Cooperazione di Basi di Dati Autonome: Criteri di Classificazione e Strumenti di Middleware *

Paolo Atzeni[1], Luca Cabibbo[1], Giansalvatore Mecca[2]

[1] Dipartimento di Discipline Scientifiche, Sezione di Informatica
Università degli Studi di Roma Tre
[2] D.I.F.A. — Università della Basilicata, Potenza
{atzeni,cabibbo,mecca}@dis.uniroma1.it

Abstract. Vengono discusse alcune problematiche relative alla realizzazione di sistemi informativi cooperanti, con particolare riferimento al ruolo delle basi di dati. Viene notato in primo luogo come l'eterogeneità, l'autonomia e la distribuzione, pur essendo presenti in misura maggiore o minore, costituiscono, soprattutto nel breve termine, vincoli sui sistemi cooperanti. Di conseguenza, vengono individuati nuovi criteri di classificazione: (a) il livello di trasparenza; (b) la complessità delle operazioni distribuite; (c) il livello di attualità dei dati. Ciò porta ad individuare tre categorie principali di sistemi: (i) sistemi multidatabase; (ii) collezioni di dati replicati; (iii) sistemi informativi locali con dati esterni. Per ciascuna di queste categorie vengono individuate soluzioni architetturali basate su strumenti di middleware disponibili sul mercato.

1 Introduzione

Grazie alla affermazione delle tecnologie delle reti di telecomunicazione ed alla effettiva realizzazione di tali reti in molti contesti applicativi, è divenuto sempre più attuale il problema di fare cooperare i sistemi informativi di una o più organizzazioni, eventualmente distribuiti nel territorio, connessi tramite una rete. In questo lavoro, ci occupiamo di aspetti architetturali, progettuali e metodologici che la presenza di una rete di telecomunicazione pone rispetto al disegno di singole applicazioni cooperanti, in particolare di quelle che interessano più organizzazioni, o più dipartimenti di una grande organizzazione, dotati di sistemi informativi realizzati autonomamente. Infatti, la presenza di una rete di telecomunicazione, oltre alla *connettività* tra i sistemi, offre direttamente solo degli strumenti di *interoperabilità*, quali, ad esempio, l'accesso da terminale remoto (*telnet*), il trasferimento di file (*ftp*), la posta elettronica, lo scambio di dati e

* Questo lavoro è stato svolto nell'ambito di un contratto di consulenza fra l'Autorità per l'Informatica nella Pubblica Amministrazione e il Dipartimento di Discipline Scientifiche dell'Università di Roma Tre. Gli autori sono stati anche supportati dal *MURST*, nell'ambito del progetto "Basi di dati evolute: modelli, metodi e strumenti," e dall'Università di Roma Tre, nell'ambito del progetto "Sistemi intelligenti," e dal *Consiglio Nazionale delle Ricerche*.

documenti in formato standard (*EDI*), l'accesso a informazioni multimediali ed ipertestuali (*http*). Tali strumenti non sono però sufficienti a dare una soluzione alle esigenze di *cooperazione* tra i diversi sistemi, che solitamente prevedono lo sviluppo di applicazioni complesse che operino sul patrimonio informativo distribuito tra le varie organizzazioni o dipartimenti.

La cooperazione fra sistemi informatici può presentarsi sotto due aspetti fondamentali:

- cooperazione centrata sui dati: i dati di un sistema sono visibili ad altri sistemi (si può trattare di una comunicazione fra pari o mediante un coordinatore);
- cooperazione centrata sui processi: i sistemi offrono l'un l'altro (ancora fra pari o con un coordinatore) servizi, attraverso scambio di messaggi, informazioni, documenti, o innescamento di attività.

In questo documento ci soffermiamo sul primo aspetto, ovvero sulla cooperazione centrata sui dati. È evidente che cooperazione centrata sui processi e cooperazione centrata sui dati sono due aspetti ortogonali delle applicazioni tra più sistemi informativi. Essi non sono invece aspetti indipendenti, per cui in generale in molte applicazioni si possono riscontrare entrambe le forme di cooperazione. Ad un primo livello di approssimazione, comunque, si può considerare che le applicazioni cooperanti siano caratterizzate dalla forma di cooperazione prevalente in esse, quella che ne qualifica le funzionalità e dà forma alla loro architettura.

Le problematiche che esaminiamo sono relative alla realizzazione di sistemi informativi cooperanti, con particolare riferimento al ruolo delle basi di dati, nei casi in cui i sistemi componenti siano stati realizzati autonomamente. Questa situazione si presenta effettivamente nei casi in cui lo sviluppo di varie basi di dati sia avvenuto in tempi e luoghi diversi, dando luogo a situazioni estremamente eterogenee, e per lo più non integrate, nel senso che non è possibile in molti casi accedere da un sistema informativo a dati contenuti nella base di dati di un altro sistema. Situazione analoga si ha nei casi in cui le esigenze di cooperazione siano emerse dopo la realizzazione dei vari sistemi informativi. Si pensi, ad esempio, al caso in cui avviene una fusione tra diverse aziende, dando luogo ad una nuova organizzazione, di cui si voglia avere un unico sistema informativo, ottenuto come integrazione dei sistemi aziendali preesistenti. Come già osservato, la presenza di una rete di telecomunicazione fornisce l'infrastruttura di comunicazione per realizzare la connettività tra i sistemi, ed un insieme di servizi di base, atti a realizzare l'interoperabilità tra tali sistemi. In alcuni casi, i servizi di interoperabilità sono sufficienti e risolvere le esigenze di comunicazione tra i vari sistemi. In altri casi, invece, sarà necessario dare il via a vere e proprie iniziative di cooperazione, per lo sviluppo di applicazioni complesse che prevedano l'integrazione dei dati esistenti. Al fine di risolvere efficacemente tali iniziative, è necessario individuare una opportuna architettura applicativa, solitamente basata su una specifica tecnologia di comunicazione tra sistemi possibilmente eterogenei, corrispondente ad una classe di prodotti software commerciali. Queste tipologie di

prodotti, solitamente chiamati di *middleware*, corrispondono a strati software che, da un punto di vista logico, operano tra lo strato applicativo e quello di rete. Ogni prodotto di middleware fornisce funzionalità, di alto oppure di basso livello, che permettono la comunicazione e l'interoperabilità tra le applicazioni eseguite ai diversi estremi della comunicazione, con lo scopo di mascherare una specifica eterogeneità tra tali applicazioni.

Questo documento è organizzato come segue. Nel tentativo di classificare le problematiche di cooperazione centrate sui dati, individuiamo alcuni criteri guida (Sezione 2) e, sulla base di questi criteri, suggeriamo alcune categorie principali di applicazioni cooperanti (Sezione 3). In un contesto così complesso è infatti possibile pensare a diversi livelli di integrazione tra sistemi eterogenei; ogni livello è caratterizzato da costi, vincoli tecnologici e organizzativi diversi. Vengono poi discusse le tecniche, ovvero gli strumenti di middleware, disponibili sul mercato (Sezione 4), nonché le possibili soluzioni architetturali alla cooperazione (Sezione 5). Vengono infine proposte alcune considerazioni di carattere metodologico (Sezione 6); riteniamo infatti importante, nell'ambito dell'analisi costi-benefici di una iniziativa di cooperazione, una riflessione di carattere metodologico sulle effettive esigenze di integrazione dei dati.

2 Criteri per la Classificazione della Cooperazione

L'analisi di applicazioni cooperanti richiede solitamente la valutazione di alcuni parametri caratteristici, e cioè il livello di distribuzione, il livello di eterogeneità e il livello di autonomia, che possono essere sommariamente definiti come segue [12].

- Il *livello di distribuzione* è una misura del decentramento delle informazioni. Più in generale, possiamo definirlo come una misura del decentramento di tutte le risorse, quindi non solo dei dati, ma anche delle applicazioni.
- Il *livello di eterogeneità* è una misura delle differenze tra i vari sistemi. Le eterogeneità possono essere dovute a: differenze nell'ambiente hardware e software, in particolare riguardo ai sistemi di gestione di basi di dati (con differenze nei modelli dei dati, nei linguaggi di manipolazione e nelle tecniche e negli strumenti per la gestione delle transazioni) e a differenze nella semantica dei dati, cioè alla possibilità di utilizzare rappresentazioni diverse per gli stessi concetti o rappresentazioni uguali per concetti diversi.
- Il *livello di autonomia* è una misura della possibilità da parte delle basi di dati componenti di mantenere la propria struttura (cioè il proprio modello per la descrizione dei dati e il proprio linguaggio di interrogazione, eventualmente diversi dagli altri) e di continuare a soddisfare le richieste di utenti locali, pur concorrendo ad eseguire funzionalità globali.

Questi tre criteri, in generale interessanti, sono stati però proposti [12] con riferimento a contesti in cui abbia senso parlare di un sistema informativo integrato, con una forma più o meno esplicita di controllo centrale, e quindi con la possibilità di intervenire sul grado di distribuzione, eterogeneità e autonomia dei componenti. Viceversa, è di interesse considerare anche e soprattutto situazioni in

cui si vogliono far cooperare sistemi completamente indipendenti l'uno dall'altro, sviluppati nel tempo secondo modelli ed architetture estremamente difformi. In generale, quindi, la cooperazione deve poter essere realizzata senza interferire con le funzionalità dei sistemi esistenti, preservandone la struttura e la distribuzione, con limitata possibilità di coordinamento centrale. Ovviamente, in alcuni casi, ed auspicabilmente molti, le iniziative di cooperazione potranno costituire una importante occasione per razionalizzare e riorganizzare questi sistemi, modificandone anche il grado di eterogeneità, di distribuzione e di autonomia. Il tutto richiede una valutazione dei costi e dei benefici, nonché dei requisiti normativi e della molteplicità delle iniziative di cooperazione, in quanto vi potrebbero essere sistemi coinvolti in più di una. Comunque, la discussione deve considerare anche il caso peggiore in cui i livelli di distribuzione, eterogeneità e autonomia siano alti e non modificabili, e costituiscano quindi vincoli per la cooperazione, soprattutto nel breve termine. Questo fatto rende inadeguati i tre criteri tradizionali e impone l'adozione di nuovi criteri, che permettano di raffinare la classificazione dei sistemi cooperanti. Noi abbiamo individuato i tre criteri seguenti, che sono discussi nelle prossime sezioni: (i) il livello di trasparenza nell'accesso ai dati distribuiti; (ii) la complessità delle transazioni distribuite; e (iii) il livello di attualità dei dati.

2.1 Livello di Trasparenza nell'Accesso ai Dati Distribuiti

Il *livello di trasparenza* nell'accesso ai dati distribuiti è una misura di quanto la distribuzione e l'eterogeneità dei dati siano mascherati, e cioè di quanto l'insieme delle basi di dati cooperanti appaia all'esterno come un'unica base di dati.

- Al livello di trasparenza più alto ci sono quei sistemi che integrano basi di dati eterogenee e distribuite permettendo di vedere il sistema integrato attraverso una *interfaccia unica*, e cioè come una base di dati virtualmente unica. Questa interfaccia può specificare l'insieme delle funzionalità definite sulla base di dati, oppure una vera e propria descrizione integrata dei dati, cioè uno *schema globale* dei dati. L'applicazione cooperante verrà definita relativamente a questa interfaccia; il sistema integrato si incarica di soddisfare le richieste, attivando le funzionalità ed accedendo i dati contenuti nelle singole basi di dati.
- Al livello più basso, invece, ci sono sistemi in cui la distribuzione e l'eterogeneità dei dati non sono mascherati. Le basi di dati componenti mettono a disposizione una serie di funzionalità ed è compito del programmatore delle applicazioni localizzare i dati per le elaborazioni, e trasformarli e correlarli opportunamente.

È importante notare che la definizione di un'interfaccia unica, in particolare se costituita da uno schema globale, prevede lo svolgimento di attività di analisi e progettazione assai delicate. La metodologia tradizionalmente utilizzata è di tipo *bottom-up*: gli schemi delle singole basi di dati vengono prima tradotti in un modello di dati comune e poi integrati per dare origine allo schema globale [2]. L'attività di traduzione richiede la definizione di opportune trasformazioni

tra modelli di dati. L'attività di integrazione, ancora più delicata, richiede la soluzione dei potenziali *conflitti*, strutturali e semantici, tra i vari schemi componenti. Per esempio, in molti casi lo stesso concetto può essere descritto nei vari schemi di esportazione utilizzando costrutti diversi del modello comune. È dunque necessaria un'accurata analisi delle sovrapposizioni tra i diversi schemi per risolvere queste forme di eterogeneità dei dati. Al tempo stesso, la definizione di uno schema globale porta alla risoluzione dei conflitti e delle eterogeneità una volta per tutte, mentre nei sistemi a bassa trasparenza, gli stessi problemi vanno riaffrontati ogniqualvolta si acceda ad una o più specifiche base di dati non locali.

2.2 Complessità delle Transazioni Distribuite

La *complessità delle transazioni distribuite* è una misura del grado di coordinamento necessario per effettuare operazioni (interrogazioni e aggiornamenti) sulle basi di dati cooperanti. La gestione di transazioni su basi di dati distribuite presenta numerose problematiche [5] relative all'integrità ed alla affidabilità dei dati, tanto da rendere il livello di complessità delle transazioni distribuite un criterio importante di classificazione.

- I sistemi che prevedono lo svolgimento di complesse transazioni distribuite (con aggiornamenti oppure interrogazioni che coinvolgono più componenti in modo significativo) su basi di dati eterogenee ed autonome richiedono lo sviluppo di sofisticate componenti software che garantiscano la corretta esecuzione delle operazioni.
- I sistemi, invece, in cui le transazioni da svolgere sulle singole basi di dati sono semplici, non richiedono una gestione sofisticata. Il caso più semplice è quello degli accessi in sola lettura effettuati separatamente sulle singole basi di dati.

2.3 Livello di Attualità dei Dati

Il *livello di attualità dei dati* è tanto più alto quanto meno l'informazione associata ai dati è superata. In particolare, in un sistema cooperante abbiamo due possibilità:

- accesso ai dati originali (nel sistema che li gestisce);
- accesso a copie o più in generale a dati derivati (in un sistema più facilmente accessibile rispetto a quello che gestisce i dati originali), per i quali l'allineamento con gli originali non è garantito. Sono possibili gradi diversi di obsolescenza (e quindi inversamente di attualità) a seconda delle modalità seguite per il riallineamento (ad esempio, relativamente alla frequenza del riallineamento stesso).

L'ipotesi più comune alla base dei meccanismi di replicazione dei dati prevede che ciascun dato abbia un responsabile unico, cui corrisponde l'unica *copia primaria* del dato stesso; è viceversa possibile che esistano più *copie secondarie* di ciascun dato. Accessi in lettura ad un dato possono avvenire tramite la copia

primaria (di cui deve essere garantita l'attualità) oppure attraverso una delle copie secondarie (delle quali possono essere garantiti solo livelli controllati di attualità). L'aggiornamento di un dato deve invece riferirsi sempre alla copia primaria del dato, per essere propagata, secondo regole e tempi opportuni, alle sue copie secondarie.

L'accesso in tempo reale a dati aggiornati è spesso delicato:

- la distribuzione può causare un traffico non trascurabile sulla rete (in particolare se le interrogazioni coinvolgono volumi di dati significativi e/o più di una base di dati) oltre che richiedere una elevata affidabilità della rete stessa e del sistema remoto;
- l'eterogeneità può richiedere trasformazioni dei dati non banali (realizzabili spesso solo tramite prodotti software molto costosi o programmi difficili da mantenere).

D'altra parte, la gestione delle copie richiede la tollerabilità di un livello ridotto di attualità (e l'individuazione del livello accettabile) e può ugualmente causare un traffico significativo sulla rete se le operazioni di riallineamento sono frequenti e/o coinvolgono molti dati. In generale, il grado di attualità può variare nell'ambito di uno stesso contesto, in quanto le esigenze relative alle varie classi di dati possono essere diverse.

3 Classificazione delle Modalità di Cooperazione

Sulla base dei criteri presentati nella sezione precedente, individuiamo alcune categorie principali per la cooperazione tra basi di dati. Allo scopo, notiamo che i tre criteri non sono completamente indipendenti, e quindi non è necessario considerare tutte le possibili combinazioni. Infatti, un alto grado di complessità delle operazioni richiede, al fine di una realizzazione efficace ed efficiente, la presenza di una infrastruttura per l'integrazione, quindi un alto livello di trasparenza. In questo contesto, vi è spesso una generale esigenza di un alto grado di attualità, cui talvolta si rinuncia, a favore dell'utilizzo di copie, con riferimento ad una porzione più o meno grande della base di dati globale, tenendo conto di una opportuna valutazione dei benefici e delle necessità rispetto ai costi e ai rischi. Abbiamo individuato in questo modo una prima categoria, quella dei *sistemi multidatabase*. Riducendo al minimo il grado di attualità e mantenendo l'alto livello di trasparenza e di complessità (delle sole interrogazioni, perché gli aggiornamenti non hanno senso in questo caso) individuiamo una seconda categoria, quella delle *collezioni di dati replicati*, che, con il supporto di replicazioni (e trasformazioni), permettono l'accesso in sola lettura a copie secondarie e derivate, ed elaborazioni fuori linea. Non consideriamo esplicitamente il caso di sistemi con alto grado di trasparenza e basso grado di complessità delle operazioni, in quanto può rientrare come caso particolare, più semplice, dei precedenti. Infine, un basso grado di trasparenza è compatibile solo con un basso grado di complessità, con un grado di attualità ancora variabile a seconda delle esigenze specifiche; parliamo in questo caso di *sistemi informativi locali*

con dati esterni. Vale la pena notare che è immaginabile che le esigenze reali di cooperazione in molti casi rappresentino tipologie intermedie, che presentano aspetti ora dell'una, ora dell'altra categoria. Tuttavia, le categorie individuate permettono di discutere efficacemente le caratteristiche fondamentali della cooperazione. Esaminiamo ora in dettaglio le varie categorie.

3.1 Sistemi Multidatabase

Con *sistema multidatabase* intendiamo una collezione di basi di dati distribuite, autonome ed eterogenee, dotata di un ulteriore sistema, detto *sistema globale*, che permette l'accesso dall'esterno alle singole basi di dati componenti. La complessità di questo sistema è variabile, a seconda della modalità di accesso alle singole basi di dati. In un sistema multidatabase, le singole basi di dati mettono a disposizione una serie di funzionalità; il sistema globale si incarica di integrare opportunamente queste funzionalità, permettendo l'accesso agli utenti esterni. Relativamente ai tre criteri sopra discussi, questi sistemi presentano certamente un alto grado di trasparenza nell'accesso ai dati distribuiti, dal momento che il sistema di integrazione s'incarica di mascherare l'accesso alle singole basi di dati. In linea di principio, dovrebbero fornire anche la possibilità di effettuare transazioni complesse (interrogazioni e aggiornamenti) e garantire un alto grado di attualità dei dati, consentendo l'accesso alla copia più aggiornata del dato. In realtà, le transazioni complesse in molti casi non sono necessarie e sono comunque difficili da gestire, mentre il requisito di attualità viene spesso rilassato, in un'ottica di rapporto fra costi (e rischi) e benefici.

Questo tipo di cooperazione è naturale in tutti i casi in cui sia necessaria l'integrazione stretta tra basi di dati, con significative operazioni che coinvolgano basi di dati diverse. Le basi di dati componenti possono essere omogenee nelle tipologie dei dati contenuti (ad esempio una ipotetica anagrafe integrata della popolazione, che integri, rispettandone l'autonomia, le basi di dati dei vari comuni, e permetta di svolgere, in tempo reale, attraverso una cooperazione forte, sia operazioni puntuali sia operazioni globali, ad esempio statistiche, con letture e scritture articolate) o disomogenee (ad esempio un ipotetico sistema informativo che integri informazioni anagrafiche e sanitarie). Il livello di trasparenza di questi sistemi può essere variabile. Infatti, in alcuni casi le basi di dati componenti mettono a disposizione del sistema di integrazione i loro dati, in altri casi solo una serie predefinita di funzionalità o servizi. Naturalmente, al crescere del livello di trasparenza del sistema, cresce la flessibilità del sistema integrato ma crescono anche il costo di realizzazione e la complessità tecnologica.

In questa scala di trasparenza è significativo esaminare l'estremo più alto, quello dei cosiddetti *sistemi di basi di dati federate ad accoppiamento forte* [12, 7, 8], in cui la collezione di basi di dati appare di fatto all'esterno come un'unica base di dati, sulla quale è possibile effettuare interrogazioni ed aggiornamenti complessi. L'eleganza e la flessibilità della soluzione federata la rendono ideale per l'integrazione di basi di dati eterogenee, autonome e distribuite, per le quali sia necessaria una cooperazione stretta. Allo stesso tempo, però, la realizzazione di un'architettura federata comporta un elevato rischio tecnologico ed alti costi.

Nel seguito esaminiamo la struttura di un sistema federato ad accoppiamento forte, e poi discutiamo le difficoltà associate alla realizzazione di queste architetture.

Sistemi federati ad accoppiamento forte Chiamiamo *base di dati federata* la collezione delle basi di dati componenti che partecipano alla federazione in modo autonomo e definendo un unico schema globale della federazione. Le basi di dati preesistenti della collezione sono dette basi di dati *componenti*. Ogni base di dati componente ha un proprio modello dei dati (potenzialmente diverso da quello delle altre); esiste però un *modello dei dati comune*, utilizzato per lo schema della federazione. L'architettura dei sistemi federati ad accoppiamento forte viene di solito descritta secondo un'architettura a cinque livelli che estendono i due livelli superiori (*logico* ed *esterno*, trascurando quello *fisico* che rimane immutato) delle tradizionali architetture a tre livelli:

- lo schema logico di ciascuna delle basi di dati componenti (espresso nel rispettivo modello dei dati) viene detto *schema locale* della base di dati componente;
- la traduzione di ciascuno schema locale nel modello dei dati comune viene detta *schema componente*;
- la porzione dello schema componente messa a disposizione della federazione viene detta *schema di esportazione*;
- lo *schema federato* (o *globale*) è l'integrazione degli schemi di esportazione delle varie basi di dati componenti;
- uno *schema esterno* (come nelle architetture tradizionali) è una porzione (eventualmente riorganizzata) dello schema federato a disposizione di una classe di utenti.

Un sistema di basi di dati federate richiede la presenza di un *amministratore della federazione*, che svolga, a livello federato, funzioni analoghe a quelle svolte, sulle basi di dati locali, dai rispettivi amministratori. L'amministratore della federazione, quindi, è responsabile dello schema federato, della sua produzione come integrazione degli schemi esportati dalle componenti e delle autorizzazioni sugli schemi esterni. Relativamente alla federazione stessa, può essere considerato una entità di livello superiore rispetto agli amministratori locali, anche se, tenendo conto dell'autonomia, non ha alcuna autorità sulle basi di dati componenti (in linea di principio, non conosce nemmeno gli schemi locali e componenti, ma solo gli schemi di esportazione). In effetti, questa forma di cooperazione richiede un accordo forte fra le parti, che può essere realizzato attraverso il coordinamento esplicito di una di esse o di una istituzione esterna, oppure attraverso un protocollo ben definito di interazione fra pari.

Le varie trasformazioni (traduzioni e integrazioni di schemi, decomposizione di operazioni ecc.), definite dai rispettivi amministratori con opportuni linguaggi, vengono poi eseguite all'atto dell'accesso ai dati attraverso opportuni processori, ovvero moduli software che trasformano i dati da una rappresentazione all'altra.

Un sistema federato richiede un componente software specifico, detto *sistema di gestione della federazione*, in cui risiede lo schema federato ed al quale

gli utenti della federazione richiedono l'esecuzione delle transazioni. Il sistema di gestione rende di fatto trasparente l'eterogeneità e la distribuzione dei dati. Una volta ricevuta una interrogazione o una richiesta di transazione, s'incarica di scomporla in parti che sottomette alle singole basi di dati componenti. In particolare, il sistema si incarica sia di effettuare le opportune traduzioni, in modo da rendere le sotto-transazioni eseguibili da ciascuna base di dati, sia di garantire la correttezza dell'esecuzione distribuita (ad esempio, l'atomicità) attraverso opportuni protocolli, che di gestire l'ottimizzazione dell'esecuzione, scegliendo tra i possibili piani di esecuzione quello a costo ritenuto minimo. Inoltre, il gestore raccoglie le risposte e le integra, presentando il risultato all'utente.

Anche le basi di dati componenti devono essere dotate di un opportuno strato software che consenta la cooperazione. In particolare, ogni base di dati componente deve essere dotata di un modulo per la gestione dei dati che permette di rendere visibili, all'esterno, i dati locali nel formato del modello comune; questo modulo contiene gli schemi ed i processori necessari per tradurre lo schema locale nello schema di esportazione.

Limiti della tecnologia per le basi di dati federate e approcci intermedi Come già detto in precedenza, le basi di dati federate ad accoppiamento forte rappresentano idealmente l'architettura che risolve i problemi legati all'integrazione di basi di dati eterogenee, autonome e distribuite, per le quali sia necessaria una cooperazione stretta. Purtroppo, però, la realizzazione di un'architettura federata comporta un elevato rischio tecnologico ed alti costi, in quanto non vi è ancora una tecnologia matura sul mercato, da vari punti di vista.

Innanzitutto, non esistono, come sarebbe invece auspicabile, prodotti integrati che realizzino direttamente l'architettura a cinque livelli che abbiamo discusso. Come si vedrà meglio nella successive Sezioni 4 e 5, dedicate alle soluzioni tecnologiche ed alle architetture, il mercato propone prodotti che svolgono alcune delle funzionalità (ad esempio, per realizzare traduzioni da un ambiente ad un altro), e di validità solo parziale (ad esempio, con riferimento ad alcuni sistemi di gestione di basi di dati e non ad altri). Si può quindi pensare di realizzare una architettura simile a quella federata integrando prodotti diversi disponibili sul mercato e sviluppandone direttamente alcuni. Si tratta evidentemente di una soluzione che, con l'evoluzione della tecnologia, può presentare alti costi di manutenzione.

Inoltre, anche con riferimento a sistemi omogenei, la tecnologia attuale non presenta ancora soluzioni completamente soddisfacenti riguardo alle funzionalità di elaborazione ed ottimizzazione delle interrogazioni distribuite e gestione delle transazioni distribuite [5]. Di fatto, nessuno dei sistemi attualmente disponibili sul mercato fornisce ottimizzazione delle interrogazione ed efficienti protocolli per la gestione di transazioni distribuite. Questo vuol dire che spesso l'utente del sistema multidatabase deve esplicitamente prendere in considerazione, in fase di scrittura delle applicazioni, l'efficienza delle interrogazioni o la consistenza delle transazioni, con il risultato di ridurre le prestazioni del sistema o produrre dati inconsistenti.

In conclusione, possiamo dire che i sistemi federati costituiscono la forma di cooperazione sui dati più complessa da realizzare e più costosa e rischiosa sia dal punto di vista tecnologico sia da quello organizzativo. Deve quindi essere utilizzata solo nei casi in cui i benefici un sistema informativo integrato siano realmente significativi. Peraltro, le funzionalità di accesso trasparente a basi di dati integrate, con gestione di transazioni complesse e disponibilità di dati attuali sono indiscutibilmente significative, in termini di efficacia nell'utilizzo delle basi di dati e quindi di servizi resi agli utenti. Pertanto, si può affermare che le architetture federate ad accoppiamento forte debbano rappresentare un obiettivo di medio-lungo periodo, a cui avvicinarsi progressivamente sfruttando l'evoluzione della tecnologia per la realizzazione di soluzioni intermedie, che sono possibili attraverso una attività di integrazione di prodotti esistenti eventualmente con l'utilizzo di soluzioni ad hoc. Peraltro, i *Database Gateway Server*, cui si accennerà nella Sezione 5.1, sono strumenti software che forniscono un contributo importante, ma non risolutivo, in questa direzione. Per quanto riguarda le soluzioni intermedie, abbiamo già notato come la gestione delle transazioni complesse sia un'esigenza presente in una frazione relativamente limitata delle applicazioni di nostro interesse. Il grado di complessità è piuttosto basso e quindi una gestione ad hoc è ragionevolmente possibile. Inoltre, i problemi legati alla difficoltà di realizzare in modo efficiente interrogazioni distribuite complesse possono essere risolti, qualora i requisiti di attualità non siano stringenti, attraverso l'utilizzo di copie dei dati. In questo modo si limitano anche i danni conseguenti ad eventuali indisponibilità della rete o del sistema remoto. Gli strumenti di gestione di dati replicati, che verranno brevemente descritti nella Sezione 4.2, forniscono servizi abbastanza flessibili sia per quanto riguarda il riallineamento delle copie sia per quanto riguarda la traduzione fra sistemi eterogenei.

3.2 Collezioni di Dati Replicati

L'utilizzo delle copie secondarie può essere portato all'estremo quando necessità di pianificazione, di supporto alle decisioni o di analisi richiedono l'accesso in lettura a collezioni di informazioni integrate. In questi casi, spesso, non è strettamente necessario l'accesso alla copia più aggiornata del dato, e può essere accettabile accedere a dati con un livello controllato di attualità. Al tempo stesso, le interrogazioni sono spesso complesse (anche con correlazioni voluminose fra basi di dati diverse) e non sono realizzabili in modo efficiente in un contesto distribuito ed eterogeneo su cui le transazioni ordinarie svolgono le loro operazioni in tempo reale. In questi casi, il sistema è caratterizzato da:

- alto livello di trasparenza nell'accesso ai dati distribuiti, perché il sistema si incarica di rendere omogenei i dati e di integrarli opportunamente;
- possibilità di accesso in sola lettura;
- livello di attualità ridotto, ma opportunamente controllato.

Collezioni di dati replicati, talvolta indicate con il termine *Data Warehouse*, sono ottenute estraendo le informazioni dalle singole basi di dati e replicandole,

creando una copia integrata locale dei dati di interesse. La possibilità di lavorare su copie secondarie dei dati, piuttosto che sulla copia primaria, permette una grande flessibilità nell'attività di integrazione. È possibile, per esempio, tradurre i dati estratti da basi di dati eterogenee (in particolare basi di dati con organizzazione antiquata, i cosiddetti sistemi legacy, eredità del passato) in un formato comune. Sull'insieme dei dati replicati ed opportunamente integrati è possibile poi effettuare elaborazioni fuori linea, per esempio sfruttando strumenti di supporto alle decisioni. L'utilizzo di copie secondarie (di solito su un sistema diverso) permette di diminuire il carico del sistema su cui risiede la copia primaria.

La replicazione di un dato può essere periodica oppure avvenire su richiesta. Gli strumenti di replicazione dei dati possono agire secondo due modalità: totale (il dato viene replicato *ex novo*) ed incrementale; le soluzioni incrementali sono preferibili nei casi un cui la dinamicità del dato sia bassa.

3.3 Sistemi Informativi Locali con Dati Esterni

In molti casi è necessario costituire sistemi informativi che accedono a dati di altri sistemi, messi a disposizione in modo esplicito e semplice: il sistema locale deve poter accedere ad una o più basi di dati esterne al sistema stesso attraverso un insieme di funzionalità predefinite. Questi sistemi sono caratterizzati come segue.

- Il livello di trasparenza nell'accesso ai dati distribuiti è basso. Ciascuna base di dati mette a disposizione un insieme predefinito di funzionalità (stabilite a seguito di opportuni accordi di cooperazione) a livello più o meno alto, ed è compito dell'applicazione far sì che il sistema locale fornisca all'utente una vista integrata del contenuto delle varie basi di dati, permettendogli di effettuare alcune interrogazioni ed aggiornamenti. Il grado di integrazione delle basi di dati esterne è quindi elementare.
- Le transazioni distribuite sono in numero limitato e di relativa semplicità. Questa è la caratteristica fondamentale della categoria di applicazioni in esame, che si adatta particolarmente bene ai casi in cui un'applicazione debba accedere a dati distribuiti, con gli accessi che avvengono per lo più ad una base di dati alla volta; in questo caso la complessità della cooperazione è ridotta e l'applicazione stessa può garantire la corretta esecuzione delle transazioni.
- Il livello di attualità dei dati è variabile, in linea di principio alto, ma con possibilità di soluzioni intermedie, nello stesso modo discusso per i sistemi multidatabase.

Sono possibili due alternative fondamentali per l'accesso da parte del sistema locale a basi di dati esterne:

- In alcuni casi, le basi di dati esterne mettono a disposizione del sistema locale i loro *dati* e lo scambio avviene attraverso un linguaggio, stabilito dallo specifico sistema esterno (ad esempio, SQL). Il sistema locale formula attraverso istruzioni del linguaggio le richieste alle varie basi di dati; queste, a

loro volta, restituiscono i dati in un formato opportuno; questi dati vengono poi elaborati localmente e presentati all'utente.
- In altri casi, invece, le basi di dati non condividono esplicitamente i dati, ma solo un insieme prefissato di *funzionalità*, ovvero un insieme di procedure. Il sistema locale può invocare, attraverso l'invio di messaggi, l'esecuzione di procedure da parte dei sistemi esterni, i quali rispondono comunicando i risultati delle elaborazioni. A differenza del caso precedente, le basi di dati, oltre che fornire dati, effettuano anche elaborazioni sui dati stessi. Ci sono due vantaggi legati a questo tipo di soluzioni: (a) il sistema locale viene scaricato di parte dell'elaborazione, che avviene a carico delle basi di dati esterne; (b) la possibilità di invocare solo un insieme fissato di procedure remote permette di "incapsulare" i dati, e quindi di rendere in qualche modo l'applicazione insensibile ai cambiamenti nella struttura delle singole basi di dati. Corrispondentemente a questi vantaggi, c'è lo svantaggio di una minore flessibilità, legato al fatto che possono essere chiamate solo le procedure esplicitamente concordate.

Vale la pena sottolineare che queste soluzioni sono applicabili solo nei casi in cui la complessità delle transazioni distribuite è bassa. Infatti il controllo dell'atomicità delle transazioni è completamente a carico del sistema locale (a differenza del caso di basi di dati federate, in cui, nel modello ideale, il sistema di gestione della federazione ha il compito esplicito di controllo delle transazioni).

4 Tecniche di Interscambio di Dati

Un sistema di base di dati cooperante può essere realizzato tramite diverse architetture. In questa sezione vengono esaminate le tecniche di base utilizzate per i singoli componenti di tali architetture; alcune soluzioni architetturali per la cooperazione centrata su dati sono proposte nella successiva Sezione 5.

Le tecniche che esaminiamo fanno riferimento all'interscambio di dati tra sistemi distribuiti con modalità *Client/Server*, facendo quindi una distinzione tra il sistema su cui risiedono le informazioni (dati e/o applicazioni) e quello che richiede l'accesso a dati e servizi. Una tale ipotesi è sicuramente accettabile: il coinvolgimento di un sistema informativo in una attività di cooperazione consiste proprio nell'esportazione di propri servizi applicativi e/o nell'accesso ai servizi forniti da altri sistemi. Inoltre, nella successiva Sezione 5, vedremo che le soluzioni architetturali possono essere effettivamente ricondotte ad architetture basate su comunicazioni di tipo Client/Server.

La progettazione di una applicazione Client/Server [6] prevede la definizione delle interfacce dei Server, che definiscono le funzionalità che ogni Server rende disponibile ai suoi Client, con i relativi parametri di ingresso e dati di uscita. Una volta individuato questo insieme di funzionalità occorre definire un *formato* ed un *protocollo* per realizzare la comunicazione tra Client e Server e permettere la cooperazione; il formato stabilisce un linguaggio comune, ovvero un insieme di possibili messaggi, ed il protocollo l'insieme delle regole per l'uso di questi messaggi.

Nelle sezioni seguenti vengono esaminate le principali tecniche per la definizione della interfaccia del Server, ovvero del tipo di messaggi che fluiscono tra Client e Server, e cioè:

- trasferimento di dati in linea;
- trasferimento di dati fuori linea (con replicazione);
- scambio di messaggi in linea;
- scambio di messaggi fuori linea (code di messaggi);
- accesso a dati in linea (mediante paradigma ipertestuale);
- simulazione di maschere.

Queste tecniche corrispondono agli strumenti di base disponibili per realizzare le funzionalità principali delle architetture per la coooperazione. In realtà, questi strumenti sono stati originariamente sviluppati non tanto per permettere la cooperazione tra sistemi distribuiti ed eterogenei, quanto per permettere la *migrazione* da sistemi di basi di dati pre-relazionali (sistemi reticolari e gerarchici, o realizzati tramite file system) a sistemi più moderni (relazionali oppure object-oriented) su architetture aperte [4].

Sono disponibili vari prodotti commerciali, relativi alle singole funzionalità, che possono essere utilizzati per realizzare architetture a cooperazione lasca, come quelle relative alle applicazioni che abbiamo definito sistemi informativi locali con dati esterni. Va inoltre osservato che stanno cominciando ad apparire sul mercato dei prodotti di middleware, chiamati *Database Gateway Server*, che mettono a disposizione, in modo abbastanza integrato, funzionalità che corrispondono a varie di queste tecniche.

4.1 Trasferimento di dati in linea

Un dato di un sistema componente può venire acceduto in modo diretto da un Client mediante un comando opportuno di un linguaggio di manipolazione dei dati. Se il Client è omogeneo con il sistema componente, l'accesso può avvenire inviando un messaggio in cui, ad esempio, sia specificato un comando di un linguaggio comune (questo è tipicamente il caso dei sistemi di gestione di basi di dati relazionali, con il linguaggio SQL). Nel caso più generale, in cui non è possibile ipotizzare l'omogeneità del Client con il Server, deve essere utilizzato un *database gateway*, ovvero un prodotto software che realizza la trasformazione di comandi di un linguaggio di manipolazione (usato dal Client) in comandi del linguaggio riconosciuti dal sistema componente da accedere, e permette la presentazione dei risultati ottenuti nel linguaggio utilizzato dal Client. Nel caso tipico in cui il Client specifica richieste nel linguaggio SQL, i risultati vengono presentati in formato relazionale; un tale prodotto è chiamato *SQL gateway*. Solitamente, il linguaggio riconosciuto dal sistema componente può essere costituito da una sequenza di richieste di accesso a file, comandi SQL (spesso un diverso dialetto SQL), oppure da chiamate a sistemi non relazionali. Questo strato software può risiedere sul Client, sul Server, oppure su altro sistema, proponendo architetture alternative.

Va osservato che i database gateway sono necessari anche tra sistemi diversi di gestione di basi di dati relazionali (omogenei sul modello, ma eterogenei sul sistema); esistono prodotti che mascherano le differenza tra i tipi di dato ammessi ed il dialetto SQL utilizzato in sistemi relazionali differenti. L'uso di un SQL gateway permette dunque di vedere una base di dati, che potrebbe essere non relazionale, come se fosse relazionale. In questo caso possiamo dire che l'interfaccia verso il sistema remoto è essenzialmente lo schema dei dati che il Client può accedere. Gli SQL gateway realizzano quindi un elevato livello di trasparenza per l'accesso a dati strutturati.

Alcuni sistemi supportano come ambiente primario di sviluppo un linguaggio di manipolazione diverso da SQL, ad esempio *ODBC* (*Open Database Connectivity*, una sorta di standard di fatto adottato da alcuni produttori). In questo caso, entrambi gli estremi della comunicazione devono supportare il formato e protocollo stabilito come ODBC.

I costi legati ai database gateway sono legati sia ad un decadimento delle prestazioni, che alla difficoltà di rappresentare vincoli di integrità di modelli logici diversi (di conseguenza, alla difficoltà a preservare l'integrità dei dati). Le soluzioni basate su database gateway possono essere proficuamente utilizzate in presenza di basi di dati remote non relazionali, accedute poco frequentemente, se il tempo di risposta non è critico.

4.2 Trasferimento di Dati Fuori Linea

L'accesso ai dati di un sistema componente da parte di un Client può avvenire, come abbiamo detto, utilizzando copie, cioè *trasferendo* l'insieme dei dati di interesse (fuori linea rispetto all'applicazione) ed accedendo ad essi in modo locale. Il trasferimento dei dati può essere accompagnato da una fase di *estrazione e trasformazione*, in cui i soli dati di interesse vengono selezionati, ripuliti, ed eventualmente convertiti in un formato diverso da quello iniziale; il trasferimento può avvenire come trasferimento di file sulla rete. L'estrazione può avvenire mediante applicazioni *ad hoc*, oppure tramite l'uso di strumenti opportuni disponibili sul mercato e con funzionalità sofisticate. L'estrazione può avvenire a monte del trasferimento, oppure a valle del trasferimento (utilizzando ad esempio un database gateway). Il trasferimento può avvenire periodicamente oppure su richiesta; il trasferimento può essere globale (tutti i dati di interesse vengono trasferiti) oppure incrementale (solo i dati cambiati rispetto all'ultimo trasferimento vengono trasferiti). A seconda dei casi, parleremo di strumenti di estrazione, replicazione o propagazione.

4.3 Scambio di Messaggi in Linea

Lo scambio di messaggi in linea (sincrono) include diverse modalità di scambio di informazioni tra programmi distribuiti, tra cui l'uso di *remote procedure call* (*RPC*), *TP monitor*, e gli emergenti sistemi ad oggetti distribuiti (*object request broker*). L'interazione tra applicazioni avviene tramite un insieme finito e prefissato di funzionalità. Il Client richiede l'esecuzione al Server di una di queste

funzionalità, inviandogli un messaggio contenente la specifica della funzionalità richiesta ed i relativi parametri; il Server esegue la funzione e restituisce al Client i dati richiesti in un messaggio secondo un formato prefissato.

A differenza delle soluzioni basate sullo scambio dei dati, nel caso di scambio di messaggi l'interfaccia del Server è costituita da un insieme di funzionalità e non da uno schema; in altre parole, l'incapsulamento del Server mostra uno schema di funzioni e non uno schema di dati.

Va osservato che molti sistemi di gestione di basi di dati prevedono l'uso di un meccanismo simile all'RPC per eseguire transazioni (chiamate *stored procedures*) che contengono comandi SQL; in questo caso di parla di *SQL statico* (in quanto, essendo note a priori le operazioni da eseguire, è possibile compilare una volta per tutte il comando), utilizzando la denominazione *SQL dinamico* nel caso dei database gateway.

Meccanismi di RPC sono implementati anche nei monitor transazionali distribuiti. Un *Monitor Transazionale Distribuito* è un prodotto software che realizza il supporto alla esecuzione di transazioni su più sistemi, garantendone alcune importanti proprietà di consistenza e atomicità. Le funzionalità offerte da un tale sistema, oltre al meccanismo di chiamata di procedure remote, sono relative al controllo della concorrenza tra risorse e processi distribuiti, alla gestione della permanenza delle operazioni eseguite, delle operazioni di commit distribuito (implementando, ad esempio, meccanismi di two-phase commit), al recovery. Queste funzionalità sono offerte agli sviluppatori di applicazioni tramite opportune interfacce applicative.

La soluzione basata sullo scambio di messaggi in linea è quella che più si avvicina allo spirito delle architetture Client/Server sviluppate negli ultimi anni. È una soluzione aperta al passato, in quanto permette di riutilizzare funzioni e applicazioni esistenti a costi relativamente bassi; grazie alla sua flessibilità, è allo stesso tempo anche una soluzione aperta al presente ad al futuro.

4.4 Scambio di Messaggi Fuori Linea

Un inconveniente dello scambio di messaggi in linea tra applicazioni è costituito dall'impossibilità di usufruire di servizi distribuiti qualora ci sia un malfunzionamento della rete di telecomunicazioni o un sovraccarico del Server. Tuttavia, in molti casi è possibile che la cooperazione tra applicazioni avvenga, utilizzando una modalità asincrona anziché sincrona, ovvero inviando dei messaggi di richiesta di servizi per ottenere il corrispondente messaggio di risposta in un momento successivo. Questa tecnica corrisponde a strumenti di gestione di code di messaggi: posta elettronica, nel caso più semplice; si parla in generale di *Message Oriented Middleware* (*MOM*).

4.5 Accesso a Dati in Linea

Una forma particolare di accesso a dati e documenti in linea è quella costituita dal paradigma ipertestuale utilizzato da *World Wide Web* (*WWW*) su *Internet*. Si tratta in questo caso di una forma particolare di scambio di messaggi in linea,

in cui il protocollo utilizzato è *HTTP* (*HyperText Transfer Protocol*) ed il linguaggio utilizzato dai messaggi è *HTML* (*HyperText Markup Language*). Questa soluzione è immediatamente attuabile ogni volta in cui i dati risiedono esplicitamente sotto forma di file HTML. È anche possibile utilizzare degli strumenti che accettano richieste di accesso ad una base di dati, convertendo i risultati in formato HTML, che sono dunque visualizzabili dal Client mediante un qualsiasi browser HTML.

I sistemi WWW presentano alcune limitazioni, legate ad una bassa sicurezza delle informazioni che viaggiano sulla rete ed al fatto che il protocollo HTTP è senza memoria. Tuttavia, questa tecnica può essere utilizzata per permettere un accesso semplice a dati ed informazioni di carattere pubblico, anche in riferimento alla grandissima diffusione di Internet. Inoltre, alcuni meccanismi di sicurezza permettono di utilizzare la stessa tecnologia anche per la diffusione di informazioni (eventualmente private) internamente ad una singola organizzazione (*Intranet*).

4.6 Simulazione di Maschere

Una forma particolare di scambio di messaggi in linea è costituito dalla *simulazione di maschere* (*screen scraper*), realizzata da prodotti software che permettono di analizzare il flusso dei dati di I/O di un terminale e di riformattarlo come insieme di parametri da utilizzare in una diversa applicazione. Le applicazioni esistenti possono essere riutilizzate senza dover essere modificate; in particolare, questa soluzione è preferibile ad altre quando non si vuole (o non è possibile, se non è più disponibile il codice sorgente) modificare l'applicazione esistente. Viceversa, questa tecnica non è consigliata per lo sviluppo di nuove applicazioni. Va osservato inoltre che la simulazione di maschere è difficile da mantenere nel caso in cui siano possibili aggiornamenti delle schermate nelle applicazioni da riutilizzare.

5 Architetture di Basi di Dati Cooperanti

In questa sezione vengono presentate alcune soluzioni architetturali per la cooperazione tra basi di dati. Il modello architetturale di riferimento è basato sul modello *Client/Server a tre livelli*, che è una estensione del modello a due livello ottenuta aggiungendo uno strato di Server intermedi destinati a supportare i servizi applicativi distribuiti. L'introduzione di un tale livello intermedio, corrispondente al livello di presentazione globale del sistema distribuito su cui va basata l'applicazione cooperante, permette di mascherare ulteriormente l'eterogeneità tra i sistemi componenti (in particolare, se ospitato su una piattaforma aperta), e di realizzare attività di supporto alla cooperazione (ad esempio, su di un tale Server possono essere concentrate le eventuali funzionalità di sicurezza e di controllo degli accessi).

Abbiamo visto nella Sezione 3 come la cooperazione fra basi di dati possa essere lasca (nel caso dei sistemi informativi locali con dati esterni) o stretta (nei

casi dei sistemi multidatabase e delle collezioni di dati replicati). In ciascuno di questi casi è necessaria la realizzazione di un vero e proprio sistema distribuito, che sarà più semplice per la cooperazione lasca e più complesso per la cooperazione stretta. Descriviamo nelle sezioni che seguono le soluzioni architetturali realizzabili con gli strumenti attualmente disponibili sul mercato. In particolare, facciamo riferimento a quattro soluzioni, ognuna basata su una specifica tipologia di strumenti di middleware: tre corrispondono sostanzialmente alle tre categorie discusse nella Sezione 3, mentre l'altra è intermedia.

5.1 Architetture Basate su Database Gateway Server

Un multidatabase, come abbiamo visto, è una collezione di basi di dati preesistenti, autonome ed eterogenee e presenta il livello globale del sistema che stabilisce la cooperazione fra le basi di dati componenti. Ogni sistema componente si presenta verso il gestore del sistema globale tramite le interfacce dei servizi che rende disponibili; ogni sistema inoltre provvede ad una implementazione di tali servizi.

Una soluzione architetturale per la realizzazione di un multidatabase è basata sui *Database Gateway Server*, strumenti che presentano in forma abbastanza integrata servizi corrispondenti a varie tecniche descritte nella Sezione 4: accesso in linea mediante SQL, replicazione, scambio di messaggi con modalità RPC (e probabilmente MOM nel prossimo futuro), gestione delle transazioni distribuite. In questo modo è possibile raggiungere in qualche misura gli obiettivi legati alle basi di dati federate, con gestione della trasparenza (anche a livello di schema globale), complessità delle operazioni (sia pure ancora non a livelli altissimi, soprattutto per quanto riguarda l'efficienza delle interrogazioni complesse) e grado di attualità variabile a seconda delle necessità. Vale la pena notare come, peraltro, questi strumenti siano in continua evoluzione e siano molto costosi.

5.2 Architetture Basate su Strumenti per la Replicazione

Per la realizzazione di collezioni di dati replicati, sono disponibili sul mercato strumenti specifici per la gestione delle operazioni di estrazione e replicazione. In particolare, ne sono stati recentemente proposti alcuni utilizzabili con riferimento a sorgenti eterogenee e distribuite. È possibile in questo modo realizzare architetture per il *Data Warehousing*, con la collezione di dati replicata gestita sul al livello del sistema globale per mezzo di un sistema di basi di dati relazionali in ambiente aperto e supportata da appositi strumenti per l'amministrazione (che gestiscono lo schema globale), l'accesso e l'analisi dei dati (per realizzare le finalità di supporto alle decisioni tipiche di queste applicazioni).

5.3 Architetture Basate su Monitor Transazionali Distribuiti

Una differente tipologia di integrazione può essere definita nel caso in cui i sistemi componenti, pur cooperando in modo stretto, non rendono visibili gli

schemi delle proprie basi di dati, ma solo specifiche funzionalità per la manipolazione dei dati stessi, realizzando quindi una forma di incapsulamento. Il sistema distribuito definisce servizi complessi ottenuti tramite composizione delle funzionalità elementari. In questo caso, il middleware utilizzato è di tipo monitor transazionale distribuito, ed il valore aggiunto dai servizi complessi consiste essenzialmente nella correttezza ed atomicità delle transazioni definite, quindi con un forte grado di coordinamento. Osservando che gli ambienti transazionali in uso sono relativamente pochi, è possibile ipotizzare situazioni in cui i sistemi componenti utilizzino un monitor transazionale distribuito comune o compatibile. In questi casi, i modelli di integrazione possono quindi essere relativamente semplici, anche grazie alla evoluzione che gli ambienti transazionali proprietari stanno effettuando verso sistemi aperti.

Questi ambienti si propongono dunque come strumenti di middleware completi per realizzare la cooperazione tra applicazioni basata sull'integrazione ed il coordinamento di funzionalità eseguite dai singoli sistemi componenti, possibilmente in contesti eterogenei. Va inoltre osservato che gli attuali monitor transazionali distribuiti implementano funzionalità per lo scambio di messaggi fuori linea, offrendosi dunque anche come strumenti di MOM. Infine, se questi strumenti sono adeguati per la realizzazione di architetture in cui la cooperazione è centrata sui dati, va sottolineato che i monitor transazionali distribuiti possono essere utilizzati proficuamente anche per applicazioni la cui cooperazione è centrata sui processi.

5.4 Architetture Basate su Strumenti Elementari

Per la realizzazione di sistemi informativi locali con accesso a dati esterni sono disponibili, oltre agli strumenti generali sopra descritti (e alle loro versioni semplificate), anche diversi strumenti di livello relativamente basso oppure dedicati a singole funzionalità. Non è possibile indicare soluzioni generali, perché si tratta spesso di risolvere un problema specifico di cooperazione fra pochi sistemi (di solito due alla volta) e possono quindi essere utilizzati strumenti che forniscono una soluzione ad hoc. Si possono utilizzare SQL gateway specifici (tra sistemi omogenei o tra coppie di sistemi) oppure tecniche di scambio di messaggi di basso livello (quali quelle basate sui socket, su API proprietarie o sulla simulazione di maschere). Vale la pena notare che soluzioni di questo genere sono difficili da manutenere e degenerano al crescere della complessità della cooperazione, e vanno quindi limitate a casi semplici.

6 Considerazioni Metodologiche

L'obiettivo di questa sezione è quello di ribadire alcuni dei concetti emersi in questo documento, cercando di individuare delle linee guida per la progettazione di applicazioni cooperanti, dove la cooperazione sia centrata sulla condivisione dei dati. Riteniamo estremamente importante, infatti, una riflessione sulle varie tipologie di interoperabilità e cooperazione possibili in presenza di una rete di

telecomunicazione. La scelta di una delle tipologie piuttosto che le altre dovrà essere fatta con attenzione ai costi e ai benefici della cooperazione. L'analisi dei benefici deve partire dall'individuazione dell'utenza di un sistema cooperante e prevedere un esame attento delle effettive necessità di utilizzo da parte di questa utenza. Corrispondentemente, devono essere ben presenti i costi stimati della cooperazione. In tale analisi riteniamo che possano essere utili i tre criteri proposti nella Sezione 2, e cioè il livello di trasparenza, la complessità delle transazioni ed il livello di attualità dei dati. Per ognuno dei criteri sono possibili scelte con costi e benefici diversi.

Vale la pena osservare che la presenza di una rete di telecomunicazione deve essere considerata una significativa opportunità per le organizzazioni che vogliano cooperare o far cooperare i loro dipartimenti, in quanto fornisce connettività e servizi senza richiedere investimenti onerosi in termini di ristrutturazione dei sistemi informativi esistenti. È possibile infatti pensare ad architetture di rete, basate sulla presenza di sistemi aperti, nel pieno rispetto dell'autonomia dei sistemi informativi preesistenti, che consentono l'interoperabilità e la cooperazione anche tra sistemi proprietari ed eterogenei. Sistemi aperti possono essere utilizzati per interfacciare un sistema esistente con la rete, incapsulando il sistema e mascherando le eterogeneità, tramite l'impiego di strumenti diversi, quali quelli descritti nella Sezione 4. Il progetto di una tale interfaccia deve ovviamente essere inquadrato in precisi accordi di cooperazione tra le varie organizzazioni.

La possibilità di far cooperare sistemi appartenenti ad organizzazioni diverse richiede una riflessione, da parte di ciascuna organizzazione, sul cosa esportare sulla rete, in particolare su quali dei propri dati condividere. La decisione di esportare parte dei propri dati deve rispondere a considerazioni legate alla sicurezza e all'opportunità, e deve condurre ad una serie di scelte di carattere progettuale. Come discusso in precedenza, infatti, esistono diverse modalità per condividere i dati. La maggiore distinzione è quella tra condivisione dei dati veri e propri e condivisione di funzionalità. Nel primo caso, il sistema aperto che si interfaccia alla rete potrà utilizzare un insieme di SQL gateway che consentono l'accesso alle proprie basi di dati; nell'altro caso sarà necessario creare le opportune procedure e consentire l'accesso remoto ad esse.

Sul sistema distribuito saranno poi realizzate applicazioni cooperanti, e, nell'ambito di ciascuna iniziativa, dovrà essere individuata la soluzione architetturale, basata su una delle architetture sopra descritte, coerentemente con le caratteristiche della cooperazione.

Le iniziative di cooperazione che possono rappresentare anche occasioni per ripensare a scelte progettuali ed organizzative fatte nel passato. Infatti, malgrado la possibilità di evitare la migrazione verso un qualche standard, è innegabile che tecnologie antiquate ed irrazionalità nell'organizzazione dei dati possano e debbano comunque essere messe in discussione. Dal punto di vista dei modelli di dati, è opportuno individuare come obiettivo di medio-lungo periodo la migrazione da modelli antiquati e sistemi cosiddetti *legacy* (e cioè file tradizionali e sistemi gerarchici e reticolari) almeno verso il modello relazionale. Contemporaneamente, il progetto di iniziative di cooperazione e la presenza di un ambiente distribuito

per la condivisione dei dati saranno occasioni per eliminare duplicazioni e inconsistenze nella distribuzione dei dati, in particolare tra i diversi dipartimenti di una unica organizzazione.

Bibliografia

1. Autorità per l'Informatica nella Pubblica Amministrazione. La Rete Unitaria della Pubblica Amministrazione: Studio di fattibilità, 1996. Reperibile (con gli allegati) anche via rete: `http://www.aipa.it`.
2. C. Batini, M. Lenzerini, and S.B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, December 1986.
3. Y. Breitbart, H. Garcia-Molina, and A. Silberschatz. Transaction management in multidatabase systems. In W. Kim, editor, *Modern Database Systems*, pages 573–591. ACM Press, 1995.
4. M.L. Brodie and M. Stonebraker. *Migrating Legacy Systems: Gateways, Interfaces & the Incremental Approach*. Morgan Kaufmann, San Mateo, California, 1995.
5. H. Garcia-Molina and M. Hsu. Distributed databases. In W. Kim, editor, *Modern Database Systems*, pages 477–493. ACM Press, 1995.
6. J. Gray and A. Reuter, editors. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, San Mateo, California, 1993.
7. D. Hsiao. Federated databases and systems: Part I – a tutorial on their data sharing. *The VLDB Journal*, 1(1):127–179, 1992.
8. D. Hsiao. Federated databases and systems: Part I – a tutorial on their resource consolidation. *The VLDB Journal*, 1(2):285–310, 1992.
9. W. Kim, editor. *Modern Database Systems: the Object Model, Interoperability, and Beyond*. ACM Press and Addison Wesley, 1995.
10. W. Kim, I. Choi, S. Gala, and M. Scheevel. On resolving schematic heterogeneity in multidatabase systems. In W. Kim, editor, *Modern Database Systems*, pages 521–550. ACM Press, 1995.
11. W. Meng and C. Yu. Query processing in multidatabase systems. In W. Kim, editor, *Modern Database Systems*, pages 551–572. ACM Press, 1995.
12. A.P. Sheth and J.A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, September 1990.
13. R.L. Stout. EDA/SQL. In W. Kim, editor, *Modern Database Systems*, pages 649–663. ACM Press, 1995.