Approfondimento: i sistemi di gestione delle basi di dati (DBMS)

Prerequisito essenziale della funzionalità delle basi di dati è il controllo e la fruibilità dell'informazione in esse contenuta: a tale scopo, la nascita dei database è stata accompagnata fin da subito dallo sviluppo e dall'implementazione di programmi preposti alla memorizzazione, modifica ed estrazione delle informazioni contenute nei database. L'utilizzo di tali **sistemi di gestione delle basi di dati** (**DBMS, *Data Base Management System***) garantisce quindi di poter compiere quelle operazioni (non supportate dalle basi di dati, che utilizzano semplicemente *flat file)* indispensabili per una gestione e una fruizione ottimale dei dati. Tra le funzionalità fornite dai DBMS possiamo citare:

- **Affidabilità e persistenza dei dati**. I DBMS forniscono solitamente specifiche funzionalità di salvataggio (*backup*) e ripristino (*restore*) dei dati, conservando intatto il contenuto d'informazione del database e limitando il verificarsi di eventi che possano accidentalmente modificare tale contenuto.

- **Condivisione dei dati**. I DBMS riuniscono l'informazione in un sistema di amministrazione unico, per poi permettere l'accesso e la condivisione dei dati a diverse applicazioni e a diversi utenti, evitando possibili problemi di **ridondanza** (duplicazione del dato in seguito a richieste contemporanee di più utenti) e **inconsistenza** del database (qualora fossero presenti copie multiple di una base di dati, è possibile che la parziale modifica dei dati di una sola di esse possa generare un disallineamento dei contenuti).

- **Controllo dell'accesso ai dati**. I DBMS forniscono solitamente alcuni sistemi di controllo dell'accesso ai dati da parte degli utenti, per evitare che possa essere ottenuta o modificata informazione da parte di persone non autorizzate.

L'architettura di un DBMS può essere schematizzata in tre livelli tra loro connessi: **livello fisico**, **livello logico** e **livello esterno** (Figura 1): il primo, invisibile all'utente, si occupa essenzialmente della memorizzazione e dell'organizzazione dei dati nei file; il secondo, anch'esso invisibile all'utente, si occupa della rielaborazione dei dati in base al modello logico (gerarchico, reticolare, relazionale o a oggetti; Paragrafo 1) adottato dal database; il terzo livello

rappresenta l'interfaccia visibile all'utente attraverso la quale è possibile consultare, memorizzare e modificare (qualora si disponga delle autorizzazioni necessarie) l'informazione presente nella base di dati. Il terzo livello si occupa inoltre della comunicazione con programmi esterni che interagiscono con il DBMS per ricavare le informazioni.

Progettare, realizzare e mantenere un DBMS è un processo complesso e laborioso; ciononostante, accanto ai prodotti commerciali (come *Access* e *SQL Server* di Microsoft o *ORACLE* della ORACLE Corporation), alcuni DBMS vengono rilasciati gratuitamente con un tipo di licenza che permette a eventuali sviluppatori di modificare e ridistribuire liberamente il software (***open source software***): tra questi citiamo *MySQL*, *PostgreSQL* e *InterBase*.

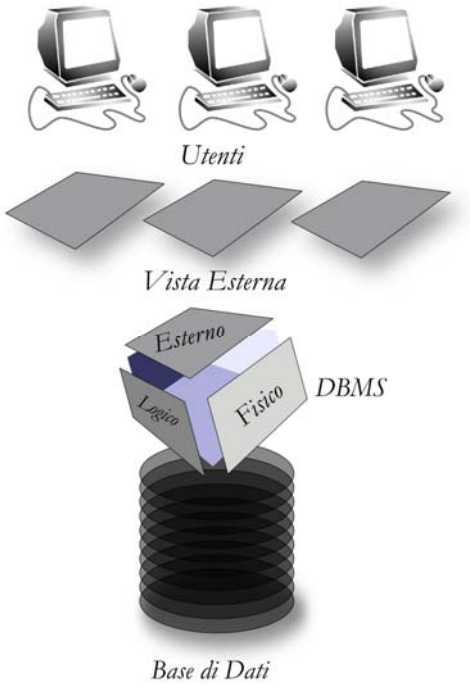

Figura 1 Struttura di un DBMS.

1. Struttura e modelli logici dei DBMS

Sebbene i dati contenuti in un DBMS possano essere, in linea di principio, di qualsiasi tipo, solitamente in relazione all'insieme dei concetti utilizzati per organizzare i dati di interesse e descriverne la struttura (**modello dei dati**) vengono individuati quattro insiemi distinti di basi di dati: i database, infatti, possono essere **gerarchici**, **reticolari**, **relazionali** o **a oggetti**.

Nelle basi di dati gerarchiche i dati sono organizzati secondo una struttura ad albero: a partire da un dato "padre" è possibile accedere a dati "figli", i quali a loro volta possono rappresentare i padri di altri dati (*parent/child relationship*). Questo tipo di architettura limita solitamente il numero possibile di relazioni che possono sussistere tra i dati stessi. Tutti i record vengono riportati in una lista (più precisamente una "lista collegata", o *linked list*, per via dei collegamenti con altri record) sui campi della quale (nodi) sono riportati gli attributi del record. Ogni singola lista può essere collegata a diverse altre liste attraverso una relazione di tipo "uno a molti".

Un esempio di base di dati gerarchica può essere fornito dalla struttura dei corsi di laurea universitari (Figura 2): in questo caso, è possibile avere come singoli record i corsi di laurea offerti da una struttura universitaria. Gli attributi dei record potrebbero essere costituiti dalla durata del corso, dal numero e dalla tipologia di esami previsti. A questa struttura sarebbe possibile associarne altre che contengono il numero di iscritti per ogni corso di laurea (record), il loro nome, il cognome, l'età e l'indirizzo (attributi). Infine, a ciascuno studente sarebbe possibile assegnare ulteriori strutture, per esempio il piano degli studi. I record di una base di dati di questo tipo verrebbero quindi a formare una relazione gerarchica, in cui un corso di laurea (*parent*) può annoverare diversi iscritti (*child*), ma ogni iscritto può appartenere a un unico corso di laurea.

Nate intorno al 1970 (il primo database gerarchico, **Information Management System** o **IMS**, fu elaborato dalla IBM nel 1966 per catalogare le componenti del veicolo spaziale Apollo), le basi di dati gerarchiche furono presto sostituite da altre architetture, principalmente a causa delle numerose limitazioni imposte dal modello gerarchico stesso: l'implementazione di una tale struttura è spesso complessa e può risultare in una forzatura del modello reale che il database tenta di descrivere.

Ogni volta che la relazione uno a molti non è applicabile, si rende necessaria la sostituzione del modello gerarchico con quello **reticolare**. Questo modello nacque proprio grazie allo sforzo compiuto da **Charles Bachman** nel superare le limitazioni della relazione uno a molti tipica del modello gerarchico e di introdurre

basi di dati caratterizzate da relazioni “molti a molti”. In tali relazioni, l’albero del modello gerarchico prende la forma di una griglia di connessioni tra i record delle diverse liste. Nel modello reticolare i record sono quindi collegati tra loro attraverso particolari campi, detti **connettori** o **puntatori**, che permettono all'utente di accedere ai dati più facilmente senza dover sottostare ai vincoli rigidi della struttura gerarchica. Ogni connettore può essere il punto di partenza per raggiungere un determinato campo. A titolo di esempio, una base di dati reticolare può essere utilizzata per modellare le connessioni tra insegnanti, studenti e classi di un corso universitario: ogni insegnante può impartire lezioni in numerose classi, popolate da studenti che a loro volta possono seguire le lezioni di altri insegnanti (Figura 3).

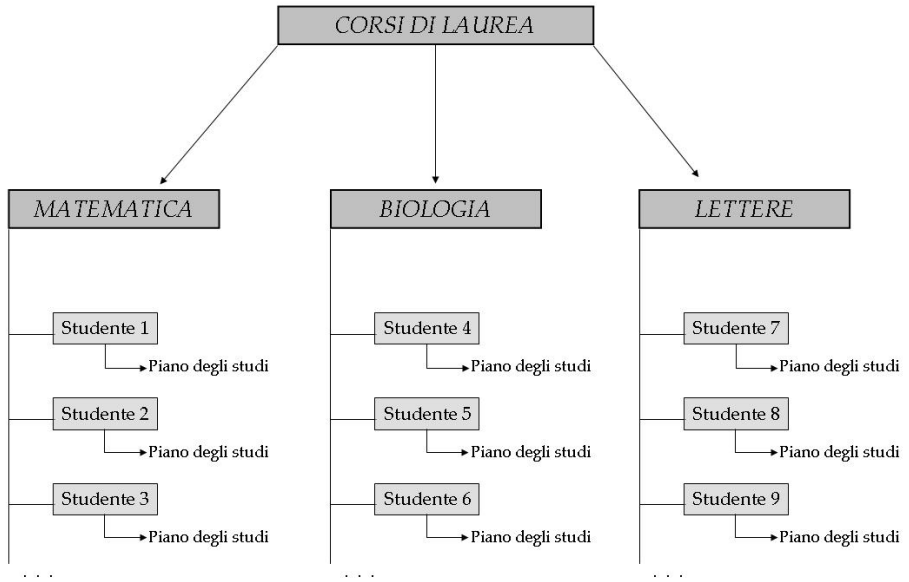

Figura 2 Un esempio di base di dati gerarchica può essere fornito dalla struttura dei corsi di laurea universitari. Il primo livello è costituito dai corsi di laurea di un ateneo. Il secondo livello è invece rappresentato dagli iscritti a ogni corso. Infine, a ciascuno studente è associato un terzo livello di dati, il piano degli studi. I record di una base di dati di questo tipo formano quindi una relazione gerarchica, nella quale un corso di laurea può annoverare diversi iscritti, ma ogni iscritto può appartenere a un unico corso di laurea.

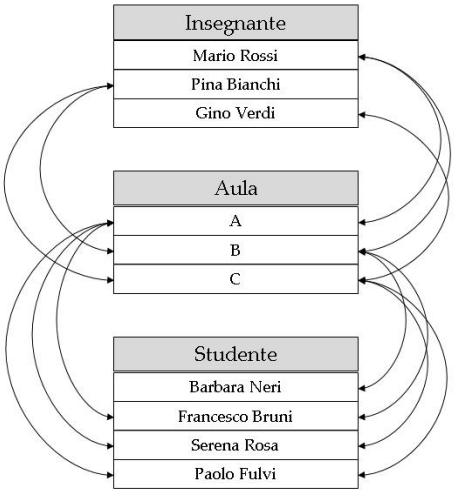

Figura 3 Un esempio di base di dati reticolare che descrive le relazioni tra insegnanti, studenti e classi di un corso universitario. Nel modello reticolare i record sono collegati tra loro attraverso puntatori (frecce).

La teoria dei **database relazionali** venne presentata per la prima volta da Edgar F. Codd nel 1970 nell'articolo *A Relational Model of Data for Large Shared Data Banks*. Il lavoro di Codd prese spunto dalle critiche rivolte all'approccio reticolare, considerato inefficiente e complesso (il numero di connessioni cresceva esponenzialmente al crescere della quantità dei dati), e dalla necessità di favorire l'indipendenza dei dati presenti in un database. Il fulcro del sistema relazionale è il **collegamento**: per ogni record di una **tabella** si definisce una **chiave primaria** (*primary key*), cioè un identificatore univoco della riga. La chiave può essere uno dei dati stessi che vengono memorizzati o un campo aggiunto specificatamente allo scopo (spesso chiamato OID, *Object IDentifier*). La ricostruzione delle relazioni avviene grazie alla corrispondenza delle chiavi appartenenti a tabelle diverse. La Figura 4 illustra un esempio di database relazionale: la tabella *Docenti* contiene i record relativi ai docenti e ai loro dati anagrafici, identificati univocamente da una chiave (*Chiave 1*); la tabella *Corsi* contiene invece gli insegnamenti tenuti all'interno di un corso di laurea, con le informazioni relative al corso e una chiave univoca (*Chiave 2*). Le relazioni tra docenti e corsi sono riportate, attraverso le rispettive chiavi, in una terza tabella (*Aule*), che contiene le informazioni relative alle aule in cui sono impartite le relative lezioni.

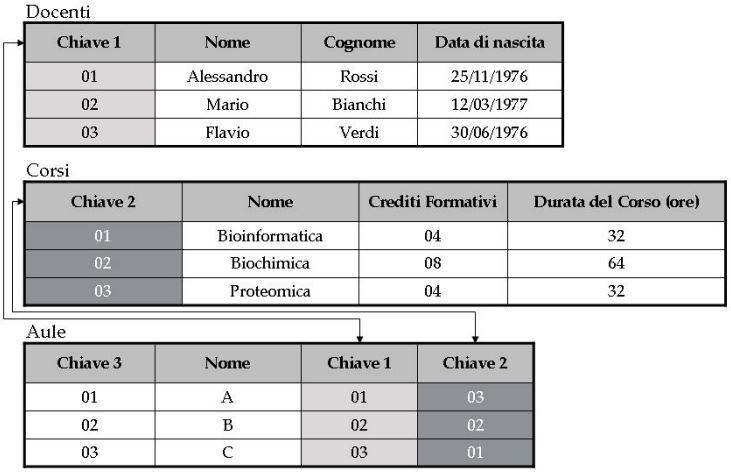

Figura 4 Un esempio di base di dati relazionale che descrive le relazioni tra docenti, insegnamenti e aule di un corso universitario. Nel modello relazionale, per ogni record contenuto in una tabella si definisce una chiave primaria (primary key), cioè un identificatore univoco della riga. La tabella Docenti contiene la chiave primaria Chiave 1; la tabella Corsi contiene invece la chiave primaria Chiave 2. Le relazioni tra docenti e corsi sono riportate, attraverso le rispettive chiavi, nella tabella Aule.

I database relazionali rappresentano attualmente le architetture più diffuse, grazie ai numerosi vantaggi offerti rispetto ai modelli gerarchico e reticolare, primo tra tutti l'**indipendenza logica** dai dati, ossia la possibilità di modificare le relazioni sussistenti tra i dati senza dover cambiare la logica e il codice delle applicazioni che li manipolano. Per questa ragione alcuni centri che storicamente hanno utilizzato *flat file* per la conservazione dei dati (EBI, NCBI o SIB solo per citare alcuni tra i più importanti) hanno sviluppato recentemente anche modelli relazionali dei loro database (si veda il Paragrafo 2.2.1).

Infine, negli anni recenti la rapida diffusione di linguaggi di programmazione **orientati agli oggetti** (come C++ e Java), il cui **codice sorgente** (l'insieme delle istruzioni utilizzate per realizzare un programma informatico) è costituito da oggetti "virtuali" che modellano gli attributi e i comportamenti di oggetti o concetti reali, ha favorito lo studio e l'implementazione di **database a oggetti**, nei quali il singolo record non è costituito esclusivamente da dati, ma contiene anche le istruzioni e le operazioni che possono essere effettuate sui dati stessi. Per esempio, l'oggetto *Alunno 1* presente nella "classe" *Alunni* potrà contenere i dati anagrafici

dell’alunno e un’operazione *visualizza indirizzo* che fa riferimento all’oggetto *Indirizzo 1* di una seconda classe (Figura 5).

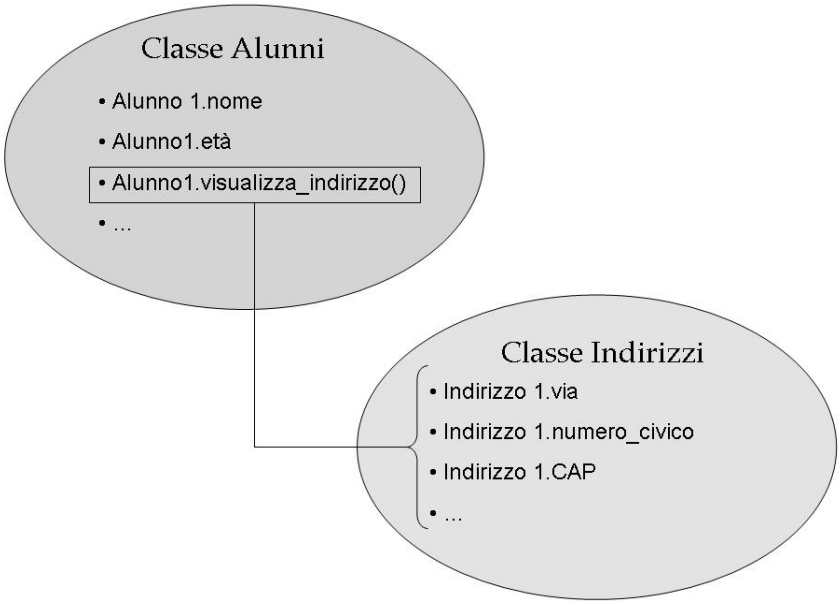

Figura 5 Un esempio di base dati a oggetti. L'oggetto Alunno 1 presente nella “classe“ Alunni contiene i dati anagrafici dell’alunno e una funzione visualizza indirizzo che fa riferimento all'oggetto Indirizzo 1 di una seconda classe.

Contenendone la struttura e il supporto, i database a oggetti stanno seguendo oggi la rapida ascesa dei linguaggi di programmazione a oggetti. In particolare, le basi di dati biologici si prestano molto bene a essere modellate secondo una tale struttura logica, in grado di rappresentare mappe genetiche, geni o proteine come oggetti ai quali viene associata una serie di funzioni per l’analisi e la visualizzazione e di attributi (come i numeri d’accesso e i riferimenti bibliografici). A tale scopo, L’*Object Management Group* (OMG, un consorzio internazionale nato con l’obiettivo di promuovere la teoria e l’utilizzo degli oggetti nello sviluppo di applicazioni informatiche) ha fondato un gruppo di ricerca sulle scienze della vita (*Life Science Research Group*) che si occupa, tra l’altro, di rappresentare la struttura e le relazioni dei dati biologici secondo un insieme di procedure *standard*,

definite nel linguaggio di modellazione unificato (**UML**, *Unified Modeling Language*).

2. Interrogazione in rete dei DBMS: un esempio

Per comprendere meglio come gli elementi che abbiamo descritto interagiscano fra loro nel processo di interrogazione in rete di una base di dati, illustreremo qui un esempio. Nonostante le modalità di conservazione, organizzazione e visualizzazione dei dati, i linguaggi di interrogazione e le applicazioni utilizzate possano cambiare a seconda del database interrogato, i principi generali alla base di tale processo sono i medesimi.

Supponiamo di visualizzare sul nostro browser un sito (che chiameremo www.proteine.org) attraverso il quale è possibile interrogare una base di dati proteica molto semplice, gestita da un DBMS Access (Figura 6). La pagina principale (*homepage*) di questo sito è un file di testo (*index.html*) il cui contenuto è il seguente:

```
<HTML>
<HEAD>
<TITLE>Homepage di proteine.org</TITLE>
</HEAD>
<BODY>
<H2>Benvenuto nel sito proteine.org. Inserisci nel campo sottostante il numero di accesso della proteina della quale vuoi conoscere la sequenza</H2><br>
<form action="risultati.asp" method="post">
Il tuo numero di accesso: <input type="text" name="accession_number" size="20">
<input type="submit" value="Invio">
</form>
</BODY>
</HTML>
```

Questo file sarà mostrato dal browser come in Figura 7.

ID	numero_accesso	sequenza
1	17262772	ADRAFAESCNMKLAIANIMSFERQWPLIMNACSFERMNCNCVSFDGERIP
2	24353721	FDRAFSSCNMKLFNMSFRMSFERQWPLIMLLLIIMANDERNHLAMALA
3	37383838	QRAWEWRMSFERHDNAMANSNKAKSCELIIMANDE
4	42624644	DRAFSSCIANIMSFHHHILARETERETEFARADEADQEQCAVDMMDL
5	75753755	KKAKAKAKSCERECQRAWEWMNVPPALAPLPPPERLRMNQRET
6	24242426	CNCMCMMDASEQWRTYILKHFGFHDNAMANSNPPPEREFCFFAC

Figura 6 Un esempio banca dati di sequenze proteiche molto semplice. A ogni sequenza è associata una chiave primaria (ID) e un numero di accesso.

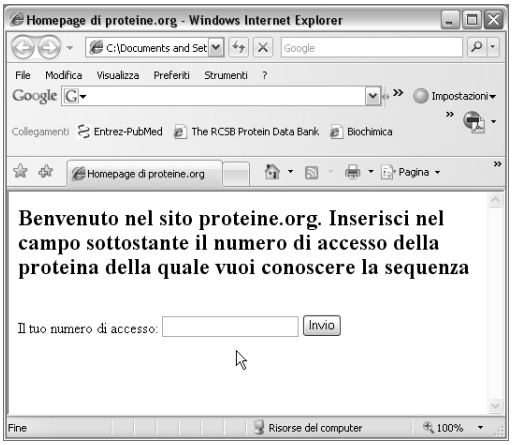

Figura 7 L'homepage del sito (di nostra invenzione) www.proteine.org.

L'utente inserisce nel *form* il numero di accesso della proteina di interesse (che, per semplicità, consideriamo noto); al momento della pressione del tasto *Invio*, tale numero viene inviato al server indicato come variabile *accession_number*; il server esegue quindi il seguente codice ASP, contenuto nel file *risultati.asp*, indicato precedentemente dal tag <form></form> all'interno di *index.html* e riportato di seguito:

```
<HTML>
<HEAD>
<TITLE>Homepage di proteine.org</TITLE>
</HEAD>
<BODY>
La tua sequenza:
<%  'Qui inizia il codice ASP
```

```
'Nella riga seguente il numero di accesso inserito viene salvato nella
variabile acc_num
acc_num=Request.Form("accession_number")
'Nelle righe seguenti viene effettuato l'accesso al database:
Dim conn
Set conn = Server.CreateObject("ADODB.Connection")
conn.Open"driver={MicrosoftAccessDriver(*.mdb)};dbq="&Server.MapPath("dat
abase.mdb")
Dim rs
Set rs = Server.CreateObject("ADODB.Recordset")
'Nella riga seguente il database è interrogato tramite SQL:
rs.Open "SELECT ALL FROM database WHERE numero_accesso = '"& acc_num
&"'", conn
'Nella riga seguente viene visualizzata la sequenza
response.write rs("sequenza")
%>  'Qui finisce il codice ASP
</BODY>
</HTML>
```

Il codice appena descritto si compone di quattro parti fondamentali:

- **Salvataggio delle variabili introdotte dall'utente**. Il numero d'accesso inserito dall'utente viene salvato nella variabile `acc_num` attraverso l'istruzione:
  `acc_num = Request.form(accession_number)`.
- **Accesso al database**. I dettagli di questo passaggio sono stati, per semplicità, omessi.
- **Interrogazione del database tramite SQL**. Attraverso la dichiarazione `SELECT ALL FROM database WHERE numero_accesso = '& acc_num &'` viene selezionato il record del database che presenta un numero d'accesso uguale a quello memorizzato nella variabile `acc_num`.
- **Visualizzazione dei risultati**. La sequenza viene stampata sullo schermo dell'utente grazie all'istruzione `response.write rs(sequenza)` (Figura 8).

Come già sottolineato, queste semplici operazioni, anche se con dettagli differenti, sono alla base delle complesse richieste che possono essere effettuate nelle moderne basi di dati biologici.

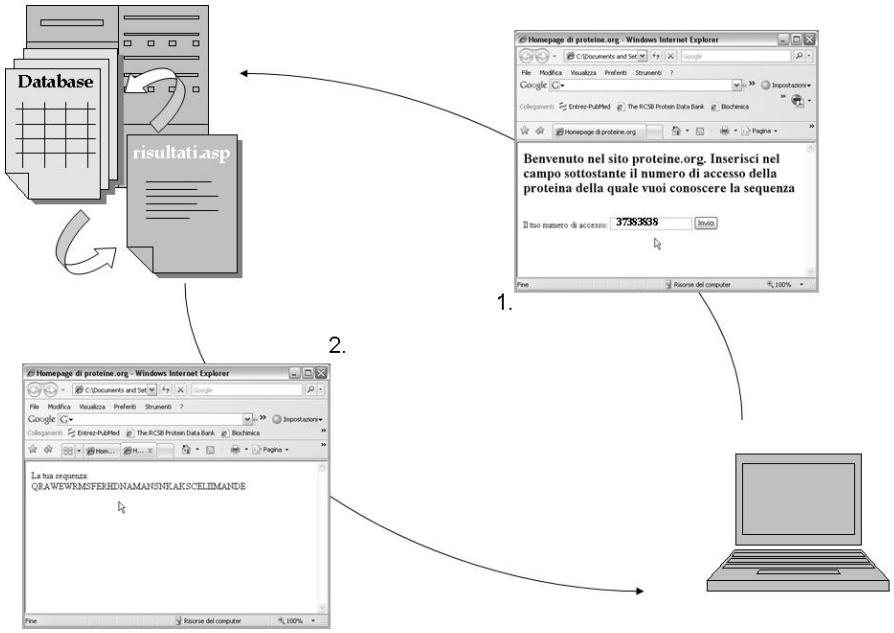

Figura 8 Schema di comunicazione tra server e client. Quando l'utente si collega all'indirizzo del server viene visualizzata la pagina *index.html* (1), all'interno della quale è presente un campo attraverso cui è possibile interrogare una base di dati proteica. L'utente inserisce il codice di accesso e preme il tasto Invio; tale codice viene inviato al server, che esegue il codice ASP contenuto nel file *risultati.asp*. Il codice è composto da quattro parti principali: salvataggio del codice introdotto dall'utente, accesso al database, interrogazione del database tramite SQL e visualizzazione dei risultati (2).