C. Bolchini, G. Orsi, E. Quintarelli, F. A. Schreiber, L.Tanca

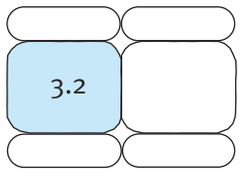
PROGETTAZIONE DEI DATI CON L'USO DEL CONTESTO

M O N D O D I G I T A L E • n . 3 - s e t t e m b r e 2 0 0 8
pp.11 -26

PROGETTAZIONE DEI DATI CON L'USO DEL CONTESTO

Cristiana Bolchini
Giorgio Orsi
Elisa Quintarelli
Fabio A. Schreiber
Letizia Tanca

Gli utenti dei sistemi informativi e di Internet si trovano oggi ad affrontare la necessità di contenere il "rumore informativo" in modo da non essere subissati e confusi da una grande quantità di dati, spesso poco legati al contesto di fruizione. L'attività di adattare i dati alle necessità delle applicazioni può essere assimilata al problema di confezionare un vestito su misura: da qui il nome Data Tailoring. In questo articolo si mostra come le "forbici" usate per questa operazione possano essere rappresentate dal contesto, inteso come un insieme di variabili che caratterizzano l'ambiente specifico nel quale l'applicazione si trova ad operare.



1. INTRODUZIONE

Fin dall'inizio degli anni '80, i sistemi informativi di molte organizzazioni si sono estesi ed evoluti per rispondere alle esigenze di un mercato che richiede sempre più la presenza di un'*impresa allargata* comprendente diversi partner. Anche tecnologicamente, sono state introdotte funzionalità di *Knowledge Management* per permettere una facile condivisione dell'informazione tra i diversi membri dell'organizzazione; l'enorme fonte di informazioni costituita da Internet ha richiesto l'introduzione di strumenti quali XML, ontologie e servizi web, che devono essere utilizzati e gestiti insieme a quelli di altre fonti di dati più tradizionali. Questo aumento di disponibilità, se non opportunamente controllato, porta ad un sovraccarico di dati riducendo drammaticamente i benefici di un ricco sistema informativo. Tuttavia, distinguere l'informazione utile dal rumore, cioè da tutta quella non rilevante allo specifico problema applicativo, non è una questione di facile soluzione; una stessa "unità di informazione" (un oggetto, un'entità) può essere considerata in modo diverso – anche dallo stesso utente – in posti e situazioni diverse; in una parola, in un *contesto* diverso [7, 8, 9].

Si noti che lo scenario appena delineato suggerisce anche un problema ortogonale, altrettanto pressante, che è quello dell'integrazione di dati provenienti da sorgenti eterogenee; esso tuttavia costituisce un argomento di ricerca di per sé assai vasto [3, 10] che – seppur strettamente correlato a quanto esporremo nel seguito – non verrà affrontato in questo lavoro.

Un'ulteriore spinta verso la focalizzazione delle informazioni deriva dall'introduzione di dispositivi mobili; in questo caso la risposta a una stessa interrogazione può cambiare a seconda della localizzazione dell'utente e della situazione ambientale istantanea. Al fine di evitare interruzioni del lavoro dovute alla connessione intermittente alla rete e la ridotta capacità di memorizzazione, tipica dei dispositivi palmari e dei telefoni cellulari, si ha la necessità di mantenere una copia locale della quantità minima di dati in grado di soddisfare le esigenze legate al contesto corrente.

La nozione di contesto, all'inizio emersa nell'ambito di discipline diverse quali la psicologia e la filosofia, sta assumendo oggi notevole importanza anche nelle attività collegate all'informatica. Nel senso comune, il contesto è visto come un insieme di variabili che possono interessare un attore e influenzarne le azioni: un cambiamento del contesto può infatti causare un cambiamento nella rappresentazione mentale che l'attore si crea della realtà, anche se quest'ultima non è cambiata. La parola stessa, derivata dal latino *cum* (con o assieme) e *texere* (tessere), descrive il contesto non solo come un profilo, ma anche come un processo *attivo relativo a come gli uomini utilizzano la loro esperienza all'interno del proprio ambiente per fornirle un significato*.

1.1. Data tailoring basato sul contesto

Scopo di questo lavoro è mostrare come si possa usare il contesto per definire delle viste (*data views*) sui dati di grandi sistemi informativi, ritagliando solamente informazioni rilevanti per una determinata applicazione, in modo da ottenerne un sottoinsieme personalizzato. Da un punto di vista concettuale, le necessità dell'utente di un sistema informativo possono dipendere da due diversi aspetti: il *dominio applicativo*, che rappresenta la realtà in esame, e l'*ambiente di lavoro*, in altre parole, il contesto.
Nel mondo dei sistemi informativi e delle metodologie di progettazione, sono ben noti i classici meccanismi di astrazione attraverso i quali, dall'analisi della realtà di interesse, si genera una rappresentazione concettuale della stessa, e quindi del *dominio applicativo*. L'astrazione che rappresenta il "punto di vista" (*viewpoint abstraction* [11]), e quindi, in buona sostanza, il *contesto di fruizione* dei dati, ha invece ricevuto scarsa attenzione dalla comunità dei ricercatori. In particolare, la strategia di progettazione mista, basata sulla produzione delle viste d'utente e sulla loro successiva integrazione, vede la produzione delle viste come prioritaria all'esistenza della base di dati. Oggi però occorre tener conto della necessità di individuare, e opportunamente configurare, viste su masse di dati già esistenti, delle quali sia eventualmente noto lo schema. In tal caso, l'individuazione del punto di vista non è necessariamente prioritaria alla costruzione della base di dati, e costituisce un'attività a sé, parallela o addirittura successiva alla progettazione della stessa.
Coerentemente, anche l'attuale generazione di sistemi commerciali per la gestione di basi di dati (DBMS), basati su SQL, non permette una definizione ed una gestione esplicita delle informazioni di contesto, ma offre solamente il meccanismo delle *viste di utente* (view) per determinare la porzione di dati gestibile da ciascun utente. Nel seguito, mostreremo come la nostra metodologia sfrutti queste caratteristiche dei DBMS per introdurre la dipendenza dal contesto senza richiedere alcun cambiamento al processo di progettazione delle sorgenti dei dati e neppure richiedere funzionalità particolari al DBMS. Nel paragrafo 4 viene mostrata l'architettura di un sistema dipendente dal contesto che realizza l'approccio metodologico proposto.
Dato uno scenario applicativo, la nostra *metodologia guidata dal contesto* aiuta il progettista nella fase di identificazione dei vari contesti possibili e nella fase successiva di definizione delle viste sull'intero insieme di dati, che risultano rilevanti per ciascun contesto. Tre sono i componenti fondamentali della metodologia:

1. un ***modello di contesto*** che cattura tutti gli aspetti – le cosiddette *dimensioni* – che permettono una rappresentazione implicita dei possibili contesti applicativi;
2. una ***strategia*** che identifica, per ciascuna dimensione in modo indipendente da tutte le altre, una porzione rilevante dell'intero schema dei dati - le cosiddette *viste parziali*;
3. un ***insieme di operatori*** che combinano tra loro le viste parziali in modo da derivare la vista definitiva, associata a ciascun contesto.

1.2. La nozione di contesto nella letteratura

Recentemente, sono stati proposti modelli, a vari livelli di sofisticazione, che permettono di realizzare applicazioni sensibili al contesto (*context-aware applications*). Possiamo elencare i diversi significati che, in essi, sono stati attribuiti alla parola "contesto":

- *Orientato alla presentazione*: il contesto è percepito come la possibilità del sistema di adattare la presentazione dei contenuti a diversi canali informativi o a diversi dispositivi. Questi modelli sono spesso rigidi in quanto

progettati per specifiche esigenze applicative e si basano su un insieme di variabili di presentazione perfettamente noto [16].

❑ *Orientato alla località*: la caratteristica di questi modelli consiste nel permettere di gestire coordinate spaziali e temporali con grande precisione e flessibilità [18].

❑ *Centrato sull'utente*: questi modelli mettono a fuoco l'attività dell'utente: diventa importante la storia e l'evoluzione del suo contesto, e vengono fornite alcune funzioni di ragionamento contestuale; se disponibili, si possono usare funzioni di apprendimento automatico per inferire l'attività dell'utente dalle rilevazioni di sensori [14].

❑ *Basato sul concetto di comunità*: è un approccio interessante che considera il contesto come un insieme di variabili condivise da un gruppo di utenti (*peer*), per i quali costituisce un presupposto alla cooperazione. Diversamente dagli approcci precedenti, si raggiunge una definizione di contesto comune in modo cooperativo [13].

L'uso del contesto per i sistemi informativi e per le basi di dati è stato introdotto in [4, 5, 11, 12, 15, 17, 19], ma è lungi dall'essere stato esplorato approfonditamente. Solo alcune di queste proposte sono orientate al *data tailoring*, mentre per altre lo scopo è di presentare i dati in una prospettiva diversa da contesto a contesto. Senza pretesa di completezza, ci dilungheremo un momento anche su questi altri modelli.

In accordo con l'approccio presentato in [11], Motschnig-Pitrik e Mylpoulos [12] vedono un "information base" come decomposto in sottoinsiemi non disgiunti, che rappresentano i contesti. Un contesto è caratterizzato dal suo contenuto, e d'altra parte i contesti stessi sono oggetti del sistema informativo che, insieme con gli oggetti elementari, costituiscono l'insieme delle *information units*. In questo modo, i contesti vengono considerati alla stregua di qualunque altro elemento del sistema informativo, e il contenimento può essere ricorsivo. Mediante questa astrazione vengono supportati:

- la denominazione e la rappresentazione, basate sul contesto, di tutte le entità concettuali del sistema informativo;
- l'esecuzione di transazioni relativizzata al contesto;
- operazioni di costruzione, manipolazione di contesti e propagazione dei cambiamenti tra contesti contigui.

Per definire un contesto, occorre quindi definire: il suo contenuto, i nomi locali degli oggetti del contesto (o lessico), le regole di autorizzazione relative a diversi utenti e diverse transazioni e, infine, il modo in cui si propagano i cambiamenti tra un contesto e l'altro.

Una tipica applicazione di questo modello, suggerita dagli autori, è lo scenario di cooperazione, dove soggetti diversi utilizzano contesti sia personali sia condivisi per aggiornare in modo cooperativo dei manufatti, come articoli o moduli software. La proposta non si basa sull'ipotesi di un singolo schema globale del database; piuttosto, vi sono schemi diversi, non necessariamente disgiunti, per i diversi contesti. La relativizzazione al contesto di tutte le operazioni risolve il problema dell'inconsistenza, che è molto rilevante nel mondo dei sistemi informativi cooperativi e integrati.

L'approccio di [19] è molto simile al precedente, ma ha il vantaggio di aver introdotto la non-unicità dei nomi per gli oggetti. In questo modo, il modello è in grado di gestire omonimie, sinonimie e oggetti anonimi, risolvendo i conflitti di nome, tipici degli ambienti integrati. Anche in questo lavoro si prevedono primitive di manipolazione dei contesti: in particolare, sono molto ben formalizzate la creazione di contesto e le operazioni insiemistiche sui contesti, tenendo proprio conto della necessità di gestire i conflitti di nome.

In [15] gli autori propongono una visione contestuale di una base di dati relazionale. Diversamente dal nostro approccio, la loro soluzione non suddivide la base di dati in porzioni, bensì valorizza diversamente alcuni attributi, ottenendo varianti (*facets*) diverse per la stessa tupla, in contesti differenti. Un attributo potrebbe addirittura non esistere in qualche contesto. Esempi di applicazione di questo modello sono: un prodotto le cui specifiche cambiano in base al Paese in cui viene commercializzato, oppure una pagina Web che deve essere visualizzata diversamente a seconda del dispositivo. Anche qui un contesto (o *mondo*) è specificato assegnando valori a un opportuno insieme di dimensioni. Una tabella contestuale è quindi una sequenza di tabelle relazionali normali, una per

ogni mondo possibile, alla stregua del modello delle basi di dati temporali, dove però agli istanti di tempo si sostituiscono i vari contesti di validità delle tabelle.
Simile, ma non uguale, è l'approccio di [17]: qui il modello dei dati è analogo a quello dei sistemi OLAP, ma le dimensioni del cubo non sono quelle scelte per l'analisi dei dati, bensì le dimensioni da valorizzare nella definizione del contesto. Gli autori propongono un modello di contesto da associare alle preferenze dell'utente, il quale, all'interno di ogni contesto, può privilegiare determinati valori dei dati. I parametri (dimensioni) del contesto possono essere statici o dinamici: per esempio, il tempo atmosferico, in una certa applicazione, può essere un parametro statico perché il suo valore viene esplicitamente aggiornato, mentre la locazione geografica viene computata in maniera automatica, e quindi varia senza necessità di aggiornamento esplicito. Analogamente al caso OLAP, in questo modello le dimensioni vengono considerate in gerarchia, per esempio la dimensione spaziale potrebbe prevedere i valori ***città, regione, stato***.
Le dimensioni del contesto proposte in questo modello sono abbastanza simili alle nostre; poiché però qui lo scopo è gestire le preferenze dell'utente, la ricerca è orientata a definire in maniera operazionale il rapporto tra contesti e preferenze, e non a ritagliare i dati su misura. Per una rassegna più articolata su questi argomenti si veda [2].

1.3. Lo scenario applicativo

Tra i diversi scenari e casi di studio presi in considerazione per lo sviluppo del modello e della metodologia, ci si riferisce in questo lavoro al mondo delle agenzie immobiliari, ed in particolare ad una rete di franchising in cui sono presenti diverse agenzie sparse sul territorio, che fanno riferimento ad un'unica rete commerciale. Ciascuna agenzia dispone di un insieme di agenti, ognuno dei quali segue i clienti nel vendere o affittare il loro immobile (sia residenziale sia commerciale) o nel trovare l'immobile che fa per loro. Il sistema informativo deve quindi gestire tutte le informazioni relative a agenti, clienti ed immobili, oltre alle agende degli agenti per quanto concerne le visite che essi organizzano per far visionare gli immobili.
L'accesso a queste informazioni avviene dai terminali dell'ufficio oppure dai dispositivi palmari e smartphone, quando gli agenti effettuano le visite presso gli immobili volendo avere a disposizione non solo la propria agenda degli appuntamenti, ma anche tutte e sole le informazioni relative agli immobili oggetto della visita.
Come abbiamo già detto, la mobilità introduce un nuovo aspetto nella fruizione dei dati: se da una parte, il recente progresso tecnologico ha reso sempre più capaci i dispositivi mobili personali (palmari, smartphone ecc.), dotandoli di buone capacità di elaborazione e memorizzazione, e offrendo diverse alternative per quanto riguarda la connettività, dall'altra, i dispositivi sono ancora di dimensioni limitate (display piccolo e interazione a menù quantomeno lunga), e il costo dell'accesso ai dati è ancora piuttosto alto. Questi aspetti spingono alla realizzazione di applicazioni e servizi in grado di fornire esattamente le informazioni cui l'utente è interessato, per semplificarne l'accesso.
Tornando al nostro esempio, oltre che tramite dispositivo mobile, gli stessi dati relativi agli immobili sono anche visibili dal sito web della rete commerciale, per offrire un altro punto di accesso a potenziali *clienti* (che quindi possono semplicemente consultare i dati pubblici degli immobili) e ai clienti delle agenzie (che quindi avranno accesso ad un insieme più ricco di informazioni).
In questo scenario applicativo, dunque, le figure dei possibili utenti (o ruoli) individuate sono le seguenti: gli ***agenti*** delle agenzie immobiliari, ciascuno con il proprio portafoglio di immobili e di clienti, i ***clienti*** che hanno dato il mandato all'agenzia per vendere o affittare l'immobile di loro proprietà, insieme a quelli che desiderano acquistare o affittare un immobile, i ***visitatori*** del sito web, che si distinguono dai precedenti ***clienti*** in quanto caratterizzati da uno stato di anonimato nei confronti dell'agenzia, il ***manager*** dell'agenzia ed infine l'amministratore delegato (***CEO***) della rete commerciale, interessato principalmente alla produttività dei propri agenti ed al portafoglio degli immobili.
È possibile distinguere macro *aree di interesse* dei dati cui possono voler accedere i vari tipi di utenti: dati relativi agli ***immobili*** (con

Agenzia(***agenziaID***, indirizzo, centroOperativo, nome, orario)
Dipendente(***dipID***, nome, cognome, dataNascita, tel, email, indirizzo, ***managerID***, agenziaID)
Immobile(***immobileID***, clienteID, descrizione, metri2, mappaID)
Multimedia(***contenutoID***, tipo, ampiezza, altezza, dato)
ImmobileMultimedia(***immobileID***, ***contenutoID***)
Cliente(***clienteID***, nome, cognome, indirizzo, CAP, tel, email, categoria)
RichiestaCliente(***ricID***, ***acquirenteID***, budget, AcquistoAffitto)
Agenda(***data***, ***ora***, ***agenteID***, immobileID, clienteID)
Visita(***immobileID***, ***data***, ***agenteID***, ***clienteID***, durata)
Affitto(***immobileID***, ***clienteID***, dataIn, agenteID, rateo, condizioni, durata)
Vendita(***immobileID***, ***proprietarioID***, ***acquirenteID***, ***data***, agenteID, prezzoIniziale, prezzoFinale, condizioni, percAgenzia)

TABELLA 1
Schema logico della base di dati del caso considerato

la possibilità di raffinare la selezione in base, per esempio, alla categoria commerciale o residenziale, al fatto che siano in vendita oppure in affitto), dati relativi ai ***clienti*** (di rilievo per gli agenti e i manager delle agenzie), informazioni relative ai propri ***agenti*** o più in generale alle proprie ***agenzie***.
La selezione dei dati di interesse è ulteriormente influenzata – in determinati frangenti, quale per esempio la consultazione delle informazioni degli immobili – dalla posizione: tale localizzazione può essere sia determinata in modo automatico mediante strumenti in grado di rilevare le coordinate geografiche, sia esplicitamente dichiarata dall'utente. L'ambito applicativo in esame, infatti, mostra l'esigenza di poter disporre di dati filtrati rispetto alla propria posizione. Per esempio per un agente che si trova a dover reperire i dati degli immobili da lui gestiti entro un certo raggio rispetto alla posizione in cui si trova, durante i sopralluoghi. D'altra parte, è possibile voler selezionare i dati in base ad una posizione in cui ci si verrà a trovare in seguito per poter scaricare in anticipo le informazioni sul proprio dispositivo. Quindi la dimensione spaziale, come quella temporale, contribuisce alla nozione di contesto in due modi diversi: mediante una determinazione automatica di posizione (o istante di tempo), oppure mediante una richiesta esplicita da parte dell'utente. Infine, come nella maggior parte delle situazioni in cui la mobilità gioca un ruolo importante, il tipo di dispositivo (e quindi l'interfaccia, il canale ecc.) mediante il quale si accede alle informazioni ne influenza la ricchezza e il livello di dettaglio, prima ancora della modalità di presentazione che dovrà essere adottato. Dunque, anche nell'esempio proposto, l'interfaccia utilizzata determina un tipo di filtraggio dei dati. Parte dello schema logico della base di dati di questa applicazione è riportato nella tabella 1.
Nel seguito del lavoro verranno presentati il nostro modello di contesto e la sua utilizzazione per il ritaglio dei dati, insieme con alcuni esempi di applicazioni.

2. IL MODELLO DI CONTESTO

Il modello di contesto, chiamato *Context Dimension Tree* (CDT) [4] (riquadro), è stato definito con lo scopo di filtrare i dati relativi ad una particolare applicazione, in base alle reali necessità dell'utente. Esso è una rappresentazione ad albero dell'insieme delle possibili prospettive e dei relativi valori, uti-

La struttura di un Context Dimension Tree
In un Context Dimension Tree vi sono tre tipi di nodi (si veda la Figura 1):
- la *radice*: rappresenta l'insieme dei contesti descritti dall'intero Context Dimension Tree;
- i *nodi dimensione*: sono graficamente rappresentati come nodi neri e descrivono le dimensioni (per esempio, ***interest-topic***), o le sotto-dimensioni (per esempio, ***prezzo***, ***categoria*** ecc.), utili per il ritaglio dei dati;
- i *nodi concetto*: sono rappresentati come nodi bianchi e identificano i valori della dimensione alla quale sono collegati (per esempio, i valori della dimensione ***role*** sono i concetti *CEO*, *manager*, e *agente*).

Le dimensioni figlie del nodo radice rappresentano le cosiddette *dimensioni di contesto*, ossia le principali prospettive utili per la descrizione dei possibili contesti dell'utente nello scenario applicativo scelto. Ogni concetto (nodo bianco) dell'albero può essere ulteriormente analizzato con un livello di dettaglio crescente, aggiungendo al suo sottoalbero altre sotto-dimensioni di analisi che raffinano il concetto stesso.

li alla descrizione dei diversi aspetti del contesto nel quale un utente si colloca; un esempio di CDT, relativo al nostro esempio applicativo, è riportato nella parte superiore della figura 1.
Si noti che nella rappresentazione gerarchica dei contesti così definita, ogni generazione è formata da nodi dello stesso tipo (dimensioni o concetti) e ogni nodo specifica in modo più dettagliato gli aspetti descritti dai suoi antenati. Si noti che alcune sotto-dimensioni non hanno figli che sono nodi concetto (per esempio, *prezzo*). In tal caso, il numero di diversi valori per tali dimensioni sarebbe elevato: per non indicare esplicitamente l'insieme di tutti i possibili valori, si introduce un nodo parametro che verrà istanziato al momento dell'esecuzione con il reale valore assunto dalla dimensione nel contesto corrente dell'utente. Nel caso del nodo *prezzo*, quindi il parametro potrà assumere come valore uno dei prezzi degli immobili. I parametri possono essere aggiunti anche a nodi concetto (nodi bianchi) quando si desidera che l'istanza del relativo concetto venga filtrata al momento dell'esecuzione in base ad un opportuno valore: per esempio, il parametro ***$aid***, identificatore del ruolo agente.
Nel nostro modello, *gli elementi di contesto* rappresentano l'attualizzazione delle dimensioni di contesto e corrispondono a nodi di un Context Dimension Tree che possono essere: nodi concetto senza parametro (per esempio, *clienti*), nodi concetto con parametro (per esempio, agente($aid)) o nodi dimensione con parametro (per esempio, *n_locali*(***$lid***)).
Un contesto è formalizzato come una congiunzione di proposizioni sempre relative agli elementi di contesto che lo descrivono, ognuno dei quali è espresso mediante la proposizione ***dim_name = value*** dove ***dim_name*** rappresenta il nome di una (sotto-)dimensione e ***value*** il valore per quella dimensione. Tale valore quindi può essere un nodo concetto oppure un nodo concetto filtrato dal valore del suo parametro, o ancora il valore di un parametro di un nodo sotto-dimensione foglia. Un esempio di possibile contesto estratto dal Context Di-

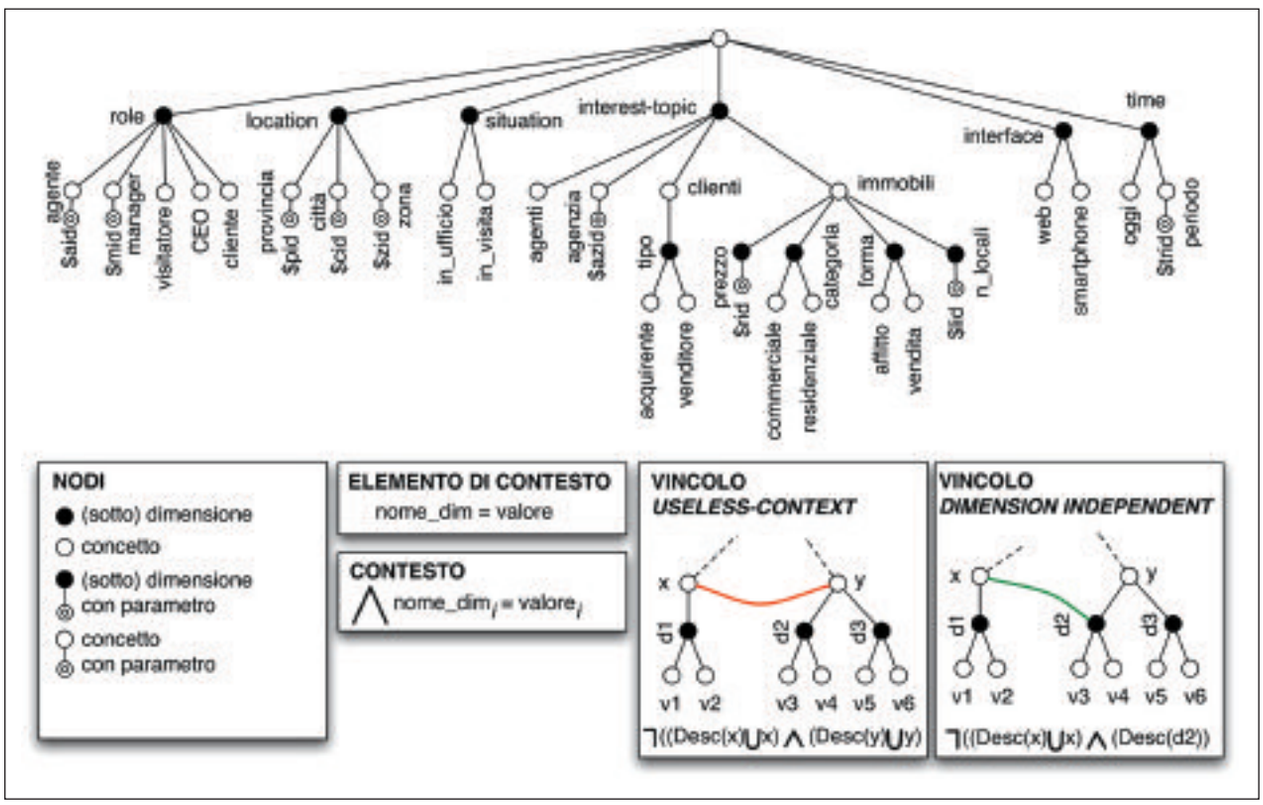


FIGURA 1
Il modello Context Dimension Tree

mension Tree di figura 1 è il seguente:

$$(\text{role} = \text{manager}(\$mid)) \wedge (\text{interest-topic} = \text{agenzia}(\$azid)) \wedge (\text{situation} = \text{in\_ufficio}) \quad (1)$$

Tale contesto è relativo ad un manager – identificato al momento dell'esecuzione istanziando il suo identificatore ***$mid*** – che vuole consultare, durante il suo lavoro d'ufficio, le informazioni relative alle agenzie delle quali è responsabile.
La profondità del Context Dimension Tree per un'applicazione dipende dal livello di dettaglio utilizzato per definire i diversi aspetti che contribuiscono alla selezione delle informazioni disponibili; alcune dimensioni di contesto si prestano maggiormente ad essere rappresentate con un sotto-albero a vari livelli di dettaglio, come l'***interest-topic***, mentre altre hanno una rappresentazione più piatta (si veda per esempio, la dimensione ***situation***). È compito del progettista scegliere il livello di dettaglio adeguato nella specifica del Context Dimension Tree, e quindi dei contesti, in quanto più specifici sono i contesti più selettiva sarà l'interrogazione per selezionare le informazioni di interesse per ogni contesto. Alcune dimensioni di contesto da noi proposte sono utilizzabili nella maggior parte delle applicazioni, ma non tutte sono sempre necessarie e altre potrebbero essere aggiunte a quelle elencate in questo lavoro a seconda delle reali necessità. La tabella 2 riporta le

Dimensioni di contesto	Significato	Esempio Agenzia Immobiliare
Role	Gli utenti che utilizzano l'applicazione di interesse.	"*CEO*", "*manager*", "*agente*".
Interest Topic	Gli interessi dei potenziali utenti dell'applicazione.	"*agenzia*", ossia le informazioni su una particolare agenzia utili per il CEO; "*agenti*", informazioni sugli agenti utili per il CEO o per il manager di quell'agenzia; "clienti", informazioni su venditori e acquirenti che possono essere utilizzate dall'agente o dal manager; "*immobili*", tutte le informazioni riguardanti gli immobili da vendere o affittare. Quest'ultimo argomento può essere ulteriormente decomposto rispetto altri criteri: per esempio, *immobili commerciali/residenziali* o proprietà in *vendita/affitto*.
Situation	Fasi in cui l'applicazione verrà usata.	L'utente consulterà le informazioni di interesse durante il normale lavoro d'ufficio, "*in ufficio*", oppure nella situazione "*in visita*", quando, ad esempio, un agente è in visita ad immobili con potenziali clienti.
Time	Informazioni temporali. Il tempo può essere espresso in modo relativo o assoluto con livelli di granularità differenti.	Abbiamo scelto una visione relativa del tempo, permettendo due scelte di analisi possibili: il giorno corrente, o un intervallo di tempo variabile centrato sull'istante di tempocorrente.
Space	Informazioni spaziali, normalmente riferite alla posizione dove l'utente è situato. Lo spazio puòessere relativo o assoluto e anche la sua granularità a può variare.	"*qui*" o "*questa città*" sono valori spaziali relativi, mentre "*Roma*" è un valore assoluto.
Interface	Informazioni sul canale, o tipo di presentazione, per trasmettere le informazioni.	In alcuni casi le informazioni devono essere consultate da utenti umani, mentre in altre situazioni le informazioni vanno gestite da dispositivi elettronici che richiedono solo poche informazioni descritte in formati specifici.

TABELLA 2
Dimensioni del Context Dimension Tree: descrizioni e uso

principali dimensioni che abbiamo individuato, il loro significato intuitivo ed alcuni esempi relativi al nostro scenario applicativo.
Inoltre, gli aspetti di privatezza dei dati sono importanti in moltissime applicazioni: a tale scopo al Context Dimension Tree può essere aggiunta la dimensione *ownership*, che non è direttamente legata al ritaglio dei dati, ma può essere usata per definire i diritti di accesso, per esempio con istruzioni di *grant* e *revoke* definite sulle viste finali, quando le sorgenti informative sono relazionali. I diritti associati ai valori della dimensione *ownership* servono quindi per "oscurare" le parti sensibili delle viste finali, ottenute combinando le viste parziali associate agli altri elementi di contesto. Poiché la funzione di tale dimensione – pur potendo contribuire alla riduzione dei dati – è diversa dalle altre, non verrà approfondita in questo lavoro.
A partire dal Context Dimension Tree di un'applicazione, i contesti vengono generati prendendo un insieme (anche vuoto) di elementi per ogni dimensione, in modo tale che i vincoli del modello vengano rispettati. L'ipotesi che un elemento di contesto descriva concetti che comprendono quelli specificati nei suoi discendenti, e che i nodi concetto fratelli siano mutuamente esclusivi, porta a definire vincoli di parentela che impongono che, scelto un concetto, non vengano selezionati nel contesto altri nodi concetto presenti nel sottoalbero la cui radice è il genitore del nodo scelto, e gli ascendenti di tale nodo.
Inoltre, poiché non tutte le combinazioni di valori delle dimensioni sono significative per l'applicazione scelta, è opportuno eliminare i contesti non significativi con l'introduzione di ulteriori *vincoli*. I principali vincoli considerati nell'analisi (indicati mediante linee colorate nella Figura 1) sono:
- *useless contexts*: vengono introdotti per eliminare le combinazioni di valori di due o più dimensioni che non hanno senso per l'applicazione. Per esempio, un vincolo di questo tipo può impedire di generare i contesti nei quali il ruolo CEO compaia con la situazione "*in visita*";
- *dimension independent*: vengono introdotti quando i valori di una dimensione non influiscono sulla vista finale prodotta per i contesti che presentano un altro (o altri) valore per un'altra dimensione. Nel nostro scenario, la dimensione ***location*** non influisce sui dati associati ai contesti relativi al ruolo CEO, in quanto questo ruolo richiede dati generici contabili che non contengono informazioni spaziali.

Nella figura 1 viene indicata la formalizzazione generica dei due tipi di vincoli.

Viste su una base di dati

Una delle motivazioni principali per la nascita dei sistemi di gestione di basi di dati è stata l'esigenza di disaccoppiare il modello generale dei dati dell'azienda dalla visione che degli stessi dati hanno i singoli programmi applicativi. Il modello ANSI-SPARC definisce pertanto tre distinti livelli di rappresentazione: lo *schema logico* al livello delle strutture dei dati, lo *schema fisico*, che definisce le strutture dei dati e le relative operazioni in termini delle strutture e delle primitive del file system, e lo *schema esterno* o *vista* sullo schema logico, al livello dell'applicazione.
Nella sua pratica realizzazione, quale si ritrova nei moderni sistemi per la gestione di basi di dati, una vista è costituita da una particolare forma di interrogazione, che estrae dallo schema logico tutte e sole le informazioni che interessano un determinato programma (o insieme di programmi) applicativo, nella forma a questo più consona. Poiché nel corso della vita di una base di dati lo schema logico e i suoi contenuti sono soggetti a modifiche, si preferisce mantenere le viste in forma *virtuale*: ciò significa che l'interrogazione che realizza la vista viene eseguita ogni volta che viene richiesta da un'interrogazione dell'utente applicativo, consentendo così una grande flessibilità nell'evoluzione degli schemi al prezzo di un piccolo allungamento nel tempo di risposta. Tuttavia, qualora lo schema logico sia stabile e sia bassa la volatilità dei dati o quando considerazioni di efficienza o di affidabilità siano prevalenti, è possibile eseguire in anticipo una volta per tutte la vista e memorizzare permanentemente i risultati dell'interrogazione in tabelle, sulle quali poi agiscono le interrogazioni degli utenti applicativi; in tal caso si parla di *vista materializzata*.

3. LA METODOLOGIA PER IL RITAGLIO DEI DATI DIPENDENTE DAL CONTESTO

Dopo aver progettato il Context Dimension Tree per l'applicazione di interesse, e aver ricavato da tale rappresentazione grafica tutti i contesti significativi, il progettista dovrà associare ad ognuno di essi la porzione di dati rilevanti.
In tale fase, partendo da una rappresentazione integrata dei dati contenuti nelle sorgenti informative disponibili, per esempio uno schema relazionale globale o un'ontologia, il progettista dovrà selezionare sullo schema globale, per ogni contesto, lo schema dei dati ritenuti rilevanti, espresso mediante un'interrogazione (vista) sullo schema globale. A partire dalla vista relativa ad ogni contesto, i dati verranno successivamente estratti dalle sorgenti informative in modo automatico per il contesto corrente dell'utente (riquadro).
Poiché il numero di contesti per un'applicazione reale è solitamente elevato, nonostante il relati-

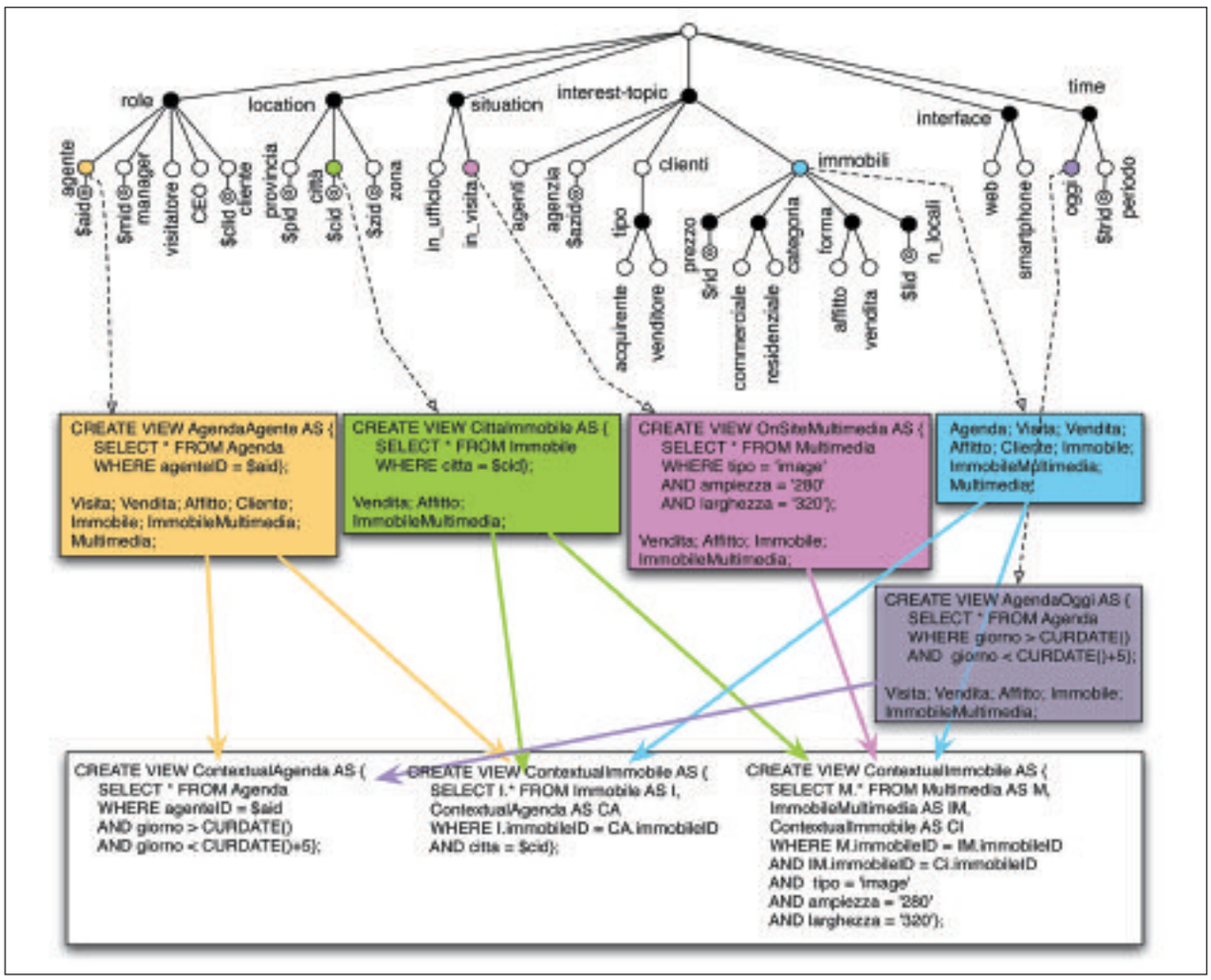


FIGURA 2

Viste associate ai singoli nodi dell'albero e derivazione per composizione della vista associata ad un contesto

vo Context Dimension Tree sia abbastanza limitato e alcune combinazioni non significative vengano eliminate con l'introduzione dei vincoli[1], l'associazione tra i contesti e le informazioni a disposizione richiede un intervento oneroso da parte del progettista. Per rendere semi-automatica tale procedura, la metodologia che viene descritta in questa sezione prevede di associare una *vista parziale* ad ogni elemento di contesto. Mediante operatori appositi, si può quindi, in modo automatico, ottenere un'approssimazione della vista relativa ad uno specifico contesto C, come combinazione delle viste parziali associate agli elementi di contesto presenti in C. Un ulteriore intervento del progettista potrà eventualmente perfezionare la vista così ottenuta. Si consideri, per esempio, il contesto relativo ad un agente interessato alle informazioni sugli immobili, mentre è in visita con possibili clienti in una specifica città:

$$(\text{role} = \text{agente}(\$aid)) \wedge (\text{interest-topic} = \text{immobili}) \wedge (\text{situation} = \text{in visita}) \wedge (\text{time} = \text{oggi}(\text{getdate}())) \wedge (\text{location} = \text{città}(\$cid)) \quad (2)$$

Il parametro ***$aid***, che identifica lo specifico agente, verrà istanziato al momento dell'esecuzione, come pure la città.
Nella figura 2 vengono mostrate, sotto il Context Dimension Tree dell'applicazione considerata, le viste SQL associate agli elementi

[1] Nel nostro esempio applicativo i contesti sono circa 1400; di questi, circa 400 vengono eliminati con l'introduzione di vincoli.

Contesto	Informazioni rilevanti
role=agente($aid) ^ interest_topic=immobili ^ situation=in_visita ^ time=oggi ^ location=città($cid)	Informazioni di uso quotidiano per un agente: informazioni sugli immobili da visitare (inclusi i dettagli degli immobili e le informazioni sulle aree circostanti), informazioni multimediali compatibili con il dispositivo mobile dell'agente, i suoi appuntamenti, le informazioni sui proprietari degli immobili e sui clienti.
role=CEO ^ interest_topic=vendita ^ time=periodo($trid)) ^ location=città($cid)	Dati aggregati relativi alle vendite: per esempio, importi medi e somme delle vendite, commissioni medie pagate sulle vendite, aggregate per intervalli di tempo e zona geografica.
role=manager($mid) ^ interest_topic=agenti ^ agende situation=in_ ufficio ^ time=periodo($trid)	Informazioni per assegnare gli immobili agli agenti: degli agenti, dati anagrafici degli agenti, immobili da assegnare e richieste dei clienti.
role=agente($aid) ^ interest_topic=clienti ^ situation=in_ ufficio ^ location=città($cid)	Informazioni per porre in contatto acquirenti e venditori: informazioni sui clienti, richieste non evase e dettagli sulle offerte.

TABELLA 3
Esempi di contesto e relative informazioni

presenti nel contesto (2); le frecce tratteggiate e colorate associano ogni elemento di contesto con la rispettiva vista parziale. L'agente sarà interessato alle informazioni relative a:

i. immobili presenti nell'area in cui lui effettua la visita con i clienti;

ii. informazioni multimediali relative agli immobili e compatibili con il dispositivo mobile in sua dotazione;

iii. gli appuntamenti fissati in un intervallo di 5 giorni, necessari per poter organizzare eventuali ulteriori impegni lavorativi.

Altri esempi di contesti per lo scenario applicativo scelto sono illustrati nella tabella 3, con una descrizione informale delle informazioni rilevanti per ogni contesto.

Le viste parziali vengono associate ad ogni elemento *n* del Context Dimension Tree, a partire dalla radice dell'albero, secondo una *politica* denominata *massimale*, che prevede di selezionare *tutte le informazioni ritenute di interesse per n*. Per esempio, i dati potenzialmente rilevanti per l'elemento ***role = agente($aid)***, come mostrato nella figura 2, saranno tutti gli appuntamenti dell'agente considerato, le informazioni sia multimediali che testuali relative agli immobili e alle relative caratteristiche, i dati anagrafici dei clienti dell'agenzia, i dettagli relativi alle visite e alle operazioni di acquisto o affitto di immobili. Inoltre, poiché il Context Dimension Tree nasce per fornire una rappresentazione gerarchica dei valori e delle dimensioni, procedendo dalla radice verso le foglie dell'albero i concetti vengono raffinati, e quindi il progettista dovrà selezionare le viste parziali in modo che la vista associata ad un nodo *n sia contenuta nella vista dei suoi antenati*.

Per esempio, la vista per l'elemento ***interest-topic = affitto*** deve essere contenuta in quella di ***interest-topic = immobili***; in particolare, dovrà contenere le sole informazioni e operazioni relative ad immobili in affitto.

Nella parte bassa della figura 2 vengono mostrate le viste SQL finali relative al contesto (2): le frecce colorate e i colori nelle viste stesse indicano i contributi delle viste parziali nella formazione di quelle finali[2].

L'operatore insiemistico usato per combinare le viste parziali associate agli elementi di contesto è la *doppia intersezione*, che estende l'operazione di ***INTERSECT*** di SQL a insiemi di relazioni. Intuitivamente, la doppia intersezione tra due insiemi di relazioni A e B, corrisponde, per ogni coppia di relazioni *RA* e *RB* (in A e B rispettivamente) che hanno in comune una parte di schema, all'applicazione dell'INTERSECT al massimo sottoschema comune. Le re-

2 In [3] gli operatori per combinare le viste parziali sono estensioni di operatori dell'algebra relazionale, banalmente traducibili nei corrispondenti operatori SQL. In questo lavoro mostreremo solo questi ultimi.

lazioni che non hanno sotto-schemi comuni non contribuiscono al risultato della doppia intersezione. Nell'esempio riportato nella figura 1 la relazione Agenda compare nelle viste parziali di tutti gli elementi presenti nel contesto (2). Nella vista SQL associata a tale contesto avremo pertanto la relazione Agenda, che verrà però filtrata per l'applicazione dell'intersezione dell'algebra relazionale, dalle condizioni imposte per gli elementi ***role = agente($aid)*** e ***time = "oggi"***. Nelle viste parziali di questi due elementi le informazioni sugli appuntamenti selezionano rispettivamente le tuple relative ai soli impegni dell'agente ***$aid***, in un intervallo di cinque giorni.
Si noti che la definizione della vista finale associata ad un contesto dipende:
i. dalla precisione e dal livello di dettaglio con cui sono state definite dal progettista le viste parziali e;
ii. dagli effetti dell'applicazione dell'operatore di doppia intersezione alle viste parziali.
Pertanto il progettista può ritenere opportuno modificare manualmente tale vista, ricavata automaticamente, aggiungendo o escludendo manualmente informazioni, ma tale intervento sarà sicuramente meno oneroso della definizione manuale di tutte le viste definitive associate ad ogni contesto significativo.

4. IL SISTEMA CONTEXT-ADDICT

Nel paragrafo precedenti abbiamo visto come la metodologia proposta permetta di realizzare sistemi per l'accesso "intelligente" ai dati, che sfruttino le informazioni proprie del dominio applicativo e della situazione corrente dell'utente, per ridurre opportunamente i dati prima di interrogarli o memorizzarli su un dispositivo client. Il sistema che abbiamo progettato si chiama Context-ADDICT, e la sua architettura è costituita da (Figura 3): le sorgenti di informazione, il *Context-ADDICT Server* (CAS), il server di Integrazione (IS) e i dispositivi client. Infatti, nel caso più generale si prevede che il sistema informativo a disposizione possa essere composto da diverse sorgenti informative, che vanno anche opportunamente integrate.

4.1. Le sorgenti informative

Le sorgenti informative comprendono tutti i sistemi che contengono le informazioni effettivamente disponibili nei diversi contesti in cui l'applicazione può operare. È possibile classificare queste sorgenti in base a criteri quali:
❑ *Il modello dei dati* - definisce se le informazioni sono rappresentate in modo strutturato (relazionale), semi-strutturato (XML o RDF) o non-strutturato (testo o HTML).
❑ *Stabilità* - rappresenta la volatilità della sorgente in termini di disponibilità durante le interrogazioni. La stabilità può essere relativa anche ai dati ovvero alla frequenza con cui i dati della sorgente vengono aggiornati.
❑ *Collaboratività* - è una misura di quanto una sorgente rende i propri dati accessibili a sistemi di terze parti. Ricordiamo infatti che la semantica dei dati è fornita dallo schema. Quanto più una sorgente rende disponibili i propri schemi in un formato direttamente elaborabile, tanto più sarà considerata collaborativa. Nel nostro contesto, la collaboratività è intesa come la disponibilità da parte della sorgente a fornire le meta-informazioni necessarie per rendere i dati accessibili in senso più generale.
❑ *Fiducia* (*trust*) - definisce quanto le risposte alle interrogazioni devono essere considerate affidabili. Tale misura comprende sia la possibilità che i dati siano errati o non aggiornati, sia il fatto che alcuni sistemi, gestiscano dati di sintesi o approssimati.
❑ *Tecnologia* - la tecnologia influenza le capacità di calcolo, i tempi di risposta, l'espressività delle interrogazioni e le modalità di co-

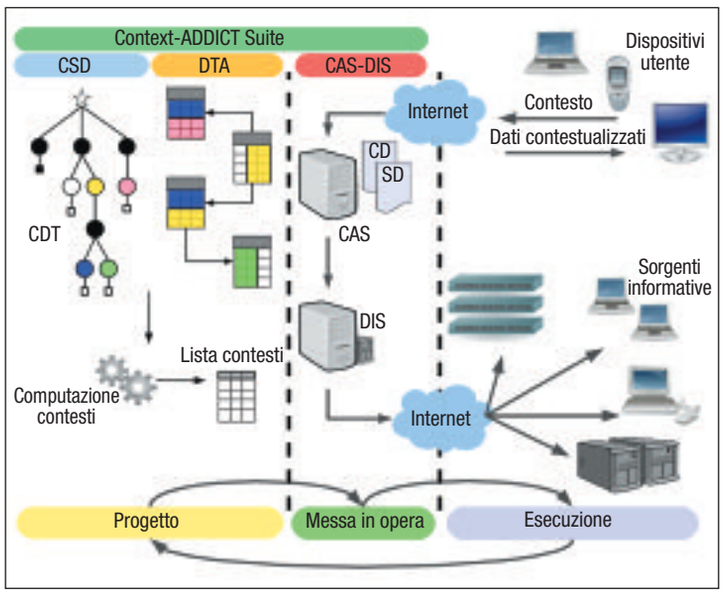


FIGURA 3
Architettura di un sistema Context-ADDICT

municazione. Tali caratteristiche influenzano la generazione del piano di esecuzione delle interrogazioni e devono dunque essere considerate.
Indipendentemente dalla collaboratività delle sorgenti, per poter generare un ritaglio coerente, occorre analizzarne gli schemi e produrre una rappresentazione degli stessi usando un modello dei dati comune, che permetta il confronto e la mappatura, con quelli delle altre sorgenti. Il modello scelto deve essere sufficientemente espressivo per rappresentare gli altri modelli dei dati e soprattutto i vincoli che devono sussistere fra le sorgenti ed il dominio modellato.

4.2. Il Context-ADDICT server

Il *Context-ADDICT Server* (CAS) rappresenta il nucleo dell'intera architettura e realizza i processi di *ritaglio dei dati* (context-aware data tailoring) e di esecuzione delle *interrogazioni contestualizzate* (context-aware query answering). Si tratta di un servizio che puó essere situato su un opportuno server applicativo come Apache Tomcat [20], JBoss [22] o IBM Websphere [21] e fornisce il supporto alla distribuzione delle interrogazioni mediante un dizionario dei contesti (*Context Dictionary* - CD). Ogni elemento del CD è una tripla (*c*, *s*, *q*) dove *c* è una specifica di contesto, *s* una sorgente informativa e *q* l'insieme di interrogazioni nel linguaggio nativo della sorgente che permette il ritaglio del contenuto (*formal query*). Il dizionario contiene anche le credenziali di accesso alla sorgente e una sua descrizione mediante meta-informazioni (*Source Descriptors* - SD). Le informazioni contenute nei SDs sono utilizzate in fase di formulazione del piano delle interrogazioni sia in un'ottica di ottimizzazione che di manipolazione delle stesse[3]. A fronte di una richiesta di informazioni di contesto da parte di un dispositivo client, il CAS:

1. individua l'elemento nel dizionario corrispondente alla specifica di contesto ricevuta;
2. per ogni sorgente coinvolta, recupera la definizione dell'interrogazione e ne istanzia le variabili con i valori contenuti nella richiesta (*actual query*);
3. recupera le credenziali di accesso alla sorgente e vi si connette;
4. nel caso la connessione sia avvenuta con successo, esegue l'interrogazione concreta e restituisce i valori al dispositivo interrogante.

Un esempio di questo processo è riportato nella figura 4.
Il CAS si avvale anche di un sistema di archiviazione di supporto (*support storage*) nel

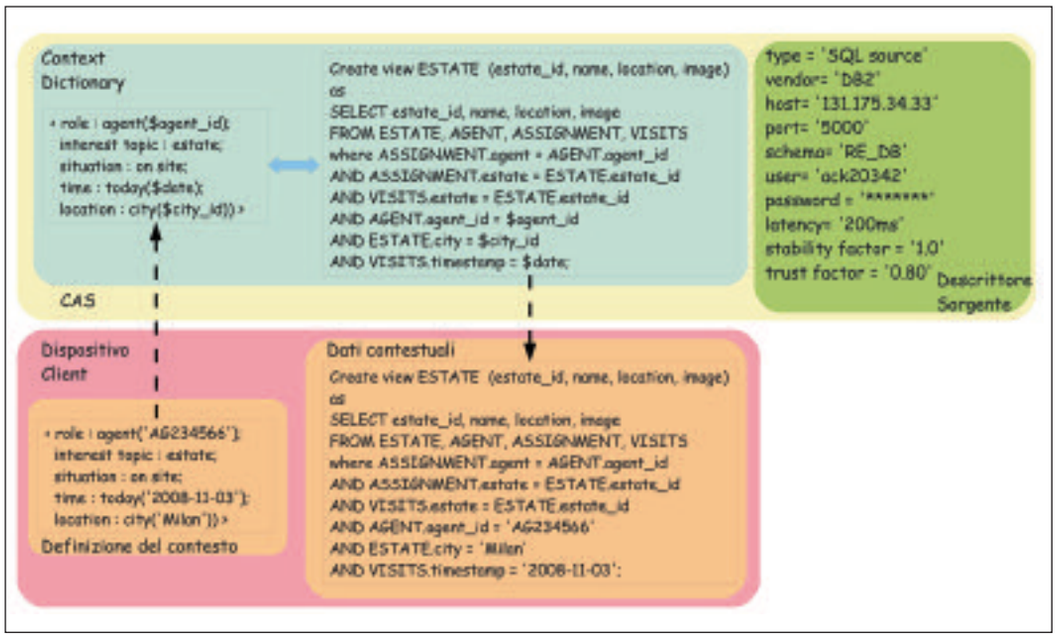


FIGURA 4
Esempio di interrogazione contestuale

[3] Per manipolazione si intende l'arricchimento di un'interrogazione con informazioni di contesto non espressamente indicate nell'interrogazione originale.

caso in cui sia più conveniente memorizzare localmente alcune informazioni piuttosto che effettuare una richiesta on-line ad una sorgente. Consideriamo per esempio il caso degli orari dei musei in un dominio applicativo di natura turistica; se la sorgente non è collaborativa, risulta necessario estrarre tali informazioni dalla pagina web del museo o dei punti di informazione turistici usando dei wrapper per pagine web. L'analisi delle pagine web mediante algoritmi di elaborazione del linguaggio naturale è un processo oneroso e non è conveniente, dal nostro punto di vista, effettuare tali operazioni ad ogni richiesta. Inoltre, informazioni cosiffatte sono in genere molto stabili (per esempio variano al cambiare della stagione o dell'anno) e possono essere soggette ad un'archiviazione locale (*local caching*) efficiente. All'atto dell'esecuzione di un'interrogazione che richieda queste informazioni, il CAS farà dunque riferimento alla memoria locale invece che alla sorgente originale. In quest'ottica, l'indice di stabilità della sorgente è uno strumento utile per definire con quale frequenza occorra aggiornare i dati memorizzati localmente.

4.3. Server di integrazione

Data la natura fortemente eterogenea delle sorgenti informative, non è sempre possibile determinare a priori se le informazioni contenute in esse appartengano ad insiemi disgiunti, ovvero vi è la possibilità che alcune sorgenti si riferiscano alla stessa porzione di dominio e quindi contengano informazioni comuni. La descrizione del processo di riconciliazione e integrazione di tali informazioni va oltre lo scopo di questo lavoro; il progetto Context-ADDICT prevede comunque un opportuno sistema di integrazione dei dati (*Data Integration System* - DIS) residente su un server di integrazione[4].

4.4. Dispositivi client

I client delle applicazioni dipendenti dal contesto che vengono realizzate con la metodologia Context-ADDICT possono risiedere su un qualsiasi dispositivo fisso o mobile. Ad ogni dispositivo saranno associate una o più specifiche di contesto, le cui variabili saranno opportunamente istanziate con i valori attuali al momento della richiesta al CAS. La gestione delle informazioni di contesto risulta quasi del tutto trasparente, fatta esclusione per le ovvie interazioni iniziali in fase di configurazione delle applicazioni client. Per i dispositivi mobili, le informazioni riguardanti la localizzazione spazio-temporale possono essere recuperate rispettivamente dal clock del sistema e da un dispositivo GPS[5]. Dimensioni come il ruolo e gli argomenti di interesse possono essere recuperate dal profilo utente o dall'analisi dei log delle interrogazioni effettuate, mentre i valori delle dimensioni più tecnologiche come l'interfaccia di comunicazione e il formato dei dati sono automaticamente determinate dalle tecnologie disponibili nel dispositivo (OBEX, WiFi, UMTS).

4.5. Il software di modellazione CADD

CADD (*Context-aware Data Designer* [6]) è il software di supporto alla modellazione di sistemi dipendenti dal contesto sviluppato al Politecnico di Milano sempre nell'ambito del progetto Context-ADDICT. CADD (Figura 5) supporta il progettista nella definizione del CDT (*Context Space Designer* - CSD), nell'associazione di ogni nodo del CDT alle porzioni di interesse degli schemi delle sorgenti informative (*Data Tailoring Assistant* - DTA) e nella fase di messa in opera del sistema (CAS Deployer - CASD).

4.5.1. Progetto del modello di contesto: CSD

Il progettista del CDT è guidato nella sua costruzione da un'interfaccia grafica intuitiva che ne permette la creazione in maniera visuale (Figura 6).
Come descritto in precedenza, è possibile esprimere dei vincoli fra i valori delle dimensioni del modello di contesto per eliminare a priori le combinazioni prive di significato (per esempio, un cliente che visita una sede aziendale, non sarà interessato alle informazioni sulla logistica in ingresso). I vincoli so-

[4] Un esempio di sistema commerciale per l'integrazione dei dati è IBM Websphere Information Integrator.

[5] Nokia prevede che entro il 2010 tutti i dispositivi cellulari e smartphones di fascia medio alta saranno dotati di dispositivo GPS integrato.

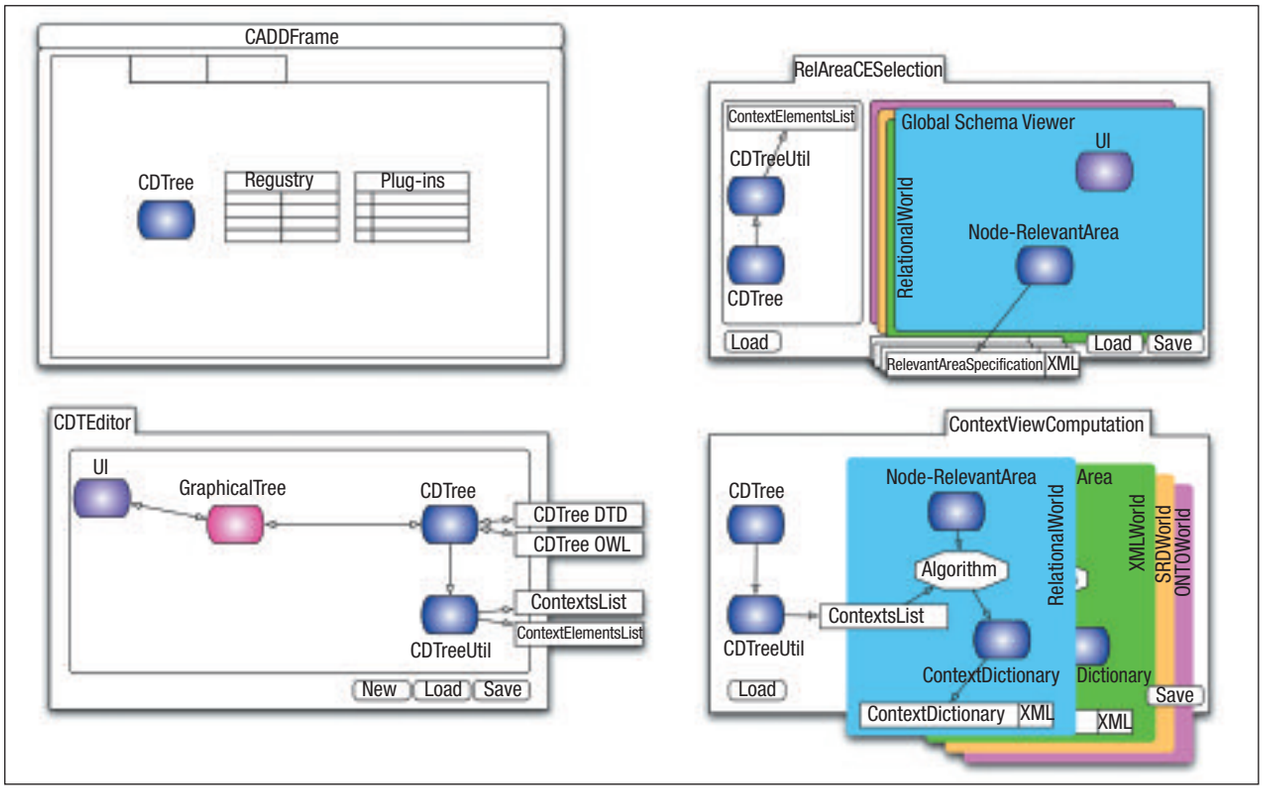


FIGURA 5

Architettura dello strumento CADD

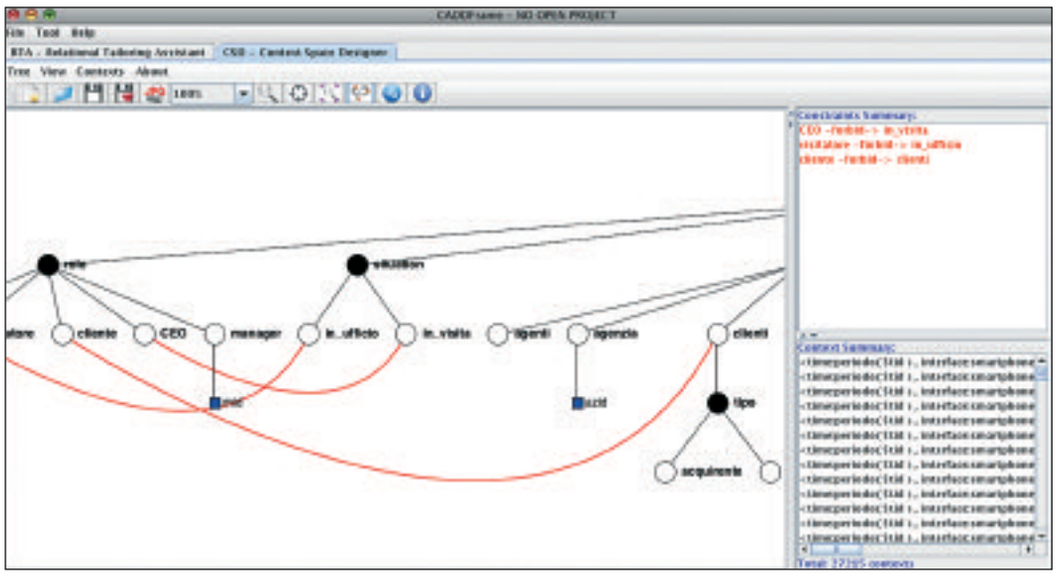


FIGURA 6

Il Context Space Designer - CSD

no espressi mediante formule logiche che predicano sulle dimensioni e sui loro valori e che possono essere create graficamente. Una volta completato il modello di contesto, CADD permette la generazione automatica di tutti i contesti validi (ovvero i contesti che rispettano i vincoli di progetto), ai quali possono essere associate le aree rilevanti degli schemi delle sorgenti.

4.5.2. Definizione delle aree rilevanti

La definizione delle aree rilevanti (*Relevant Areas*) degli schemi delle sorgenti avviene per mezzo del DTA. Il progettista associa gra-

ficamente ad ogni nodo del modello di contesto la porzione rilevante dello schema delle sorgenti; questa associazione si traduce poi in una vista sui dati della sorgente codificata nel linguaggio nativo della stessa (SQL, XQuery o SPARQL).
Come descritto precedentemente, una specifica di contesto è il risultato della combinazione di più dimensioni e quindi di più nodi del modello di contesto. Dato un contesto (cioè un insieme di nodi del CDT) CADD combina automaticamente le viste sulla sorgente dati in modo da ottenere una *vista contestuale*, determinata cioè da una specifica di contesto mediante combinazione delle viste associate ad ogni singolo nodo. Le associazioni (*contesto-vista*) tra contesti validi e interrogazione di ritaglio sono poi memorizzate nel dizionario dei contesti (Contexts Dictionary).

4.5.3. Messa in opera del sistema

CADD supporta anche la messa in opera (deployment) automatica del CAS su un server applicativo come Apache Tomcat. In una fase successiva all'installazione, è possibile modificare le viste contestuali con il DTA e aggiornare il dizionario dei contesti, in modo trasparente all'utente applicativo. È opportuno notare come l'architettura Context-ADDICT si appoggi su tecnologie esistenti senza richiedere modifiche ai sistemi di memorizzazione pre-esistenti come i database relazionali o XML. L'esecuzione delle interrogazioni contestuali atte a ritagliare le informazioni sulle sorgenti prescinde dalla natura delle stesse che rimangono meri esecutori delle interrogazioni. La complessità della federazione del sistema rimane all'interno dello strumento di modellazione ed è semplificata dall'interfaccia grafica del CADD.

5. CONCLUSIONI E SVILUPPI FUTURI

La nozione di contesto sta assumendo un ruolo sempre più rilevante nello sviluppo dei moderni sistemi informativi. Il sistema presentato in questo lavoro richiede la progettazione preventiva del modello di contesto, seguendo una precisa metodologia, mentre il valore delle variabili relative al contesto stesso viene stabilito dinamicamente in fase di esecuzione. Tale modo di procedere è reso ancora più efficace dalla disponibilità e diffusione di sensori di vario tipo e natura (dall'RFID al Telepass, al GPS ed altri ancora), che consentono di rilevare i valori direttamente dal mondo fisico circostante. Si rendono così possibili funzionalità di apprendimento, di analisi storiche e di ragionamento sui contesti [1], che costituiscono un importante passo verso la possibilità di modificare a tempo di esecuzione il modello stesso del contesto, definito per una certa applicazione. Questa é la sfida con la quale ci si dovrà confrontare nei prossimi anni.

Bibliografia

[1] AA.VV. IEEE Electronic Proc. of 5th Workshop on Context Modeling and Reasoning (CoMoRea'08), p. 406-452, Hong Kong, 2008.

[2] Bolchini C., Curino C.A., Quintarelli E., Schreiber F.A., Tanca L.: A data-oriented survey of context models. *ACM SIGMOD Record*, Vol. 36, n. 4, 2007, p. 19-26.

[3] BolchiniC., Curino C.A., Schreiber F.A., Tanca L.: *Context integration for mobile data tailoring. Electronic Proc.* of 7th Int. Conf. on Mobile Data Management (MDM'06), Nara (Japan), 2006, p. 1-8.

[4] Bolchini C., Quintarelli E., Rossato R.: *Relational data tailoring through view composition*. In: Proc. Intl. Conf. on Conceptual Modeling (ER'2007), Springer LNCS, Vol. 4801, 2007, p. 149-164.

[5] Bolchini C., Schreiber F.A., Tanca L.: A methodology for a Very Small Data Base design. *Information Systems*, Vol. 32, n. 1, 2007, p. 61-82.

[6] Bolchini C., Curino C., Orsi G., Quintarelli E., Schreiber F.A., Tanca L.: *CADD: a tool for context modeling and data tailoring*. Proc. IEEE Intl. Conf. on Mobile Data Management (MDM'07), Mannheim (Germany), 2007, p. 221-223.

[7] Chalmers M.: A historical view of context. *Computer Supported Cooperative Work*, Vol. 13, n. 3, 2004, p. 223-247.

[8] CoutazJ., Crowley J.L., Dobson S., Garlan D.: Context is key. *Communications of the ACM*, Vol. 48, n. 3, 2005, p. 49-53.

[9] Dey A.K.: Understanding and using context. *Personal Ubiquitous Computing*, Vol. 5, n. 1, 2001, p. 4-7.

[10] Lenzerini M.: Data Integration: *A Theoretical Perspective*. Proc. Of ACM PODS, 2002, p. 233-246.

[11] Motschnig-Pitrik R.: A Generic Framework for the Modeling of Contexts and its Applications. *Data & Knowledge Engineering*, 2000, p.145-180.

[12] Mylopoulos J., Motschnig-Pitrik R.: *Partitioning Information Bases with Contexts*. Proc. of the 3rd Intl Conf. on Cooperative Information Systems, 1995, p. 44-54.

[13] Ouksel A.M.: In-context peer-to-peer information filtering on the web. *SIGMOD Record*, Vol. 32, n. 3, 2003, p. 65-70.

[14] Petrelli D., Not E., Strapparava C., Stock O., Zancanaro M.: *Modeling context is like taking pictures*. In: Proc. of the Workshop "The What, Who, Where, When, Why and How of Context-Awareness". In: Proc. of CHI2000 - The future is here, 2000.

[15] Roussos Y., Stavrakas Y., Pavlaki V.: *Towards a Context-Aware Relational Model*. In: Contextual Representation and Reasoning Workshop (CRR'05), held in conjunction with CONTEXT'05, Paris, 2005.

[16] Sridharan H., Mani A., Sundaram H., Brungart J., Birchfield D.: *Context-Aware Dynamic Presentation Synthesis for Exploratory Multimodal Environments*. Proc. of IEEE Intl Conf. on Multimedia and Expo (ICME'05), 2005, p. 1014-1017.

[17] Stefanidis K., Pitoura E., Vassiliadis P.: *Modeling and storing context-aware preferences*. In: Proc. of 10th East European. Conf. on Advances in Databases and Information Systems (ADBIS'06), Springer LNCS, Vol. 4152, 2006, p. 124-140.

[18] Strimpakou M., Roussaki I., Anagnostou M.E.: *A context ontology for pervasive service provision*. In: 20th Intl. Conf. on Advanced Information Networking and Applications (AINA'06), 2006, p. 775-779.

[19] Theodorakis M., Analyti A., Constantopoulos P., Spyratos N.: A theory of contexts in information bases. *Information Systems*, Vol. 27, n. 3, 2002, p. 151-191.

[20] The Apache Software Foundation: Apache tomcat. http://tomcat.apache.org/.

[21] The International Business Machines Corporation: Ibm websphere application server, http://www-306.ibm.com/software/webservers/appserv/was/.

[22] The JBoss Community. Jboss application server, http://labs.jboss.com/jbossas/.

Cristiana Bolchini è Professore Associato presso il Politecnico di Milano. Laureata in Ingegneria Elettronica, indirizzo Automatica, nel Febbraio nel 1993, ha conseguito il Dottorato in Ingegneria Informatica e Automatica (IX ciclo) nello stesso ateneo. Gli interessi di ricerca riguardano sia problematiche di progetto di dispositivi digitali, con particolare attenzione agli aspetti di affidabilità, sia metodologie di progetto, ritaglio e gestione di dati.
E-mail: bolchini@elet.polimi.it

Giorgio Orsi è Dottorando di Ricerca presso il Dipartimento di Elettronica e Informazione del Politecnico di Milano. Nell'ottobre del 2006 ha ottenuto la laurea specialistica in Ingegneria Informatica presso il Politecnico di Milano. Il tema di ricerca maggiore riguarda lo studio di sistemi di integrazione di dati eterogenei e dinamici basati su ontologie, con particolare enfasi sulla gestione e l'evoluzione delle dipendenze dei dati.
E-mail: orsi@elet.polimi.it

Elisa Quintarelli è Ricercatrice presso il Dipartimento di Elettronica e Informazione del Politecnico di Milano, dove insegna Informatica A e Progetto di Basi di Dati. Gli interessi di ricerca attuali riguardano l'utilizzo di linguaggi di interrogazione e tecniche di datamining per documenti XML, la definizione di metodologie di progettazione di basi di dati per dispositivi portabili e di piccole dimensioni. Su questi e precedenti argomenti di ricerca è co-autrice di articoli, pubblicati su atti di conferenze e riviste internazionali. È inoltre autrice del libro "*Model-Checking Based Data Retrieval: an application to semistructured and temporal data*" pubblicato da Springer Verlag. Partecipa ai comitati di programma di workshop internazionali ed è revisore per riviste internazionali.
E-mail: quintarelli@elet.polimi.it

Fabio A. Schreiber è Professore Ordinario di Basi di Dati e di Technologies for Information Systems e vice Preside della Facoltà di Ingegneria dell'Informazione presso il Politecnico di Milano. I suoi interessi di ricerca riguardano i Sistemi di Elaborazione Distribuiti e i Sistemi Informativi; su questi argomenti ha partecipato a vari progetti di ricerca nazionali e internazionali. Ha fatto parte del Comitato Metropolitano per la Pubblica Amministrazione presso la Prefettura di Milano come consulente per i Sistemi Informativi nella P.A. Attualmente si occupa di problematiche relative ai Sistemi Informativi pervasivi, dipendenti dal contesto e comprendenti dispositivi molto piccoli (PDA, telefoni cellulari, sensori, Rfid ecc.). È autore di numerosi articoli scientifici ed è associate editor di Data & Knowledge Engineering.
E-mail: schreibe@elet.polimi.it

Letizia Tanca è Professore Ordinario di Basi di Dati e di Technologies for Information Systems presso il Politecnico di Milano. È autrice di diverse pubblicazioni internazionali sulle basi di dati e sulla teoria delle basi di dati, e del libro "*Logic Programming and Databases*", scritto con S. Ceri e G. Gottlob. Ha partecipato a vari progetti di ricerca nazionali e internazionali. I suoi interessi di ricerca riguardano tutta la teoria delle basi di dati, in particolare le basi di dati deduttive, attive e orientate agli oggetti, i linguaggi a grafi per basi di dati, la rappresentazione e l'interrogazione di informazione semistrutturata. Gli interessi più recenti vertono sul progetto di Basi di Dati sensibili al contesto, con particolare riferimento alle basi di dati per piccoli dispositivi mobili, e sulle basi formali e gli strumenti per il web semantico.
E-mail: tanca@elet.polimi.it