INDICE

Premessa

Il termine informatica deriva dalla contrazione delle parole INFORmazione autoMATICA. L'attuale linea di sviluppo che caratterizza l'informatica e ne determina le applicazioni, è particolarmente attenta alle tematiche concernenti l'archiviazione e la gestione delle informazioni.
La progettazione di basi di dati rigorose, il cui contenuto sia facilmente oggetto di elaborazione, rappresenta un punto di snodo per le strategie aziendali e per le organizzazioni in generale.
Ogni azione di formazione, informazione e istruzione prevede flussi di dati che si spostano, si trasformano e s'influenzano in modo reciproco.
È centrale in questa dinamica comunicativa:

- definire schemi logici per l'organizzazione dei dati;
- definire linguaggi dedicati alla gestione e alla manipolazione dei dati e delle relative strutture;
- definire procedure automatiche per il trattamento dei dati;
- progettare, realizzare ed aggiornare sistemi per la condivisione di dati in rete (con particolare attenzione ad Internet e alle Intranet)

In questo quadro di riferimento, la progettazione di sistemi complessi per l'archiviazione e la gestione dei dati è un'applicazione di tipo orizzontale sulla quale si innestano applicazioni verticali relative alla comunicazione e alla trasmissione delle informazioni.
Diventa importante saper analizzare, gestire ed indirizzare i flussi di informazione in un'ottica di supporto alle procedure di "*Decision Making*".
È con queste linee guida che i modelli di data base saranno descritti:

- modelli di data base;
- linguaggi per la gestione e l'interrogazione delle basi di dati;
- utilizzo e concetti base dei più diffusi linguaggi per la gestione dei dati.

1. Introduzione ai data base

1.1 I dati e le informazioni

Necessità primarie delle organizzazioni e delle strutture aziendali sono l'archiviazione, l'elaborazione, l'estrazione e la manutenzione di dati.
I dati rappresentano un patrimonio che caratterizza la natura stessa delle organizzazioni e che ne influenza le scelte e le strategie.
Le tipologie di documento da cui i dati, oggetto di archiviazione ed elaborazione, sono estratti sono essenzialmente di tre tipi:

- cartacei, ad esempio moduli d'ordine, schede bibliografiche e fatture;
- "analogici", ad esempio telefonate o relazioni;
- multimediali, ad esempio filmati, registrazioni, progetti tecnici ed illustrazioni.

Le basi dati delle organizzazioni sono popolate con i dati estratti con processi di elaborazione da questi tre tipi di documenti. Un esempio di processo di estrazione dati è l'estrapolazione di dati destrutturati da dati strutturati. Per esempio, da una fattura, dato strutturato, si possono estrarre i dati destrutturati: importo, quantità, descrizione dei prodotti, aliquote IVA. Ai dati ottenuti con queste procedure possono essere applicati processi di elaborazione e di trasformazione dei dati stessi.
L'insieme delle azioni e dei processi che concorrono alla definizione della base dati aziendale, unitamente alle risorse impiegate, é ciò che si definisce il *sistema informativo*.
La crescente importanza che il sistema informativo ricopre nella vita delle organizzazioni e la necessità di gestire grandi masse di dati in tempo reale, ha incentivato lo sviluppo di tecniche e tecnologie per l'archiviazione e la elaborazione delle informazioni. In particolare ha fatto sì che si definissero strutture di archiviazione sempre più efficienti e flessibili, unitamente a strumenti informatici che le rendessero facilmente utilizzabili da operatori non specializzati.
I primi passi mossi in questo campo hanno visto uno sviluppo settoriale per quanto concerne le applicazioni aziendali; in pratica ogni settore, dalla contabilità alla gestione del personale così come biblioteche e archivi documentali, ha richiesto lo sviluppo di software utilizzando linguaggi di alto livello e definendo strutture di archivi particolarmente utili in campi di applicazione specifici.
In questa fase di iniziale dell'implementazione di sistemi informativi complessi, per quanto riguarda ad esempio le applicazioni gestionali, é stato ampiamente utilizzato il linguaggio COBOL[1].
Con lo sviluppo delle tecnologie e la necessità di integrazione di dati disomogenei per tipo e formato, non solo tra diversi settori di una stessa organizzazione, ma tra settori di organizzazioni diverse, unitamente alla necessità di archiviare ed elaborare dati di diversa origine, si sono evidenziati numerosi problemi di questo modo di interpretare la gestione dei dati.
Tra i problemi emersi si possono individuare come principali:

- la duplicazione di dati in più archivi indipendenti;
- la forte dipendenza tra programmi di gestione e modalità di archiviazione e gestione dei dati.

Duplicazione degli archivi
Lo sviluppo di applicazioni settoriali all'interno di organizzazioni tra loro indipendenti aveva reso necessario la disponibilità del medesimo dato elementare in più settori della

[1] **CO***mmon* **B***usiness* **O***riented* **L***anguage*, linguaggio per l'implementazione di applicazioni gestionali, definito ed implementato tra gli anni '50 e '60 in ambienti militari statunitensi. Il COBOL è utilizzato principalmente su Mini e Mainframe, e dispone anche di versioni per PC.

medesima organizzazione, producendo così una ridondanza di dati, laddove non aveva richiesto la duplicazione di dati o di interi archivi da utilizzare in fasi diverse del flusso informativo.
Tale sviluppo rendeva particolarmente oneroso qualsiasi processo di aggiornamento sia dei dati sia delle procedure che li gestivano. L'architettura che scaturiva da questo sviluppo delle modalità di archiviazione non dava garanzie in relazione al fatto che il dato o le procedure di elaborazione fossero aggiornate in maniera omogenea in ogni processo del sistema informativo che le vedeva coinvolte.
Questo diminuiva il valore dell'informazione e l'efficacia dello stesso sistema informativo.

Dipendenza di programmi e dati
La dipendenza dei dati dai programmi che li gestivano si esplicitava particolarmente dove erano utilizzati linguaggi di alto livello, nel fatto che il dato non era disponibile come entità autonoma rispetto al programma che lo gestiva. Inoltre non era stata definita una netta separazione tra la struttura logica di gestione dei dati e la struttura fisica di memorizzazione dei dati.
Il riscontro delle serie problematiche che questo tipo di approccio induceva, era chiaro nel momento in cui era necessario un aggiornamento o una modifica dell'intero sistema informativo che si mostrava così scarsamente flessibile se non estremamente macchinoso.

Da queste problematiche hanno preso avvio una riflessione ed una ricerca approfondita nel campo della tecnologia delle **basi dati**, che ha portato alla definizione di un processo di risoluzione di problemi inerenti ad essa.
I punti di forza su cui la tecnologia delle basi dati ha fondato il suo sviluppo possono essere riassunti attraverso tre concetti:

- dati memorizzati in archivi condivisi;
- software per la gestione degli archivi;
- software di gestione di accessi multipli alle basi dati.

La memorizzazione in archivi condivisi fa sì che il valore di ogni singolo dato sia definito in funzione della sua essenzialità e della possibilità di essere aggiornato in maniera univoca. In questo modo viene tracciata la via per la risoluzione del problema della ridondanza e quindi della consistenza[2] dei dati presenti negli archivi.
La generazione di software dedicati alla gestione degli archivi svincola i dati dai software che li gestiscono e permettono a più applicazioni di interrogare gli archivi e di utilizzare i dati in essi contenuti in maniera flessibile e standard.
L'implementazione di software per la gestione di accessi multipli agli archivi permette ai processi di archiviazione, elaborazione ed estrazione dati di lavorare contemporaneamente ed in particolare di interagire in tempo reale.
In pratica con quest'ultima caratteristica, se l'ufficio amministrativo modifica la ragione sociale di un cliente, l'ufficio vendite, che deve emettere una fattura, dispone immediatamente della nuova dicitura.
Le basi dati con queste linee di sviluppo hanno colmato gli svantaggi derivanti dalle precedenti forme di memorizzazione dei dati e impresso un forte impulso alla progettazione di software e sistemi di archiviazione, orientati secondo le linee fondamentali di questa tecnologia innovativa.
I sistemi di software per la gestione centralizzata dei dati sono chiamati **Data Base Management Systems** (**DBMS** - sistemi per la gestione dei dati).

[2] I dati si possono definire *consistenti* se ad ogni istante sono allineati e aggiornati in maniera simultanea.

Lo sviluppo dei DBMS ha come fondamento teorico il rapporto pubblicato nel 1971 dal Data Base Task Group, istituito negli Stati Uniti nel 1969 nell'ambito della "*Conference on Data Base System and Languages"*, con il compito specifico di affrontare il problema dell'organizzazione delle basi dati.
Il rapporto stilato alla fine dei lavori definiva, tra l'altro, alcuni standard che si ritenevano indispensabili per una base dati:

- **consistenza dei dati**: i dati non debbono essere ridondanti e comunque debbono essere aggiornati con un'unica operazione;
- **indipendenza tra struttura fisica e struttura logica dei dati**: deve essere possibile modificare la modalità fisica di registrazione dei dati, supporto e struttura, lasciandone inalterata la struttura logica;
- **integrazione dei dati**: deve essere possibile definire legami e relazioni tra i dati archiviati;
- **centralizzazione del controllo dei dati**: creazione della figura del gestore dei dati, che si deve occupare dell'organizzazione dei dati ed evitare che siano gli esperti del software ad interessarsene;
- **sicurezza dei dati**: creazione di un sistema di sicurezza che permetta di creare liste di utenti e liste di autorizzazione; in modo che sia possibile definire le operazioni che ogni utente può eseguire sui dati;
- **accessi multipli ai dati**: creazione di un sistema che permetta la connessione simultanea di più utenti e diverse applicazioni alla base dati.

1.2 Data Base Management Systems

Obiettivo dei DBMS é la definizione di un sistema ***centralizzato*** per la gestione dei dati delle organizzazioni e delle aziende.
L'analisi della parola "centralizzato" mette in evidenza quelle che sono le principali innovazioni che il DBMS ha introdotto e che sono riassumibili nei seguenti punti:

- controllo della ridondanza dei dati e consistenza dei dati;
- accessi simultanei ai dati;
- integrità dei dati;
- sicurezza delle informazioni;
- facile accesso ai dati per non esperti.

La scelta di un sistema centralizzato di gestione permette un'implementazione efficace ed efficiente di ciascuno dei punti caratterizzanti il DBMS.

Ridondanza e consistenza dei dati
Disporre di un archivio centralizzato, anche se formato da più tabelle, permette un controllo efficiente della ridondanza dei dati: in pratica, anche laddove fosse necessario duplicare dati elementari utilizzati da applicazioni diverse, il loro aggiornamento sarebbe garantito in maniera omogenea.
In questo modo si raggiunge anche lo scopo di mantenere la consistenza dei dati, i quali risulteranno, in ogni istante, allineati ed aggiornati in maniera simultanea, sicura e controllabile.

Accessi simultanei ai dati
La centralizzazione degli archivi implica il vantaggio di permettere l'accesso simultaneo a tutti i dati presenti negli archivi stessi. In particolare é possibile implementare e controllare funzioni e applicazioni che attingono agli archivi già presenti senza dovere creare copie o nuovi archivi che riaprirebbero nuove problematiche di ridondanza e consistenza dei dati.

L'accesso simultaneo ai dati in archivi centralizzati presenta l'indubbio vantaggio dell'aggiornamento e dell'elaborazione controllata sulle diverse applicazioni che operano sui dati archiviati, eliminando contemporaneamente il rischio di avere stati in cui gli archivi non risultino consistenti. In altri termini, con la gestione centralizzata degli archivi, si garantisce l'integrità dei dati.

Integrità dei dati
L'integrità dei dati é da mettere in relazione con lo stato in cui i dati degli archivi sono resi disponibili alle applicazioni. In particolare, l'integrità dei dati di un archivio é definibile come la capacità di quell'archivio di rendere disponibili dati consistenti alle applicazioni e agli utenti.
La centralizzazione degli archivi del DBMS permette la definizione di vincoli che possono riguardare la sequenza delle esecuzioni delle operazioni, gli stati in cui i dati sono resi disponibili oppure le operazioni di inserimento, aggiornamento, modifica e cancellazione dei dati.
Tali vincoli definiscono e gestiscono i possibili stati in cui l'archivio si può trovare e sono formalizzati attraverso funzioni preposte a:

- controllo di accesso;
- controllo di azione.

Sicurezza delle informazioni
La situazione di centralizzazione delle informazioni rappresenta un oggettivo valore per l'organizzazione che ne fa uso, poiché rende l'informazione disponibile in modo completo e in tempo reale per ogni settore dell'organizzazione, ma fa emergere il problema della sicurezza delle informazioni.
Il problema della sicurezza é di due livelli:

- **livello dell'integrità fisica dei dati**, minacciata da inconvenienti tecnici, che necessita di operazioni di copia degli archivi (operazione facilitata dalla centralizzazione degli archivi);
- **livello di sicurezza di accessi**: il DBMS è dotato di sistemi di permessi in grado di controllare gli accessi ai dati, di selezionare i tipi di dati accessibili in funzione dell'utente che ne fa richiesta (ad esempio un dipendente potrà vedere solo i propri dati anagrafici, mentre il responsabile dell'ufficio del personale potrà vedere i dati anagrafici di tutti i dipendenti), fino a selezionare solo alcuni dati all'interno di uno stesso archivio in funzione del profilo dell'applicativo o dell'utente che ne fa richiesta (una **vista** sui dati).

Facile accesso ai dati per i non esperti
Le problematiche finora analizzate prendono in esame l'organizzazione degli archivi e le funzioni di manipolazione dei dati ma non esaminano le problematiche relative agli utenti che devono accedere ai dati, tuttavia il DBMS, attraverso la centralizzazione di dati e procedure, è in grado di consentire la definizione di linguaggi non procedurali che permettono all'utente, anche inesperto nell'uso di tecnologie informatiche, di accedere direttamente ai sistemi di gestione dei dati.

I DBMS non sfruttano solo le grandi opportunità derivanti da una gestione centralizzata degli archivi ma tendono anche a sviluppare applicazioni altamente indipendenti dalle strutture dati, affinché la modifica della struttura di un archivio non comporti la modifica strutturale del software di gestione e, viceversa, la modifica o la sostituzione di software di gestione non richieda una migrazione dei dati in nuovi archivi.

Ciò che le funzioni dei DBMS si prefiggono é di configurarsi in funzione della struttura logica degli archivi per realizzare l'indipendenza fisica dei dati.

1.3 Rappresentazione dei dati e delle relazioni nei DBMS

Nell'ambito delle applicazioni dei DBMS parleremo di *entità* intendendo un elemento oggetto della memorizzazione. Ad esempio, nella progettazione della base dati di una biblioteca, un'entità può essere il libro.

Le entità e gli attributi
Le entità saranno descritte attraverso i loro **attributi** che possono assumere valori diversi in relazione alle diverse entità.
Ad esempio l'entità libro potrà avere, tra gli altri, l'attributo titolo che assumerà valori diversi per ogni entità ("I promessi sposi", "Il milione", ...).
Le definizioni dei concetti di entità ed attributo permettono di classificare i dati come:

- ***dati elementari**:* elementi non ulteriormente scomponibili (ad esempio il titolo di un libro) caratterizzati dal un nome e dal tipo di dati che contiene (tipo testo, tipo numerico, tipo logico ...);
- ***dati aggregati**:* insieme di dati elementari. I dati aggregati si definiscono:
  - *vettoriali* se sono composti da elementi omogenei (ad esempio i titoli dei libri di un determinato autore);
  - *ripetuti* se composti da un insieme di dati ripetuti un numero di volte non predeterminato (ad esempio l'elenco dei nominativi degli utenti di una biblioteca che hanno preso in prestito un determinato libro).

I livelli di rappresentazione
Un DBMS, da un punto di vista schematico, rappresenta una informazione strutturandola su tre livelli:

1. ***livello interno***
2. ***livello concettuale***
3. ***livello esterno***

Il **livello interno** descrive i dati elementari (campi) e la struttura dei record[3] di memorizzazione delle informazioni. Queste descrizioni sono formalizzate attraverso il *tracciato record*, cioè la descrizione analitica della struttura fisica della memorizzazione.
Tale descrizione é necessaria e sufficiente per il DBMS, in relazione alle operazioni di memorizzazione e manipolazione dei dati.
Se, ad esempio, si deve strutturare un archivio di libri, i dati elementari possono essere "Titolo", "Autore" e "Editore". Questi tre elementi compongono il record LIBRO.

Il **livello concettuale** formalizza la descrizione della logica che governa la struttura dati nel suo complesso, esplicita le entità che si vogliono memorizzare, le caratteristiche di ogni entità che si intende prendere in esame, le relazioni esistenti tra le entità che compongono la base dati ed il modello dei dati che si intende applicare.

Il **livello esterno**, infine, é la formalizzazione della descrizione delle viste logiche sui dati che si vogliono rendere disponibili per gli utenti e, parallelamente, le diverse tipologie di utenti previste, definendo per ciascuna tipologia i dati a cui possono accedere e quelli che possono manipolare.

[3] Lista di campi

L'insieme delle definizioni riguardanti ciascuno dei livelli sono raggruppate in insiemi detti **schemi** che, per il livello esterno, prendono il nome di **sottoschemi**, uno per ogni vista logica.

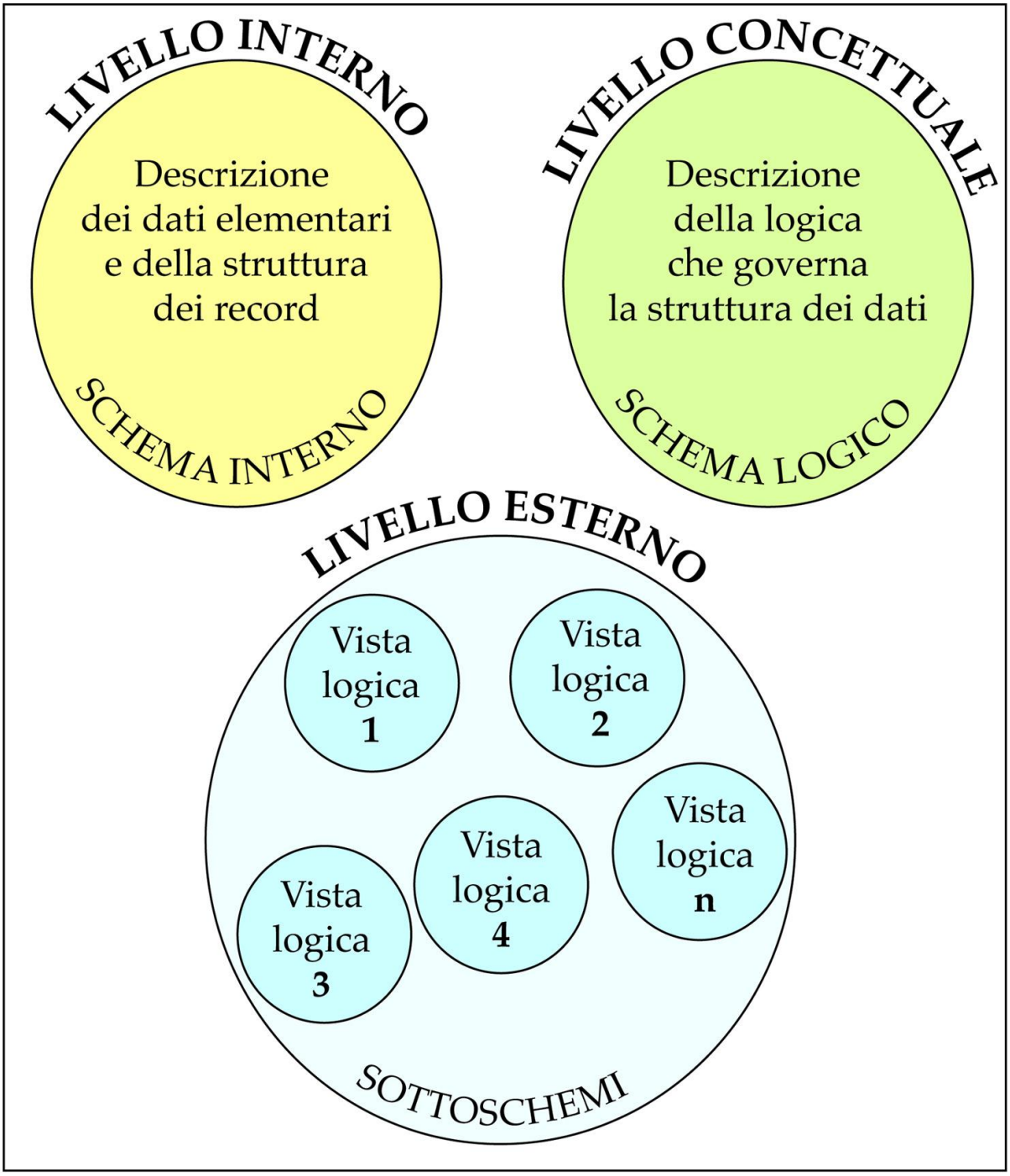

Figura 1. DBMS - Rappresentazione dell'informazione

A completamento di questo schema strutturale esistono le diverse interfacce, che permettono il passaggio ordinato tra i livelli degli schemi. Nell'analisi dei livelli é interessante rilevare che solo il livello concettuale é interessato alla modifica strutturale degli archivi. Per questo si può parlare, per i DBMS, d'indipendenza fisica dei dati.
Con questa architettura è possibile anche affermare che la definizione dei sottoschemi nel livello esterno definisce l'indipendenza logica dei dati. Costruendo opportune viste logiche

dell'archivio é possibile permettere ad un utente o ad un'applicazione di continuare a trattare i dati secondo una logica funzionale, senza per questo influire sulla struttura fisica dei dati. Per esempio, se anagrafica cliente e listino prodotti sono due archivi diversi é possibile definire un sottoschema che permetta all'impiegato di emettere una fattura considerando i dati anagrafici del cliente e i dati dei prodotti acquistati come provenienti da un unico archivio, anche se, nella realtà, la struttura fisica degli archivi é differente.

Lo schema logico legato al livello concettuale, definendo le entità e le loro relazioni, permette di fornire tutte le informazioni necessarie al DBMS per le operazioni che salvaguardano la consistenza e l'integrità dei dati.
Dall'unione delle caratteristiche generali del DBMS e dalla conseguente schematizzazione in livelli della rappresentazione dell'informazione, si ottiene uno schema generale, per i DBMS, composto da:

- archivi,
- dizionari (insieme degli schemi),
- software di gestione,
- funzioni di amministrazione,
- accessi (utenti e applicativi).

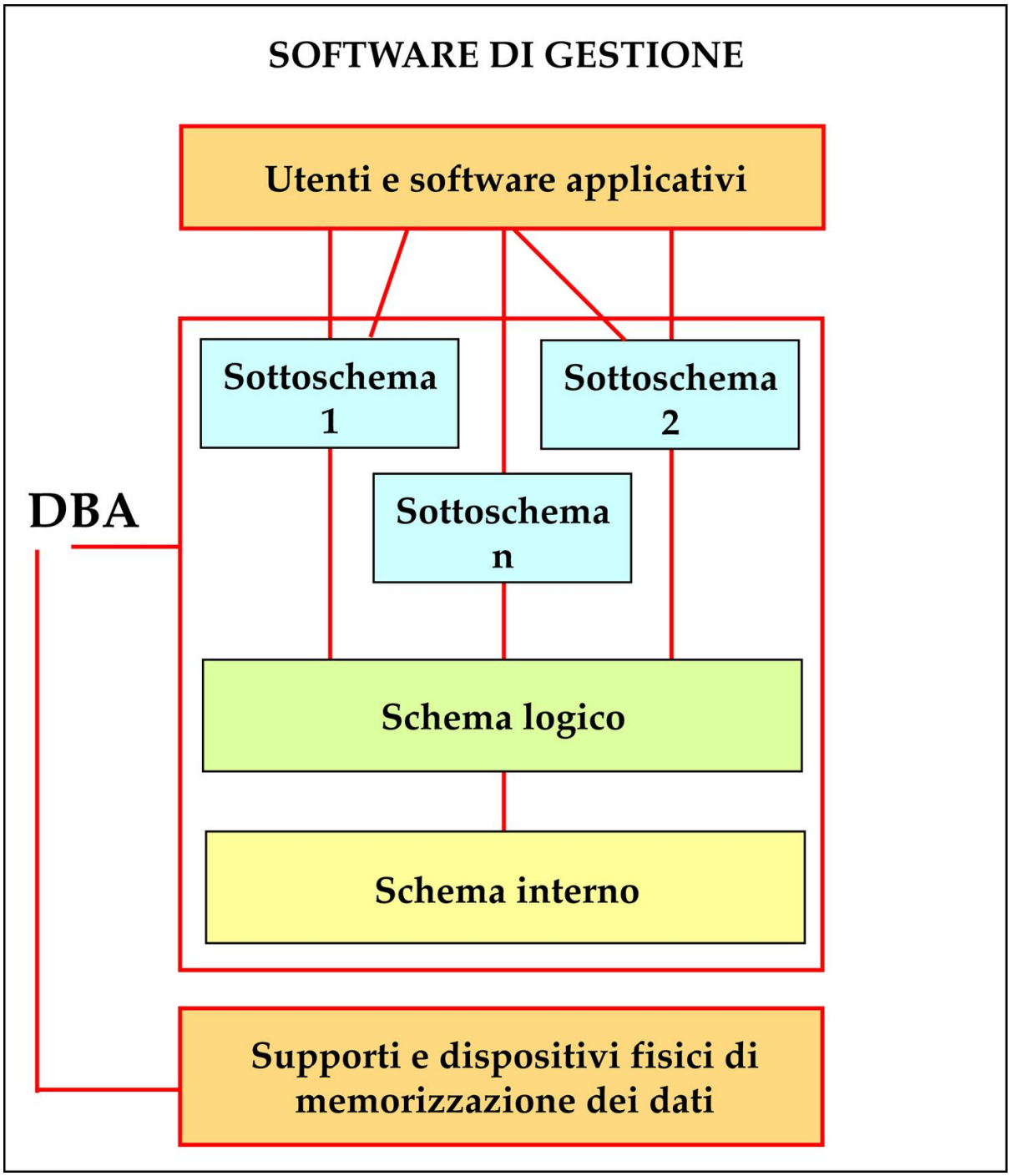

Figura 2. DBMS - Rappresentazione della struttura del software di gestione

Per i DBMS, il software di gestione comprende il Data Definition Language (DDL) che è utilizzato per la descrizione dettagliata e completa dei dati e delle loro relazioni.
Attraverso il DDL è possibile formalizzare la descrizione della struttura logica dei dati, definendo in modo convenzionale gli schemi che si intendono utilizzare.
Il DDL, inoltre, permette il collegamento tra la struttura logica e la struttura fisica dei dati, rappresentando così l'interfaccia software tra la struttura fisica dei dati e gli applicativi che li gestiscono.
Del linguaggio per la descrizione dei dati é parte integrante un linguaggio dedicato alla descrizione della memorizzazione su supporto fisico dei dati, il Device Media Control

Language (DMCL). L'utilizzo di questo linguaggio dedicato é finalizzato alla realizzazione delle basi dati per quanto concerne la struttura fisica e il conseguente utilizzo delle memorie di massa.
Per la completa gestione dei DBMS é utilizzato un Data Manipulation Language (DML), indirizzato alla definizione di interfacce software che permettano ad utenti e applicazioni di accedere e di gestire i dati.

Figura 3. DBMS - Linguaggi e strutture

Come DML é possibile utilizzare:

- *linguaggi algoritmici*, che permettono la creazione di funzioni per l'immissione, l'aggiornamento e la gestione dei dati così come funzioni di utilità più generali per la gestione del DBMS;
- *linguaggi di interrogazione* (query language) che permettono, attraverso istruzioni strutturate, di interrogare le basi dati per estrarre informazioni senza il necessario supporto di esperti informatici;

- *host language*, linguaggi di programmazione di ampio utilizzo applicativo, che permettono la completa gestione di archivi dati (come ad esempio COBOL, Visual Basic, C, PL/1 e Java).

2. I MODELLI DI DATA BASE

2.1 La memorizzazione dei dati: strutture e organizzazione

Operando una astrazione concettuale, i DBMS considerano ogni archivio memorizzato come una struttura costituita da un insieme di **record**.
Il record è definito attraverso la descrizione dei suoi elementi costitutivi: i **campi**.
I record contengono i valori che caratterizzano un'entità, come ad esempio può essere un libro, cioè l'insieme dei dati che permettono di estrapolare informazioni sull'entità stessa.
I campi invece contengono ciascuno il valore di un attributo dell'entità, come ad esempio il titolo dell'entità "libro" memorizzata in un record determinato.
Lo schema interno di un DBMS permette, in queste condizioni, di riconoscere se su un archivio é definita una regola di ordinamento dei dati e, se esiste, quale combinazione di campi la definiscono.
Ad esempio l'archivio libri può essere ordinato in ordine alfabetico crescente per "autore" oppure per "autore e titolo del libro"; in quest'ultimo caso troveremo i record in ordine alfabetico per autore, inoltre ogni gruppo di record contenenti i dati relativi ai libri di uno stesso autore è disposto in ordine alfabetico rispetto al titolo del libro.
In modo analogo, il DBMS é in grado di definire la possibilità di accesso diretto ad un record, identificandolo attraverso il valore di un attributo o di una combinazione di attributi.
L'attributo o la combinazione di attributi che permettono tale funzione di accesso diretto é detto **chiave**, cioè *un attributo o una combinazione di attributi il cui valore non si ripete in modo uguale in record distinti.*
I DBMS fondano la memorizzazione fisica dei dati sulle caratteristiche fisiche di quattro tipologie di strutture, che trovano ampio utilizzo nel campo dell'informatica:

- sequenziale;
- casuale;
- indicizzata;
- concatenata.

La struttura sequenziale
La struttura sequenziale opera una memorizzazione dei record in aree di memoria fisica contigue[4].
Questo tipo di struttura presenta l'inconveniente di avere bisogno di operazioni macchinose per la cancellazione o, negli insiemi ordinati, per l'inserimento di dati.
In particolare, se l'ordine fisico non coincide con l'ordine logico, il reperimento dei dati é il risultato di una ricerca sequenziale sui blocchi di memoria, per cui è sicuramente più lento di qualsiasi altra tecnica di ricerca.
Per questa inefficienza intrinseca, la struttura sequenziale é utilizzata dai DBMS prevalentemente per copie di backup[5] dei dati.

La struttura casuale
La struttura casuale opera la memorizzazione di record in blocchi di memoria, non necessariamente contigui, avvalendosi di un principio in base al quale i valori del campo chiave sono trasformati per determinare l'indirizzo di memoria fisica in cui memorizzare il record.

[4] Due aree di memoria si definiscono contigue quando il loro indirizzo differisce di l (l'area con indirizzo 123 e l'area con indirizzo 124 o 122)
[5] Copie di sicurezza periodiche che consentono, in caso di danni al sistema, di ripristinare la situazione di normalità, con minima perdita di dati.

Gli algoritmi che regolano questo sistema di memorizzazione devono gestire i problemi di collisione, vale a dire la possibilità che due valori della chiave generino lo stesso indirizzo fisico.
Il problema è generalmente risolto con metodi che portano alla generazione di liste di puntatori, che si allungano ad ogni nuova immissione. L'effetto più immediato é l'allungamento progressivo dei tempi di inserimento dei dati.
La struttura casuale, per come si é configurata in relazione ai problemi di collisione, permette esclusivamente l'acceso diretto ai dati.

La struttura indicizzata
La struttura indicizzata opera memorizzando l'archivio dati e creando parallelamente uno o più archivi indice che facilitano il reperimento dei dati.
Un esempio della funzione é dato dalla biblioteca dove, accanto al posizionamento fisico dei libri sugli scaffali, vengono creati schedari per argomento, per autore o per genere. La presenza di tali schedari permette la ricerca della scheda di un libro in modo rapido e in base a criteri di ricerca efficienti. Una volta trovata la scheda, su di essa é annotata la posizione del libro sullo scaffale, rendendo così facile la sua localizzazione.
La struttura indicizzata prevede diverse tipologie di indicizzazione che possono essere applicate alle possibili strutture di archivio per mantenere alta l'efficienza delle operazioni di inserimento, ricerca e elaborazione dei dati; questo fa sì che la struttura indicizzata, pur difficile da implementare e gestire, sia usata frequentemente.
La struttura indicizzata, si é rivelata utile per archivi dinamici, dove sono frequenti le operazioni sui dati.

La struttura concatenata
La struttura concatenata é la struttura concreta che traduce la struttura dati logica definita **lista**. La memorizzazione degli archivi avviene, in questo tipo di struttura, in blocchi di memoria che non hanno una precisa collocazione, ma sono uniti attraverso puntatori che permettono la definizione di ordinamenti logici.
La struttura concatenata é notevolmente flessibile, offre vantaggi in termini di efficienza nell'inserimento e nella cancellazione dei dati e permette che uno stesso blocco di memoria possa appartenere a sequenze logiche diverse.

Le strutture dati analizzate non sono le uniche definite e definibili per un DBMS, tuttavia qualsiasi altra struttura più complessa che si può definire é riconducibile alla combinazione di due o più delle quattro strutture individuate.

2.2 La memorizzazione dei dati: i modelli

Nel livello concettuale della rappresentazione dell'informazione nei DBMS si definisce il modello[6] che si intende applicare nella descrizione degli schemi e dei sottoschemi.
Per modello di dati si può definire l'insieme delle categorie e delle funzioni per la memorizzazione e la gestione delle entità che si intendono memorizzare, rispettando le caratteristiche di consistenza e integrità dei dati.
I tre modelli che sono stati individuati ed utilizzati nella definizione, progettazione e realizzazione dei DBMS sono:

- gerarchico;
- reticolare;
- relazionale.

[6] Un modello è un insieme di concetti e regole per l'organizzazione e la descrizione della struttura dei dati.

Ciascuno di questi modelli definisce strutture in cui memorizzare i dati e le relazioni con cui correlarli. È interessante notare che, presa una base dati, essa può essere rappresentata, in maniera più o meno efficace, utilizzando uno qualsiasi dei tre modelli, sottolineando in questo modo la loro importante valenza concettuale.

2.2.1 Il modello gerarchico

Il modello gerarchico é storicamente il primo modello utilizzato nella definizione e strutturazione dei DBMS. Il primo utilizzo di questo modello risale al 1969, nel sistema chiamato IMS (Information Management System) prodotto dall'IBM, e ne caratterizza a tutt'oggi la terminologia.
L'elemento portante del modello gerarchico é il **segmento**, un record per ogni dato da memorizzare. Ogni dato é un'entità e il record é costituito da tutti gli attributi dell'entità che si intendono memorizzare. Un attributo o una combinazione di attributi può essere considerato la **chiave** dell'entità o utilizzando un termine tipico del modello: il **descrittore**.
Lo schema di questo modello é costruito definendo un legame ***uno a molti*** tra un record padre e più record figli.
Prendiamo ad esempio uno schema per gli studenti universitari in cui individuiamo quattro segmenti:

1. **Facoltà** (*attributi*: Denominazione, Preside, Indirizzo Sede)
2. **Corsi di Laurea** (*attributi*: Denominazione, Durata, Numero di esami)
3. **Esami** (*attributi*: Denominazione, Anno di corso, Propedeuticità)
4. **Studenti** (*attributi*: Matricola, Nome, Cognome, Anno di corso)

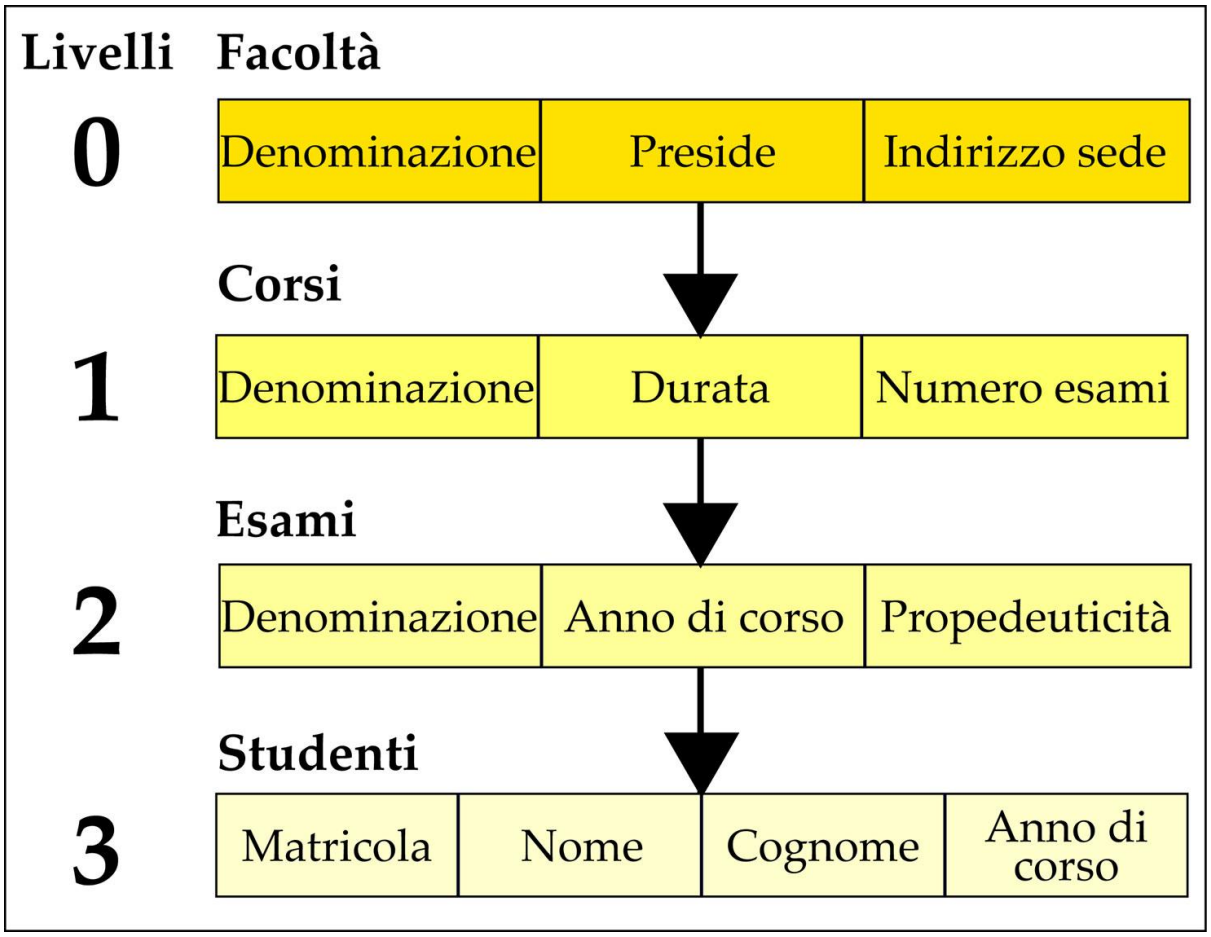

Figura 4. Modello gerarchico - Schema

In questa schematizzazione, ogni elemento del segmento Facoltà é radice di una struttura ad albero, in cui non é necessario associare nomi ai legami padre-figlio, poiché siamo in una tipologia di legame *uno a molti* dove ogni elemento figlio possiede solo un *elemento padre*, che a sua volta può possedere più *elementi figlio*.
Nella struttura ad albero introdotta per il modello gerarchico, é definito il concetto di ***livello di un record***, come la distanza del record dalla radice, definita di livello **0**.
Nell'esempio in esame, il record Studenti é di livello **3**, perché due sono i livelli di distanza dal record radice (Facoltà).
Ricorrendo alle proprietà tipiche della struttura ad albero é possibile affermare che ogni segmento (record) può diventare radice dell'albero, trasformandolo in una nuova struttura gerarchica.

La rigidità di questo schema diviene evidente nei casi in cui è necessario trasformare un legame uno a molti in un legame *molti a molti*.

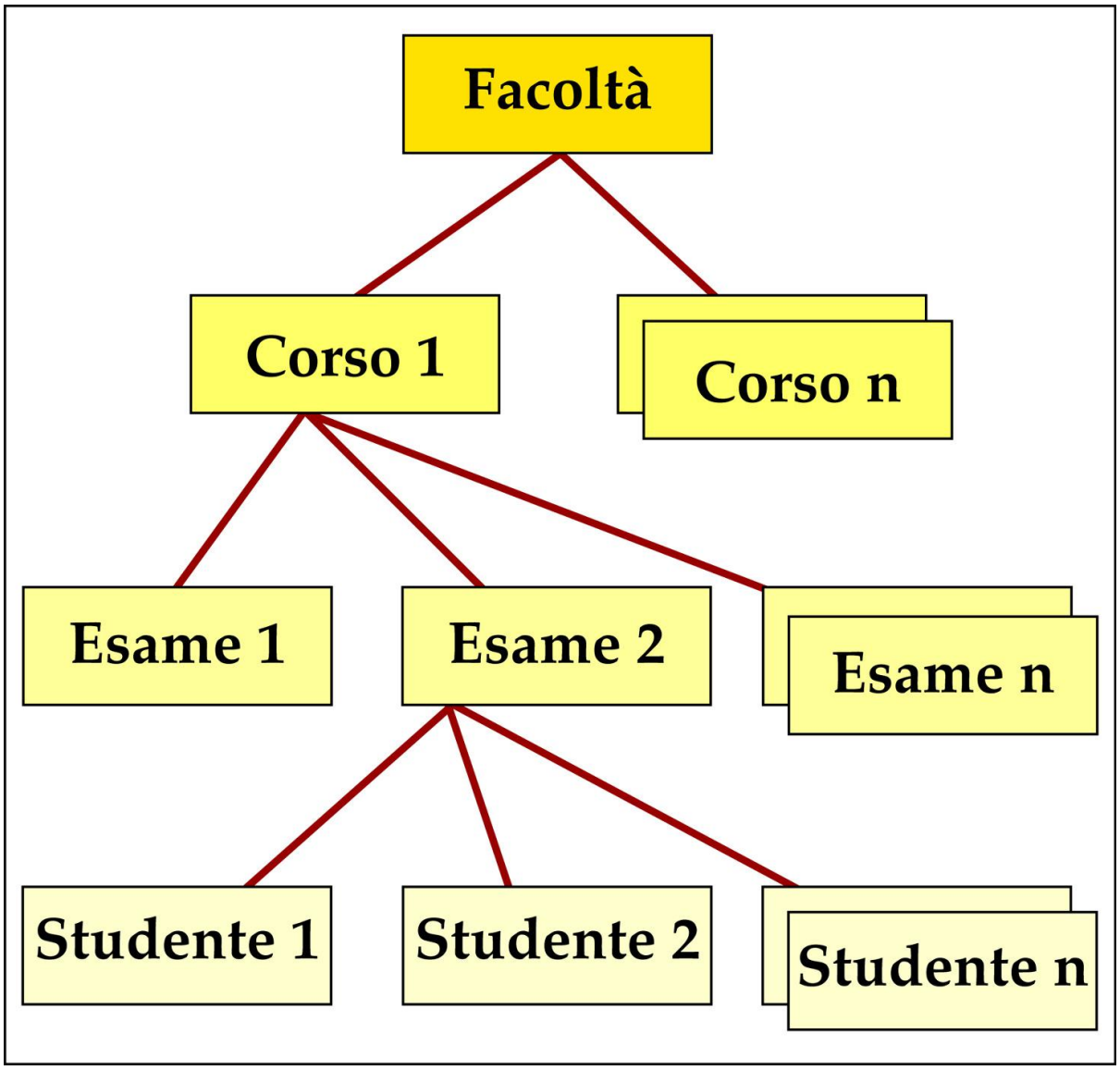

Figura 5. Modello gerarchico - Albero e relazione uno a molti

Nel caso dell'esempio, un legame *uno a molti* é esame-studenti (diversi studenti sostengono quell'esame), ma é legame *molti a molti* se si considera che diversi studenti sostengono diversi esami. In questa condizione é relativamente semplice formulare interrogazioni per identificare studenti a partire dagli esami, ma notevolmente più complesso formulare interrogazioni a partire dai dati degli studenti.
La struttura gerarchica presenta anche difficoltà gestionali nel caso di eliminazione di record. L'eliminazione di un elemento padre determina l'eliminazione di tutti i rami dell'albero che dipendevano da quell'elemento record (sottoalbero).
Nel caso del DBMS preso come esempio, la cancellazione di un corso di laurea implicherebbe la cancellazione di tutti i dati relativi agli esami di quel corso di laurea e di tutti i dati relativi agli studenti che avevano sostenuto quegli esami.
L'esempio che stiamo considerando evidenzia anche problemi concernenti l'inserimento di nuovi dati, infatti deve essere definito un padre per inserire un record figlio: ad esempio, non possono essere inseriti dati relativi ad uno studente senza che prima abbia sostenuto almeno un esame.

Le operazioni di ricerca ed individuazione di record, nel modello gerarchico, possono avvenire in due modi diversi:

- **accesso diretto *ad un record***: il record deve essere descritto attraverso la gerarchia dei record padre (per cercare il record Esame é necessario fare esplicito riferimento alla sua subordinazione a Corsi di Laurea e Facoltà);
- **attraversamento dell'albero**: si avvale della proprietà, implicita nella struttura ad albero, di ordinamento dei segmenti. Questo metodo permette, a partire dal posizionamento nella gerarchia, di individuare l'elemento successivo allo stesso livello o ai livelli inferiori dell'albero.

Nella realizzazione di modelli gerarchici si é, ad esempio, ovviato alle problematiche collaterali al passaggio da relazioni *uno a molti* a relazioni *molti a molti*, introducendo il concetto di **genitore logico**.

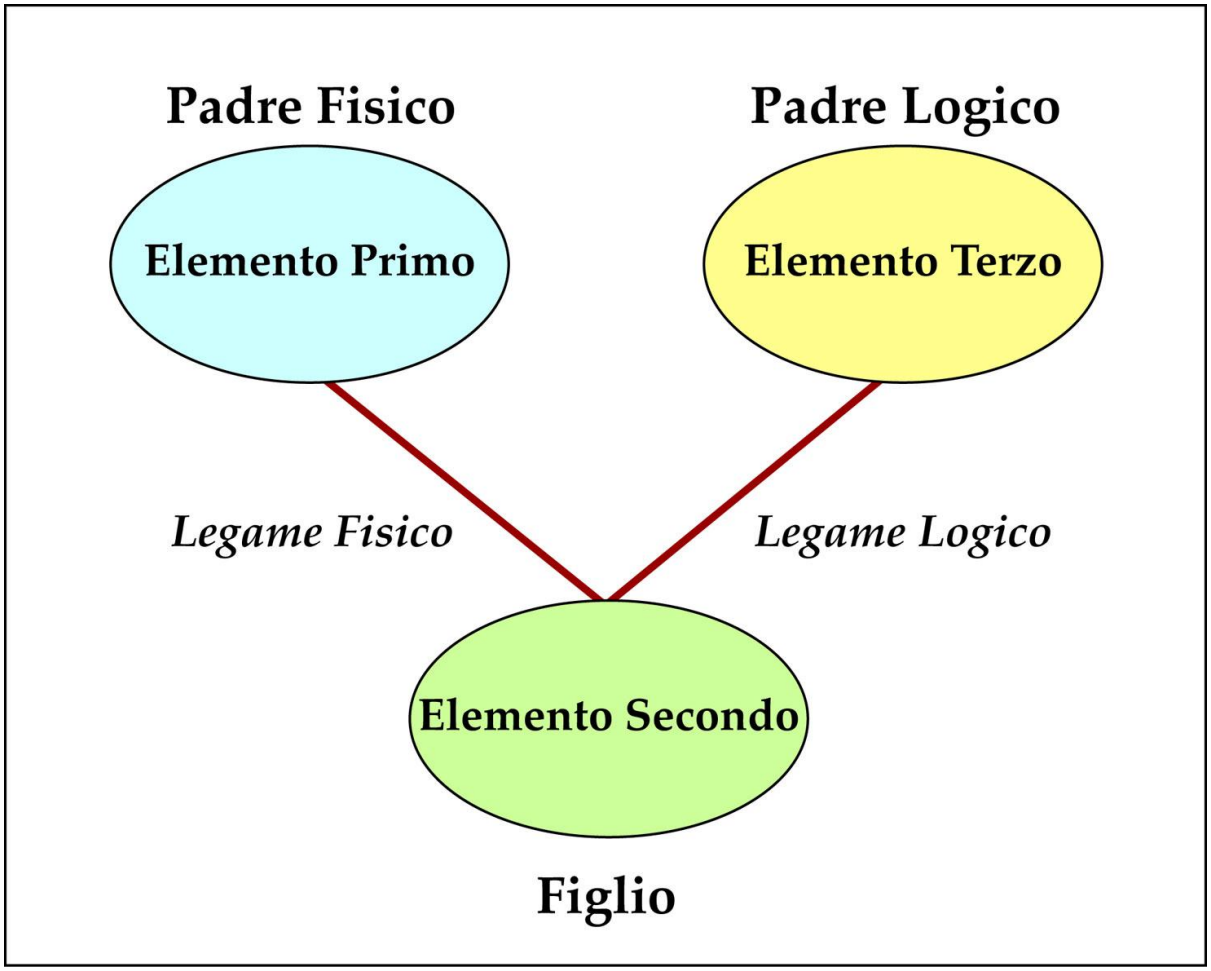

Figura 6. Modello gerarchico - Padre Logico e Padre Fisico

Il *genitore logico* permette di aggirare il vincolo determinato dalla condizione che ogni figlio ha un solo padre, consentendo di mantenere una definizione parimenti rigorosa, modificando la definizione del vincolo in: "ogni figlio può avere uno ed un solo padre fisico".
Questa non é una soluzione così flessibile, come é possibile trovare in altri modelli di dati, tuttavia é una notevole facilitazione nella definizione di legami molti a molti.

2.2.2 Modello reticolare

Il modello reticolare é, dal punto di vista della struttura, un'evoluzione del modello gerarchico, e definisce lo schema dei dati attraverso due elementi portanti:

- **record**
- **set**

Il **record** è, per definizione e per utilizzo, il medesimo elemento introdotto nel modello gerarchico, ma ancora più in generale nella definizione di archivio.
Il **set** invece é l'elemento innovativo, l'ampliamento del modello gerarchico, attraverso il quale è possibile superarne alcuni limiti. È la rappresentazione esplicita dei legami logici esistenti tra i diversi tipi di record.
I set sono record speciali (record di collegamento) che permettono più collegamenti tra le diverse entità; i record di collegamento schematizzano i legami logici tra i record di tipo diverso.
In particolare sono esplicitati i legami gerarchici *uno a molti* tra ***record owner*** e ***record member***, dove un record owner é collegato ad un insieme di record member, e tale insieme può essere vuoto.

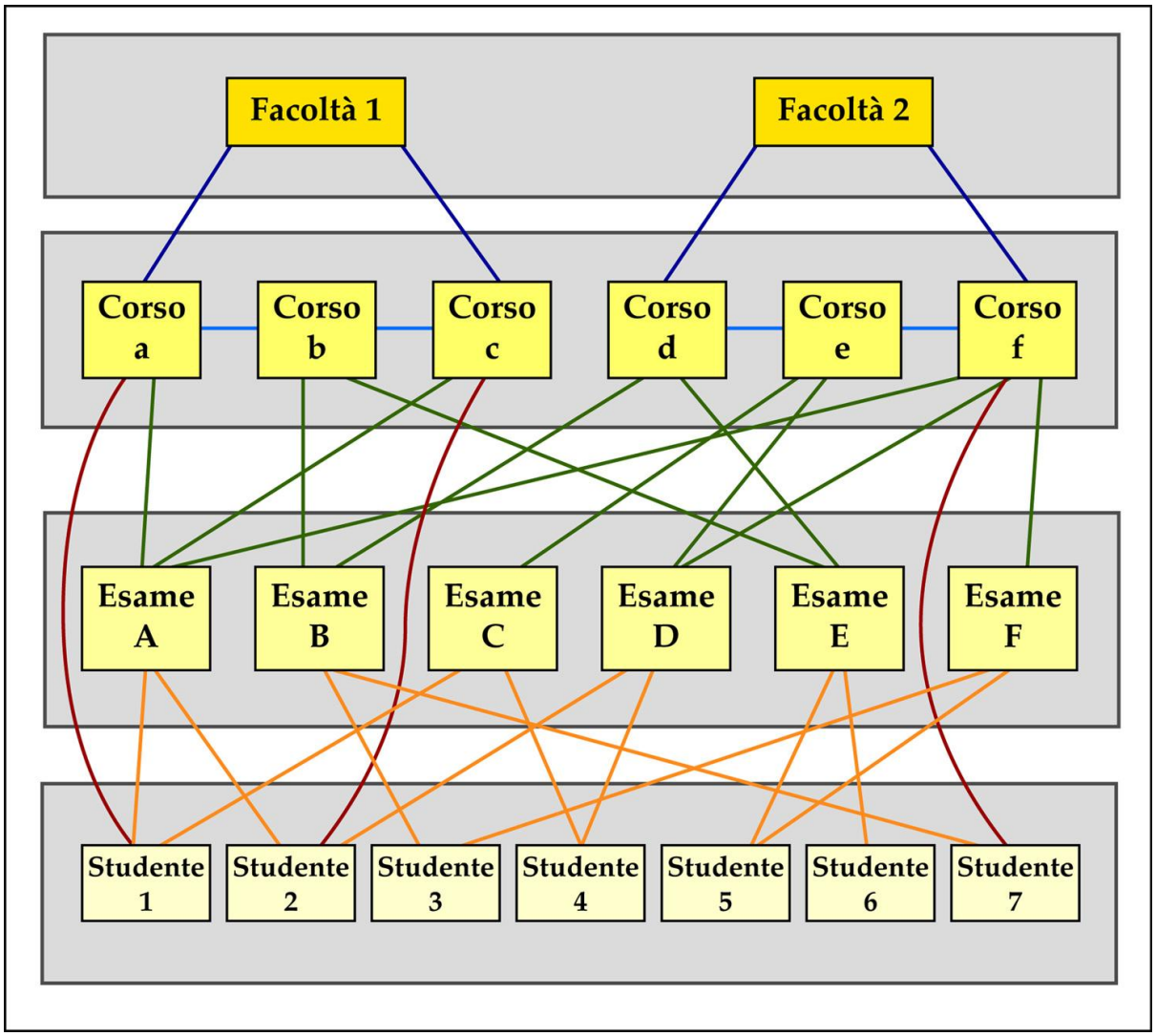

Figura 7. Modello reticolare - Record e relazioni

Nell'esempio preso in esame per i modelli gerarchici, possiamo definire quattro tipi di record:

1. Facoltà
2. Corso di Laurea
3. Esami

4. Studenti

e quattro set
1. *"appartiene a"*
2. *"è del corso di"*
3. *"è iscritto a"*
4. *"ha sostenuto"*

In questa configurazione il set "ha sostenuto" ha come owner *Esame* e come member *Studente*.

Il modello reticolare nella sua configurazione completa, per godere delle proprietà messe a disposizione dalla struttura, deve possedere le seguenti proprietà:
- ogni tipo di record appartiene ad un set;
- un tipo di record non può essere member e owner del medesimo set (ma lo può essere per set diversi);
- un esemplare di record non può appartenere a due esemplari distinti dello stesso set.

Ogni tipo di record appartiene ad un set
Nel modello reticolare non é possibile definire un record che non possieda nessun tipo di relazione con un altro record presente nel modello.
Se emerge la necessità di definire record per il quale non é possibile definire relazioni e, nonostante questo, è necessaria la presenza di tale record, il DBMS possiede un set di sistema, definito in modo standard attraverso l'entità SYSTEM, di cui si considera member il record in oggetto.

Un tipo di record non può essere member e owner del medesimo set
Nel modello reticolare non é possibile definire relazioni che vedano lo stesso record come owner e come member, ma situazioni in cui si deve affrontare questo tipo di problema sono tutt'altro che casi particolari.
Considerando la base dati dell'esempio, se volessimo descrivere la relazione gerarchica tra studenti, come "studente" e "rappresentante degli studenti" (chiamiamo tale set RAPPRESENTA), giungeremmo a definire un set che ha il record "studente" come owner e come member.
In queste situazioni si ricorre alla definizione di un *record di collegamento* e di due set, nei quali il record di collegamento compare nel primo come owner e nel secondo come member.
A questo tipo di artificio é necessario ricorrere in tutti i casi in cui è determinata, tra due tipi di record, una relazione *molti a molti*.

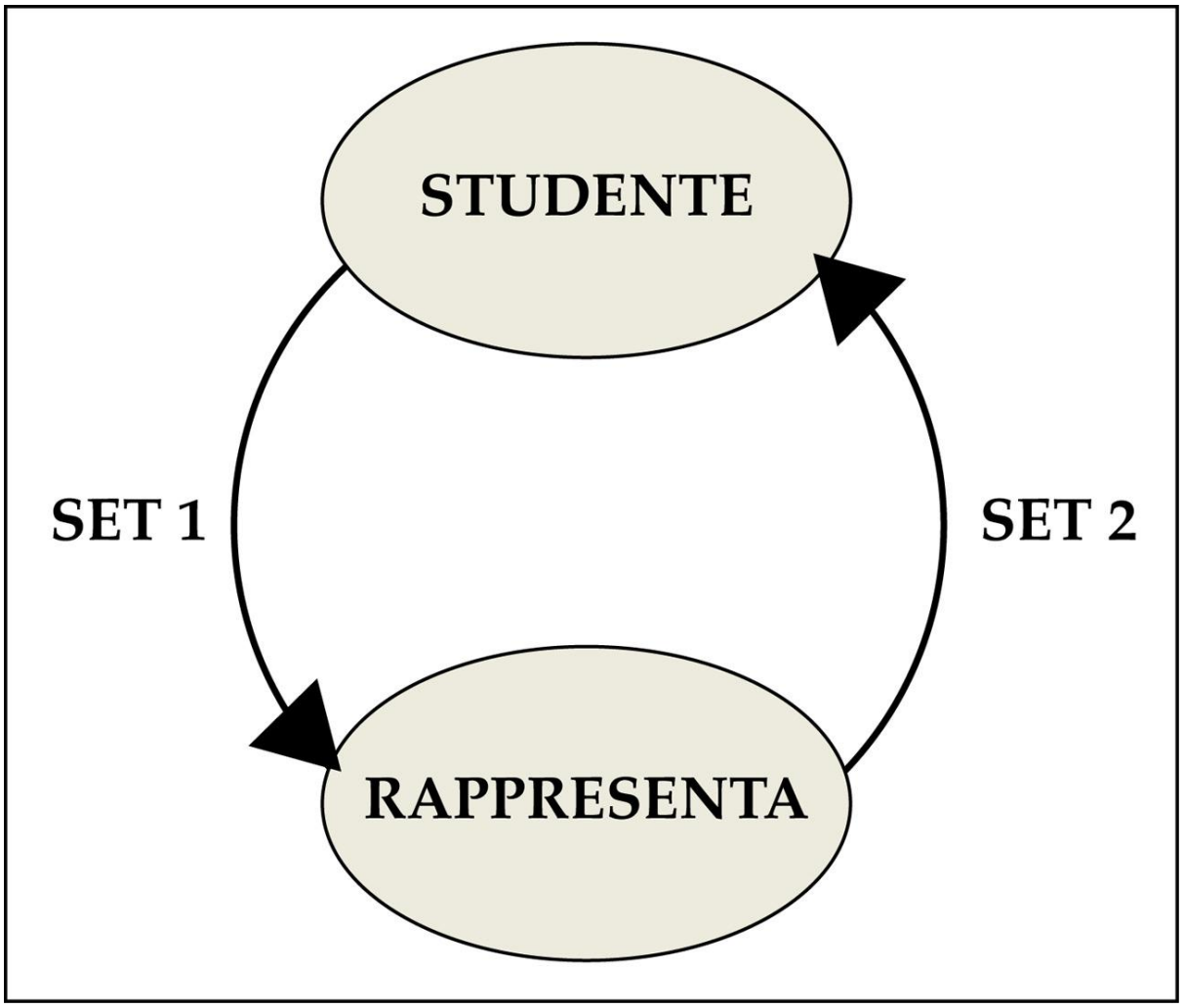

Figura 8. Modello reticolare - Record di collegamento

Un esemplare di record non può appartenere a due esemplari distinti dello stesso set
La corretta strutturazione di un modello reticolare è verificabile elencando record e set che compongono il DBMS e verificandone la loro congruenza rispetto alle proprietà del modello.
Per una corretta ed efficiente gestione dei dati, nel modello reticolare é necessario definire regole e algoritmi implementati all'interno del DBMS, che permettano di individuare ogni singolo esemplare dei tipi di record definiti.
Per ottenere questo risultato un metodo è quello di avere un archivio interno di chiavi (database key) che rende possibile identificare, in maniera univoca, gli esemplari dei record.
Il database key non é l'unico modo attraverso il quale é possibile eseguire localizzazioni di esemplari, ma é possibile anche utilizzare i set a cui il record appartiene.
È compito delle procedure che svolgono mansioni di localizzazione di esemplari scegliere il metodo di localizzazione più efficace: la **location mode**.

2.2.3 Modello relazionale
Il modello relazionale é, in ordine di tempo, l'ultimo modello di dati sviluppato ed introdotto nell'ambito dei DBMS.

Il modello si fonda sulla definizione del concetto matematico di *relazione*[7] ed opera organizzando i dati relativi alle diverse entità in tabelle come le seguenti:

SCHEMA RELAZIONALE

Facoltà

Chiave Facoltà	Denominazione	Preside	Indirizzo

Corsi di Laurea

Chiave Corso	Denominazione	Durata	Chiave Facoltà

Esami

Chiave Esami	Denominazione	Anno di corso	Chiave Corso

Studenti

Chiave Studenti	Matricola	Nome	Cognome	Chiave Esame

SCHEMA SOTTOINSIEMI DI DATI

Facoltà

Chiave Facoltà	Denominazione	Preside	Indirizzo
Fac1	Scienze	Cino Rossi	Via Verdi 5
Fac2	Lettere	Paolo Bianchi	Via Pini 8

Corsi di Laurea

Chiave Corso	Denominazione	Durata	Chiave Facoltà
Cor1	Matematica	3	Fac1
Cor2	Pedagogia	3	Fac2
Cor3	Fisica	4	Fac1

Esami

Chiave Esami	Denominazione	Anno di corso	Chiave Corso
Esa1	Analisi	1	Cor1
Esa2	Statistica	3	Cor1
Esa3	Storia Antica	2	Cor2
Esa4	Filologia	4	Cor2
Esa5	Metodi e misure	3	Cor1

Studenti

Chiave Studenti	Matricola	Nome	Cognome	Chiave corso
Stu1	001234	Aldo	Verdi	Cor1
Stu2	000123	Pino	Bianchi	Cor1

[7] Dati gli insiemi $\mathbf{A_1, A_2, A_3..., A_n}$ non necessariamente ad intersezione nulla, una relazione su tali insiemi, indicata con $\boldsymbol{R(A_1, A_2, A_3..., A_n)}$, é un insieme di n-ple ordinate $\boldsymbol{(a_1,a_2,a_3...., a_n)}$, tali che ogni elemento $\mathbf{a_i}$ appartiene all'insieme $\mathbf{A_i}$. A_i sono detti i domini di R e n il suo grado.

Stu3	012345	Paolo	Rossi	Cor2
Stu4	003456	Giorgio	Cera	Cor3
Stu5	000567	Elena	Riva	Cor2

Le tabelle così strutturate, una per ogni entità, sono composte da un numero di colonne pari agli attributi dell'entità e da un numero di righe pari agli esemplari dell'entità memorizzata.
Per la teoria delle relazioni in matematica, ogni tabella é una ***relazione***, ogni colonna é un ***attributo*** e ogni riga è una ***tupla*** (oppure ***n-pla***).
Per una corretta memorizzazione dei dati e della struttura, il modello relazione richiede due condizioni fondamentali, cioè

- le relazioni devono avere nomi diversi;
- gli attributi di ogni singola relazione devono avere nomi diversi.

Il modello relazionale richiama la struttura sequenziale, dove le tabelle possono essere messe in corrispondenza diretta con le relazioni e i record con le tuple.
In questo caso, tuttavia, si sta parlando di strutture di dati astratte, per questo il DBMS utilizzerà diverse modalità di memorizzazione fisica.
Altre sono le differenze che determinano la specificità di questa struttura:

- la tupla deve essere unica nella relazione mentre la struttura sequenziale ammette ripetizioni di record;
- l'ordine delle tuple è fondamentale per le strutture sequenziali mentre non caratterizza i DBMS;
- l'ordine di memorizzazione degli attributi non é determinante laddove nella struttura sequenziale é una caratteristica del record.

I modelli relazionali si caratterizzano e differenziano da tutti gli altri modelli dati, per la capacità di esplicitare i legami logici tra le diverse entità e, in modo implicito, attraverso la presenza di attributi comuni (come ad esempio "Chiave Corso" tra le entità "Studenti" ed "Esami").
In particolare, questo permette al modello relazionale di non dover ricorrere a strutture di memorizzazione per i legami tra entità.
In questa caratteristica risiede la semplicità di strutturazione per tali modelli, di DML non procedurali.
Si fa riferimento al contenuto delle relazioni per definire stringhe di interrogazione degli archivi, rimanendo totalmente svincolati dalla struttura fisica nella quale sono memorizzati.

Nell'esempio strutturato per introdurre il modello relazionale si può notare che esistono legami impliciti di tipo *uno a molti*: una facoltà ha più corsi oppure un corso di laurea prevede più esami.
Se, all'interno del modello utilizzato come esempio, introduciamo una nuova entità a cui diamo nome "Sessioni", che ha la struttura schematizzata nel modo seguente:

Sessioni

Chiave Esame	Chiave Studente	Voto riportato

si può introdurre un'ulteriore relazione *molti a molti* tra "Esami" e "Studenti": "uno studente sostiene più esami ed un singolo esame é sostenuto da più studenti".
Implementando queste caratteristiche, il modello relazionale sarà indifferente anche all'ordine con cui i domini sono definiti, anche se nella definizione rigorosa di relazione matematica si parla di n-ple ordinate.

La gestione dei dati in un modello relazionale risente dell'influenza della matematica delle relazioni. Per questo le caratteristiche del modello fino ad ora enucleate si possono elencare come azioni necessarie per la *normalizzazione*[8] dei dati:

- ogni n-pla é composta da dati elementari;
- ogni attributo é istanziato[9] con un solo valore;
- la combinazione di attributi di una n-pla che identifica un esemplare, deve essere unica (*chiave*[10]) e non contenere dati ridondanti;
- gli attributi appartenenti a *chiave* non possono assumere valore nullo;
- gli attributi dell'n-pla che non rientrano nella *chiave* (attributi dipendenti) non devono essere ridondanti e non deve esistere dipendenza funzionale tra gli attributi dipendenti.

Esempi di relazioni non normalizzate e normalizzate
Prendiamo in esame alcuni semplici esempi di relazioni non normalizzate e una possibile forma di normalizzazione che, in molti casi, si avvale della definizione di nuove relazioni.

Esempio 1

Relazione non normalizzata per ridondanza Libri

Autore	Nazione	Titolo
Benni	Italia	Spiriti
Benni	Italia	Stranilandia

Relazioni normalizzate
Autori

Autore	Nazione
Benni	Italia

Libri

Autore	Titolo
Benni	Spiriti
Benni	Stranilandia

[8] Nei modelli relazionali esistono alcune proprietà, chiamate *forme normali*, che permettono di assicurare che uno schema di un database sia relazionale. Per schemi che non soddisfino queste proprietà è possibile applicare processi detti di *normalizzazione* che permettano la trasformazione degli schemi in esame in schemi che soddisfano la forma normale.
[9] Il termine "istanza" è paragonabile al termine "oggetto", differenziandosi da un punto di vista del significato, poiché tende a porre in evidenza l'appartenenza dell'oggetto a un dato tipo ("istanza di"). In tale contesto, "istanziare" un attributo (o "creare un'istanza" di un attributo) significa attribuire un valore all'attributo stesso. Ad esempio, "professore" è un attributo, "docente" è una sua istanziazione.
[10] Attributo o combinazione di attributi che non si ripete uguale per due entità diverse di una medesima relazione.

Esempio 2
Relazione non normalizzata per dati ridondanti
Libri

Autore	Titolo	Casa Editrice	Indirizzo Casa Editrice
Benni	Spiriti	Stamperie & C.	Via Respighi
Benni	Stranilandia	Stamperie & C.	Via Respighi

Relazioni normalizzate
Libri

Autore	Titolo	Casa Editrice
Benni	Spiriti	Stamperie & C.
Benni	Stranilandia	Stamperie & C.

Case Editrici

Casa Editrice	Indirizzo Casa Editrice
Stamperie & C.	Via Respighi

Esempio 3
Relazione non normalizzata per dipendenza funzionale
Libri

Autore	Titolo	Casa Editrice	Codice Autore
Benni	Spiriti	Stamperie & C.	0001
Benni	Stranilandia	Stamperie & C.	0001

Relazioni normalizzate
Libri

Autore	Titolo	Casa Editrice
Benni	Spiriti	Stamperie & C.
Benni	Stranilandia	Stamperie & C.

Codici Autore

Autore	Codice Autore
Benni	0001

Il concetto di *chiave* é centrale nella gestione dei modelli relazionali e specificatamente si parla di **chiave primaria** e **chiavi secondarie** (o alternative).
Analizzando le relazioni, é possibile identificare più attributi o più combinazioni di attributi che soddisfano le caratteristiche di *chiave*. Tali combinazioni sono chiamate **candidati chiave**.
La scelta nell'insieme dei *candidati chiave*, della *chiave primaria* e delle *chiavi secondarie* é una scelta che dipende dalle strategie che definisce il responsabile della gestione dell'archivio.
La presenza di *chiavi* nei modelli relazionali non implica necessariamente la definizione di archivi fisici di indici, poiché, come é evidente in tutti gli aspetti di questo modello, anche il concetto di *chiave* é puramente logico e la *chiave* é completamente trasparente all'utente che accede e manipola gli archivi.
In questo modello, l'utilizzo di indici fisici é subordinato a necessità di efficienza e velocità di reperimento dei dati.

Il modello relazionale é attualmente il modello predominante nei DBMS, in virtù di una sua rigorosa base matematica che rende semplici e intuitive le operazioni di creazione di schemi, interfacce e funzioni di manipolazione dei dati.
Queste sue caratteristiche sono state accentuate dalla rapida evoluzione dei tipi di base dati (si pensi alle basi dati multimediali) e degli strumenti che si collegano con i DBMS (ad esempio gli strumenti dell'intelligenza artificiale, gli strumenti di rete per i dati distribuiti e l'interazione base dati - Internet).

2.3 Conversione tra modelli di dati

L'analisi dei tre modelli di dati utilizzati nei DBMS ha evidenziato la forte caratterizzazione di ciascuno di essi, in funzione degli strumenti di manipolazione e correlazione dei dati.
È intuitivo comprendere che, per questo motivo, convertire un sistema DBMS da un modello ad un altro è, nella realtà delle aziende e delle organizzazioni un'operazione che richiede grande dispendio di tempo e di risorse.
Da un punto di vista didattico e per la migliore comprensione delle caratteristiche e delle affinità tra i tre modelli possibili di dati, é un utile esercizio determinare alcuni passaggi fondamentali per convertire un modello dati in uno degli altri modelli possibili.

Modello Gerarchico in modello Reticolare
Questa trasformazione é sicuramente di semplice esecuzione e non presenta alcuna difficoltà concettuale; i passi che la conversione richiede sono i seguenti:

- segmenti → tipi di record
- legami gerarchici → set
- nominare i set così ottenuti

Modello Reticolare in modello Relazionale
Analogamente a quanto è avvenuto per la trasformazione da *modello gerarchico* a *modello reticolare*, bastano pochi semplici passi per ottenere questa trasformazione. Si tratta della formalizzazione di un processo cognitivo che ha portato all'evoluzione del *modello reticolare* in *modello relazionale*:

- record che contengono una *chiave* (un campo o più) → relazioni con la stessa *chiave primaria*;
- record *member* di *set* → relazione composta con il campo di tipo owner, del medesimo set, che diventa così, *chiave primaria*, se lo era per il record owner, della relazione o attributo dipendente.

Modello Reticolare in modello Gerarchico
La caratteristica che differenzia questi due modelli é che nel modello gerarchico ogni segmento ha un unico padre, mentre nel modello reticolare, é possibile che un segmento abbia più di un padre (in pratica può essere *member* di più *set).*
Si ricorre dunque al concetto di *padre fisico* ed al concetto di *padre logico*, privilegiando la scelta dell'owner come padre fisico, essendo la struttura con meno limitazioni di utilizzo e maggior efficienza in fase di manipolazione dati.

Modello Relazionale in modello Reticolare
La sequenza di operazioni che formalizzano il processo di trasformazione dal modello relazionale al modello reticolare é la seguente:

- relazione → tipo di record con campi gli attributi della relazione;
- attributi comuni a più relazioni → set (owner del set è il record relativo alla relazione che possiede quell'attributo come chiave).

Come esemplificazione di quest'ultimo processo di trasformazione prendiamo in esame la struttura relazionale[11]:

Facoltà

Chiave Facoltà	Denominazione	Preside	Indirizzo

Corsi di Laurea

Chiave Corso	Denominazione	Durata	Chiave Facoltà

Esami

Chiave Esami	Denominazione	Anno di corso	Chiave Corso

Studenti

Chiave Studenti	Matricola	Nome	Cognome	Chiave Esame

In questa condizione, nella relativa struttura reticolare si avranno quattro tipi di record, uno per ogni relazione, ciascuno dei quali avrà come campi gli attributi della corrispondente relazione.
Sono così definiti, in maniera uniforme rispetto alla descrizione del processo di trasformazione, tre set:

Nome del set	Owner	Member
Appartiene alla facoltà	Facoltà	Corsi di Laurea
È un esame del corso	Corsi di Laurea	Esami
Ha sostenuto	Esami	Studenti

Ogni altra trasformazione tra modelli é definibile come combinazione delle trasformazioni di cui è stato descritto il processo.

2.4 Elaborazione dei dati e Data Manipulation Language

La definizione di un modello di dati é completa se comprende anche l'elenco delle funzioni messe a disposizione dal DML per le operazioni di inserimento, cancellazione, modifica di dati e strutture del modello stesso.
La definizione rigorosamente matematica del modello relazionale permette di postulare due tipi di linguaggio di manipolazione dei dati:

1. algebra relazionale
2. calcolo relazionale

Algebra relazionale
L'algebra relazionale si fonda sulla possibilità di applicare alcune delle operazioni tra gli insiemi alle relazioni che hanno struttura analoga.
Le operazioni dell'algebra relazionale permettono di ottenere nuove relazioni risultanti dalle operazioni su relazioni esistenti. Le operazioni sono le seguenti:

1. **Operazioni base**
   1.1. Unione
   1.2. Intersezione

[11] In grassetto i campi *chiave* per ogni relazione

1.3. Differenza

2. Operazioni di associazione

2.1. Prodotto cartesiano

2.2. Divisione

3. Operazioni di selezione

3.1. Proiezione

3.2. Giunzione

3.3. Selezione

Le operazioni che sono definite sulle relazioni sono in parte una trasposizione delle operazioni equivalenti per l'algebra degli insiemi e in parte operatori specifici delle operazioni sulle relazioni.

Le operazioni di base

Date due relazioni R_1 ed R2, dello stesso grado n si definisce:

- **unione** tra R_1 ed R_2 la relazione R che contiene tutte le n-ple di R_1 e tutte le n-ple di R2, con l'eliminazione delle ripetizioni;
- **intersezione** tra R1 ed R2 la relazione R_u che contiene tutte le n-ple che appartengono ad R_1 e ad R_2;
- **differenza** tra R_1 ed R2 la relazione che contiene tutte le n-ple che appartengono ad R_1 ma non appartengono ad R_2;

R_1 - Libri di avventura

Titolo	Autore
Libro 1	Autore X
Libro 2	Autore Y
Libro 3	Autore X
Libro 4	Autore Z

R_2 — Libri di autori italiani

Titolo	Autore
Libro a	Autore A
Libro b	Autore B
Libro c	Autore B
Libro 4	Autore Z

$R_1 \cup R_2$ (unione)

Titolo	Autore
Libro 1	Autore X
Libro 2	Autore Y
Libro 3	Autore X
Libro 4	Autore Z
Libro a	Autore A
Libro b	Autore A
Libro c	Autore B

$R_1 \cap R_2$ (intersezione)

Titolo	Autore
Libro 4	Autore Z

R_1 — R_2 (differenza)

Titolo	Autore
Libro 1	Autore X
Libro 2	Autore Y
Libro 3	Autore X

R_2 — R_1 (differenza)

Titolo	Autore
Libro a	Autore A
Libro b	Autore A
Libro c	Autore B

Le operazioni di associazione
Date due relazioni R_1 e R_2, di grado rispettivamente *n* ed *m*, non necessariamente uguale, si definisce:

- **prodotto cartesiano** tra R_1 e R_2 la relazione R_u le cui tuple (di grado $n+m$) si ottengono associando ogni *n*-pla di R_1 con ogni *m*-pla di R_2.

Date due relazioni R_1 e R_2, di grado rispettivamente $m+n$ ed *m* e tali per cui gli attributi, nelle posizioni da $m+1$ fino ad $m+n$ di R1 siano definiti sui medesimi domini degli attributi da 1 ad *n* della relazione R_2, si definisce:

- **divisione** tra R_1 ed R_2 la relazione R_u di grado *m*, le cui *m*-ple sono composte da tutti i valori degli attributi di R_1, non comuni ad R_2, per i quali esistono, delle $(m+n)$-ple di R_1 e nei domini comuni a R_2, tutti i valori degli attributi di R_2.

R_1 — Libri

Titolo
Libro 1
Libro 2

R_2 — Edizione

Edizione
Economica
Tascabile
Illustrata

R_1 x R_2 (Prodotto)

Titolo	Edizione
Libro 1	Economica
Libro 1	Tascabile
Libro 1	Illustrata
Libro 2	Economica
Libro 2	Tascabile
Libro 2	Illustrata

R_1 — Libri

Titolo	Edizione
Libro 1	Autore X
Libro 2	Autore Y
Libro 3	Autore X
Libro 4	Autore Z

R_2 — Autori

Autore
Autore X
Autore Y

R_1 / R_2 (Divisione)

Titolo
Libro 4

Le operazioni di selezione
Data una relazione R_1 di grado *n*, si definisce:

- **proiezione** una nuova relazione R_2 di grado $m \leq n$ che si ottiene conservando solo *m* domini della relazione R_1 e mantenendo, nella relazione risultante, le *m*-ple senza ripetizioni. La relazione risultante é dunque un sottoinsieme dei domini della relazione di origine ed in tali domini non presenta righe multiple.

R_1 - Libri

Titolo	Autore	Nazionalità
Libro 1	Autore X	Italiana
Libro 2	Autore Y	Italiana
Libro 3	Autore X	Inglese
Libro 4	Autore Z	Francese

R_2 - Proiezione

Autore	Nazionalità
Autore X	Italiana
Autore Y	Italiana

Date due relazioni R_1 e R2, di grado rispettivamente *n* ed *m*, non necessariamente uguale e con un dominio in comune, si definisce:

- **giunzione sul dominio** (il dominio comune) tra R_1 ed R_2 la relazione R_u, formata da tutte le tuple (di grado $n+m-1$) che si ottengono giustapponendo ogni *n*-pla di R_l, con tutte le *m*-ple di R2 che hanno valori, nel dominio comune, che soddisfino una o più condizioni (ad esempio una condizione di uguaglianza);

R_1 - Libri

Titolo	Autore
Libro 1	Autore X
Libro 2	Autore Y
Libro 3	Autore X
Libro 4	Autore Z

R_2 - Autori

Autore	Nazionalità
Autore X	Italiana
Autore Z	Inglese

R_u - Giunzione

Titolo	Autore	Nazionalità
Libro 1	Autore X	Italiana
Libro 2	Autore X	Inglese
Libro 3	Autore Z	Francese

Data una relazione R_1 di grado *n*; un valore *x* di un attributo *a* della relazione; ed una condizione *c* per i valori *x* assunti dall'attributo *a*, si definisce:

- **selezione** la relazione R_u, formata da tutte le *n*-ple che si ottengono selezionando le *n*-ple di R_1 per le quali la condizione *c* é soddisfatta dal valore *x* assunto dall'attributo *a*;

R_1 - Libri

Titolo	Autore	Nazionalità
Libro 1	Autore X	Italiana
Libro 2	Autore Y	Italiana
Libro 3	Autore X	Inglese
Libro 4	Autore Z	Francese
Libro 5	Autore L	Italiana
Libro 6	Autore X	Inglese
Libro 7	Autore M	Francese

Condizione c: Nazionalità = Italiana
R_u - Selezione

Titolo	Autore	Nazionalità
Libro 1	Autore X	Italiana
Libro 2	Autore Y	Italiana
Libro 5	Autore L	Italiana
Libro 6	Autore X	Inglese

Calcolo relazionale

Il **calcolo relazionale** é un DML che fonda le sue definizioni e le sue funzioni sul calcolo dei predicati, così come é postulato nella logica. Nell'ambito dei modelli relazionali i predicati trovano corrispondenza nelle *n*-ple delle singole relazioni.

Le premesse teoriche del linguaggio così definito determinano una struttura sintattica che può essere riassunta, in maniera rigorosa, in due passaggi:

1. DICHIARAZIONE DELLA *N*-PLA
2. <*COMANDO*> <*LISTA DEGLI ATTRIBUTI DELLA NUOVA RELAZIONE*> **WHERE** <CONDIZIONI PER LE *N*-PLE>

A questa sintassi si fa riferimento nei linguaggi di interrogazione delle basi dati, in particolare per quanto riguarda SQL (Structured Query Language) e QBE (Query By Example).

3. Linguaggi per la manipolazione dei dati

Definito un modello per i dati e strutturati tutti gli elementi necessari alla corretta utilizzazione e gestione degli archivi, l'apertura e la manipolazione di una base dati sono operazioni possibili mediante le caratteristiche di un DML.
Tra i possibili linguaggi di diverso tipo e generazione utilizzabili per tale scopo, ha assunto un ruolo di primo piano lo Structured Query Language (SQL, pronunciato |*siquel*|).
SQL é un DML definito per il modello relazionale, costruito contestualmente al primo DBMS relazionale, realizzato negli anni 70 dall'IBM (SYSTEM/R).
La flessibilità e semplicità di utilizzo hanno fatto sì che SQL diventasse uno standard per l'interrogazione dei DBMS.
SQL, per le caratteristiche intrinseche che lo definiscono é:

- un linguaggio DML autonomo;
- un linguaggio DML integrato in tutti i più diffusi linguaggi di programmazione ad alto livello (C++, Visual Basic, Java) ed in linguaggi server-side ed applicativi per Internet (ASP e JSP).

SQL si candida dunque ad essere un linguaggio universale per la manipolazione dei dati nei DBMS; infatti, tutti i principali DBMS utilizzati attualmente nelle gestione delle grandi masse di dati (ad esempio SQL Server od Oracle), pur mantenendo molte caratteristiche proprietarie, riconoscono i comandi SQL.
In tale contesto strutturale é possibile implementare funzioni di manipolazione dei dati basate su SQL, rendendo quasi irrilevante la conoscenza del DBMS di riferimento, se non per l'elenco delle relazioni e degli attributi a cui ci si può riferire.
Tra i linguaggi per la manipolazione dei dati, riveste un interesse da un punto di vista logico il QBE (Query By Example), un DML con le caratteristiche di un linguaggio visual[12].
La sua minor diffusione é dovuta principalmente alla difficoltà dell'integrazione di tale DML nei linguaggi evoluti di programmazione.

3.1 Structured Query Language (SQL)

SQL é un linguaggio di tipo non procedurale che consente di operare sui dati di un database ricorrendo a sostantivi e verbi comuni nel linguaggio corrente della lingua inglese, come parole chiave, e consente di inserire un'interrogazione dentro un'altra, generando queries nidificate.
SQL possiede buona parte delle caratteristiche dei DML basati sul calcolo relazionale, per cui mostra stretti legami con gli operatori algebrici.
L'istruzione standard per la selezione, nel linguaggio di SQL, che permette l'estrazione di dati da un archivio secondo criteri predefiniti é l'istruzione **SELECT**.
L'utilizzo di tale istruzione permette di definire la sintassi base per una frase di interrogazione in SQL[13].

[12] L'utente che utilizza QBE crea un esempio della risposta che intende ottenere ed il DBMS, attraverso questo DML, identifica tutte le *n*-ple presenti nelle diverse relazioni, che soddisfano i principi definiti tramite l'esempio.
[13] Per convenzione sono scritte in maiuscolo le parole chiave del linguaggio SQL, tra **<** e **>** le descrizioni del tipo di oggetti da utilizzare nei comandi. Le sezioni di comando racchiuse tra **[** e **]** sono da intendersi opzionali. Se le opzioni, o parti del comando, sono separate da **|** significa che non possono essere presenti contemporaneamente (sono mutuamente esclusive).

SELECT <lista di attributi>
FROM <lista di relazioni>
[WHERE <lista di predicati>]

WHERE, in relazione al suo utilizzo é un operatore di selezione che opera allo stesso modo dell'operatore Selezione dell'algebra relazionale.

Un esempio della funzione di selezione di WHERE, se si prende in esame la seguente relazione, é l'interrogazione "elencare tutti i libri pubblicati dall'editore E1", che si traduce con la seguente frase SQL:

SELECT Libri FROM Catalogo WHERE Editore = "E1"

Catalogo

Libri	Autore	Editore	Anno di Pubblicazione
Libro 1	Autore 1	E1	1998
Libro 2	Autore 1	E2	1998
Libro 3	Autore 3	E1	1999
Libro 4	Autore 1	E2	1999
Libro 5	Autore 2	E3	2000
Libro 6	Autore 1	E1	1998
Libro 7	Autore 3	E2	1999
Libro 8	Autore 2	E1	1999

Alla funzione di SELECT, é possibile far eseguire un'azione che permette di ottenere una relazione, in maniera analoga all'operatore dell'algebra relazionale ***Proiezione***.
Un'interrogazione che esemplifica l'operatore Proiezione é "elencare tutti gli autori presenti nel catalogo". Tale interrogazione si traduce con la frase SQL:

SELECT DISTINCT Autore FROM Catalogo

Se, invece, l'interrogazione che intendiamo eseguire é "elencare tutti i libri dell'autore Autore 1 pubblicati dall'editore E2" la frase SQL che si deve scrivere é:

SELECT *[14] FROM Catalogo
WHERE Autore = "Autore 1" AND Editore = "E2"

SQL, con l'utilizzo della clausola FROM é in grado di definire una relazione che é il **Prodotto Cartesiano** di due relazioni di base. Supponiamo ora di avere, oltre alla relazione "Catalogo" utilizzata negli esempi di selezione, anche una seconda relazione:

[14] Il simbolo ***** sta ad indicare che tutti gli attributi della relazione di partenza debbono essere inseriti nella relazione risultante (in questo caso i quattro attributi della relazione "Catalogo").

Autori

Autore	Nazionalità	Ultimo Libro
Autore 1	Italiana	Libro X
Autore 2	Inglese	Libro Y
Autore 3	Francese	Libro Z

Definiamo ora la seguente richiesta "definire una relazione che contenga gli attributi Autore e Nazionalità della relazione Autori e l'attributo Libri della relazione Catalogo per tutti i libri pubblicati da E2", questa formulazione genera la seguente frase SQL:

```
SELECT Autori.Autore, Autori. Nazionalità, Catalogo.Libri
FROM Autori, Catalogo
WHERE Catalogo.Editore = "E2"
```

A completamento dell'elenco degli operatori dell'algebra relazionale, che SQL implementa attraverso le proprie parole chiave, esaminiamo un esempio dell'operatore UNION, che permette di tradurre frasi di interrogazione che comprendano il concetto di ***unione***. Un esempio di frase di richiesta, che sottende il concetto di unione, é "elencare tutti gli autori che hanno pubblicato libri con l'editore E2 o sono di nazionalità italiana", la quale si traduce nella frase SQL:

```
SELECT DISTINCT Autore
FROM Catalogo WHERE  Editore= “E2” UNION
SELECT DISTINCT Autore
FROM Autori WHERE Nazionalità= “Italiana”
```

SQL, attraverso la sintassi del comando SELECT, permette molti altri tipi di operazioni sulle relazioni risultanti, così come può generare relazioni includendo selezioni all'interno di altre selezioni.
Scrivere la frase SQL:

```
SELECT DISTINCT Autore
FROM Catalogo
WHERE NOT EXIST
        (SELECT * FROM AUTORI
         WHERE Catalogo.Libri = Autori. Ultimo_Libro)
```

equivale a dire "estrai l'elenco di tutti gli autori per cui il libro sul catalogo non é l'ultimo libro scritto", il che equivale a dire, esprimendosi con i termini rigorosi dell'algebra relazionale:
"estrai, senza valori doppi, l'attributo Autore delle *n*-ple della relazione Catalogo, per le quali, l'insieme delle *n*-ple della relazione Autori, che hanno lo stesso valore su quell'attributo, é l'insieme vuoto ".
All'operatore SELECT é possibile associare anche gli operatori, come BY GROUP (identifica i diversi sottoinsiemi di valori per le n-ple), cui é associato l'operatore HAVING. L'operatore HAVING permette di esprimere predicati per i gruppi determinati da GROUP BY, in modo del tutto analogo a ciò che permette WHERE per le singole *n*-ple.
SELECT é associabile all'operatore ORDER BY, per consentire che le *n*-ple della relazione risultante siano ordinate, rispetto a uno o ad una combinazione di attributi; l'ordinamento può essere crescente (operatore ASC) o decrescente (operatore DESC).

Queste specifiche permettono di definire la seguente sintassi completa per le frasi di interrogazione:

SELECT [ALL | DISTINCT] <lista di attributi>
FROM <lista di relazioni>
[WHERE <lista di predicati per le n-ple>]
[GROUP BY <lista di attributi>]
[HAVING <lista di predicati per i sottoinsiemi di *n*-ple>]
[ORDER BY <lista di attributi> [ASC | DESC]]

La sintassi del linguaggio SQL che permette la formalizzazione di frasi di interrogazione di database evidenziano il naturale utilizzo di SQL come strumento di estrapolazione di dati.

SQL come DML
Le potenzialità, ampiamente esplorate attraverso l'istruzione SELECT, del linguaggio SQL come DML per i DBMS è espressa dalle funzioni INSERT, UPDATE e DELETE.
La funzione **INSERT** permette di inserire nuove *n*-ple in una relazione esistente.
La sintassi[15] essenziale della funzione INSERT é:

INSERT INTO <nome della relazione>
[«lista di attributi»]
VALUES «lista di valori»

Ad esempio per aggiungere una nuova *n*-pla nella relazione "Autori" si scriverà

INSERT INTO Autori (Autori, Nazionalità, Ultimo_Libro)
VALUES (Autore 4','Italiana','Libro T')

Se nella definizione del comando si omette la lista degli attributi, la lista dei valori viene inserita negli attributi seguendo l'ordine di definizione, all'interno della relazione.
La funzione **UPDATE** permette di modificare i valori esistenti in una o più *n*-ple di una relazione. La sintassi essenziale della funzione UPDATE é:

UPDATE <nome della relazione>
SET
Nome attributo = valore | espressione
[Nome attributo = valore | espressione]
.
.
[Nome attributo = valore | espressione]
[WHERE <elenco di predicati>]

Ad esempio, per aggiornare una *n*-pla nella relazione "Autori", l'ultimo libro dell'Autore 1 é W, si scriverà:

UPDATE Autori
SET Ultimo_Libro = `W`
WHERE Autore = Autore 1'

[15] La parentesi '**(**' e la parentesi '**)**' che compaiono nella definizione sintattica sono parte integrante del comando e non separatori di metalinguaggio.

La funzione **DELETE** permette la cancellazione di *n*-ple da una relazione e la sua sintassi essenziale é la seguente:

```
DELETE FROM <nome della relazione>
[WHERE <lista di predicati>]
```

Ad esempio, per cancellare una *n*-pla nella relazione "Autori", i dati relativi all'Autore 1, si scriverà:

```
DELETE FROM Autori
WHERE Autore = 'Autore 1'
```

Tutte le istruzioni esaminate modificano il contenuto del database. È per questo che SQL rende possibile iniziare una sessione di lavoro (*transazione*) e decidere alla conclusione della sessione di lavoro se confermare i cambiamenti apportati alle relazioni o ripristinare gli archivi come si presentavano all'inizio della transazione.
In particolare questo é possibile attraverso i comandi:

BEGIN TRANSACTION <nome della transazione>
(apre una transazione)
COMMIT TRANSACTION <nome della transazione>
(conferma le modifiche)
ROLLBACK TRANSACTION <nome della transazione>
(annulla i cambiamenti)

SQL come DDL
La capacità di SQL di definire relazioni ed attributi lo rende, un ottimo DDL, con tutti i vantaggi derivanti dalle integrazioni con i linguaggi di alto livello.
I comandi specifici, che sottolineano questa sua potenzialità, sono CREATE, ALTER e DROP.
La funzione **CREATE** crea oggetti di un database (come relazioni o indici).
Per meglio comprendere le azioni rese possibili da tale funzione, è importante esaminare in quale modo SQL operi, per la definizione degli schemi delle basi dati.
SQL dispone di famiglie di domini elementari (domini base) con i quali é possibile definire ulteriori famiglie di domini da utilizzare per associare valori agli attributi delle *n*-ple, delle relazioni della base dati.
Le famiglie di domini elementari sono:

- **Carattere**: stringhe di lunghezza fissa o variabile
- **Bit**: per valori o stringhe di valori (da SQL-2)
- **Tipi numerici esatti**: valori interi e valori decimali a virgola fissa
- **Tipi numerici approssimati**: valori interi e decimali memorizzati in virgola mobile
- **Data e ora**
- **Intervalli temporali** (da SQL-2)

La sintassi del comando CREATE che permette di generare una nuova relazione da aggiungere al DBMS é la seguente:

```
CREATE TABLE <nome della relazione>
(
```

<nome dell'attributo> <tipo dell'attributo> <vincolo dell'attributo> [,<nome dell'attributo> <tipo dell'attributo> <vincolo dell'attributo>]
[
[,<nome dell'attributo> <tipo dell'attributo> <vincolo dell'attributo>]
)

Il tipo dell'attributo può essere scelto tra le famiglie di domini elementari, o generando nuove famiglie attraverso la combinazione delle famiglie elementari.
Per quanto concerne il **vincolo dell'attributo**, specifica le caratteristiche che si intendono attribuire a quell'attributo.
I vincoli che é possibile definire utilizzando i parametri che SQL mette a disposizione sono i seguenti:

- **IDENTITY** [base, incremento]: come valore dell'attributo viene inserito un valore progressivo calcolato a partire da una base e con incrementi definiti.
- **NULL/NOT NULL**: l'attributo considerato può o non può assumere valore nullo.
- **CONSTRAINT** <nome constraint dell'attributo> **UNIQUE** (<nome dell'attributo>): i valori dell'attributo non si possono ripetere in *n*-ple diverse della relazione.
- **CONSTRAINT** <nome dell'attributo chiave> **PRIMARY KEY** <nome dell'attributo>: l'attributo è una chiave primaria (NOT NULL + UNIQUE).
- **CONSTRAINT** <nome foreign key dell'attributo> **FOREIGN KEY** (<nome dell'attributo>) **REFERENCES** <relazione> (<elenco attributi da collegare>): l'attributo è una foreign key[16]. Questo vincolo può essere definito in due modi:
  - con l'uso del costrutto sintattico **REFERENCES** si specifica la relazione esterna a cui l'attributo deve essere legato.
  - con l'uso del costrutto **FOREIGN KEY**.
- **CONSTRAINT** <nome constraint dell'attributo> **DEFAULT** <espressione della costante>: inserisce un valore di default nell'attributo.
- **CONSTRAINT** <nome constraint dell'attributo> **CHECK** <espressione>: esegue un test sul contenuto dell'attributo utilizzando l'espressione fornita.

Attraverso l'utilizzo del comando CREATE é possibile generare indici e viste (Relazioni logiche). Le sintassi per queste tipologie di oggetti é:

```
CREATE [UNIQUE[17]] INDEX <nome dell'indice>
ON <nome della relazione> «lista degli attributi»

CREATE VIEW <nome della vista>
        [ <elenco degli attributi>]
        AS subquery
```

Il comando **ALTER** permette di modificare la definizione di una relazione. In particolare, permette di aggiungere attributi e togliere vincoli agli attributi, con una sintassi che risulta essere:

ALTER TABLE <nome della relazione>

[16] Questo vincolo (chiave esterna) impone che ogni valore dell'attributo di una relazione (se diverso da null) sia presente tra i valori di un attributo delle *n*-ple appartenenti ad un'altra relazione (detta relazione esterna).
[17] UNIQUE significa che l'indice creato è unico.

[ADD[18] <nome dell'attributo> <nome dell'attributo>
[<vincolo dell'attributo>]
[, ...]
[,<nome dell'attributo> <nome dell'attributo>
[<vincolo dell'attributo>]]]
[DROP[19] <predicato drop>]

Il comando **DROP** permette di cancellare oggetti esistenti nel DBMS come relazioni, indici e viste.

DROP TABLE <nome della relazione>
DROP INDEX <nome dell'indice>
DROP VIEW <nome della vista>

È possibile cancellare più di un oggetto utilizzando un solo comando, separando gli oggetti da cancellare, con virgole.

SQL come DCL
Attraverso i comandi GRANT e REVOKE, si esplica la possibilità di usare il linguaggio SQL, come linguaggio per la gestione degli accessi ai dati, e per la loro manipolazione.
Con il comando **GRANT**, il proprietario di un oggetto del DBMS (relazioni, indici e viste) é in grado di accordare o revocare privilegi agli altri utenti, al fine di controllare le possibili operazioni che, ciascun utente é autorizzato ad eseguire.
La funzione GRANT possiede la seguente sintassi:

GRANT ALL | <lista dei privilegi>}
ON <nome della relazione>
[<lista degli attributi>] | <nome della vista>
[lista degli attributi>]
TO <lista degli utenti> | PUBLIC

I privilegi che possono essere accordati (ALL significa accordarli tutti) agli utenti sono:

- **SELECT** (l'utente può interrogare l'oggetto);
- **INSERT** (l'utente può aggiungere *n*-ple all'oggetto);
- **DELETE** (l'utente può cancellare *n*-ple dall'oggetto);
- **UPDATE** (l'utente può modificare *n*-ple dell'oggetto).

Se viene omessa la lista degli attributi il privilegio viene concesso su tutto l'oggetto. Il valore PUBLIC concede i privilegi a tutti gli utenti.

In maniera opposta rispetto al comando GRANT, il comando **REVOKE** revoca i privilegi concessi in precedenza ad uno o più utenti.
Per come é stato definito, la sintassi del comando é la seguente:

REVOKE ALL | <lista dei privilegi>}
ON { <nome della relazione>
[lista degli attributi>] | <nome della vista>

[18] ADD aggiunge attributi e vincoli sugli attributi.
[19] DROP cancella attributi e vincoli sugli attributi.

[<lista degli attributi>]
FROM <lista degli utenti> | PUBLIC

In termine di linguaggio ed in relazione alle funzioni che sono supportate, SQL può essere considerato come:

- **DML (Data Manipulation Language) per le funzioni**
  - ***SELECT***
  - ***INSERT***
  - ***UPDATE***
  - ***DELETE***
- **DDL (Data Definition Language) per le funzioni**
  - ***CREATE***
  - ***ALTER***
  - ***DROP***
- **DCL (Data Control Language) per le funzioni**
  - ***GRANT***
  - ***REVOKE***

3.2 Query By Example (QBE)

Il QBE non é dotato di strutture sintattiche paragonabili a quelle di un linguaggio strutturato di programmazione.
L'utente é chiamato a modellare un esempio, riempiendo uno schema di *n*-pla e rispettando alcune regole e codifiche. In risposta, il sistema estrae tutte le *n*-ple che soddisfano l'esempio modellato, cioè soddisfano tutte le condizione esplicitate. Descriviamo modelli di operazioni eseguibili attraverso il linguaggio QBE e per questo consideriamo di operare sulla seguente relazione:

Catalogo

Libri	Autore	Editore	Anno di pubblicazione
Libro 1	Autore 1	E1	1998
Libro 2	Autore 1	E2	1998
Libro 3	Autore 3	E1	1999
Libro 4	Autore 1	E2	1999
Libro 5	Autore 2	E3	2000
Libro 6	Autore 1	E1	1998
Libro 7	Autore 3	E2	1999
Libro 8	Autore 2	E1	1999

L'utente che ha bisogno di estrarre tutti i libri dell'autore "Autore 1", dovrà eseguire le seguenti operazioni:

- visualizzare lo schema della relazione “Catalogo”;

Libri Autore	Editore	Anno di pubblicazione

- inserire, seguendo le regole del linguaggio, i dati necessari all'interno dello schema della relazione "Catalogo";

Libri	Autore	Editore	Anno di pubblicazione
P.X	**Autore 1**[20]	**P. X**[21]	

- il sistema restituisce la seguente relazione:

Libri	Editore
Libro 1	E1
Libro 2	E2
Libro 4	E2
Libro 6	E1

Lo stesso risultato sarebbe stato fornito dalla frase interrogativa SQL

SELECT Libri, Editore WHERE Autore = 'Autore 1'

Un altro esempio interessante é l'operazione di aggiornamento di una *n*-pla di una relazione utilizzando QBE.
Impostare uno schema come il seguente significa richiedere l'aggiornamento della *n*-pla, che deve soddisfare la prima riga di valori:

Libri	Autore	Editore	Anno di pubblicazione
Libro 1	Autore 1	**C**[22].E2	1998

Lo stesso risultato é ottenibile con il comando SQL:

UPDATE Catalogo SET Editore = 'E2' WHERE Autore = 'Autore 1' And
Libro = 'Libro 1' And Anno di Pubblicazione = '1998'

4. Reti di basi dati e base dati distribuite

L'evoluzione dei DBMS, non ha coinvolto esclusivamente lo sviluppo di nuovi modelli di rappresentazione delle relazioni fra dati, ma ha naturalmente interessato i software di gestione e le strutture hardware di supporto.
Dagli inizi degli anni '70, da quando hanno cominciato il loro cammino evolutivo, notevoli sono stati i cambiamenti e le trasformazioni tecnologiche.

[20] È un valore costante che equivale alla dichiarazione di un criterio di selezione.
[21] **P** é un comando che indica la richiesta di mostrare quell'attributo nella relazione risultante, e **X** é un **valore esempio**, che significa qualsiasi valore assunto dall'attributo.
[22] Il comando **C** indica l'azione di modifica del dato che viene richiesto.
Si può definire la seguente tabella dei comandi:

Comando	Significato	Azione sull'attributo
P	Put	Mostra i valori
C	Change	Modifica i valori
I	Insert	Inserisce nuovi valori
D	Delete	Cancella i valori presenti

Da una parte si é assistito alla crescita esponenziale della capacità di calcolo dei sistemi e, dall'altra, l'affermarsi graduale e costante delle reti di comunicazione e, conseguentemente, dei sistemi distribuiti.
Lo stato dell'arte dei DBMS é conseguenza di due fondamentali linee evolutive di hardware e software:

- la graduale "globalizzazione" dei DBMS, che forniscono versioni omogenee per versioni operanti su PC e su main-frame;
- il consolidamento delle tecniche di trasmissione dati, con il conseguente sviluppo e diffusione delle **reti**, dalle reti locali alle reti geografiche, per giungere ad Internet come rete delle reti.

4.1 Le basi dati e le reti

Il termine **DDBMS** (Distributed Data Base Management System) tende ad identificare le basi dati distribuite in aree geografiche.
I DDBMS si possono applicare a situazioni aziendali o istituzionali, in cui i dati sono memorizzati in località diverse, come avviene per società che hanno diversi stabilimenti, ciascuno dei quali possiede un DBMS attraverso il quale memorizza, elabora e modifica dati.
Un DDBMS non é tuttavia caratterizzato dalla distribuzione geografica delle basi dati, neppure dal fatto che essi siano collegati attraverso una rete di computer che gestiscono gli archivi, ma dal fatto che nella condizione di dati distribuiti il sistema gestore consideri tutte le basi dati appartenenti al sistema stesso come un'unica entità logica.
In particolare, tutte le funzioni di accesso e elaborazione dei dati sono in grado di operare su tutti i dati delle diverse basi dati collegate in rete.
È fondamentale comprendere che, tuttavia, i DDBMS non sono una semplice espansione nello spazio dei DBMS, ma contengono nuovi caratteri innovativi, avendo come obiettivo la realizzazione di un controllo centralizzato delle informazioni e dei flussi informativi.
Bisogna rilevare che, pur permanendo fondamentale l'importanza della figura del *data base administrator* come elemento locale di riferimento per i DBMS, non si riscontra frequentemente una figura analoga gerarchicamente superiore che amministra l'intero sistema distribuito e i compiti che eventualmente é chiamata a svolgere sono più di coordinamento che non di gestione nel senso stretto.
Per i DDBMS diventano notevolmente più rilevanti i problemi di sincronizzazione dei dati, per cui si pone grande attenzione nello sviluppo di sistemi e di applicativi che gestiscano efficacemente la contemporaneità dei processi di accesso e manipolazione dei dati.

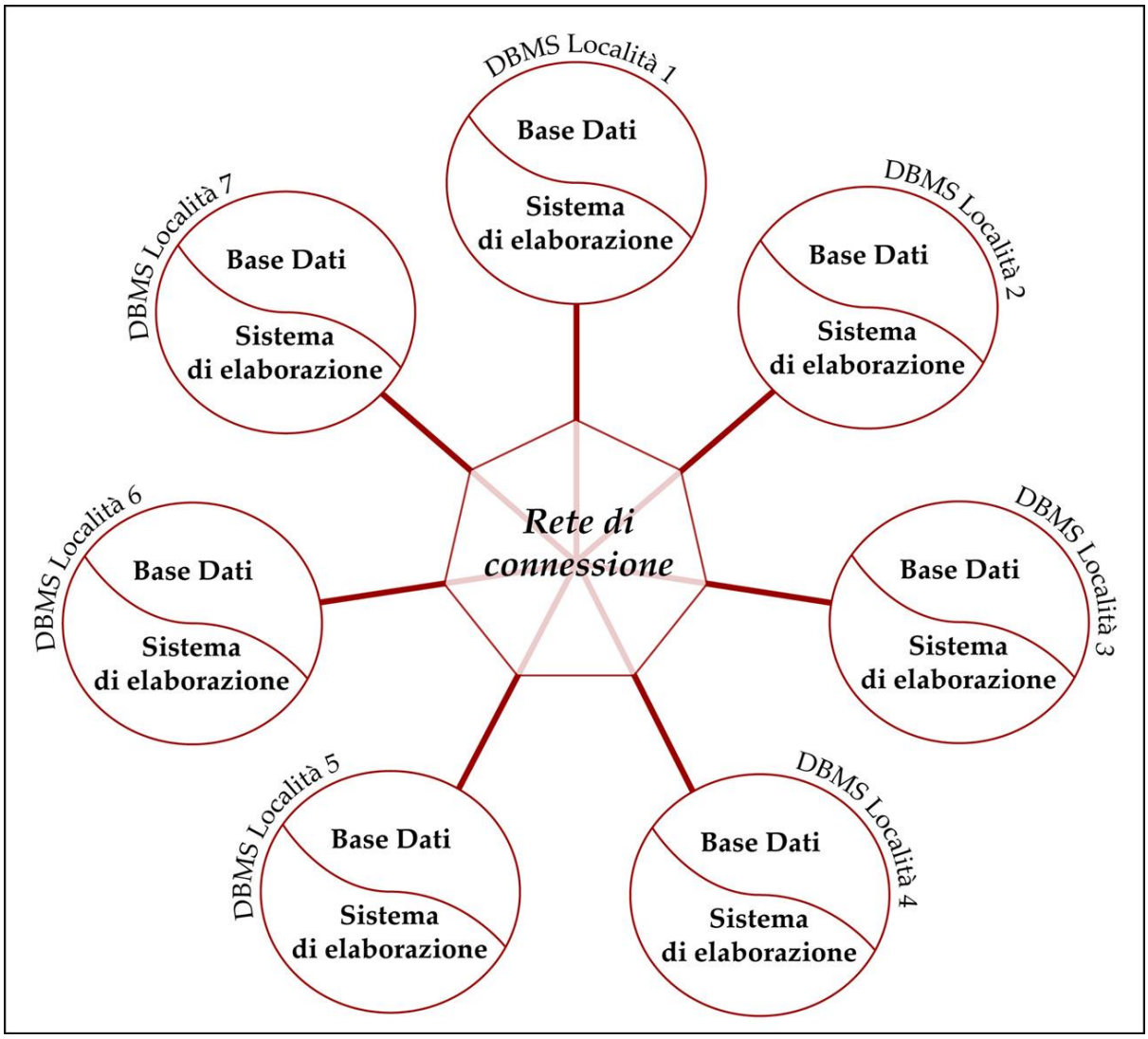

Figura 9. Basi dati distribuite geograficamente

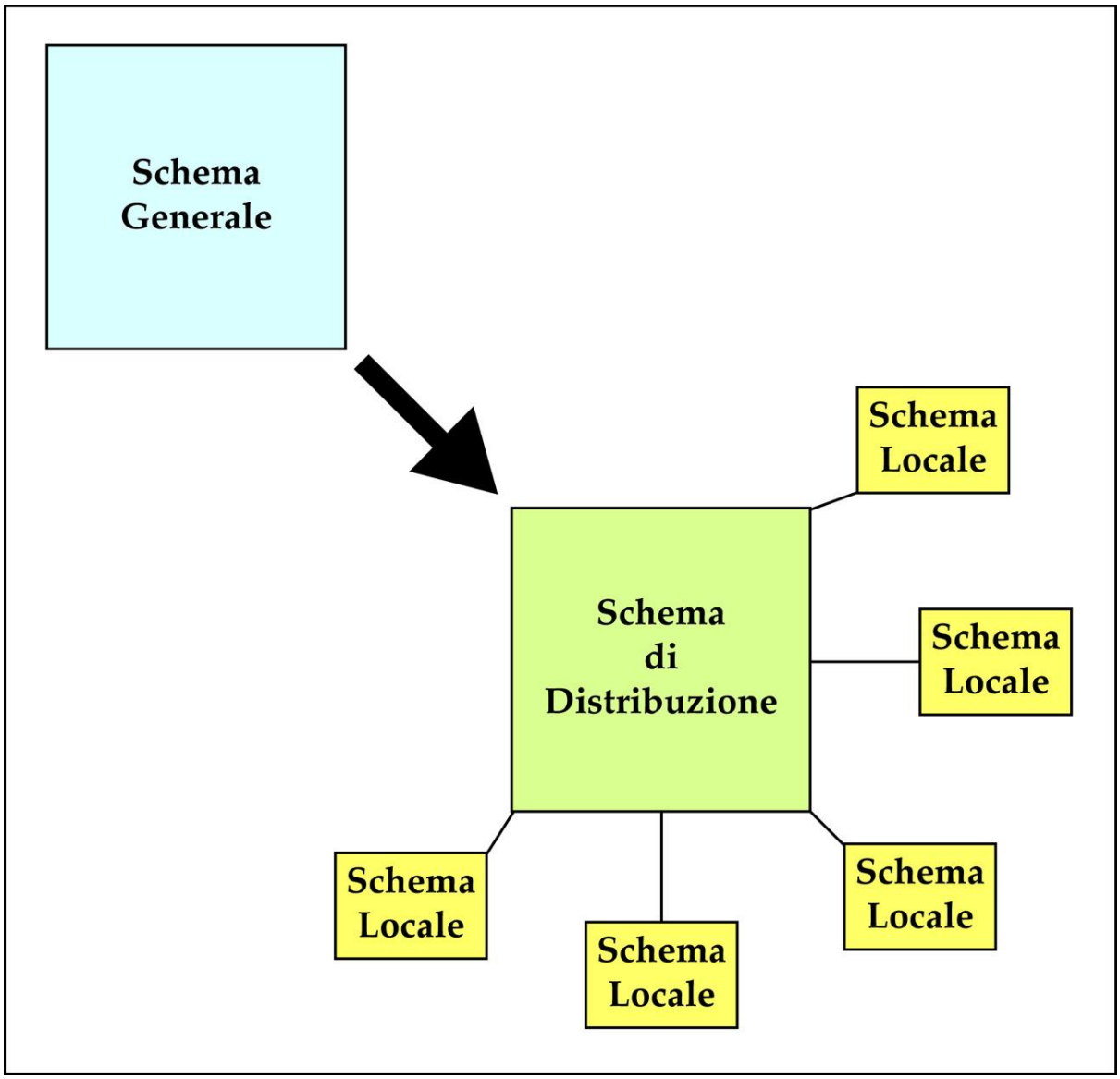

Figura 10. Basi dati distribuite - schema

In analogia con i DBMS é possibile tracciare una struttura gerarchica, a tre livelli, per la rappresentazione dei dati.

- **Schema Generale** (descrizione di tutti i dati a prescindere dalla loro distribuzione);
- **Schema di Distribuzione** (descrizione delle entità come unione di sottoentità, in numero proporzionale al livello di distribuzione dell'informazione, distribuite nei diversi sottoinsiemi);
- **Schema Locale** (uno schema dei dati per ogni sottoinsieme identificato nello Schema di Distribuzione).

4.2 Le basi dati e Internet

Con l'avvento di Internet si é assistito ad un salto generazionale per quanto concerne il mondo della comunicazione.

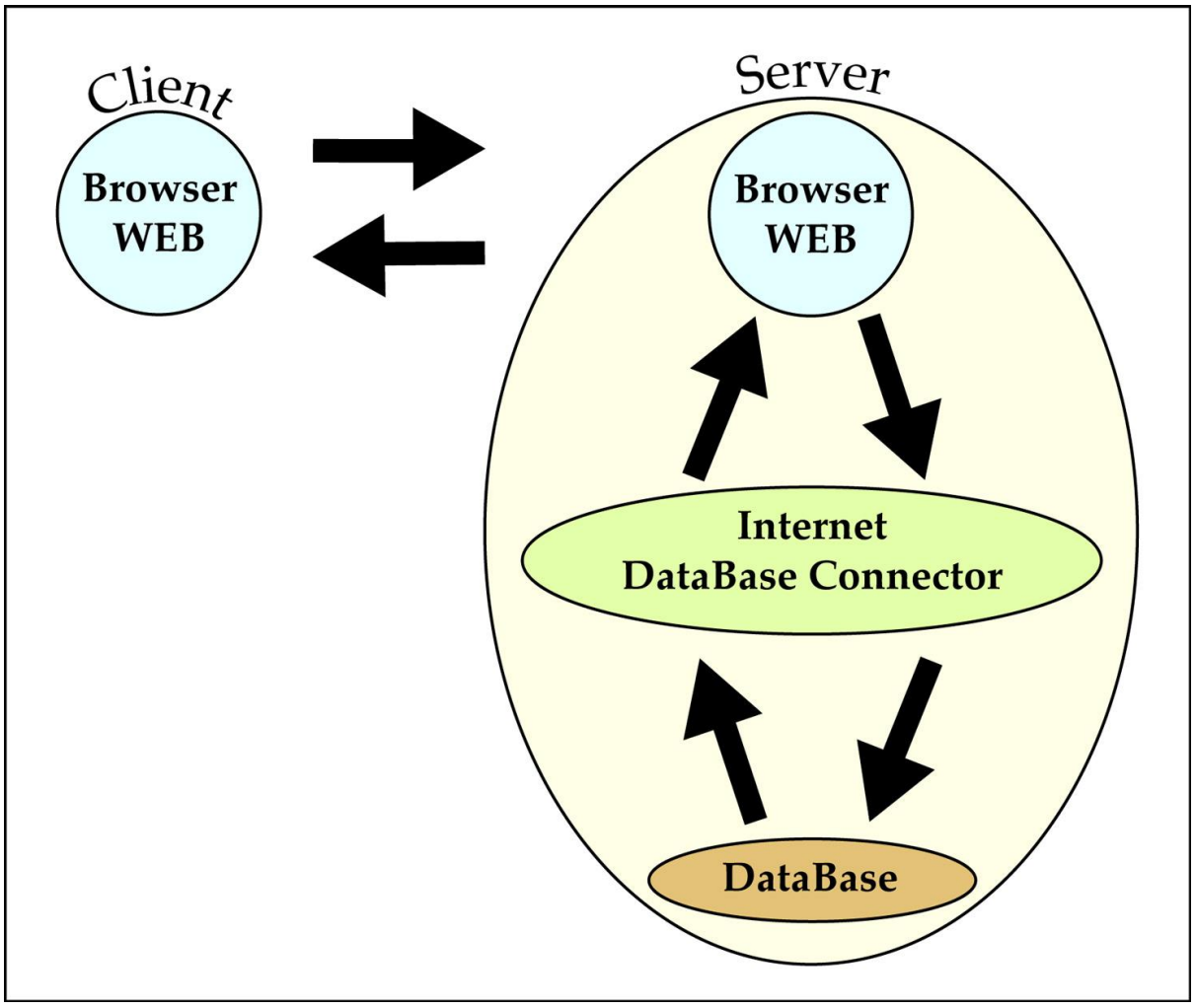

Figura 11. Flusso Data Base su Server - Connessione via browser

L'avvento di Internet ha profondamente modificato le applicazioni informatiche note per le reti e ne ha introdotte di totalmente nuove. Si possono elencare una serie di documenti Web statici che hanno rilevanza per l'utilizzo della rete come, per esempio, orari dei trasporti, cataloghi di prodotti, annunci pubblicitari, notiziari. Con un analogo processo, é possibile produrre una lista di applicazioni Web attive, come acquisto di biglietti on-line, prenotazioni alberghiere e gestione ordini.
Le applicazioni e i documenti enunciati sono strettamente legati ad operazioni di raccolta, memorizzazione e manipolazione di dati.
In questo contesto Internet e le applicazioni basate su Internet, con il supporto dei linguaggi di programmazione ad alto livello, e dei linguaggi specifici per le applicazioni client-server, sono una potente piattaforma per l'implementazione di DDBMS.
Le possibilità di gestire flussi informativi con il protocollo Internet, il linguaggio SQL e i linguaggi di programmazione ad alto livello (come Java), i protocolli di integrazione dati e formato dei documenti (XML) sono la solida base su cui poggia lo sviluppo dei sistemi dati distribuiti e dei dataware-house. In quest'area di sviluppo particolare rilievo hanno assunto linguaggi server-side (ad esempio JSP - Java Server Page) e protocolli di comunicazione come ODBC (Object Data Base Connectivity) che permettono di accedere a basi dati, senza occuparsi della loro strutturazione fisica di memorizzazione.

4.3 Problematiche sull'uso delle basi dati distribuite

Le applicazioni di basi di dati distribuite introducono nuove problematiche laddove la distribuzione dei dati influenza le tecnologie necessarie alla gestione di tali applicazioni, diventandone punti di criticità.
Le tecnologie correlate alle applicazioni di basi di dati distribuite introducono alcune problematiche specifiche:

- controllo delle concorrenze;
- interoperabilità;
- cooperazione tra sistemi;
- ottimizzazione delle interrogazioni.

Controllo delle concorrenze
L'architettura dei sistemi ha tra i nodi principali da affrontare, la gestione delle concorrenze, un problema ad alta complessità. "Gestire le concorrenze" significa regolamentare l'accesso concorrente ai dati da parte di più applicazioni. Tale tipologia di accessi è il punto caratterizzante di tali sistemi.
La concorrenza garantisce la possibilità di una gestione efficiente dei sistemi di dati distribuiti poiché consente la massimizzazione di operazioni eseguite nell'unità di tempo correlata ad una minimizzazione dei tempi di risposta.
La regolamentazione delle concorrenze si basa sul concetto di transazione[23] come operazione atomica[24].
L'accesso al DBMS da parte di più utenti o applicazioni genera la concorrenza e il carico di lavoro si misura in **tps** (*transazioni per second*o o *trasactions per second*).
Il problema viene risolto, in molti casi, utilizzando le proprietà della teoria della concorrenza che dimostrano come validi due metodi:

- locking a due fasi (two phase locking -2PL);
- timestamping.

Il **locking a due fasi** si sviluppa secondo il seguente schema:

- **fase crescente**: acquisizione dei lock[25] per le risorse necessarie;
- **fase calante**: rilascio dei lock acquisiti.

Il **timestamping** è un metodo semplice da implementare per il controllo delle concorrenze che si fonda sull'associazione, ad ogni evento, di un identificativo che permetta un ordinamento temporale degli eventi.
Ad esempio, nei sistemi centralizzati, l'identificativo viene generato attraverso una lettura dell'ora di sistema al momento in cui l'evento si è verificato.

---

[23] Operazione elementare dotata delle proprietà di:

- Consistenza: una transazione non viola le regole di integrità dei data base;
- Isolamento: una transazione può essere eseguita in maniera indipendente dall'esecuzione di altre transazioni;
- Persistenza: una transazione eseguita correttamente produce un effetto che non va perso.

[24] L'**atomicità** di una transazione, consiste nel fatto che la transazione è un'unità indivisibile di esecuzione. Non è possibile abbandonare il data base in uno stato intermedio ed il **rollback** (l'annullamento del risultato e degli effetti della transazione e il recupero dello stato del data base prima dell'esecuzione della transazione) può essere effettuato dalla transazione stessa o dal sistema.
[25] Il **locking**, presente in molti DBMS di tipo commerciale, si basa sul principio che tutte le operazioni di lettura e scrittura si devono proteggere attraverso l'utilizzo di tre primitive *w_lock* (protezione di scrittura), *r_lock* (protezione di lettura), *unlock* (rimozione di un vincolo)

Interoperabilità
Le basi di dati distribuite devono confrontarsi con l'eterogeneità dei sistemi che interagiscono e scambiano informazioni.
Il concetto di eterogeneità mette in evidenza la necessità di rendere possibile l'interazione tra sistemi diversi per struttura e filosofia di gestione dei dati.
Affrontare il problema di interazione e interoperabilità, in questo contesto, è riconducibile al problema di progettare ed implementare sistemi di conversione e scambio di dati e informazioni tra sistemi eterogenei, sistemi che operino attraverso reti informatiche.
In altri campi di applicazione il problema viene risolto attraverso la generazione di protocolli standard come HTTP o FTP per Internet o MIME per i servizi di posta elettronica.
Le applicazioni legate alle basi di dati distribuite vengono utilizzati in larga scala ODBC e X-Open, mentre, per i data base ad oggetti, è utilizzato lo standard CORBA.
Il protocollo ODBC, interfaccia applicativa di Microsoft, attraverso il linguaggio supporta le funzioni di interoperabilità con data base diversi per tipo attraverso l'utilizzo di driver, software specifici che permettono il dialogo con i diversi data base.
Questo sistema è completamente trasparente all'utente e rispetto al protocollo di comunicazione che si utilizza.
Il protocollo X-Open DTP (Distributed Transaction Processing) è un protocollo per l'interoperabilità su DBMS con architetture differenti, che è composto da due interfacce:

1. **client - Transaction Manager** (TM – Gestore delle transazioni);
2. **TM Resource Manager** (RM o Gestore delle risorse), definito anche *Xa-interface.*

Schematicamente il protocollo X-Open DTP è definito da un'architettura che prevede un processo *client*, diversi processi RM e un processo TM.
I DBMS che intendono rendere possibile l'accesso di processi TM, devono essere dotati e rendere disponibile la Xa-interface.

Cooperazione tra sistemi
La catalogazione e l'archiviazione attraverso l'utilizzo di data base era pratica diffusa prima dello sviluppo della rete.
Lo sviluppo delle reti avvenuto negli ultimi anni ha evidenziato la necessità di integrazione tra sistemi informativi autonomi già esistenti, come ad esempio i sistemi informativi bancari o aziendali, rapporti con i fornitori e il B2B[26].
Il concetto di cooperazione, in questo ambito di discussione, è definibile come la capacità di sistemi di operare con funzioni che permettano l'utilizzo di servizi messi a disposizione da sistemi informativi appartenenti a diverse entità. In questo contesto è bene quindi distinguere cooperazione e interoperabilità.
La cooperazione tra sistemi può avvenire a diversi livelli come, ad esempio, la condivisione di servizi basata sullo scambio di informazioni o documenti attraverso l'uso dei servizi di posta o messaggistica, oppure mediante un meccanismo che permetta la visibilità remota dei dati di un sistema da parte di un altro sistema.
La generazione e strutturazione di progetti di cooperazione tra sistemi trova diversi ostacoli dovuti soprattutto ai diversi gradi di eterogeneità che i sistemi presentano sia in termini di struttura che di filosofia di servizi.
Affrontare questi ostacoli per superarli in modo razionale richiede spesso la necessità di processi di razionalizzazione e standardizzazione dei sistemi che il più delle volte hanno buone possibilità di successo solo in presenza di authority che governino tali processi, e

[26] B2B (Business to Business) è il commercio elettronico interaziendale, riservato alle transazioni tra aziende.

che, a volte, hanno l'inconveniente immediato di ridurre l'autonomia di gestione dei singoli sistemi.
I problemi in questo ambito tendono ad emergere in maniera predominante nelle cooperazioni basate sui dati.
Questa tipologia di cooperazione può assumere forme tra loro notevolmente diverse, che possono essere classificate valutando il loro livello rispetto a tre parametri:

- **trasparenza**: la percezione come una unica base di dati, da parte dell'utente, del sistema distribuito di dati;
- **articolazione delle operazioni distribuite**: la definizione di quali operazioni di coordinamento sono necessarie affinché si possano determinare le azioni possibili sulle basi di dati cooperanti;
- **attualità dei dati**: i sistemi cooperanti possono condividere i dati dell'archivio originale o di una copia che potrebbe non essere allineata. L'attualità dei dati è un indice che misura quanto i dati debbano essere recenti perché il sistema di cooperazione funzioni.

Le considerazioni sui parametri di analisi di cooperazioni basate sui dati porta alle definizioni di alcune architetture relative a tali sistemi di cooperazione:

1. **archivi a collezione di dati replicati**: i client accedono con operazioni di sola lettura a dati replicati ottenendo così un'alta trasparenza ma una scarsa attualità;
2. **archivi multidatabase**: i client dialogano con un sistema gestore il quale esegue le operazioni di accesso ai dati degli archivi utilizzando sistemi mediatori installati sui singoli sistemi informativi. Una tale architettura vede sia un alto livello di trasparenza che di attualità, ma parallelamente anche un alto livello di articolazione o complessità;
3. **archivi informativi locali con dati esterni**: in questa architettura i client accedono direttamente ai data base locali. L'architettura così strutturata ha un basso grado di trasparenza e di complessità, con un'attualità dipendete dai singoli sistemi.

Le problematiche affrontate rivelano che il lavoro di raccolta, strutturazione e memorizzazione dei dati sono l'obiettivo principale dei sistemi informatici.
La tecnologia dei data base e dei DBMS garantisce la persistenza dei dati, il loro aggiornamento, l'elaborazione delle informazioni e l'accesso ad esse da parte di utenti ed applicazioni.
La capacità di gestire in maniera efficace il flusso e l'archiviazione di informazioni è fondamentale in relazione alla gestione di sistemi informativi aziendali.
Accade spesso che i dati siano la struttura più persistente di una applicazione rispetto alle procedure che operano su di essi.
Con questa osservazione si intende evidenziare la priorità della progettazione di un efficace modello per i dati nell'ambito dell'automazione dell'archiviazione e della gestione delle informazioni.

BIBLIOGRAFIA DI RIFERIMENTO

I testi che vengono proposti forniscono una gamma di possibilità per conoscere gli strumenti per la progettazione e la realizzazione delle basi di dati.
La conoscenza della teoria e degli strumenti software a disposizione per la realizzazione dei DBMS, consente di attuare una progettazione efficace ed efficiente, corroborata dalla capacità parallela di associare la progettazione alla realizzazione di studi di fattibilità.

BATINI Carlo, DE PETRA Giulio, LENZERINI Maurizio, SANTUCCI Gaetano, *La progettazione concettuale dei dati*, Milano, Franco Angeli, 1986.
Un testo completo che guida alla progettazione, da un punto di vista dell'astrazione e delle relazioni per la definizione di un DBMS. Sono trattate le tecniche e gli approcci teorici per la definizione degli schemi concettuali da applicare alla memorizzazione e alla manipolazione dei dati.

GARBUS Jeffrey, PASCUZZI David, CHANG Alvin, *SQL Server Progettazione database*, Milano, Apogeo, 2000.
Questo libro, scritto per i corsi di certificazione Microsoft, é un'interessante guida alla progettazione di database. In particolare é un interessante esempio di progettazione legata ad uno strumento software concreto. Inoltre é un'ottima introduzione all'utilizzo di SQL come linguaggio per la manipolazione di dati.

STANEK William, *Microsoft SQL Server 2000 Administrator's Pocket Consultant*, Mondadori Informatica, Milano, 2000.
È una guida di riferimento rapido, per la gestione attraverso gli strumenti Microsoft dei flussi informativi aziendali e nelle organizzazioni. In questa guida sono compendiati, tra gli altri argomenti:
elementi di SQL;
gestione della sicurezza dei dati;
funzioni di importazione ed esportazione dati;
replicazione di dati e creazione di tabelle ed indici.

BUYENS Jim, *Sviluppo di Database per il Web*, Milano, Microsoft Press - Mondadori Informatica, 2000.
Il crescente interesse rivolto ai database in rete, ne ha moltiplicato il numero e la complessità. In modo analogo si é, notevolmente alzato il livello di prestazioni che é richiesto, poiché in maniera sempre più evidente si é alzato il livello di competenza di utenti e fornitori di servizi. In quest'ottica con un approccio graduale si é introdotti alle tecnologie ADO (Active Data Object), ASP (Active Server Page), HTML e XML.

DYCHÉ Jill, *e-Data*, Milano, Apogeo, 2000.
Il libro é un ottima fonte di informazioni per quanto concerne Data Warehousing. In particolare analizza il suo sviluppo e il suo valore strategico, le tecnologie che supporta e la sua influenza sulle procedure di Decision Making.

ATZENI Paolo, *Basi di dati. Modelli e linguaggi di interrogazione*, Milano, McGraw-Hill Companies, 2009.
Un libro che riassume in forma aggiornata i principi di progettazione di base di dati in relazione al loro utilizzo.

ESERCIZI

Esercizio 1
Quali sono stati gli standard, considerati indispensabili per una base di dati, definiti dal Data Base Task Group, e quali conseguenze comportano, nella definizione dei dati e delle funzioni per la manipolazione degli archivi?

Esercizio 2
Quali sono i livelli di rappresentazione schematica di un Data Base Management System e quali informazioni forniscono sui dati?
In particolare che informazioni fornisce al DBMS lo Schema Logico e a quale livello di rappresentazione schematica appartiene?

Esercizio 3
Definire e descrivere, da un punto di vista funzionale, quali software di gestione comprende un DBMS e specificare che tipi di linguaggi si possono utilizzare come Data Manipulation Language (DML).

Esercizio 4
Dare una definizione delle seguenti strutture fisiche di memorizzazione, utilizzate dai DBMS, per l'immagazzinamento dei dati
1. struttura sequenziale;
2. struttura concatenata;
3. struttura indicizzata;
4. struttura a lista.

Esercizio 5
Quali sono le caratteristiche e gli elementi base del modello gerarchico dei dati?
Definire una struttura gerarchica di esempio e descrivere il problema che ha portato, per il modello gerarchico, alla definizione di genitore logico. Come questo elemento risolve il problema per cui é stato generato?

Esercizio 6
Definire e mettere in relazione tra loro il modello gerarchico e il modello relazionale.
In particolare, dare una descrizione dettagliata del SET di un modello gerarchico ed elencare le caratteristiche necessarie affinché si possa parlare, nel modello relazionale, di dati normalizzati.

Esercizio 7
Descrivere e dettagliare con esempi i processi di conversione tra il modello reticolare ed il modello relazionale, e tra il modello relazionale ed il modello gerarchico.

Esercizio 8
Quali sono e che caratteristiche distinguono i due principali linguaggi per la manipolazione dei dati?
Se entrambi i linguaggi permettono l'interrogazione di archivi, mostrare attraverso un esempio, in che modo specifico ciascuno di essi adempie alla funzione.

Esercizio 9
Considerando gli operatori dell'algebra lineare, definire quelli di selezione e darne una definizione rigorosa. Preso poi come riferimento la seguente relazione

Itinerari

P a r t e n z a	A r r i v o	Interesse	D u r a t a
Località 1	Località 4	Archeologico	2 gg
Località 2	Località 3	Naturalistico	5 gg
Località 4	Località 2	Faunistico	1 gg
Località 2	Località 1	Ambientale	3 gg
Località 1	Località 6	Naturalistico	4 gg
Località 1	Località 3	Ambientale	1 gg

Scrivere un esempio per ogni operazione di selezione.

Esercizio 10

Quali sono i comandi che caratterizzano SQL come DCL (Data Control Language)?
Per ciascuno scrivere il paradigma sintattico, cioè la struttura che deve rispettare per essere compreso ed eseguito.