

Capitolo sesto

Gli Ambienti Distribuiti

6.1. Introduzione

Questi ultimi anni hanno visto l'affermarsi degli ambienti distribuiti. Ciò deriva soprattutto dall'apparizione e diffusione del personal computer che ha spostato l'elaborazione e i dati verso l'utente finale. Si è passati da ambienti completamente centralizzati in cui sia l'elaborazione, sia i dati risiedevano tutti su Mainframe (host), cui ci si collegava tramite terminali "stupidi" (detti in questo modo perché non avevano nessuna capacità d'elaborazione), ad ambienti dove tali elementi sono disposti su più mezzi fisici.

La diffusione del personal è dovuta sia all'affermarsi di un nuovo concetto di "informatica personale", che si basa sulla realizzazione di "macchine" hardware con discrete capacità elaborative, ma anche sulla realizzazione di software nelle aree di lavoro monoutente (ad esempio video scrittura, fogli elettronici) e in quelle dove parte del patrimonio informativo è condiviso e dove le procedure automatizzate interessano differenti aree organizzative delle imprese.

Si sono così sviluppati i software di rete che danno la possibilità di connettere macchine diverse, di diversa potenza, rendendo direttamente accessibili i dati indipendentemente dalla loro locazione.

Però non garantiscono sempre una loro corretta disponibilità.

I dati utilizzati localmente, e che dovevano essere condivisi, venivano in passato copiati o estratti da archivi centralizzati o residenti su altre stazioni di lavoro. Questo introduceva ridondanze e inconsistenze difficili da controllare. I dati, consistenti e aggiornati, devono invece essere condivisi tra più utenti con un accesso efficiente, sicuro e indipendente dalla loro localizzazione.

E' per tali ragioni che l'architettura client-server si è diffusa separando i ruoli di utilizzatore dei dati e di gestore.

Il gestore risiede sul server, mantiene un'unica copia dei dati rendendoli disponibili, su richiesta, alle applicazioni residenti su sistemi client. Tali

applicazioni sfruttano strumenti di carattere individuale dotati di sviluppate interfacce grafiche che aiutano l'utente non esperto alla realizzazione del lavoro.

Per indicare questa separazione delle funzionalità si utilizzano spesso i termini back-end e front-end. S'indica con back-end il software di gestione delle basi di dati (DBMS) che realizza le funzioni relative alla struttura, alla manipolazione e integrità dei dati. Con front-end s'indica, invece, l'insieme degli strumenti per l'accesso (interfacce, comandi, maschere video) e la presentazione (reporting) dei dati, per la generazione di applicazioni e di strumenti d'informatica individuale.

In questi tipi di strutture si sono affermati i gestori di base dati relazionali, i quali si fondano su di un modello "solido", vale a dire su di un modello che poggia su basi matematiche certe. Inoltre, la sua struttura permette alle basi di dati di crescere, integrarsi senza creare un grande stravolgimento nella struttura dei dati. Altro vantaggio è dato dal fatto che esiste uno standard d'accesso ai dati relazionali che è riconosciuto universalmente dal mercato informatico : l'SQL.

L'utilizzo di SQL consente di combinare in un solo linguaggio l'accesso sia ai database locali sia ai database locali e remoti. Questo è, quindi, un grosso strumento di connettività in architetture client-server e distribuite in genere.

Va notato, però, che lo standard attuale di SQL lascia alcuni gradi di libertà nell'implementazione dello stesso da parte dei diversi produttori. Ciò consente la realizzazione di dialetti di SQL da parte delle case produttrici, i quali, però, non sono pienamente compatibili fra loro. I dialetti hanno in ogni caso una base comune rispetto agli altri linguaggi d'accesso. SQL resta perciò la base per la connettività tra i diversi server di basi di dati su piattaforme differenti.

6.2. Le architetture Client-Server

L'architettura client-server è un "sistema" che realizza una distribuzione sia dell'elaborazione sia dei dati, ma solo in alcune delle sue tipologie. L'elemento centrale dell'architettura è il server. Il server può essere considerato sia da un punto di vista logico, sia da un punto di vista fisico. Dal punto di vista fisico, cioè hardware, il server è una macchina dedicata all'esecuzione di un server software¹. Da un punto di vista logico un server è

¹“ Affinché una stazione client possa accedere alle risorse del server occorre, nei due sistemi, un insieme di funzioni (i livelli cui si farà in questo testo riferimento sono quelli dello standard OSI che permettono di far comunicare sistemi che sono fra loro diversi. Si realizzano per questo delle regole che, per ogni livello nei sistemi comunicanti, dal 7 all'1, devono produrre gli stessi risultati. Per chiarimento su i protocolli di rete si veda nota 11):

Stazione Client

1. *Interfaccia utente.* Negli attuali sistemi operativi per PC una apposita interfaccia (Windows, per esempio) permette un accesso facilitato alle risorse locali¹.
2. *Instradatore o redirector.* Il sistema operativo locale deve essere in grado di riconoscere le richieste di risorse locali da quelle remote. Queste ultime saranno passate all'instradatore, che deve provvedere ad inoltrare le richieste verso il sistema server.
3. *Livelli funzionali 3-5.* I sottostanti livelli comunicativi devono accettare la richiesta, provvedere ad instaurare, controllare e mantenere fino a quando necessario la connessione fra la stazione client e il server, e definire la modalità di scambio (5° livello, di sessione), di controllo dell'interscambio, quali quelle per gestire le condizioni eventuali di errore (4° livello, di trasporto), gestire l'indirizzamento e l'instradamento di rete (3° livello, di rete)
4. *Driver di rete locale.* E' il software corrispondente ai primi due livelli funzionali, quello di collegamento dati e di interfaccia fisica permettendo l'effettiva comunicazione via LAN.

Stazione Server

Deve avere esattamente gli stessi livelli funzionali dal 1° al 5° della stazione client. Grazie ad essi le due stazioni interagiscono, le richieste del client vengono passate al sistema operativo del server e le risposte di questo vengono trasmesse successivamente indietro. Il sistema operativo è il centro di tutto questo sistema. Riceve ed interpreta le richieste delle altre stazioni, le gestisce in multitasking, accede alle risorse necessarie per erogare il servizio, delegando di volta in volta le richieste ad appositi servizi. Vi saranno così un sottosistema per lo spool di stampa, un altro per l'accesso a file su disco, un altro per l'accesso e la gestione del database, un altro per la comunicazione remota e così via. Per valutare i vari servizi offerti da un sistema operativo di rete le caratteristiche principali sono :

- ? La gamma delle funzioni
- ? Le prestazioni
- ? Le funzioni di protezione delle risorse per l'integrità dei dati e per la protezione da accessi abusivi.
- ? Le opzioni di condivisione delle risorse (un solo utente per volta, contemporaneità di accesso in lettura, in lettura/scrittura, e così via).
- ? Il livello di multiprogrammazione :ad esempio quanti task possono essere attivi simultaneamente sul server.

Un altro elemento essenziale dei sistemi operativi di rete è che devono essere scalabili, - ovvero in grado di essere trasportati su configurazioni hardware differenti. Questo svincola

un software. Il server come processo logico fornisce servizi ad altri processi che assumono il ruolo di *requester* (richiedenti) o *client*. In genere, il server non invia al requester i risultati sino a che quest'ultimo non glielo chiede. Esistono diversi tipi di server software: di rete, di file, di terminale, di database. Un *database server* è un processo logico responsabile dell'elaborazione di richieste d'interrogazione su basi dati. I processi che richiedono servizi al server vengono definiti client.

Una caratteristica che distingue il client dal suo server è che il client può dare inizio ad una transazione (non necessariamente un accesso al database) con il server, mentre il server non può mai dare inizio di sua iniziativa ad una transazione con il client. Infatti, i compiti specifici del client sono quelli di dare avvio alle transazioni, richiedere servizi specifici, notificare l'avvenuto completamento del servizio e ricevere risultati dal server.

Le comunicazioni fra client e server possono avvenire utilizzando una varietà di meccanismi: dalle reti geografiche a quelle locali sino ai servizi di comunicazione tra applicazioni a livello di sistema operativo. Un'architettura client-server deve essere indipendente dall'utilizzo di uno piuttosto che di un altro di questi metodi di collegamento fisico.

Occorre notare, inoltre, come non sia necessario che un processo client server risieda su sistemi fisicamente separati. In effetti, il processo server e quello client possono risiedere sulla stessa piattaforma di calcolo².

L'obiettivo principale dell'architettura client-server per la gestione dei dati è consentire alle applicazioni client di accedere ai dati gestiti dai server. Il

il NOS dalla componente hardware rendendolo applicabile su diverse piattaforme." Lan e sistemi operativi- R. Adinolfi -Masson 1992.

² "In questa configurazione si ha egualmente un vantaggio nel ricorrere ad una architettura client server in quanto si ha un risparmio di risorse dovuto al fatto che il server può fornire centralmente i servizi a diverse applicazioni (si tratta a bes guardare di una variazione sul tema dei servizi di sistema operativo, come il file system, che conviene accentrare piuttosto che delegare a ogni singola applicazione). Si ha pur sempre un lieve degrado delle prestazioni in quanto è necessario ogni volta instaurare una comunicazione fra client e server. Le funzionalità offerte sono quelle di un sistema di gestione di basi di dati classico. Quando sulla stessa piattaforma di calcolo sono presenti diversi client e diversi server, possono essere utilizzate architetture multiprocessore per migliorare le prestazioni. In questo primo caso non si può parlare ne di elaborazione distribuita ne di database distribuito." D.McGoveran e C. White DBMS 1991.

server (inteso in senso logico come software) può essere in esecuzione su di un sistema remoto (ad esempio in un'altra città o su di una rete locale). Per questo le applicazioni client-server sono spesso associate all'elaborazione distribuita. Un client fisicamente separato dal server deve effettuare dell'elaborazione distribuita per colloquiare con quest'ultimo, ma questo non implica per forza la presenza di un database distribuito. Quest'ultimo non è necessario in un'architettura client server come non lo è la separazione fisica fra client e server (vedere nota n : 2).

Bisogna distinguere fra elaborazione distribuita e le sue forme e database distribuito. L'elaborazione distribuita implica la suddivisione d'alcuni compiti elaborativi su diverse risorse di calcolo. In un'applicazione di gestione di base di dati ciò potrebbe voler dire eseguire l'interfaccia utente, il codice dell'applicazione, l'elaborazione logica dei dati su di un sistema e l'elaborazione e gestione dei dati vera e propria su di un altro.

Un database distribuito, invece, implica la suddivisione della base dati su diverse risorse di calcolo.

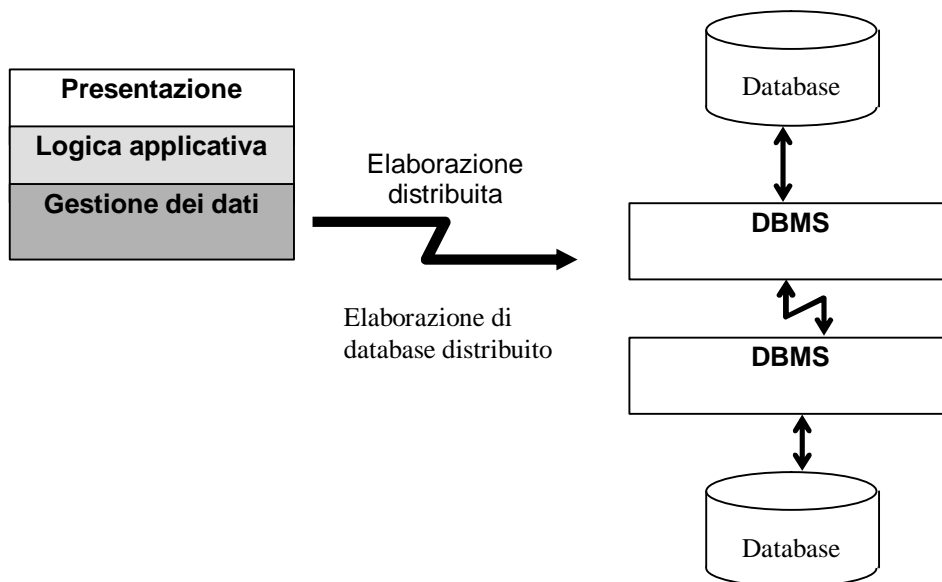


Figura 6.1. Differenza fra elaborazione distribuita e database distribuiti

Tutto ciò può avvenire, ad esempio, memorizzando diverse tabelle di una base di dati relazionale su diversi sistemi, o addirittura diverse parti o

frammenti di tabelle su diversi computer. Indipendentemente dal modo in cui viene effettuata la suddivisione, essa deve essere trasparente all'applicazione, cioè quest'ultima (e l'utente) non deve essere cosciente del fatto che i dati sono distribuiti. Scopo, quindi, dei sistemi distribuiti è quello di garantire alle applicazioni l'accesso ai dati sia remoti sia locali.

6.2.1. Tipologie di Client-Server

Si possono distinguere diverse tipologie di client-server che danno luogo a particolari forme di ambiente distribuito. Si può utilizzare per questo lo schema ormai divenuto famoso della Gartner Group.

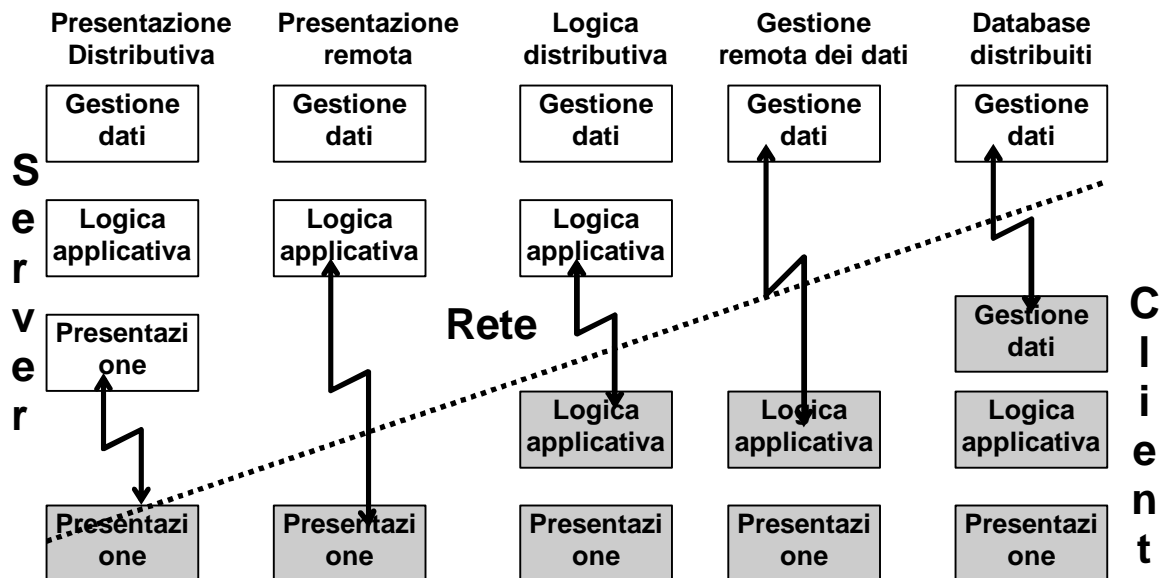


Figura 6.2. Schema della Gartner Group

Questo schema sintetizza le componenti di un'applicazione in tre elementi :

- ? presentazione
- ? logica applicativa
- ? gestione dei dati

Nella **presentazione** rientrano tutti i componenti applicativi che consentono all'utilizzatore del programma di interagire con esso per ottenere determinati risultati (interfaccia utente). Questa sarà realizzata in un ambiente grafico

tale da permettere all'utente di svolgere le sue funzioni più facilmente possibile senza che questo abbia particolari nozioni tecniche.

Le componenti applicative (**logica applicativa**) fanno sì che l'esecuzione di un programma porti a determinati risultati. Può essere costituita da componenti di calcolo, di stampa, d'elaborazione di un testo o di quanto altro realizza la funzione per la quale l'utente ha richiamato in esecuzione un determinato componente software.

La terza componente, infine, vale a dire quella della **gestione dei dati**, riguarda la gestione dei dati in un ambiente come quello client server. Tali operazioni sono normalmente svolte da un DBMS dovendo questo garantire, integrità fisica dei dati, accesso concorrente, ottimizzazione dei tempi di lettura e scrittura e indipendenza nei confronti del sistema hardware utilizzato.

Le componenti nello schema Gartner Group vengono tagliate in diagonale dalla rete, e le tipologie di client-server dipendono dalla disposizione che hanno dalla parte server, in alto, o da quella client, in basso, rispetto a questa rete.

La prima tipologia, **la presentazione distribuita**, è quella che più si avvicina al passato. Tutto viene sviluppato su host, mentre il PC (o in questo caso sarebbe meglio parlare di terminale stupido) ha solo alcune funzioni d'interfaccia utente. Primo ruolo di un PC può essere quello della **presentazione remota**, in cui sia la gestione dei dati che la logica applicativa, cioè il codice dell'applicazione, restano su host ma l'interfaccia viene sviluppata su personal. Al centro troviamo il modello della **logica distribuita**. Pur demandando ancora all'host la gestione dei dati, questo condivide i compiti di elaborazione, calcolo con il personal. Ovvero il lavoro viene svolto sia da una parte sia dall'altra, mentre la presentazione rimane al PC. La quarta tipologia, denominata di **gestione remota dei dati**, trasporta completamente il motore dell'applicazione e l'interfaccia su PC. Questo è

forse il caso più diffuso e più tipico, implementabile sia su piccole reti locali che in grosse reti geografiche³.

L'ultima via è quella dei **database distribuiti**. Questa ultima tipologia può riguardare sia il caso tipico dei database distribuiti in cui i dati, che risiedono su macchine differenti, devono essere visti come un insieme unico. O più semplicemente si può trattare di repliche locali di parte di un database centrale. Si può intendere, ancora, con database distribuiti il collegamento di tabelle locali, esterne cioè al database vero e proprio, con altre remote, ed è questo il caso principale che i prodotti personal devono affrontare.

6.3. Servizi d'accesso ai dati

Un'applicazione client deve mettersi in comunicazione con un server. Per questo vengono utilizzati particolari software (middleware) che cercano di far colloquiare il client con il o i server scavalcando tutte le incompatibilità che esistono fra i due sistemi⁴.

³ "Una LAN opera in un ambiente limitato come un ufficio o la sede di una società, mentre reti come le MAN (Metropolitan Area Network) e le WAN (Wide Area Network) o reti geografiche sono progettate per coprire intere città regioni stati o continenti. La differenza principale consiste nei protocolli ossia nel modo di trasmissione delle informazioni." Local Area Network -S. Buonopane -Bit- 1/1991

⁴ La spinta attuale alla realizzazione di sistemi aperti, cioè sistemi che possono comunicare con altri sistemi, ha portato sempre più alla ribalta il middleware, cioè un software o meglio un insieme di software capace di far comunicare ed interoperare tra loro applicazioni operanti su piattaforme hardware, software e di rete eterogenee⁴. L'architettura middleware consiste di un certo numero di software, diversissimi tra loro (i middleware) che però vanno a costituire una architettura che permette di fatto la comunicazione fra sistemi diversi. Più una architettura middleware si presenta eterogenea maggiore è la capacità connettiva tra i vari sistemi. Quindi, si cerca da parte dei fornitori di allargare il raggio delle componenti da tenere in considerazione.

La differenza fra l'offerta di una impresa o di un'altra sta nella scelta del middleware e/o degli standard di comunicazione delle reti (OSI, TPC/IP, ecc.). Essendo collezioni di middleware, le architetture attuali tendono a coprire gli ambiti che interessano di più i singoli fornitori.

Un esempio di architettura middleware è Open Blue print di IBM. La componente architeturale tradizionale (quella relativa alla comunicazione fra sistemi di reti differenti) proviene dal richiamo agli standard delle più varie provenienze (OSI, TPC/IP, ecc.) che garantiscono a Blue print l'aggettivo di "Open". Il resto è middleware vero e proprio che permette l'interoperabilità delle applicazioni o meglio che tutti i componenti hardware e software collegati in rete operino come un unico sistema.

I servizi applicativi sono organizzati in 3 gruppi stratificati :partendo dal basso abbiamo il *network*, il *distributed systems* e *l'application emebling*. A fianco di questi sistemi ci sono i local operating system ed i servizi di gestione. Del gruppo network fa parte fra gli altri la

Particolari middleware facenti parte delle architetture middleware permettono di accedere ai database di diverso formato. La necessità di un accesso migliore ed “aperto” a dati provenienti da diversi database è dovuta alla maggiore importanza che l’informazione ha assunto nell’azienda divenendo una componente essenziale in ambito competitivo. Queste hanno la necessità di interrogare i propri dati in maniera accurata ed in tempi brevi; questa esigenza si fa sentire di più in aree come il controllo di qualità, le analisi di mercato, la pianificazione, la gestione di magazzino e le vendite. I dati risiedono normalmente in una notevole varietà di formati diversi come il VSAM e l’ISAM o come quelli dei diversi DBMS relazionali o gerarchici e questo accade per ragioni tecnologiche, strategiche o storiche. Le varie informazioni sparpagliate su tutti questi sistemi dovrebbero poter essere confrontate ed analizzate fra loro per ricavare un andamento generale della società. Per questo è importante realizzare un metodo comune per accedere, gestire ed analizzare tutti i dati indipendentemente dal loro formato. La connettività è necessaria per realizzare un accesso integrato ai diversi database aziendali. E questa necessità si è notevolmente accresciuta nell’ambito dell’architettura Client/Server. Solitamente sulle macchine Client si trovano delle applicazioni che richiedono dati ai database posti sui server. Un apposito software (il middleware) realizza una funzione particolare nell’accesso ai dati poiché riceve le richieste d’interrogazione e di modifica dei dati direttamente dalla applicazione e le trasmette al Server. Questo software è inoltre responsabile del ritorno alla applicazione dei risultati o degli eventuali errori.

Common Transport Semantics, una interfaccia tra applicazione e trasporto di rete che consente di accedere con una unica serie di comandi a reti eterogenee.

I servizi di *distributed systems* si occupano della comunicazione tra le applicazioni distribuite, sia tradizionali che ad oggetti e per questo si impiegano particolari middleware. Degli *application embling services* fanno parte particolari prodotti middleware che consentono l’accesso della applicazione stessa a dati e applicazioni esterne.



Figura 6.3. In un sistema client/sever, per accedere ai dati, è necessario passare attraverso alcuni livelli di software (Microsoft)

Quindi, gli obiettivi della connettività d'accesso dati dovrebbero consistere :

- ? Nell'accedere a molte sorgenti di dati eterogenee da una singola applicazione (Accessibilità)
- ? L'applicazione dovrebbe essere in grado di accedere senza nessuna modifica ai vari tipi di database (Flessibilità)

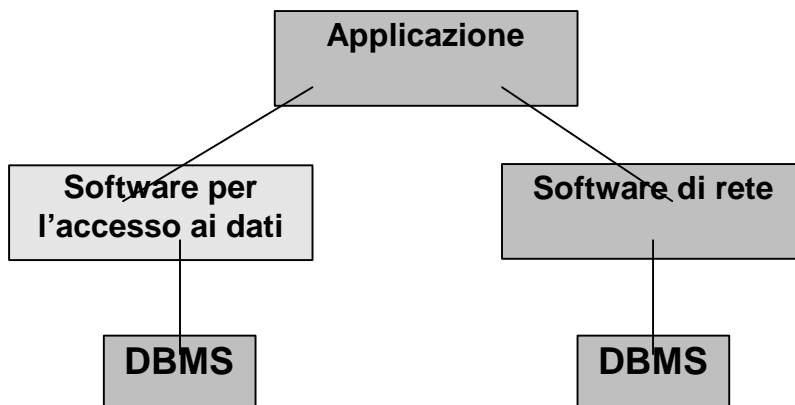


Figura 6.4. Un'applicazione può accedere a più sorgenti di dati se dispone di un apposito software per il collegamento (Microsoft).

Un primo modo di risolvere i problemi di connettività è quello di usare dei gateway. Un gateway permette ad un'applicazione di “vedere” un tipo di DBMS come fosse quello selezionato. Un gateway traduce le chiamate al database nel formato del DBMS e ne interpreta le risposte. Per esempio un'applicazione che utilizza SQL server, può accedere ad un database DB2 attraverso l'apposito gateway, però per accedere ad un altro database ha bisogno di un altro gateway. Quindi il gateway è troppo dipendente dalle

architetture dei diversi database e richiede un gateway apposito praticamente per ogni tipo di DBMS. E' logico che se si utilizzano diversi prodotti di gestione dei dati non è sufficiente una semplice interfaccia di comunicazione.

Un altro modo di affrontare il problema della connettività con diverse sorgenti di dati è quello dell'impiego di un'interfaccia comune, tramite la quale ogni applicazione può connettersi con qualunque database.

Sulle specifiche di quest'interfaccia si possono costruire le applicazioni che poi avranno accesso ad una miriade di database con formati diversi. Per cui si possono creare delle applicazioni che sono "svincolate" dalla struttura dei singoli database; *si afferma che questo fatto consenta di creare un ambiente omogeneo di sviluppo nel quale si possono realizzare programmi dall'interfaccia comune anche quando i DBMS sottostanti sono di tipo diverso*⁵

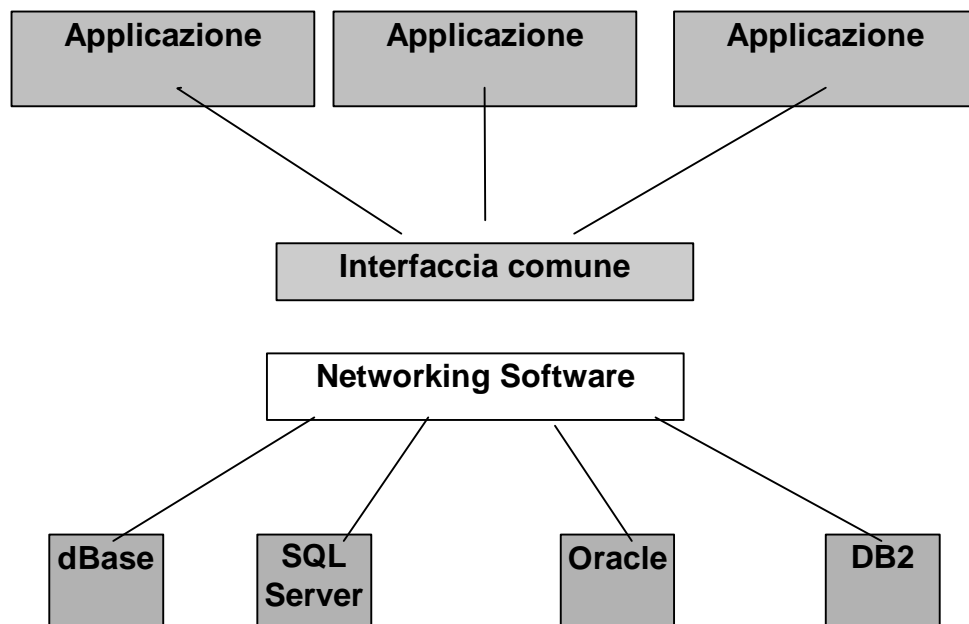


Figura 6.5. Tramite l'impiego di una interfaccia comune, ogni applicazione è in grado di connettersi con qualunque database senza sforzi aggiuntivi (Microsoft)

Ciò è reso possibile ,ad esempio, tramite la realizzazione di API⁶ standard

⁵ PC Magazine : Speciale "Il mondo Windows"

⁶ Le API (Application Programming Interface) rappresentano delle routine che sono state create e strutturate per fornire una interfaccia standard a disposizione delle applicazioni

oppure un linguaggio macro in grado di tradurre le richieste degli utenti e di interpretare le risposte. Il mezzo più utilizzato per ottenere questo risultato è di aggiungere un driver d'accesso ai DBMS, diverso per ogni DBMS. Tutto questo permette di tradurre, ad esempio, i vari dialetti di SQL dei produttori in linguaggio SQL standard in modo tale che i dati dei diversi database che supportano l'SQL siano perfettamente trasparenti ed accessibili da qualunque applicazione che usi questi software. Le applicazioni solitamente non hanno limitazioni sul numero di driver con i quali comunicare. Una singola applicazione può avere multiple connessioni con basi eterogenee, ognuna con un driver differente oppure più connessioni a database di tipo simile, tramite un unico driver. Questa è la soluzione offerta da prodotti come ODBC (Open Database Connectivity) di Microsoft o IDAPI (Integrated Database Application Programmer **Interface**) di Borland.

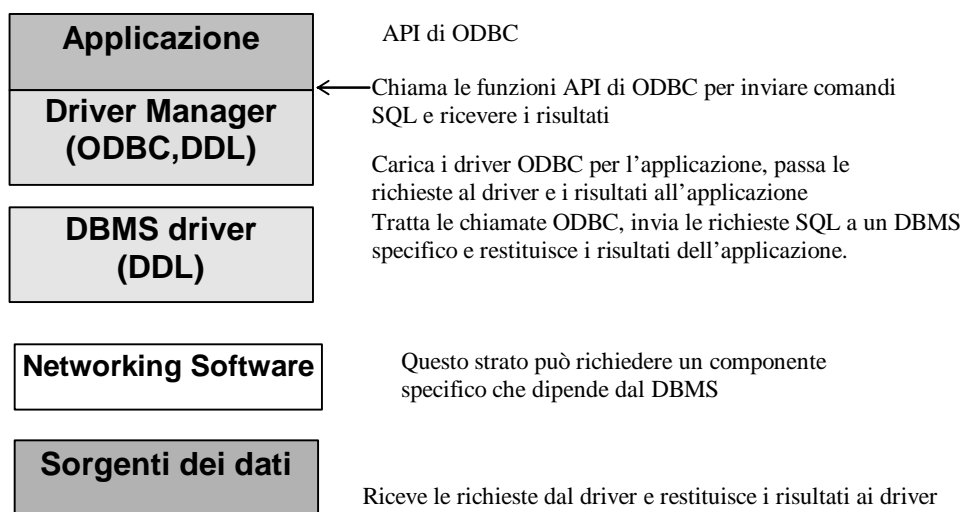


Figura 6.6. L'architettura flessibile di ODBC permette di intervenire su qualunque tipo di database, che sia o meno SQL e che si trovi in rete o localmente (Microsoft)

realizzando così una maggiore indipendenza fra le stesse, il sistema operativo e l'hardware. Si può, quindi, realizzare un ambiente più uniforme senza doversi adattare in maniera "aderente" alle caratteristiche hardware, oppure, come in questo caso specifico, al formato del DBMS.

Lo scopo dello strato di API fra applicazione e il DBMS è quello di far accedere l'applicazione alle diverse risorse dati e condividerle fra le varie applicazioni, in modo trasparente ed automatico per l'applicazione. L. Napolitano :Le API di Windows - BIT 11/1993

Un ulteriore modo di effettuare la connettività è quello della realizzazione di standard, cioè tramite la realizzazione di un protocollo comune a tutti i database. Rendendo uniforme il protocollo d'accesso ai DBMS, la sintassi dell'SQL e il protocollo di rete, le applicazioni sono in grado di comunicare in maniera semplice con tutti i DBMS.

6.4. I Database distribuiti

Un database distribuito è costituito da un insieme di siti, geograficamente distinti, detti nodi e connessi tramite una rete. Ogni sito ha un suo database, ma con la possibilità da parte di un utente di qualsiasi sito di accedere, in caso di necessità, a qualunque dato dei nodi come se il dato si trovasse nel sito dell'utente⁷.

Si può pensare al database distribuito come ad un database virtuale le cui componenti sono fisicamente memorizzate in un numero di distinti database reali che risiedono in un numero di siti geograficamente separati e connessi tra loro tramite una rete⁸.

C. Date ha cercato di riassumere attraverso dodici regole le caratteristiche essenziali di questi database distribuiti⁹. Date afferma che tali regole non

⁷ E. Schiavetti :I Database, Jackson 1989

⁸ "Generalmente, ma non necessariamente, i nodi sono geograficamente distribuiti in locazioni fisiche diverse, collegate da una rete di comunicazione dei dati. E' talvolta possibile che un grosso computer contenga diversi nodi, che hanno perciò la stessa locazione fisica. L'obbiettivo principale di un database distribuito è quello di processare i dati provenienti da database diversi, qualunque sia la natura dei loro collegamenti di comunicazione: la presenza di una rete di comunicazione dei dati non è perciò essenziale, benché solitamente presente." DEEN :Database :concetti teorici ed applicativi - FrancoAngeli

⁹ Le regole fissate da Date sono :

- ? Regola 0
- ? Regola 1 : Autonomia locale
- ? Regola 2 :Inesistenza di un sito centrale
- ? Regola 3 :Continuità operativa
- ? Regola 4 : Indipendenza di locazione
- ? Regola 5 : Indipendenza di frammentazione
- ? Regola 6 :Indipendenza dalle copie
- ? Regola 7 : Elaborazione della query distribuita
- ? Regola 8 : Gestione della transazione distribuita
- ? Regola 9 :Indipendenza dall'hardware
- ? Regola 10 :Indipendenza dal sistema operativo
- ? Regola 11 :Indipendenza dalla rete
- ? Regola 12 :Indipendenza dall'DBMS

sono tutte indipendenti fra loro e che l'ordine d'enunciazione non è indice del diverso grado d'importanza.

Un database distribuito deve presentarsi all'utente esattamente come un sistema non distribuito (**Regola zero**). Questo vuol dire che tutti i problemi relativi alla distribuzione fisica dei dati devono essere risolti internamente dal sistema. In un sistema di database distribuito si possono individuare due tipi di utenti : l'utente globale che elabora i dati del database distribuito sotto il controllo di un DBMS distribuito e l'utente locale o di nodo che elabora i dati di un particolare nodo sotto il controllo di un DBMS locale teoricamente ignaro dell'esistenza di un database distribuito. La query di un sistema distribuito sarà, quindi, un'elaborazione distribuita che coinvolge più CPU, più accessi su siti locali e una rete (**Regola 7: Elaborazione della query distribuita**). I siti di un database distribuito devono essere autonomi ; ovvero dati devono essere posseduti e gestiti localmente anche se possono essere acceduti da altri database remoti. Allo stesso modo le operazioni locali dovrebbero rimanere tali, in altre parole l'utente locale non dovrebbe essere penalizzato dal fatto che il sito partecipa ad un sistema distribuito. Inoltre, nessun sito per le operazioni che gli competono deve dipendere da un altro (**Regola 1 : Autonomia locale**).

Non deve esistere un sito centrale (**Regola 2 :Inesistenza di un sito centrale**) e come per i sistemi operazionali deve essere sempre garantita la continuità di funzionamento (**Regola 3 : Continuità operativa**).

Esempio tipico di basi di dati distribuite è quello dei sistemi dei conti correnti bancari. Un'applicazione locale può essere un accredito o un addebito su un conto corrente richiesto presso il terminale di un'agenzia di uno stesso nodo (cioè area geografica :ad esempio agenzie situate nella stessa regione). Un'applicazione distribuita può essere il trasferimento di fondi tra due conti correnti appartenenti a due aree diverse¹⁰.

¹⁰ G. Pelagatti :Le basi di dati distribuite - Masson 1985

Un'architettura come quella dei database distribuiti richiede di fatto una gestione molto complessa del dizionario dei dati o Metadata. In effetti, si prevede la realizzazione di uno **schema globale** che definisce i dati che sono inclusi nel database indipendentemente dalla localizzazione fisica. Quindi, l'esistenza di uno schema globale realizza l'indipendenza dalla distribuzione. Gli utenti devono, cioè, operare come se tutti i dati del database distribuito risiedessero sul loro sito (**Regola 4: Indipendenza di locazione**).

A un livello più basso troviamo lo **schema di frammentazione** e lo **schema di allocazione** che ci danno notizie sulla distribuzione dei dati. Lo schema di frammentazione dà notizia della frammentazione fisica di relazioni su più siti. Questa frammentazione può riguardare sia le righe (frammentazione orizzontale) a sia le colonne (frammentazione verticale) della relazione. La possibilità di scindere fisicamente le relazioni e di memorizzare queste parti in più siti separati può dare dei benefici di performance. I dati possono essere memorizzati nei siti dove sono più referenziati, riducendo il traffico di rete. Si dice che questi sistemi devono supportare l'indipendenza di frammentazione (**Regola 5: Indipendenza di frammentazione**). L'allocazione dei frammenti sui nodi è descritta dallo schema d'allocazione. Infine lo schema più basso in questa gerarchia è rappresentato dallo **schema dei dati dei singoli nodi**. Tutti questi dati devono essere presenti nel Metadata.

I database distribuiti possono essere *specifici* oppure *aperti*. Nel primo caso il database distribuito è costruito dal nulla: ogni database di nodo viene progettato per soddisfare specificamente le esigenze sia locali sia globali. E' più facile da costruire rispetto al secondo tipo poiché si deve realizzare sia un'integrazione delle diverse tipologie di hardware stratificate nel tempo (**Regola 8: Indipendenza dall'hardware**), sia l'indipendenza dai diversi sistemi operativi (**Regola 8: Indipendenza dal sistema operativo**), sia l'indipendenza dalla rete (**Regola 8: Indipendenza dalla rete**), sia l'indipendenza dai DBMS scelti (**Regola 8: Indipendenza dal DBMS**). Tutte

questi tipi d'indipendenza si sono realizzate tramite l'utilizzazione di protocolli come quello OSI, TPC/IP, ecc.. e di altri software per rendere trasparenti i dati (ODBC, IDAPI) alle varie applicazioni¹¹.

¹¹ Affinché la trasmissione fisica dei dati avvenga con successo occorre utilizzare opportune regole o protocolli. In generale *il protocollo è un insieme di regole che devono essere osservate per consentire uno scambio ordinato di informazioni fra più punti in modo che l'insieme di elementi che compongono la rete obbediscano ad uno stesso codice di comportamento. Quindi un protocollo definisce le regole che governano le caratteristiche elettriche, fisiche e funzionali di collegamento* (M.G. Ceppatelli "I sistemi informativi Aziendali").

Il problema principale da risolvere è quello della compatibilità fra sistemi di elaborazione e comunicazione attualmente esistenti. Nel caso della trasmissione fisica dei dati questi protocolli sono stati standardizzati dal comitato 802 dello IEEE (International Electrical and Electronic Engineers).

Scopo del modello OSI (open system interconnection) stabilito dall'ISO (international standard organisation) è quello di definire un insieme di regole tali che i sistemi siano coerenti con queste e possano cooperare e comunicare fra loro anche se sono di produttori diversi ed impiegano regole di funzionamento differenti. L'esigenza della comunicazione sorge in una organizzazione quando, ad esempio, si sceglie un sistema del produttore X per le applicazioni gestionali, di Y per il calcolo scientifico e di Z per altre applicazioni particolari. Maggiori, poi, sono le esigenze di comunicazione fra sistemi extraziendali poiché difficilmente le aziende possiedono lo stesso sistema.

Il modello OSI stabilisce degli standard attraverso i quali realizza la comunicazione fra sistemi diversi. Questo compito di comunicazione fra sistemi diversi viene suddiviso in sette subcompiti detti livelli o strati nei quali sono definite le funzioni particolari che devono svolgere. Questi livelli non sono altro che insiemi di programmi che svolgono compiti ben identificati (a livello di applicazione, di sistema operativo, o all'interno dei chip della scheda di rete) interagendo con i programmi degli altri livelli tramite delle procedure di validità generale.

Si tratta di un sistema gerarchico in cui i primi 4 livelli (i più bassi) raggruppano le funzioni definite globalmente di trasporto, i livelli 5 e 6 le funzioni di consegna e il livello 7 le funzioni di supporto alle applicazioni dell'utente. Solitamente l'insieme dei 7 livelli viene definito "stack" o "pila". La procedura di funzionamento del sistema è la seguente: quando una applicazione deve comunicare dati alla controparte remota, essa passa al livello sottostante i dati veri e propri da trasmettere ; una testata, inserita all'inizio dei dati, in cui vengono specificati comandi o richieste o semplici informazioni destinate allo strato applicativo remoto e infine i comandi destinati allo strato inferiore. Questo considera l'insieme di testata e dati veri e propri come le informazioni da destinare al sistema remoto, quindi, basandosi sui comandi ricevuti, esegue in proprio alcune funzioni, e altre le delega al proprio omologo remoto, aggiungendo una ulteriore intestazione al messaggio. Infine passa il nuovo campo di informazioni, al quale è stata aggiunta una nuova intestazione, e un opportuno comando al livello inferiore. La tecnica che consiste nell'aggiungere ad ogni passaggio un'intestazione ad hoc per ciascun livello si chiama imbustamento (encapsulating) ed in questo modo vengono realizzati protocolli, veri e propri elementi centrali del modello. Quindi possiamo dire che i protocolli esistono solo fra i livelli omologhi di due computer collegati in rete, per cui tramite i dati contenuti in ciascuna intestazione i livelli omologhi si comunicano reciprocamente il modo corretto di trattare i dati. Il processo di imbustamento, cioè di aggiunta di una intestazione, e di passaggio a livello successivo prosegue per tutta la struttura OSI. Arrivati, poi, a livello di collegamento dati oltre alla intestazione, si aggiunge anche una coda, in cui questo livello mette il risultato della propria manipolazione matematica della struttura dei dati che deve essere trasmessa, chiamata

frame check sequence (FCS). Il messaggio viene trasmesso al livello fisico che ne cura la trasmissione in rete al suo omologo remoto. Il sistema ricevente sottopone la parte dati del messaggio ad identica manipolazione matematica (livello collegamento dati remoto) e confronta il risultato ottenuto con il contenuto del campo FCS. Se i due risultati sono uguali si può dire che la trasmissione si sia svolta quasi sicuramente senza errori. Ogni livello compie le azioni in accordo a quanto specificato nell'intestazione di sua competenza, depenna la testata di sua competenza dal blocco e passa quanto rimasto al livello superiore, che si comporta in maniera analoga. Il risultato di tutti questi passaggi è che allo strato applicativo del destinatario remoto vengono passati solo i dati veri e propri e i comandi a livello applicativo.

Lo scopo della interconnessione tra reti è quello di mettere in collegamento una stazione residente su di una rete locale con quella di un'altra rete locale. Il collegamento può essere LAN-WAN oppure LAN-LAN. I dispositivi di interconnessione realizzano l'interconnessione di due reti isolate. Questi dispositivi possono dirsi "remoti" se interconnettono due reti geograficamente isolate, si parlerà invece di dispositivi "locali" se usati in ambiti geograficamente più ristretti (ed esempio possono collegare reti locali di una società poste tutte in uno stesso edificio).

In generale, per poter realizzare una rete composta da reti tramite questi dispositivi di interconnessione, questi ultimi dovranno inglobare una serie di funzioni che vanno dalla semplice ripetizione del segnale alla realizzazione di particolari protocolli per la comunicazione di reti che hanno delle architetture totalmente differenti.

Col modello OSI si dà la possibilità a due utenti, tramite la realizzazione di protocolli (regole) di comunicare. La struttura della rete viene descritta in 7 strati o livelli a ciascuno dei quali vengono assegnate funzioni ben precise. Per cui, per ogni strato si devono eseguire queste funzioni in maniera ben precisa indipendentemente dalle diverse modalità seguite per realizzarle dalle macchine dei diversi costruttori. Ora, nell'ambito di una rete locale siamo sicuri che i pacchetti (o stack) di funzioni ai vari livelli per permettere la comunicazione fra due utenti risultino identici mentre nel caso di due reti i pacchetti (o stack) dei protocolli saranno generalmente differenti. Per fare colloquiare due macchine poste su reti diverse occorre trovare un livello di interconnessione, ovvero un livello per cui le due entità finali presentino dei protocolli sufficientemente simili. Possono presentarsi casi di complessità diversa di interconnessione.

Esistono varie tecniche utilizzate per l'interconnessione e qui ne prenderemo una in particolare detta PDU (Protocol Data Units). Per PDU si intende l'unità dati del protocollo, cioè la struttura dei messaggi gestiti da un protocollo. Il livello di interconnessione tra utenti che abbiano una suite di protocolli in parte diversa è rappresentato dall'ultimo livello (partendo dal primo al settimo) in cui le due suite differiscono.

Repeater

E' il più semplice sistema di interconnessione ed opera a livello 1 del modello OSI. Viene applicato alle reti quando raggiungono estensioni territoriali tali per cui ne va della qualità del segnale. Si preferisce, quindi, dividere la rete in più segmenti e collegarli a questo repeater che ha come solo compito quello di rigenerare il segnale.

Bridge

I bridge lavorano allo strato superiore a quello dei repeater, vale a dire a livello 2. Però rispetto a quest'ultimo il bridge svolge attività di elaborazione, mentre il repeater si occupa solo di ripetere il messaggio. Il bridge fa sì che il messaggio trasmesso su una rete, e che non abbia destinatario su quella rete, venga intercettato e trasmesso su un'altra rete. Per far ciò viene analizzata solo la testata e non l'intero blocco. E' logico che i bridge devono conoscere gli indirizzi delle reti collegate.

Infatti ogni messaggio che viaggia sulla rete porta l'indirizzo del nodo sorgente e quello del destinatario. Il bridge, analizzando gli indirizzi sorgente, costruisce in modo automatico delle tabelle con la locazione delle stazioni. Le tabelle verranno poi utilizzate per decidere se una certo messaggio deve essere trasmesso verso una data LAN oppure no in modo tale da

ridurre il traffico sulla rete al minimo. Il bridge è una macchina costituita da due parti identiche dotate ambedue di processore e quindi con capacità autonome di elaborazione, con una parte di memoria comune a fare da punto di giunzione. Possono essere sistemi appositamente progettati oppure PC con software apposito. Esistono due tipi di bridge: quelli locali e quelli remoti. I primi uniscono o separano due segmenti di rete, i secondi invece uniscono reti geografiche. Questo significa che se le reti da collegare insieme sono molto vicine le funzioni del bridge stanno tutte in una singola macchina, mentre in reti distanti le funzioni bridge possono essere poste agli estremi di una linea. Per cui si può collegare insieme due LAN remote grazie alle due metà del Bridge e ad una linea a bassa velocità.

Router

I router vengono utilizzati per connettere reti locali logicamente separate utilizzando però lo stesso protocollo di trasporto (quindi l'ultimo protocollo non eguale è il 3, quello di rete). I router agiscono a livello rete del modello OSI deducendo le informazioni di routing (instradamento) da campi trasportati dai protocolli di livello 3. L'utilizzo dei router permette un filtraggio più efficiente e la scelta dell'instradamento migliore. Dal punto di vista della installazione e della gestione i router sono però più complessi rispetto ai bridge. I router, comunque, sono complementari ai bridge poiché servono a risolvere problemi diversi. L'ideale è avere prodotti che uniscano le caratteristiche migliori di entrambi. Questo ha portato all'introduzione di apparati ibridi come i brouter e i routing bridge.

Gateway

Con gateway, in generale, si indicano dei sistemi che operano ai livelli OSI superiori: sessione, presentazione, applicazione. La grande potenza dei gateway è di connettere fra loro reti che hanno una architettura completamente diverse. Si ottiene, così, la conversione di tutti i protocolli dello stack relativo ad una architettura in quelli dell'altra senza alterare i dati.

Un particolare tipo di gateway è il protocollo TCP/IP che permette la comunicazione fra sistemi appartenenti o no alla stessa rete. Questo protocollo fra l'altro è alla base della rete delle reti, ovvero internet.

Il software TPC/IP ha come scopo quello di far comunicare fra loro sistemi incompatibili, appartenenti o no alla stessa rete. Si parla di TPC/IP come di un protocollo, ma in verità è costituito da una suite di protocolli i cui più importanti sono il TPC (Transmission Control Protocol) e IP (Internetwork Protocol). TPC/IP è una soluzione comunicativa di tipo simmetrico come quella OSI.

L'IP gestisce l'indirizzamento fra due diversi computer (a ciascuno viene assegnato un indirizzo di rete che consista in un numero formato da 4 byte) e per regolare l'eventuale frammentazione dei pacchetti di dati che vengono scambiati. IP ha all'incirca funzioni corrispondenti a quelle del terzo livello OSI. Il TPC si muove a livello più alto corrispondente al quarto livello, quello di trasporto, di OSI. E fornisce un mezzo per lo scambio di dati affidabile sotto tutti gli aspetti: controlla che i pacchetti vengano ricevuti senza corruzioni, in sequenza e provvede metodi di ritrasmissione e correzione degli errori, oltre a consentire di scegliere quale porta del computer remoto aprire per la comunicazione. A livello dell'IP troviamo una serie di protocolli d'appoggio: l'ARP (Address Resolution Protocol) si occupa di mappare gli indirizzi (numeri) di rete sugli indirizzi fisici. Il RAR (Reverse ARP) fa l'inverso, l'ICMP (Internet Control Message Protocol) si occupa di rilevare eventuali problemi in modo che possano essere corretti.

A livello TCP troviamo l'UDP (User Datagram Protocol) che mette a disposizione il trasporto dei dati senza però assicurare nulla sulla avvenuta ricezione.

Al di sopra di TPC troviamo diversi protocolli di applicazione come Telnet per l'emulazione di terminale su computer remoto, l'Ftp per il trasferimento dei file, l'SMTP per la gestione della posta elettronica, il DNS per la mappatura dei nomi in rete, l'HTTP per la gestione dei collegamenti WWW, ecc....Ciascuno di questi protocolli fornisce un servizio a quello al di sopra di lui e utilizza servizi forniti da quelli sotto.

6.4.1. Il Two phase commit e la replicazione

La maggiore difficoltà nella realizzazione di sistemi distribuiti è relativa alla realizzazione di protocolli capaci di serializzare le transazioni (**Regola 8: Gestione della transazione distribuita**). Il metodo più affermato è quello del *two phase commit* (2PC). Questo è un algoritmo che, per effettuare un aggiornamento distribuito su più siti, designa dinamicamente un gestore della transazione detto *commit manager* o sito coordinatore. Il sito coordinatore potrebbe essere, ad esempio, quello che ha lanciato la transazione. Il commit manager provvede a verificare la disponibilità di tutti i nodi interessati alla transazione. Avuta risposta positiva da tutti i sistemi coinvolti, inizia la prima fase della transazione che si limita a preparare gli aggiornamenti necessari su ogni sito, senza eseguirli realmente. Terminata questa fase di preparazione, il commit manager autorizza l'effettiva esecuzione delle operazioni necessarie. Dopo questa autorizzazione ogni sistema dovrà presentare all'utente le modifiche realmente avvenute. Un sistema di questo genere ha però come grosso inconveniente di ridurre pesantemente l'efficienza nell'accesso ai dati e rendere lente le operazioni di aggiornamento.

In verità, non sono molte le transazioni distribuite che hanno un effettivo bisogno d'aggiornamento in tempo reale. Ci sono molte attività che richiedono l'accesso ad un numero di dati molto cospicuo per le quali un aggiornamento in tempo reale o leggermente differito non ha poi una così grande importanza. Si tratta in particolare di applicazioni come quelle MIS, DSS, d'analisi statistica che utilizzano dati globali dell'azienda per le quali è essenziale soprattutto una grande velocità d'accesso ai dati. Collegare queste attività direttamente ad un sistema come quello distribuito o in generale un

Documenti di riferimento :R. Adinolfi LAN e sistemi operativi- Masson 1992 ; S. Buonopane Standard e tipologie - Bit 3/1991 ; G. Maruzzelli :TCP/IP 3/1995 Bit ; G. Barbaro : Le Matrioske delle reti Start ; Ceccarelli, giardino :L'interconnessione delle reti,10/1991 MC ; M.G.Ceppatelli :I sistemi informativi aziendali, Cedam 1992

ambiente legacy (si veda per questo il capitolo 7) può soltanto provocare un forte appesantimento dello stesso.

Si è pensato, quindi, di realizzare per queste funzioni un sito secondario detto di replicazione. Cioè, i dati che interessano queste applicazioni vengono replicati su questo sito, realizzando una fotografia dei dati d'interesse degli utenti ad una certa data, dal quale si estrarranno i dati per le varie interrogazioni. Questi dati, poi, periodicamente dovranno essere aggiornati. In questo modo si permette di non caricare troppo il sistema che deve svolgere oltre che le funzioni per le quali è stato realizzato (funzioni operative) anche quelle di natura globale che interessano l'azienda come quelle a supporto delle decisioni.

Però questo tipo d'approccio non vale per tutte quelle operazioni che richiedono necessariamente un aggiornamento in tempo reale. Un esempio può essere quello relativo alla prenotazione di un volo aereo, oppure quello relativo all'utilizzo di tessere Bancomat. Nel caso del Bancomat occorrerà evitare che l'intero importo disponibile per il prelievo venga ritirato in istanti ravvicinati di tempo da agenzie diverse ognuna dotata di un database locale. Il ritardo dell'aggiornamento potrebbe consentire operazioni non autorizzate e quindi potrebbe consentire di prelevare più danaro di quanto sia effettivamente nella disponibilità del proprietario della tessera Bancomat.