e l'attributo in questione, che darà luogo ad un'altra relazione.

Se applichiamo quanto detto alla relazione $r(A, B, C, V, Z)$ con ABC chiave primaria, V dipende funzionale da ABC e Z dipende funzionale da AB, per normalizzare la relazione dobbiamo realizzare due nuove relazioni come segue:

$$r'(A, B, C, V) \text{ e } r''(A, B, Z)$$

Esempio:

Abbiamo una relazione che rappresenta delle scrivanie da ufficio prodotte da una azienda industriale, la chiave primaria è data Tipo_tavolo e Tipo_legno.

Tipo_tavolo	Tipo_legno	Tipo_finitura
Manager	Mogano	Ottone
Manager	Ciliegio	Acciaio inox
Montecarlo	Noce	Sughero
Montecarlo	Mogano	Ottone
Top	Noce	Sughero

Tipo_finitura è dipendente funzionalmente da un sottoinsieme della chiave primaria, ovvero da Tipo_legno, dunque la tabella non è in 2NF.

È possibile normalizzare la relazione trasformandola in:

Tipo_tavolo	Tipo_legno
Manager	Mogano
Manager	Ciliegio
Montecarlo	Noce
Montecarlo	Mogano
Top	Noce

Tipo_legno	Tipo_finitura
Mogano	Ottone
Ciliegio	Acciaio inox
Noce	Sughero

Terza forma normale:

La terza forma normale ha come scopo l'eliminazione dalla relazione delle dipendenze funzionali transitive.

Una relazione è in terza forma normale (3NF) se, oltre ad essere in 2NF, ciascun attributo che non partecipa alla chiave non è transitivamente dipendente da una qualunque chiave candidata.

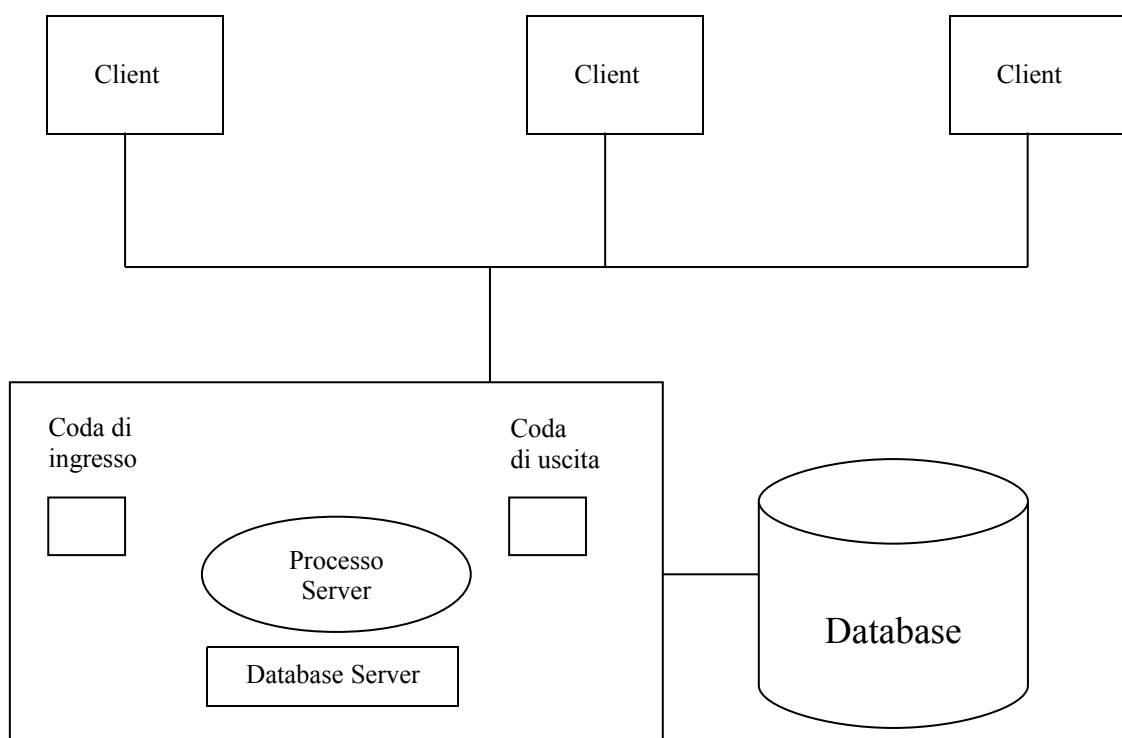
INTRODUZIONE ARCHITETTURA CLIENT SERVER

Il paradigma client server è un modello di iterazione tra processi software, ove processi interagenti si suddividono tra *client*, che richiedono servizi, e *server*, che offrono servizi.

Il processo client è tipicamente dedicato ad interagire con l'utente finale; esso svolge un ruolo attivo, in quanto genera autonomamente richieste di servizi. Invece, il processo server è reattivo: esso svolge una computazione solo a seguito di una richiesta da parte di un qualunque client. Normalmente, un processo client può richiedere in sequenza alcuni servizi a vari processi client.

Non è necessario che i processi server e client siano allocati su macchine diverse: la distinzione fra i processi client e server è un ottimo paradigma per la costruzione del software indipendentemente dalla allocazione dei processi.

- Le funzioni di client e server sono ben identificate nel contesto delle basi di dati. Esse corrispondono a una decomposizione ideale delle competenze e professionalità: il programmatore applicativo ha la responsabilità di gestire il software relativo al client rispondendo a esigenze specifiche, mentre l'amministratore della base di dati ha la responsabilità sul server che è condiviso da varie applicazioni, e deve organizzare la base di dati così che essa garantisca prestazioni ottimali a tutti i processi client.
- Oltre alla decomposizione funzionale dei processi e dei compiti, nelle basi di dati l'utilizzo di elaboratori diversi per client e server è particolarmente conveniente. L'elaboratore dedicato al client deve essere adatto alla interazione con l'utente; si usa spesso un personal computer, dotato di strumenti di produttività (posta elettronica, editor, ecc.) che sono tipici dell'automazione d'ufficio. Tra questi strumenti, spesso mascherate da una interfaccia amichevole, vi sono le applicazioni che richiedono l'uso della base di dati. L'elaboratore dedicato al server è dimensionato in funzione dei servizi che deve offrire.



L'organizzazione, illustrata nella figura della pagina precedente ci mostra nel server al presenza di due code, in una vengono accodate le richieste dei client, nell'altra vengono accodati i risultati delle richieste.

Il processo server monitorando la coda di ingresso svolge i servizi richiesti dai client.

INTRODUZIONE ARCHITETTURA DISTRIBUITA

Abbiamo visto che in una architettura client server una transazione (una operazione sul data base) coinvolge al più un server. Quando viceversa le transazioni coinvolgono più server, parliamo di basi di dati distribuita.

Le motivazioni che portano allo sviluppo di soluzioni distribuite nella gestione dei dati sono di tipo pragmatico; esse rispondono alla esigenza di adeguare la gestione dell'impresa, che è strutturalmente distribuita.

La gestione distribuita dei dati si contrappone ad una gestione centralizzata tipica dei grandi centri di calcolo, dominante fino alla metà degli anni ottanta.

Un data base distribuito è un in realtà, un data base virtuale le cui componenti (reali) risiedono in un sistema distribuito, ovvero sono data base fisicamente indipendenti ma correlati, che si trovano su nodi connessi tramite rete. Ogni utente può accedere ai dati del data base indipendentemente dalla loro collocazione fisica effettiva, come se questi fossero locali alla propria macchina. Anzi non è neppure tenuto a sapere se le informazioni che richiede arrivano dal proprio hard disk o sono inviate, tramite rete, da un sistema remoto. Così come, nel caso di query complesse, può ignorare se il processo di elaborazione è avvenuto sul suo computer (quindi usando la sua CPU, la sua memoria centrale, oltre ai suoi dati) o su altro nodo del sistema.

Note legali:

1. Queste dispense sono liberamente utilizzabili e pubblicabili sul web, a patto che:
 - a) Non vengano rimosse queste note legali.
 - b) Non venga rimosso il link al sito web della Art Net.
2. Non è permessa la pubblicazione su carta (riviste, libri, stampa, ecc.), salvo esplicita autorizzazione.
3. La Art Net (studio di consulenza informatica che si occupa di: sviluppo software conto terzi o per clienti finali, di outsourcing e body rental, di docenze e corsi d'informatica, di creazione siti web, di sviluppo applicativi gestionali tradizionali o web based) detiene tutti i diritti di copyright.
4. Nel caso di uso di questo materiale, non conforme a queste note, la Art Net perseguirà civilmente e penalmente i trasgressori.
5. Sebbene questi appunti siano stati redatti con coscienza, cura e buona fede, la Art Net non può assumersi alcuna responsabilità circa l'attendibilità del loro contenuto.

Art Net di Arrigoni Roberto via Giovanni XXIII, 4 Civita Castellana (Viterbo) P.I. 01598360566
fax 0761 51.51.11