18. TECNICHE DI ACCESSO AI DATABASE IN AMBIENTE INTERNET

Ipotesi di partenza: concetti di base del networking

Le ipotesi di partenza indispensabili per poter parlare di t**ecniche di accesso ai database in rete** e **programmazione lato server** sono le seguenti:

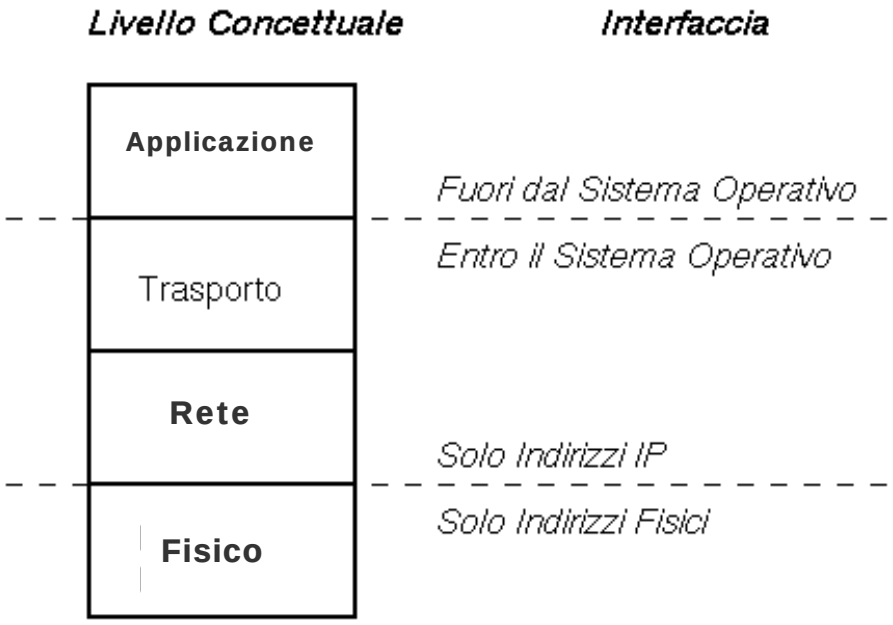
1) consideriamo il modello concettuale di **networking** (ovvero tutto ciò che riguarda la comunicazione tra reti diverse) semplificato a 4 livelli;
2) consideriamo i **protocolli** di comunicazione della famiglia **TCP/IP**, il **browser** come client universale ed il **Web server** come server universale.

Modello concettuale di networking semplificato a 4 livelli

Sappiamo che il **modello OSI** ***(Open System Interconnection)*** è il classico *modello concettuale di networking* a **sette livelli o strati** sviluppato dall'**ISO** ***(International Standards Organization).***
Il modello OSI descrive il flusso dei dati tra il collegamento fisico della rete e l'applicazione dell'utente finale. Ogni livello o strato è responsabile di una determinata funzione.
Inoltre tale modello concettuale è **indipendente** da specifici hardware e software e fornisce un'architettura da rispettare nell'organizzazione dei servizi di rete.

Generalmente **non vi è una corrispondenza biunivoca** tra i livelli del modello OSI ed i livelli di un modello realmente implementato per i servizi di rete.

Un classico esempio è quello fornito dal **modello di rete TCP/IP** che può essere rappresentato con un **modello semplificato a 4 livelli**.



In questo modello semplificato:
- il livello ***Applicazione*** ingloba le funzionalità di tre differenti livelli del modello OSI e precisamente i livelli ***Applicazione, Presentazione*** e ***Sessione***;
- il livello ***Fisico*** ingloba le funzionalità di due differenti livelli del modello OSI e precisamente i livelli ***Data Link*** *(o Collegamento Dati)* **,**e ***Fisico***;

Modello di Riferimento
Open Systems Interconnect
(International Standards Organization)

7	Applicazione
6	Presentazione
5	Sessione
4	Trasporto
3	Rete
2	Data Link
1	**Fisico**

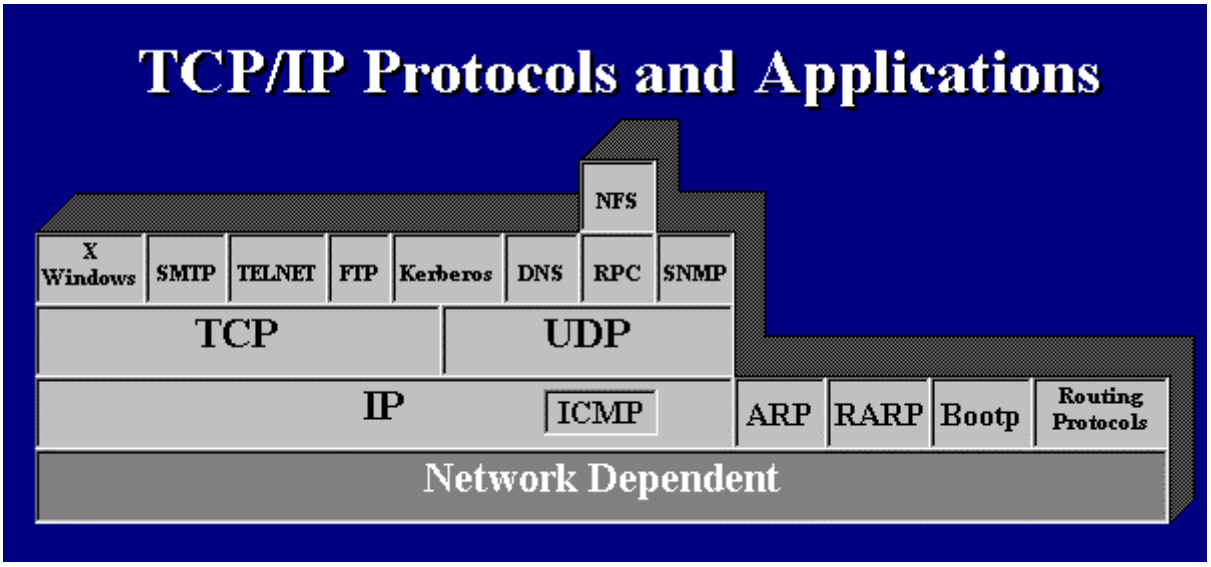
Confronto tra i modelli

OSI	TCP/IP
Application	Application
Presentation	
Session	Transport
Transport	
Network	Network
Data link	Physical
Physical	

Dettagli riguardo alla comunicazione tra due computer **A (mittente)** e **B (destinatario)** secondo il **modello concettuale di networking OSI** nell'ipotesi che **A** invii una comunicazione via rete a **B.**

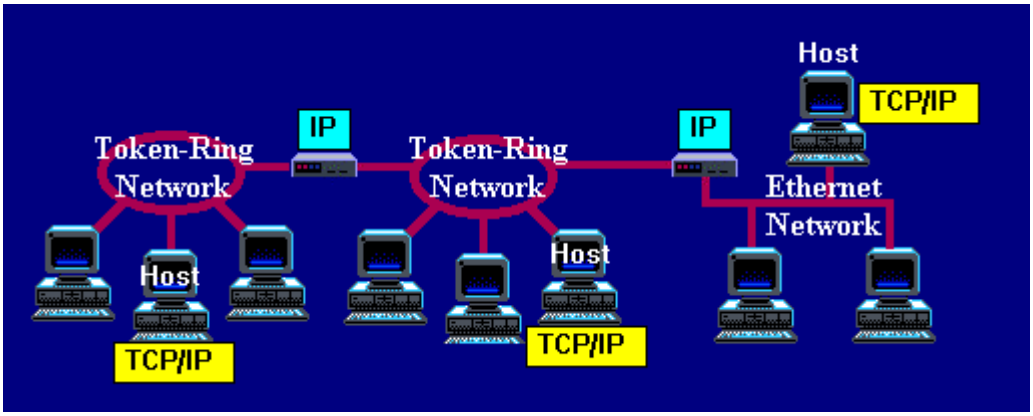
N.B.

TCP e IP sono solo due dei protocolli utilizzati in nel **modello di rete TCP/IP**.
Il termine TCP/IP si riferisce propriamente alla **suite di protocolli e applicazioni** mostrata nell'immagine seguente:



N.B.

Nei termini di utilizzo dei protocolli TCP/IP in ambiente di rete dobbiamo notare alcuni punti.
La suite di protocolli TCP/IP non indirizza i livelli di rete più bassi. Questi livelli più bassi sono dipendenti dalla rete e sono determinati dal particolare hardware che si sta utilizzando. Vale a dire che ogni singola rete che utilizzi il protocollo TCP/IP fornisce il proprio insieme di protocolli di livello più basso.
Pertanto, si possono utilizzare i protocolli TCP/IP tra o all'interno di differenti tipi di LAN e WAN in congiunzione con differenti protocolli di livello più basso.
Per le LAN questo vuol dire che si possono usare i protocolli TCP/IP in cima a Ethernet, Token-Ring e agli altri protocolli di rete.



D'ora in vanti il nostro **modello di rete** sarà quello individuato dal **modello di rete TCP/IP.**

Dato **un modello di rete** chiameremo **protocollo di rete** un procedimento *dettagliato, accettato* dal **mittente** e dal **destinatario** per lo scambio dei dati ad un determinato livello o strato del modello di rete.

Nel modello di rete semplificato a quattro livelli individuiamo, come i più importanti, i seguenti protocolli:

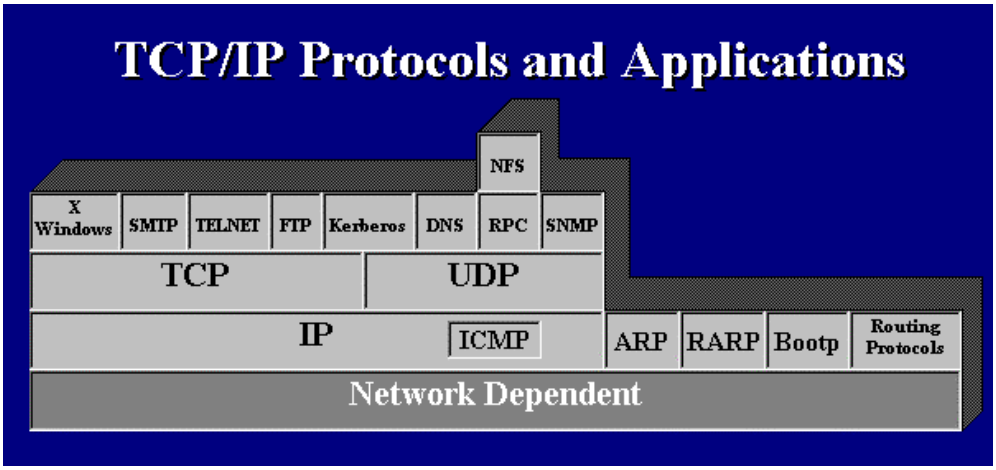
- **livello Applicazione** : comprende i protocolli **HTTP** (HyperText Transfer Protocol) , **FTP** (File Transfer Protocol) , **SMTP** (Simple mail transfer Protocol), **DNS** (Domai Name System), **SNMP** (Simnple NetworkManagement Protocol);
- **livello Trasporto** : comprende i protocolli **TCP** (Tansmission Control Protocol) e **UDP** (User Datagram Protocol);

} Fanno parte della famiglia di protocolli TCP/IP

- **livello Fisico** : comprende i protocolli **Ethernet, Fast Ethernet, Token Ring, FDDI** (Fiber Distributed Data Interface), le **ATM** (Asynchronous Transfer Mode) e le LocalTalk.

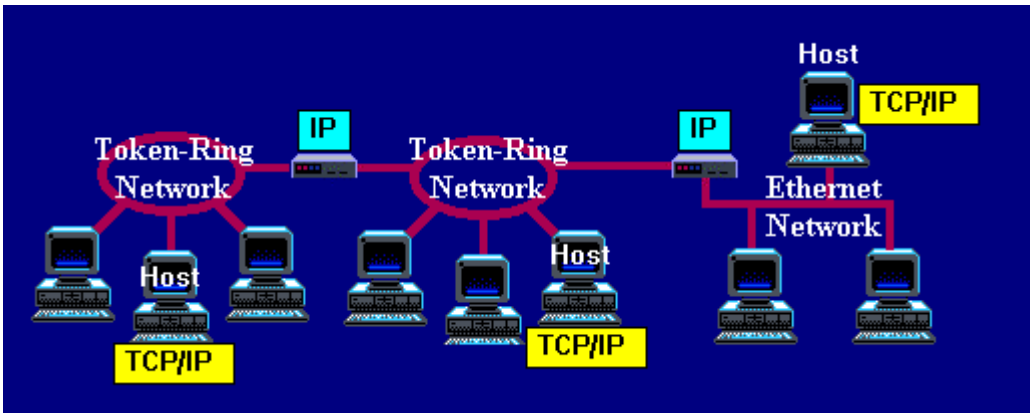
} Variano a seconda della tipologia "fisica" di rete

La **famiglia di protocolli TCP/IP** viene anche chiamata **stack** di protocolli TCP/IP o **suite** di protocolli TCP/IP p più semplicemente **Protocollo TCP/IP**.
Tale famiglia rappresenta ormai lo *standard di fatto* in **ambiente Internet** ossia per **l'internetworking** e sarà il punto di partenza per le nostre applicazioni per il web.



N.B.

Nei termini di utilizzo dei protocolli TCP/IP in ambiente di rete dobbiamo notare alcuni punti.
Il Protocollo TCP/IP non indirizza i livelli di rete più bassi. Questi livelli più bassi sono dipendenti dalla rete e sono determinati dal particolare hardware che si sta utilizzando.
Vale a dire che **ogni singola rete che utilizzi il protocollo TCP/IP fornisce il proprio insieme di protocolli di livello più basso**.
Pertanto, si possono utilizzare i protocolli TCP/IP tra o all'interno di differenti tipi di LAN e WAN in congiunzione con differenti protocolli di livello più basso.
Per le LAN questo vuol dire che si possono usare i protocolli TCP/IP in cima a Ethernet, Token-Ring e agli altri protocolli di rete.



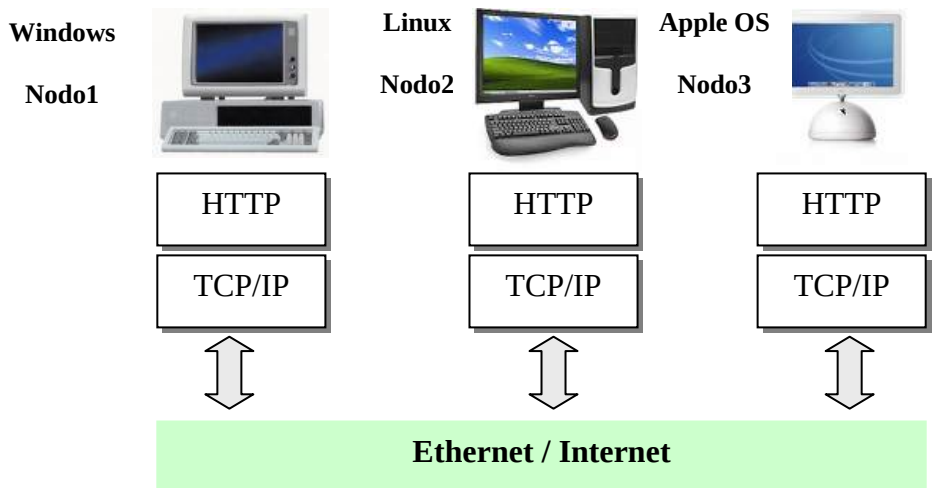
Il protocollo TCP/IP è un **protocollo aperto**: ciò significa che le descrizioni tecniche del protocollo appaiono in documenti **pubblici** *(denominati **RFC** Request For Comments)* e mette in condizione **chiunque** di creare un TCP/IP sul proprio hardware o software (Ecco il motivo del suo grande successo).

Nel corso di questo modulo saremo interessati solo alla rete **Internet** o alle **Intranet**.

Internet: (INTERnational NETworking) Rete composta da migliaia di reti WAN (Wide Area Network) e LAN (Local Area Network) di computer singoli che operano come se fossero reti. Questa rete mondiale di computer che utilizzano il protocollo TCP/IP come protocollo comune di comunicazione.
Tra i servizi disponibili su Intente ci sono il World Wide Web e la posta elettronica.

Intranet: Rete di computer interna a un'organizzazione, in grado di supportare le applicazioni ed i servizi Internet quali il World Wide Web e la posta elettronica (in altre parole la rete si basa sul protocollo TCP/IP). La maggior parte delle reti Intranet è configurata in modo da consentire ai propri utenti l'accesso all'intera Internet, senza tuttavia consentire agli utenti di Internet di accedere ai computer della rete Intranet.

Il protocollo di rete che considereremo come protocollo standard al livello Applicazione sarà HTTP.
In una rete Intranet (ma anche Internet) possono convivere computer aventi diversi sistemi operativi purchè per comunicare si basino sui protocolli TCP/IP ed http.



L'architettura **client-server** è ormai l'architettura standard parlando di **networking.**

> **Architettura di rete client-server**: In questa architettura di rete alcuni computer detti **server**, forniscono **servizi** ed altri computer detti **client** semplicemente richiedono ed utilizzano tali servizi. **Client e Server** possono essere su piattaforme diverse o addirittura coincidere con la stessa macchina, l'importante è che rispettino il protocollo predefinito stabilito per richiedere ed erogare i servizi offerti.

Nel caso del **World Wide Web** il protocollo si chiama **HTTP** ed il linguaggio standard di formato di una pagina o documento Web è **l'HTML**.

Le pagine scritte in **HTML** sono visualizzate da appositi programmi chiamati **browser** che in pratica implementano la parte client del protocollo HTTP.

> Un **browser** è un programma che consente la navigazione nella rete Internet, più precisamente nel World Wide Web. La funzione primaria di un browser è quella di interpretare il codice HTML (e più recentemente XHTML o il DHTML) e visualizzarlo in forma di ipertesto.

Considereremo il "browser" come client universale in un'architettura client-server

Ogni computer server per poter fornire i propri servizi ha bisogno di un **Web server** che è in pratica un software che implementa la parte server del protocollo HTTP.

> Un **Web server** è un programma (e, per estensione, il computer) che si occupa di fornire, su richiesta del **browser** una pagina web (spesso scritta in HTML). Le informazioni inviate dal Web server viaggiano in rete trasportate dal protocollo HTTP. L'insieme di Webserver presenti su Internet forma il WWW ossia il World Wide Web, uno dei servizi più sfruttati della Grande Rete.

Ogni Web server va **configurato** per offrire una serie di servizi ai client. Nella configurazione occorre tener presente il tipo di servizio che si vuole offrire all'utente.

Le seguenti sono comuni configurazioni per poter offrire *servizi* reali:
- gestire richieste HTTP (**Web server)**;
- gestire caselle di posta elettronica (s**erver SMTP**);
- gestire servizi di FTP per il download dei file(**server FTP**);
- gestire servizi di News (**server News**);
- gestire *streaming* audio-video;

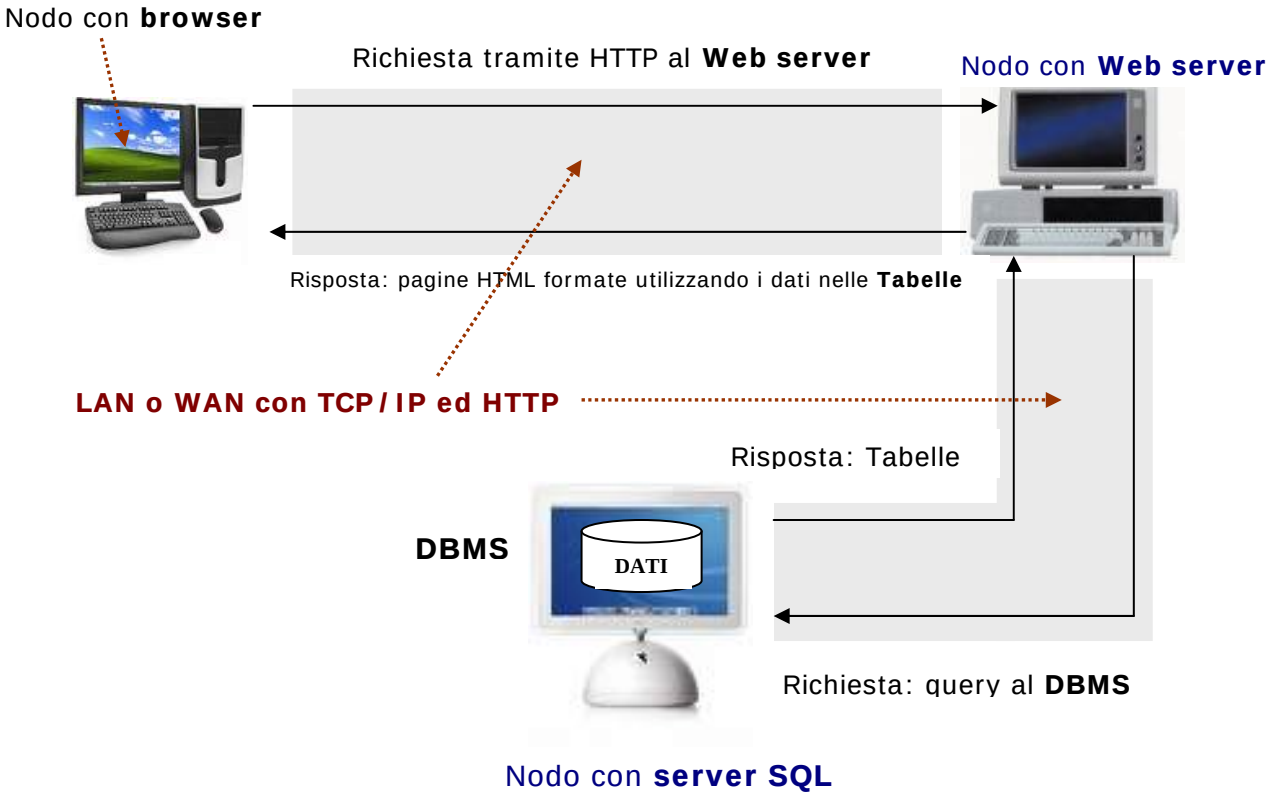
ma soprattutto

- *integrare linguaggi di programmazione o di scripting lato server* da mandare in esecuzione su richiesta dei client. Tali programmi potranno anche interagire con database remoti.

L'obiettivo in un ambiente di rete

Il nostro **obiettivo in un ambiente di rete** è quello di far interagire un database situato fisicamente su un *nodo server* con i browser situati su qualsiasi altro *nodo client* della rete (*Internet* o *Intranet*)

Scenario client-server con client (**browser**) , **Web server** e **server SQL** su nodi **separati**



IMPORTANTE

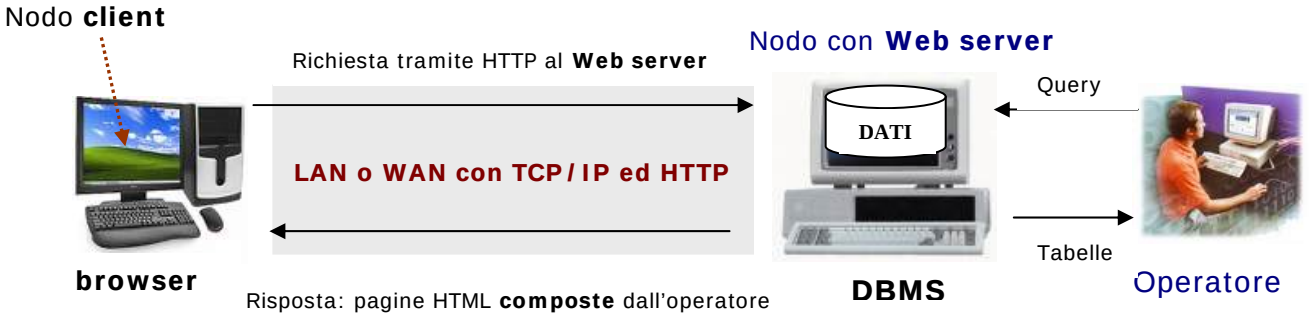
Per semplicità in seguito considereremo il Web server ed il server SQL (ossia il vero motore del database identificabile in pratica con il DBMS) SULLO STESSO NODO.
Sempre per semplicità e chiarezza utilizzeremo l'SQL come linguaggio standard di interrogazione *e* manipolazione di un database.

I possibili approcci di interfacciamento di un database in rete

Per integrare un database in un ambiente client-server sono possibili oi seguenti cinque approcci:

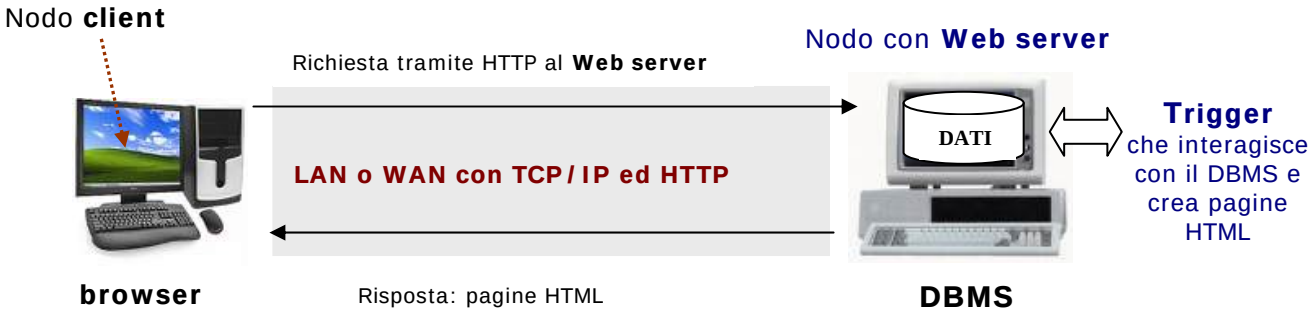
1. **primitivo** (in base alle attuali conoscenze);
2. basato su **trigger**;
3. basato su **Web server**;
4. basato su **client**;
5. basato su **ODBC** (trasversale).

1) Dettagli approccio **primitivo**: quando arriva una richiesta di interrogazione al Web server quest'ultimo la sottopone ad un operatore umano (collegato alo stesso server) che effettua *"manualmente"* l'interrogazione al DBMS (tramite client SQL).
Con il risultato l'operatore costruisce sempre *"manualmente"* la pagina HTML da restituire come risposta al client.



N.B. Tale approccio è **irrealizzabile** per tempi, costi ed inaffidabilità delle interrogazioni ma è necessario per comprendere la necessità di un *metodo* o di una *tecnologia* che **automatizzi** il processo di creazione delle pagine HTML che costituiscono il risultato dell'interrogazione.

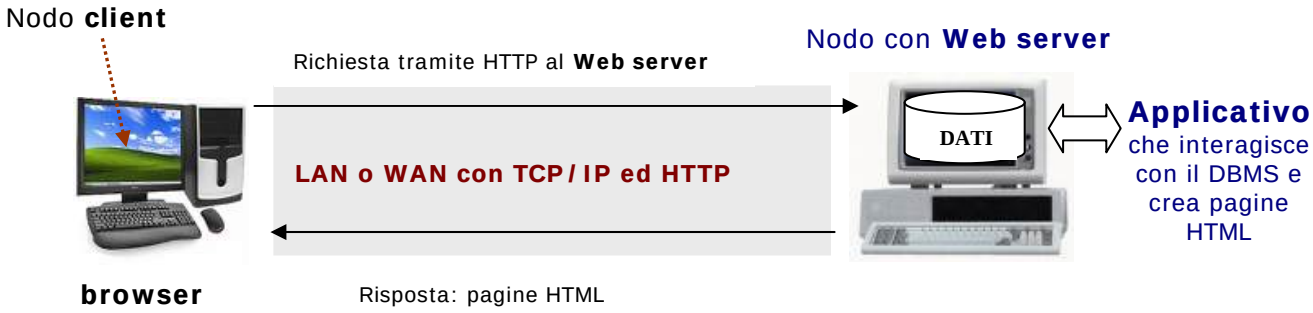
2) Dettagli approccio basato sul **trigger**: potremmo creare dei *trigger* (ossia delle procedure automatiche che vengono eseguite non appena si verificano determinati eventi che soddisfano determinate condizioni) da inserire nel database in modo che ad ogni variazione di alcuni particolari dati (ad esempio l'attributo *Prezzo* di una tabella *Articoli*) essi generino automaticamente le pagine HTML da trasferire ai client.



N.B. Anche in questo caso **ponendoci nell'ottica del Web server** le pagine HTML create sono **sempre statiche** in quanto create da parte dei trigger *ad insaputa del Web server stesso*.
In questo approccio al client arrivano pagine HTML che non ritrova *fisicamente* sul disco del server ma che viene creata *dinamicamente* da parte dei trigger del DBMS

In questo caso il **carico elaborativi della query** è esclusivamente a carico del computer sul quale è installato il server SQL.

3) Dettagli approccio basato su **Web server**: occorre avere un **programma applicativo** sul server che invia i comandi SQL al server SQL. A questo punto si crea automaticamente una pagina HTML che formatti tale risultato ovvero la renda leggibile da parte del browser.



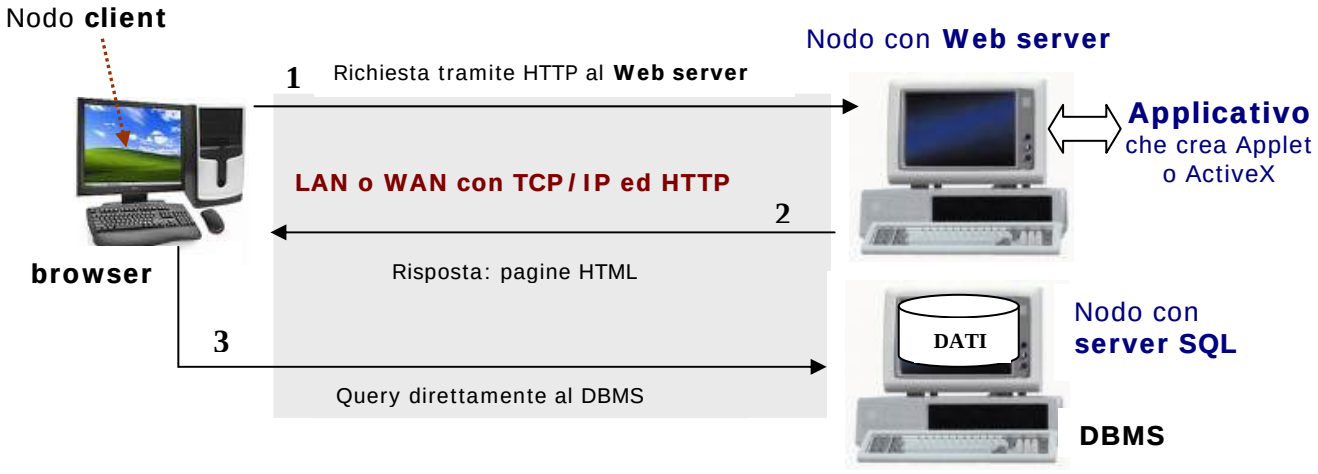
In questo approccio il **programma applicativo** deve.
- *inviare comandi SQL* al server SQL;
- *ricevere una risposta* dal server SQL;
- *creare* una pagina HTML;
- *restituire* tale pagina HTML al Web server.;

In questo caso il **carico elaborativi della query** è esclusivamente a carico del computer sul quale è installato il Web server
Il Web server invierà poi la pagina HTML così creata al client che ne ha fatto richiesta.

N.B. Anche in questo caso **ponendoci nell'ottica del Web server** le pagine HTML create sono **sempre statiche** in quanto create da parte del **programma applicativo** *ad insaputa del Web server stesso*.
La pagina creata non si trova *fisicamente* sul disco del server ma viene **generata automaticamente** dal programma applicativo nel momento in cui arriva la richiesta.

Rientrano in questo approccio:
- le pagine **ASP** di Microsoft;
- le pagine **PHP**;
- le pagine **JSP** di Java e le **Servlet** Java

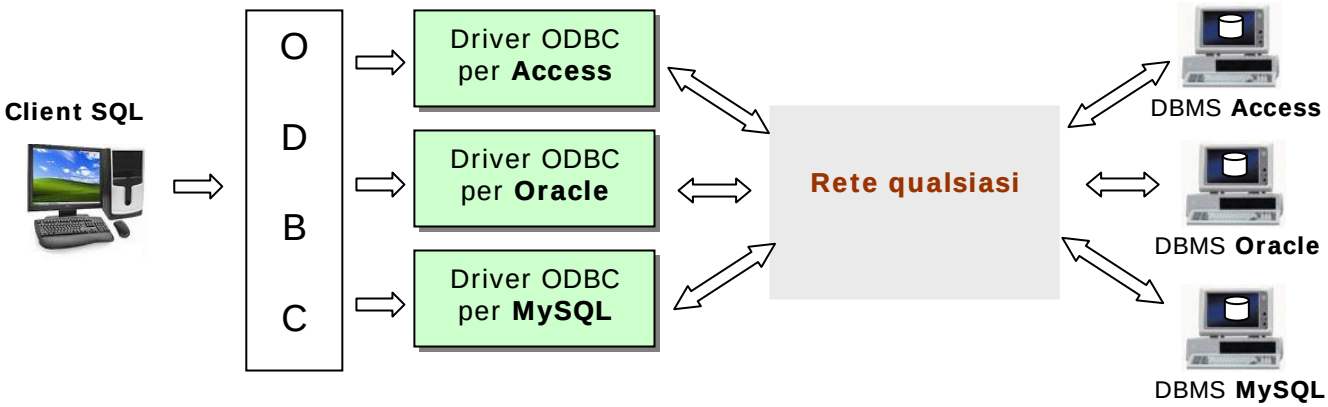
4) Dettagli approccio basato sul **client**: il Web server **invia** al client **un applicativo** (ad esempio una *Applet* o un componente *ActiveX*) che **interagisce direttamente** con il server SQL del database remoto e quindi senza alcuna intermediazione da parte del Web server.



In questo caso il **carico elaborativi della query** è esclusivamente a carico del computer client
(Rientrano in questo approccio le soluzioni JDBC di Sun Micosystem)

5) Dettagli approccio **trasversale** basato su **ODBC**

ODBC (**O**pen **D**ata**B**ase **C**onnectivity) è l'interfaccia software standard sia in ambiente Windows che in altri ambienti che consente ai programmatori di interfacciarsi a qualunque database creato da altri programmatori, purchè siano stati scritti i driver ODBC per quel database.



In pratica se si vuole accedere ad un database qualsiasi (gestito ossia da un qualsiasi DBMS) si possono sfruttare i driver ODBC per quel database ed utilizzarli all'interno delle applicazioni per connettersi ed interrogare quel database.

N.B. Questo approccio viene definito approccio **trasversale** (oppure "**non pulito"**) rispetto al client poiché bisogna effettuare alcune **operazioni di installazione direttamente sul computer client** (ad esempio aggiungere il **DSN** *Data Source Name* ossia la specificazione della sorgente dati) mentre gli altri approcci finora visti sono considerati "**puliti"** per il client in quanto non necessitano di alcuna conoscenza su dettagli implementativi del DBMS con il quale si vuole interagire

In particolare in quest'approccio il client deve:
- **conoscere il tipo di DBMS** che è installato sul server;
- **reperire i driver ODBC** per il collegamento a quel particolare DBMS;
- **installare tali driver** sul proprio sistema operativo.

Inoltre l'approccio ODBC:
- **non necessita di un browser** ma può utilizzare un client qualsiasi per l'accesso al database purchè sia un client SQL;
- **non è basato su HTML** (il risultato non deve essere necessariamente formattato in HTML);
- **non è necessario che il database si trovi su di una rete TCP/IP.**

IMPORTANTE

Sceglieremo l'approccio basato su Web server come riferimento per l'interazione con un databse in rete.

Programmazione lato client e lato server

Dato un **ambiente client-server** in un'architettura di **protocolli TCP/IP ed HTTP** per:

- **programmazione lato server** intenderemo lo sviluppo di programmi applicativi che andranno in esecuzione prevalentemente sul *server*, accettando le richieste dal client e fornendo a quest'ultimo i risultati dell'elaborazione sottoforma di pagine HTML;

- **programmazione lato client** intenderemo lo sviluppo di programmi applicativi che andranno in esecuzione prevalentemente sul *client*, inviando le richieste al server e gestendo i risultati ricevuti da quest'ultimo.

E' possibile quindi parlare di **programma lato server** e di **programma lato client** così come di **linguaggi lato server** e di **linguaggi lato client.**

Esempi di programmazione lato client: gli Applet, il codice Javascript, il codice VBScript
Esempi di programmazione lato server: il codice PHP, il codice ASP, il codice XML

Un qualsiasi Web server quando riceve una richiesta da parte del client può interpretarla:
- **semplice richiesta di invio di pagine statiche HTML** presenti sul server (o altre risorse come un'immagine, un suono, un Applet, etc.);
- **richiesta di esecuzione di un file** contenete le istruzioni del programma lato server.

Definiamo **programmazione orientata al Web o Web-oriented** l'insieme di tecniche e metodologie che si possono usare in un *ambiente client-server* con un'architettura di p*rotocolli TCP/IP e HTTP* per far interagire tra loro *programmi lato server* e *programmi lato client* con l'obiettivo di realizzare sistemi che possono essere eseguiti in una *Intranet* oppure in *Internet*.

Ripartizioni di applicazioni tra client e server

Ecco alcune **linee guide** su come ripartire un'applicazione tra client e server.

Conviene utilizzare la **parte client dell'applicazione** per i seguenti compiti:
- *convalidare* l'input dell'utente (effettuare i controlli del caso);
- *richiedere* all'utente una conferma l'input;
- *visualizzare* messaggi di errore o informativi;
- *eseguire* alcuni tipi di elaborazioni sui dati (quali somme o medie);
- *eseguire* altre funzioni che non richiedano informazioni dal server.

Conviene utilizzare la **parte server dell'applicazione** per i seguenti compiti:
- *conservare* le informazioni tra un accesso e l'altro del client;
- *mantenere* i dati tra diversi client o applicazioni;;
- *accedere* ad un database;
- *richiamare* librerie di altri linguaggi sul server;;
- *accedere* genericamente ad altre risorse presenti sul server.

Linguaggi di scripting e di programmazione lato server

Abbiamo finora parlato *genericamente* di linguaggi di programmazione lato server ma spesso occorre distinguere tra:

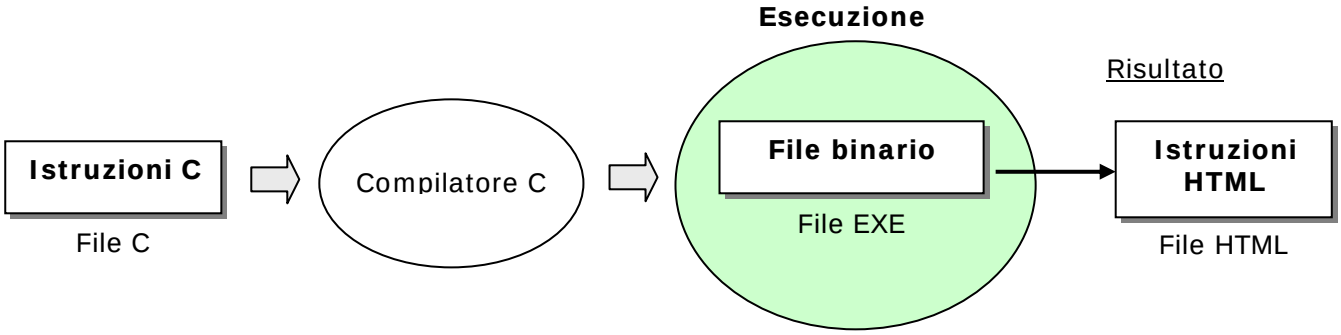
a) **linguaggi di programmazione** *(veri e propri)* **lato server:** ad esempio Java con le sue Servlet oppure il C con il quale si scrivono programmi **CGI** (uno dei primi metodi di programmazione lato server);

b) **linguaggi di scripting lato server**: ad esempio **PHP**. **PERL**, **ASP**

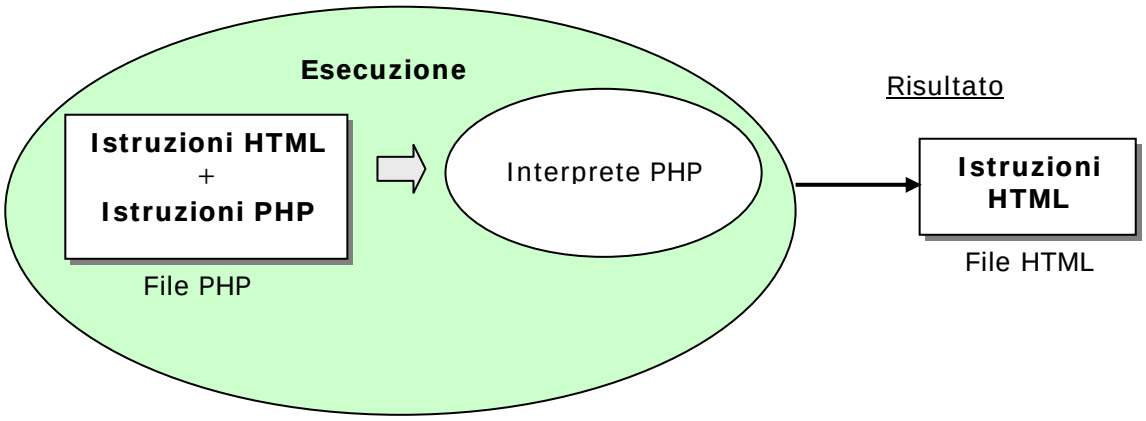
La differenza consiste nel fato che mentre i *linguaggi di programmazione veri e propri lato server* hanno vita *autonoma* anche in versione *non server*, i *linguaggi di scripting lato server* possono essere utilizzati *esclusivamente* in quel contesto.

Linguaggi interpretati, compilati e misti

a) linguaggio compilato lato server: un **programma CGI** scritto in linguaggio è un classico esempio di programma lato server compilato il cui eseguibile è inserito in un'opportuna directory del Web server.



b) linguaggio interpretato lato server: un **file di comandi PHP** è un classico esempio di programma interpretato lato server poiché infatti il Web server associa a tale file l'interprete PHP che deve essere mandato in esecuzione per poterne eseguire i comandi.



Per questo motivo è possibile avere *comandi PHP all'interno di pagine HTML* oppure *pagine PHP pure*.

c) linguaggio lato server ad approccio misto (compilato ed interpretato): un esempio classico è quello delle **JSP Java** poiché il codice java per essere eseguito deve essere prima *compilato* ottenendo del codice intermmedio *(i file .class)* e poi *interpretato* dall'*interprete Java* ossia la **JVM** Java Virtual Machine