Università degli Studi di Modena e Reggio Emilia

Facoltà di Ingegneria – Sede di Modena

Corso di Laurea Specialistica in Ingegneria Informatica – *NOD*

Tecniche per l'esecuzione efficiente di interrogazioni sul contenuto in collezioni MPEG-7

Relatore:
Chiar.mo Prof. Paolo Tiberio

Candidato:
Alessandro Tonelli

Correlatori:
Dott. Federica Mandreoli
Ing. Riccardo Martoglia

Anno Accademico 2004-2005

RINGRAZIAMENTI

Vorrei ringraziare in primis i miei genitori, mia sorella Elena ed i miei nonni che in tutti questi anni di studio mi hanno aiutato, incoraggiato, spronato fino al raggiungimento di questo importante risultato. Ringrazio il Professor Paolo Tiberio, la Dottoressa Federica Mandreoli e soprattutto l'Ing. Riccardo Martoglia per la loro squisita disponibilità e l'aiuto fornito durante questi mesi di lavoro. Ringrazio infine tutti gli amici e compagni d'università che mi sono stati vicini nel corso della mia carriera universitaria.

PAROLE CHIAVE

MPEG-7
MMDBMS
content querying
ricerche di similarità
M-Tree

Indice

Introduzione **1**

I Stato dell'arte

1 MPEG-7 ..... **5**

1.1 MPEG-7 e i Multimedia Database Systems 8

2 I MULTIMEDIA DATABASE MANAGEMENT SYSTEMS **10**

2.1 MMDMS per MPEG-7 12

2.2 Soluzioni database XML 13

2.2.1 Soluzioni database native 14

2.2.2 Estensioni database 16

2.3 Analisi 17

2.3.1 Rappresentazione delle media descriptions 18

2.3.2 Accesso alle media descriptions 22

2.3.3 Schemi delle media descriptions 31

2.3.4 Estensibilità 35

2.3.5 Funzionalità classiche DBMS 37

2.4 Conclusioni dell'analisi 41

3 QUERYING DI DATI MPEG-7 ..... **42**

3.1 Classi generiche di query 42

3.2 Feature-based querying 43

3.3 Semantic querying 44

3.3.1 Interrogazione mediante XQuery 45

3.3.2 Interrogazione di informazioni implicite 48

4 PROCESSING DI QUERY **65**

4.1 Motore di XQuery 65

4.2 Reti inferenziali 66

4.3 Alberi 74

4.4 Viste 81

II Tecniche per l'esecuzione di interrogazioni sul contenuto in collezioni MPEG-7

5 IL SISTEMA XSITER 85

5.1 Generalità 85

5.2 Architettura 85

5.3 Strutture datastore e rappresentazioni interne 88

5.4 Algoritmi specializzati, indici e caratteristiche di memorizzazione 90

6 RICERCHE DI SIMILARITÀ IN SPAZI METRICI: M-TREE 92

6.1 Alberi e spazi metrici 92

6.2 L'M-tree 94

6.2.1 La struttura dei nodi 95

6.2.2 Processing delle query di similarità 96

6.2.3 La costruzione dell'M-tree 100

6.2.4 La gestione degli split 101

6.2.5 Le politiche di split 103

7 IL PROGETTO DEL SOFTWARE 106

7.1 Scelta della libreria 106

7.2 Specializzazione di XSiter per il supporto a MPEG-7 ed al recupero multimediale 107

7.3 Modifiche al sistema XSiter 108

7.4 Interrogazioni di similarità in XSiter 110

7.4.1 Costruzione dell'M-tree 110

7.4.2 Inserimento degli oggetti 114

7.4.3 Esecuzione delle query 115

7.4.4 Chiusura e riapertura dell'albero 116

7.5 Prove sperimentali 117

Conclusioni e sviluppi futuri 119

Bibliografia 121

Indice delle figure

Figura 1 - Organizzazione generale MPEG-7 6
Figura 2 - Principali componenti dello standard MPEG-7 e le loro relazioni 8
Figura 3 - Utilizzo di MPEG-7 in un sistema database multimediale 8
Figura 4 - Architettura con MMDBMS e server video 11
Figura 5 - Selezione di soluzioni database XML 13
Figura 6 - Risultati di analisi (■ supporto, □ supporto parziale, - nessun supporto) 40
Figura 7 - Una Inference Network 52
Figura 8 - Operatori supportati dal modello IN 53
Figura 9 - La Document Network a tre strati 54
Figura 10 - Un vincolo normale 55
Figura 11 - Un vincolo complesso 56
Figura 12 - Estratti di descrizioni MPEG-7 di immagini 57
Figura 13 - Descrizione MPEG-7 della query 58
Figura 14 - Esempio dell'albero di descrizione MPEG-7 60
Figura 15 - Esempio di un albero di query MPEG-7 61
Figura 16 - Un esempio di query analizzato secondo differenti viste 61
Figura 17 - Le cinque viste 62
Figura 18 - Un esempio di query SVQL 63
Figura 19 - Link Inheritance 69
Figura 20 - Path Cropping 70
Figura 21 - Una DN di esempio 72
Figura 22 - Struttura del processo di recupero 74
Figura 23 - Concetti dell'ontologia per i ruoli di persona in una partita di pallacanestro 76
Figura 24 - Esempio di inclusione consentita 77
Figura 25 - Embedding di un query tree MPEG-7 in un data tree MPEG-7 80
Figura 26 - Architettura a strati di recupero basata sull'uso di SVQL 82
Figura 27 - Interfaccia di interrogazione SQVL in COALA 83
Figura 28 – Architettura astratta di XSiter 86
Figura 29 - Architettura del Query Engine 88
Figura 30 - Struttura e contenuto di un datastore 89

Figura 31 - Applicazione del lemma 2 per evitare calcoli di distanza 97
Figura 32 - Classi di XSiter che rappresentano indici 108
Figura 33 - Esempio di metriche di Minkovski 111
Figura 34 - Distanze di Miskowski 112
Figura 35 - Un esempio di uso dei containers 113
Figura 36 - Risultati interrogazioni di similarità 118

Introduzione

L'evoluzione tecnologica degli ultimi anni ha portato ad un aumento considerevole di informazioni audiovisive e la produzione di contenuti multimediali è in continua crescita. Senza strumenti adeguati di ricerca però, tutto questo patrimonio rischia di rimanere poco o per niente utilizzato. A tale riguardo sono stati compiuti un gran numero di iniziative e sforzi. Un primo passo verso la soluzione di questo problema è stata sicuramente la realizzazione di standard di metadati per la descrizione dei contenuti multimediali, tali da permettere un'agevole descrizione dei contenuti audiovisivi, sui quali poter fare ricerche, selezioni, sincronizzazioni temporali e simili. Tra questi lo standard MPEG-7 (formalmente chiamato "Multimedia Content Description Interface") è quello maggiormente accettato. MPEG-7 specifica un insieme standard di strumenti che possono essere usati per descrivere diversi tipi di informazioni multimediali.

Il tema della mia tesi riguarda lo sviluppo di metodi di interrogazione su dati video attraverso misure di similarità: queste ricerche vengono effettuate usando appunto l'annotazione MPEG-7. Poiché le descrizioni multimediali MPEG-7 non sono altro che documenti XML conformi alle definizioni di schema scritte in MPEG DDL (Data Definition Language), è possibile impiegare soluzioni XML per la loro gestione.

Una parte importante del mio lavoro analizza i requisiti che devono essere soddisfatti dal sistema per gestire efficientemente i documenti XML. Un sistema in grado di gestire query su dati XML dovrebbe memorizzare documenti XML, esprimere query e processarle in maniera efficiente. A tal fine è possibile sfruttare una grande varietà di soluzioni database, principalmente raggruppate in due categorie: database relazionali che supportano estensioni XML oppure database XML nativi.

Per il progetto di tesi viene utilizzato il sistema XSiter (XML SIgnaTure Enhanced Retrieval), un query processor XML nativo ed estensibile sviluppato presso l'Università di Modena e Reggio Emilia capace di immagazzinare, gestire ed interrogare collezioni di documenti XML di qualsiasi numero e dimensione. XSiter è capace di fornire supporto efficiente ad interrogazioni su digital libraries multimediali, finora limitate a ricerche sui valori di tipo esatto.

A partire da questo sistema si è cercato di compiere ulteriori passi in avanti. Per migliorare l'efficacia e l'efficienza per le applicazioni specifiche MPEG-7 e i metadati multimediali sono state incluse le astrazioni necessarie allo strato di applicazione per

controllare tipi di dato complessi quali i tipi di dati XML e le loro estensioni specifiche MPEG-7, compresi gli array e le matrici. Inoltre sono state aggiunte all'architettura del sistema le strutture specializzate di indice adatte a gestire questi tipi supplementari di dati che hanno permesso ad XSiter di andare facilmente oltre il match esatto per i valori, che non è assolutamente sufficiente in un ambiente di metadati multimediali, e di supportare la ricerca di similarità.

Per soddisfare i fabbisogni informativi degli utenti è di vitale importanza supportare efficientemente ed efficacemente il processo di recupero per determinare quali porzioni del database sono rilevanti per le richieste dell'utente. In particolare c'è un urgente bisogno di tecniche d'indicizzazione capaci di supportare l'esecuzione delle query di similarità. L'approccio più generale al problema dell'indicizzazione di similarità ha portato allo sviluppo dei cosiddetti alberi metrici che considerano solo le le distanze relative degli oggetti piuttosto che la loro posizione assoluta nello spazio multidimensionale. Tra questi è stato analizzato l'M-tree, progettato esplicitamente per essere integrato con altri metodi di accesso nei sistemi database, per indicizzare gli oggetti usando caratteristiche confrontate con funzioni di distanza. Usando gli M-tree ed algoritmi di ricerca è stato possibile processare due tipi di query di similarità di base: la range query e la k nearest neighbor query.

Infine sono stati eseguite prove sperimentali per testare le prestazioni del sistema nel processare tali query di similarità.

La tesi è strutturalmente organizzata in 7 capitoli raggruppati in due parti principali:

- la Parte I, formata dai primi 4 capitoli, presenta lo stato dell'arte e illustra gli scenari ed il contesto degli studi esistenti sul tema;
- la Parte II, formata dai restanti 3 capitoli, sviluppa e ricerca le tecniche più efficienti per effettuare interrogazioni su dati multimediali.

Entrando nel dettaglio dei singoli capitoli:

- nel capitolo 1 viene introdotto in maniera sintetica lo standard MPEG-7
- il capitolo 2 descrive i Multimedia Database Management Systems e analizza nel dettaglio le soluzioni database XML correnti con le loro capacità e le loro limitazioni in relazione alle descrizioni multimediali MPEG-7
- il capitolo 3 presenta una panoramica delle diverse tipologie di approcci possibili all'interrogazione di tali descrizioni

- il capitolo 4 tratta il processing delle query sui dati multimediali MPEG-7
- il capitolo 5 è dedicato all'illustrazione del sistema XSiter
- il capitolo 6 affronta il problema dell'indicizzazione di similarità e propone a tal fine diverse soluzioni, descrive in generale gli alberi metrici ed in particolare i principi e gli algoritmi di base della struttura M-Tree per poter effettuare ricerche di similarità efficienti
- il capitolo 7 è dedicato infine all'implementazione pratica del progetto di tesi e presenta prove sperimentali eseguite sul sistema

Parte I

Stato dell’arte

1 MPEG-7

MPEG-7 è un emergente standard ISO/IEC sviluppato da MPEG (Moving Picture Experts Group), formalmente chiamato "Multimedia Content Description Interface", che fornisce un insieme di specifiche e strumenti per la descrizione, attraverso metadati, di contenuti di tipo multimediale. A differenza degli standard MPEG precedenti che avevano come obiettivo lo sviluppo di algoritmi di compressione (MPEG-1, MPEG-2) e l'organizzazione in oggetti di realtà multimediali eterogenee (MPEG-4), esso si pone come fine ultimo la descrizione di informazioni multimediali attraverso una rappresentazione testuale (XML) che faciliti l'identificazione, il filtraggio, la ricerca, il reperimento e la gestione dei dati multimediali, siano essi file audio, filmati o immagini.

MPEG-7 permette differenti granularità nelle sue descrizioni offrendo la possibilità di avere differenti livelli di distinzione. Sebbene la descrizione MPEG-7 non dipenda dalla specifica rappresentazione del materiale (indipendenza dal formato), MPEG-7 può sfruttare i vantaggi forniti dal contenuto codificato MPEG-4. Per esempio, se il materiale è codificato usando MPEG-4 che fornisce un mezzo per codificare materiale audiovisivo come oggetti aventi certe relazioni nel tempo (sincronizzazione) e nello spazio (sullo schermo per il video o nella stanza per l'audio), sarà possibile collegare descrizioni agli elementi (oggetti) dentro la scena, così come oggetti audio e video.

I dati MPEG-7 possono essere collocati fisicamente con il materiale audiovisivo a loro associato nello stesso flusso di dati oppure nello stesso sistema di database multimediale, ma le descrizioni possono anche risiedere da qualche altra parte in Internet. Quando il contenuto e la sua descrizione non sono collocati insieme, vengono forniti meccanismi per il loro collegamento; questi collegamenti devono funzionare in entrambe le direzioni.

I metodi e gli strumenti MPEG-7 sono raggruppati in otto parti:

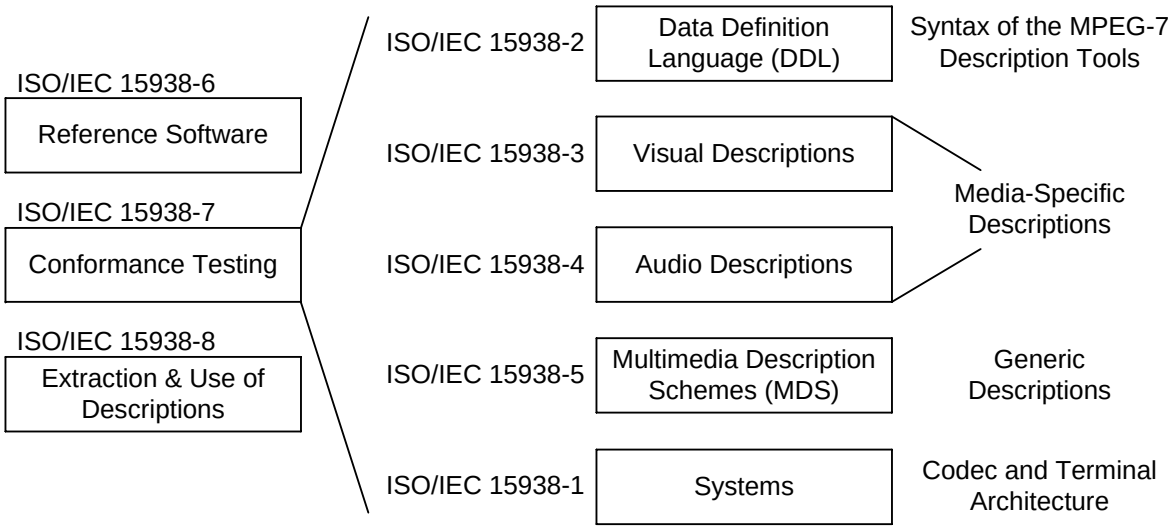


Figura 1 - Organizzazione generale MPEG-7

Le parti a destra nella figura 1 sono state progettate per creare una descrizione e trasportarla.
La prima (parte 2) contiene la descrizione del Description Definition Language che è un linguaggio che definisce la sintassi dei Description Tools MPEG-7 e che permette la creazione di nuovi Description Schemes. Gli elementi descrittivi sono definiti nelle parti da 3 a 5 distinte tra loro dal livello di genericità impiegato. MPEG-7 Video contiene i Description Tools che forniscono strutture per la descrizione di contenuti video, MPEG-7 Audio contiene i Description Tools che forniscono strutture per la descrizione di contenuti audio ed infine MPEG-7 Multimedia Description Schemes (MDS) contiene i Description Tools che forniscono strutture per la descrizione di entità generiche e/o multimediali. MPEG-7 Systems fornisce un sistema per la codifica in formato binario delle descrizioni MPEG-7 e la "Terminal Architecture" che è una specie di applicazione stand-alone per la codifica, la decodifica, la memorizzazione ecc. dell'informazione MPEG-7.

Le parti 3 e 4 forniscono il più basso livello di astrazione che può essere, per esempio, forma, dimensione, tessitura, colore, movimento (traiettoria) e posizione per il video, e chiave, tono, tempo, cambi di tempo e posizione nello spazio sonoro per l'audio. Il più alto livello si può trovare nella parte 5, nell'MDS, e permette di rappresentare il contenuto semantico di un'entità, come per esempio "Un video di calcio con il portiere sulla destra ed il giocatore con il pallone sulla sinistra". Inoltre possono anche esistere livelli intermedi di astrazione. In aggiunta agli strumenti di descrizione semantica, l'MDS fornisce un mezzo per descrivere i processi di creazione e produzione del contenuto, l'utilizzo e il contesto.

La parte 6, Reference Software, racchiude le prime cinque parti, nel senso che fornisce un'implementazione software delle parti normative più rilevanti di MPEG-7. Le ultime due parti di MPEG-7 sono la parte 7, MPEG-7 Conformance Testing, che tratta di linee guida e procedure su come testare la conformità di implementazioni MPEG-7, e la parte 8, MPEG-7 Extraction and Use of Descriptions, che fornisce materiale informativo nella forma di rapporto tecnico sull'estrazione e sull'uso di alcuni Description Tools. Tutte le parti sono standard internazionali.

Lo standard MPEG-7 utilizza XML (eXtensible Markup Language) come linguaggio per la rappresentazione testuale dei contenuti multimediali e "XML-Schema" come base per il DDL (Data Description Language) usato per la definizione sintattica dei description tools MPEG-7.

I componenti principali di MPEG-7 sono:

- *Description Tools*: formati dai Descriptors (Ds) che definiscono la sintassi e la semantica di ogni feature (ossia l'elemento metadato) e dai Descriptors Schemes (DSs) che specificano la struttura e la semantica delle relazioni tra i vari componenti (ossia le relazioni fra i vari metadati) che possono essere sia D che DS.
- *Data Description Language (DDL)*: definisce la sintassi dei Description Tools permettendo la creazione di nuovi Description Schemes e Descriptors e, possibilmente, l'estensione e la modifica di DS esistenti.
- *Classification Schema (CS)*: definisce una lista di termini tipici usati in molte applicazioni insieme al loro significato. Per esempio, esso permette la definizione di formati di file in maniera standardizzata. MPEG-7 fornisce molti CS predefiniti per caratterizzare ruoli, formati e così via.
- *System Tools*: supportano la rappresentazione dei contenuti in formato binario per una maggiore efficienza nella memorizzazione e nella trasmissione e provvedono ai necessari meccanismi di trasmissione.

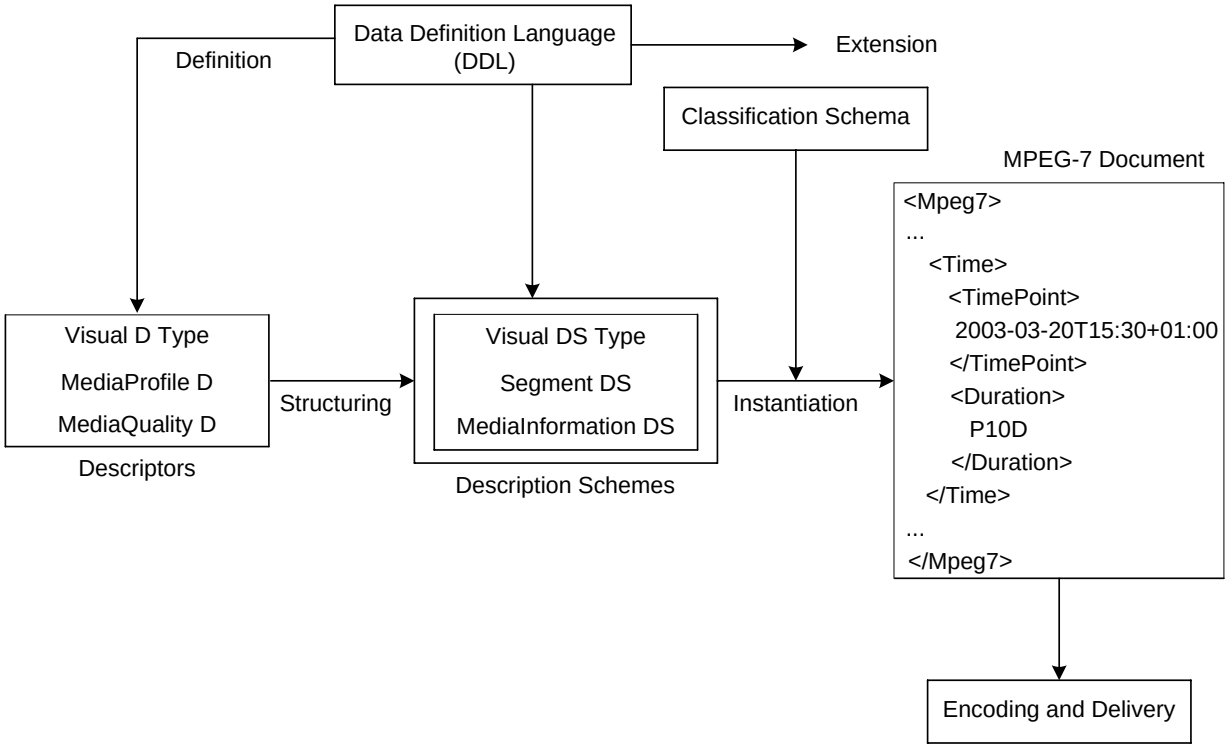


Figura 2 - Principali componenti dello standard MPEG-7 e le loro relazioni

1.1 MPEG-7 e i Multimedia Database Systems

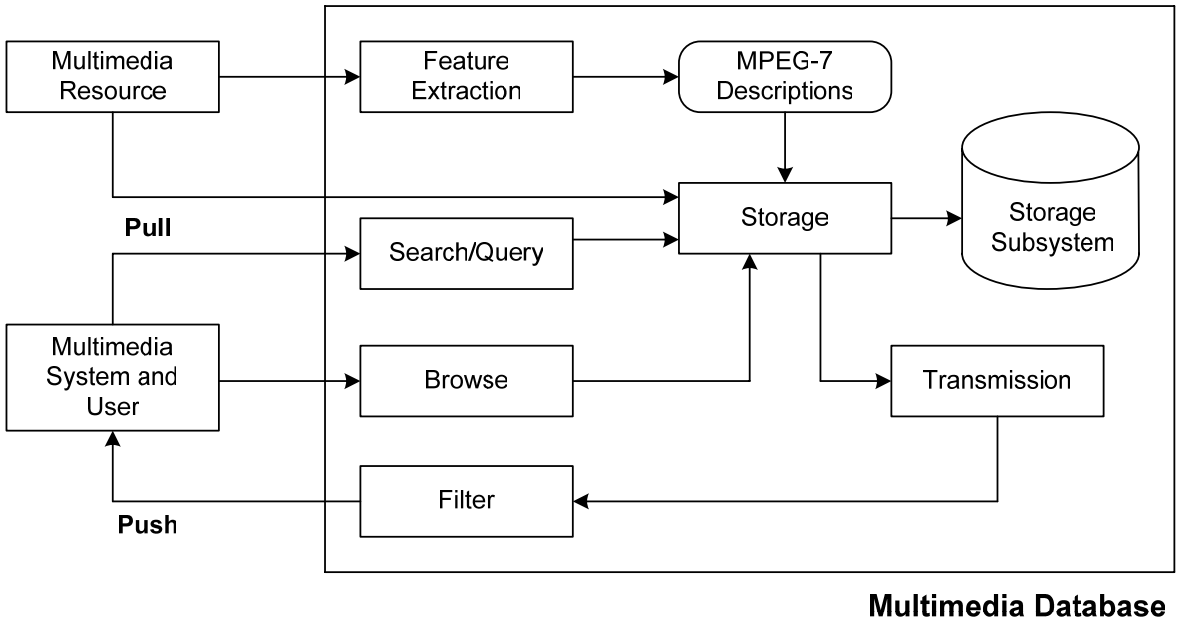


Figura 3 - Utilizzo di MPEG-7 in un sistema database multimediale

La figura 3 mostra un possibile uso di MPEG-7 in un sistema database multimediale distribuito. Partendo dal processo di annotazione (strumenti automatici o annotazione manuale) le descrizioni MPEG-7 sono generate ed immagazzinate per ulteriori accessi. Il processo di indicizzazione può essere supportato dalle informazioni memorizzate. Per esempio, per annotare un nuovo video di calcio, si può contare sulle informazioni dei giocatori già immagazzinati nel database per evitare l'annotazione multipla dello stesso giocatore. Occorre notare che i metadati multimediali possono essere disponibili prima che il video sia disponibile. Per esempio, l'informazione sulla posizione e sui giocatori della squadra è in generale disponibile prima che avvenga la partita di calcio.

Lo scenario di query distingue due casi: pull (estrarre) e push (spingere). Se si considera uno scenario pull, un utente sottomette le query al database multimediale e riceve un insieme di descrizioni che soddisfano la query per la navigazione (per controllare la descrizione, per maneggiarla, per recuperare il contenuto descritto, ecc.). Alternativamente, se è disponibile un server di streaming, l'utente riceve i media insieme alle descrizioni.

In uno scenario push, un filtro (per esempio, un agente intelligente) seleziona le descrizioni a partire da quelle disponibili ed esegue le azioni programmate in seguito (per esempio, commutando un canale broadcast o registrando lo stream descritto). Per questo scenario è conveniente codificare e indicizzare il video in formato binario per un veloce accesso casuale. In entrambi gli scenari, tutti i moduli possono trattare descrizioni codificate nei formati MPEG-7 (testuale o binario).

Un esempio di scenario pull si presenta quando un utente registra 10 secondi di una canzone radiofonica (canzone di query) e li trasmette ad un servizio mobile che conta su un database multimediale per il riconoscimento del titolo e del cantante. La base di dati multimediale estrae l'AudioSignature della canzone e genera una descrizione MPEG-7 della canzone di query. Questa descrizione è confrontata alle *signature* (firme) immagazzinate nel database audio e la firma più simile viene conservata. Il sistema ricupera le informazioni del titolo e del cantante per questa firma risultante e le restituisce all'utente. Uno scenario pull tipico è collegato con la trasmissione della televisione digitale con in più le informazioni MPEG-7. Si supponga che un telespettatore abbia espresso in MPEG-7 le sue preferenze circa il contenuto di programma (User-Preferences DS). Per esempio, ha chiesto che gli siano presentate soltanto le notizie sportive da un programma di notizie della televisione. In questo caso, i descrittori MPEG-7 nello stream trasmesso sono filtrati secondo le User-Preferences descriptions MPEG-7 disponibili e viene mostrato soltanto il programma di notizie di sport.

Dai metadati MPEG-7 disponibili con il flusso trasmesso, l'utente potrebbe identificare il materiale aggiuntivo che è specificamente di suo interesse. Per esempio, si immagini che un utente stia guardando "Mai dire mai". Egli può decidere di cercare ulteriori informazioni per quanto riguarda quel film (per esempio, altri film con Klaus Maria Brandauer) o semplicemente altri dati relativi al film disponibili su Internet (per esempio, il prezzo della colonna sonora del film su CD).

2 I MULTIMEDIA DATABASE MANAGEMENT SYSTEMS

Un MultiMedia DataBase Management System (MMDBMS) deve fornire un ambiente adatto per utilizzare e gestire dati multimediali. Quindi deve supportare i vari tipi di dati multimediali oltre che fornire facilitazioni per le funzioni DBMS tradizionali come creazione di database, modellazione, recupero, accesso, organizzazione e indipendenza dei dati.

La natura speciale dei dati multimediali ha anche reso importante supportare nuove e speciali funzioni. Queste includono la gestione di enormi volumi di dati multimediali e la gestione efficace dell'immagazzinamento e del trasporto in un server di immagazzinamento multimediale. Il server d'immagazzinamento può essere parte del DBMS, ma può anche essere un componente fisico individuale. In questo caso, per garantire efficienza, il DBMS deve essere strettamente connesso al server d'immagazzinamento.

In aggiunta, un MMDBMS deve fornire i mezzi per un'efficiente indicizzazione e recupero dell'informazione, supportare modelli concettuali dei dati multimediali e gestire l'ottimizzazione della query e le questioni di processing. MMDBMS deve anche rivolgersi ai problemi specifici dei database distribuiti, come il processing delle query distribuite.

I problemi della rete come la larghezza di banda limitata e i ritardi della rete sono considerazioni importanti poiché possono avere effetti negativi sulla qualità del servizio. A differenza di quanto accade nei tradizionali DBMS, la replicazione dei dati non viene spesso incoraggiata in un MMDBMS distribuito a causa del largo volume di dati scambiati. In questo contesto, per semplici applicazioni pull (ossia dove il client richiede dati al server), il modello di computazione Client-Server (C-S) è considerato adatto per MMDBMS distribuiti. Applicazioni multimediali più complesse potrebbero richiedere un'architettura con server video e MMDBMS, che è più dinamica.

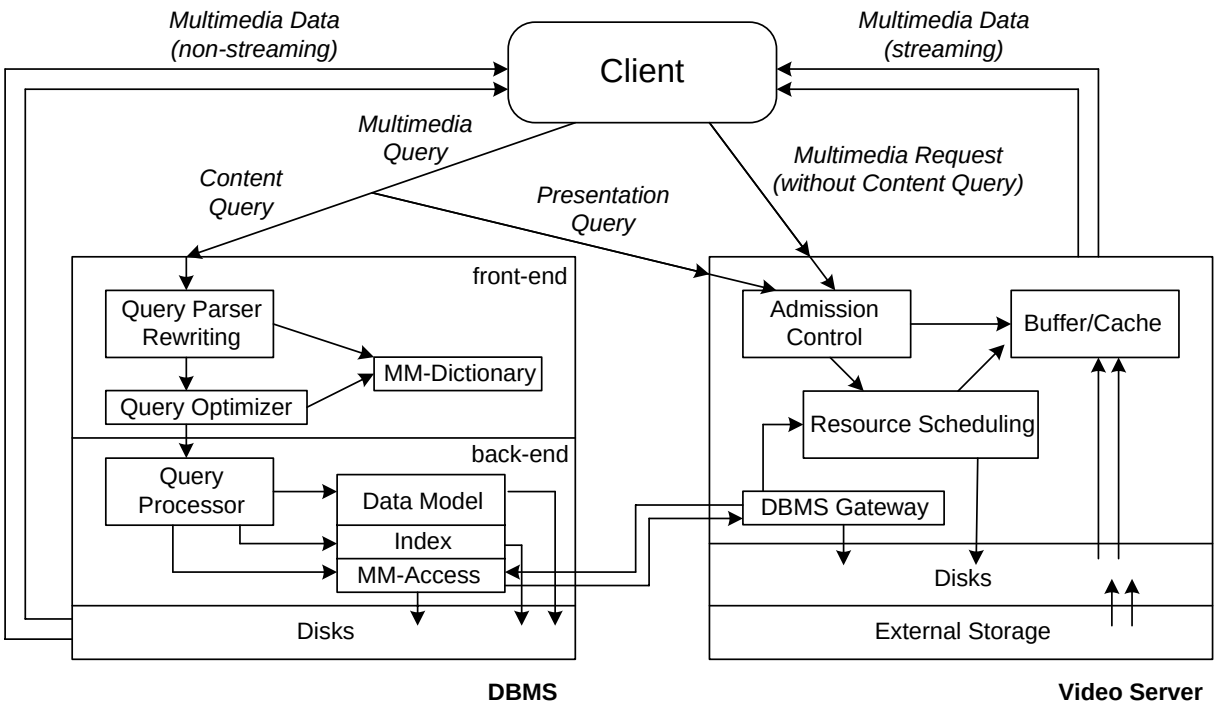


Figura 4 - Architettura con MMDBMS e server video

Concentriamoci prima sulla *content query*. La query viene inizialmente analizzata e, qualora venga rilevata un'eventuale ridondanza nella formulazione, viene riscritta. Successivamente l'ottimizzatore della query (*query optimizer*) prepara un piano di esecuzione per il processore della query (*query processor*) il quale cerca il contenuto che corrisponde alla specificazione della query. Questo processo di ricerca può coinvolgere complessi meccanismi di processing (come l'esecuzione di join multimediali, group by, ecc.). Poi, il MMDBMS informa il server video riguardo ai dati multimediali che devono essere consegnati. Se l'MMDBMS ed il server video sono strettamente accoppiati, per esempio, attraverso un gateway DBMS disponibile sul server video, non è necessaria ulteriore sottomissione di query al server video ed i dati sono quindi consegnati possibilmente mediante trasmissione dal server video al client. Se questa stretta comunicazione non è disponibile, i risultati del database sono rinviati al client e possono essere sottomessi al server video in seguito.

In questo modo, il client può contattare il server video direttamente, senza coinvolgere l'MMDBMS, oppure dopo aver ottenuto i metadati sul video dal MMDBMS. Queste query possono imporre certi vincoli di presentazione sul server video; per esempio, se sono richiesti due video, il secondo video sarà trasmesso 5 secondi dopo la fine del primo. La query viene prima analizzata dal controllo di ammissione (*admission control*).

Il lavoro del controllo di ammissione è trovare un elenco ottimale dei video richiesti tale che né i vincoli di presentazione né i vincoli della rete e del server siano violati. Ciò suppone che il server abbia abbastanza risorse disponibili (dimensione del buffer, larghezza di banda del disco, ecc.) per sostenere le richieste di tutti i flussi video attualmente supportati. Così, per ogni query nuovamente sottomessa, il controllo di ammissione verifica se essa può essere ammessa o no. Durante l'ammissione, le risorse richieste per la consegna del video sono state riservate in anticipo. A questo punto il modulo di schedulazione delle risorse (*resource scheduling*) è responsabile della gestione delle risorse per tutti i flussi video serviti concorrentemente. Il buffering è un problema importante nella gestione delle risorse del server video ed è utilizzato per ammortizzare differenti larghezze di banda delle risorse; per esempio, tra disco, memoria e rete.

2.1 MMDMS per MPEG-7

Essenzialmente, le media descriptions MPEG-7 sono documenti XML che seguono i media description schemes definiti con una variante di XML Schema. Quindi, è ragionevole studiare le soluzioni database correnti per i documenti XML per quanto riguarda la loro idoneità alla gestione di queste descrizioni.

Esiste una varietà confusionaria di soluzioni database XML con differenti maturità e possibilità: soluzioni database XML "native"così come estensioni XML dei sistemi di gestione database (DMS, Database Management Systems) tradizionali, prodotti commerciali così come progetti open source e prototipi di ricerca.

Vediamo quindi una descrizione dettagliata delle correnti soluzioni database XML con le loro capacità e limitazioni.

2.2 Soluzioni database XML

L'area delle soluzioni database XML è tuttora molto attiva. Esiste una varietà sconcertante di sistemi a differenti stadi di sviluppo e con differenti gradi di maturità che riguardano la gestione di documenti XML in un database. Sono stati ristretti gli studi ad un insieme di soluzioni database XML di importanza e rilevanza. Come si può osservare in figura 5, l'insieme selezionato di soluzioni database XML consiste di prodotti commerciali, progetti open source e prototipi di ricerca grossolanamente classificati in accordo alla terminologia comune in soluzioni database XML *native* e *estensioni* database XML che permettono l'immagazzinamento di documenti XML all'interno dei DBMS convenzionali, di solito relazionali o object-oriented.

	Native database solutions	**Database extensions**		
		Unstructured storage	Structured storage	Mapping
Commercial	eXcelon XIS GoXML DB Infonyte-DB Tamino TEXTML X-Hive/DB	IBM DB2 XML Extender Microsoft SQLXML Oracle XML DB		Oracle XML DB/ Structured Mapping
Open source	dbXML eXist Xindice		ozone/XML	
Research	Lore Natix PDOM TIMBER		Monet XML Shimura et al. XML Cartridge	

Figura 5 - Selezione di soluzioni database XML

Sebbene si incontri frequentemente in congiunzione con i database XML e appaia a prima vista intuitivamente chiara, la nozione di soluzione database XML nativa è oscura e non esiste una definizione accettata largamente. In pratica, si deve comprendere che il termine nativo è soprattutto impiegato dai venditori di soluzioni database XML specializzate come un marchio di marketing per distinguere i loro prodotti dalle estensioni database XML offerte dai maggiori venditori di DBMS tradizionali. Per dare un'idea della varietà di interpretazioni del termine nativo presenti in giro, una classe di definizioni considera come aspetto centrale il modo in cui i documenti XML vengono memorizzati che permette di distinguere le soluzioni database XML native dalle altre.

Alcune di queste definizioni ritengono che la caratteristica essenziale di una soluzione database XML nativa sia quella di memorizzare i documenti XML nel loro originario formato di file testuale, applicando al massimo tecniche di compressione per ridurre lo spazio di immagazzinamento e mantenendo indici addizionali per velocizzare le operazioni di recupero. Altre definizioni orientate all'immagazzinamento rendono malleabile questo punto di vista relativamente rigido e permettono ad una soluzione database nativa di definire un modello dei dati logico come DOM per la struttura gerarchica dei documenti XML e di memorizzare documenti in accordo con questo modello che impiega ogni immagazzinamento fisico desiderato. Ancora un'altra classe di definizioni accoglie un punto di vista molto astratto e definisce una soluzione database XML nativa come una soluzione che è stata specificamente sviluppata per la gestione dei documenti XML.

Di fronte a questa diversità di definizioni, si vorrebbe raggiungere un criterio trattabile per distinguere tra le categorie di soluzioni database XML native e estensioni database XML per lo scopo dell'analisi che ha dimostrato di essere simile alla stessa concezione dei maggiori venditori di prodotti database XML. Viene presa in considerazione come caratteristica distintiva tra entrambe le categorie il fatto che una soluzione database XML nativa permette la modellazione di dati solo per mezzo di documenti XML. Un'estensione database XML, in contrasto, offre ancora alle applicazioni le primitive di modellazione del modello dei dati del DBMS esteso. Quindi, il predicato nativo non implica che una soluzione database nativa sia stata sviluppata dall'inizio alla fine specificatamente per la gestione di documenti XML; una soluzione nativa dovrebbe essere basata sulla tecnologia di database tradizionali finché il modello dei dati del sistema sottostante è interamente nascosto.

2.2.1 Soluzioni database native

Per servire il bisogno crescente di una gestione efficiente di larghe quantità di documenti XML è apparsa sul mercato una grande varietà di soluzioni database XML native. Sostenendo che i documenti XML non potessero essere memorizzati efficientemente nei tradizionali DBMS a causa della loro natura gerarchica e semistrutturata, parecchi venditori hanno sviluppato interi DBMS specializzati nella gestione di documenti XML. Probabilmente la più rappresentativa di questo gruppo di soluzioni database XML native è Tamino; soluzioni commerciali che ricadono nella stessa categoria sono X-Hive/DB e GoXML DB. Infonyte-DB, che si è evoluta dal prototipo di ricerca PDOM, è stata specificamente sviluppata per la gestione di documenti XML.

Comunque, Infonyte-DB non fornisce un pratico database server con tutte le funzionalità standard DBMS come la gestione della transazione ed il controllo della concorrenza ma piuttosto costituisce una leggera soluzione di immagazzinamento in corso per i documenti XML.

Per di più, i venditori di DBMS object-oriented hanno sfruttato l'opportunità di prendere la loro fetta di torta dal mercato della gestione di documenti XML e dato nuova forma ai loro DBMS esistenti verso soluzioni database XML native. Un rappresentante prominente di questo approccio è eXcelon XIS, che internamente fa uso del DBMS object-oriented ObjectStore. In maniera simile, i venditori di sistemi di gestione di documenti XML hanno dato nuova forma alla loro tecnologia esistente verso soluzioni database XML native, per esempio, TEXTML.

Separati da queste soluzioni commerciali, sono emersi recentemente parecchi progetti open-source miranti allo sviluppo di soluzioni database XML native. Apache XML Project ha implementato daccapo un DBMS specializzato per i documenti XML chiamato Xindice. Xindice è il successore di dbXML precedentemente guidato dal dbXML Project. eXist è un altro esempio di progetto open source che implementa una soluzione database XML nativa. Diversamente da Xindice, comunque, eXist è costruito in cima ad un sistema database relazionale, MySQL o PostgreSQL, che serve internamente come immagazzinamento posteriore persistente.

Infine, c'è stata una considerevole ricerca riguardante soluzioni database XML native. Un esempio importante è il prototipo DBMS semistrutturato di ricerca Lore che è stato sviluppato verso una soluzione database XML nativa. L'idea è quella di sfruttare l'abilità di Lore di rappresentare strutture grafiche irregolari (includendo gerarchie) ed il potente linguaggio di query del sistema per la gestione di documenti XML. Le soluzioni database native Natix e TIMBER sono ulteriori importanti prototipi di ricerca. Mentre il cuore di Natix è un gestore di immagazzinamento dedicato che supporta un raggruppamento flessibile ed efficiente di alberi e sottoalberi di documenti XML in registri disco fisici di dimensioni limitate, TIMBER è stato costruito in cima al gestore di immagazzinamento multiuso Shore.

2.2.2 Estensioni database

Fondamentalmente, possono essere distinti tre approcci per rappresentare documenti XML nei DBMS tradizionali. Nel primo approccio, che viene chiamato *unstructured storage* (immagazzinamento non strutturato), un documento XML è immagazzinato direttamente nel suo formato testuale in un campo CLOB (Character Large Object). Questo è l'approccio supportato oggi dai maggiori DBMS relazionali: questi sistemi sono stati estesi con tipi di dati basati su CLOB per documenti XML con funzioni più o meno sofisticate per accedere al contenuto di un documento da SQL. Rappresentanti prominenti di estensioni database relazionali che supportano l'immagazzinamento non strutturato dei documenti XML sono Oracle XML DB, IBM DB2 XML Extender e Microsoft SQLXML.

Nel secondo approccio, che viene chiamato *structured storage* (immagazzinamento strutturato), viene costruito un metamodello a grana fine di documenti XML capace di rappresentare i nodi d'albero di documenti XML arbitrari impiegando le primitive di modellazione del DBMS tradizionale sottostante. Questo rende accessibile la struttura e i contenuti dei documenti XML ai mezzi di interrogazione forniti con il DBMS. C'è stata una notevole ricerca in quest'area ed è stata sviluppata una gran quantità di prototipi di ricerca di estensioni database per l'immagazzinamento strutturato di documenti XML, principalmente per DBMS relazionali. Per l'analisi, sono stati esaminati i prototipi rappresentativi XML Cartridge, Shimura e Monet XML, che estendono rispettivamente i DBMS relazionali Oracle, PostgreSQL e Monet con il supporto per la gestione di documenti XML. Ci sono anche progetti open source che forniscono estensioni di immagazzinamento strutturato per i DBMS tradizionali. Nell'analisi è stato incluso il progetto open source ozone/XML. ozone/XML è una libreria di classi persistenti per il DBMS object-oriented ozone che implementa lo standard DOM per la rappresentazione di documenti XML in un database.

Infine, nel terzo approccio per rappresentare documenti XML nei DBMS tradizionali, che viene chiamato *mapping*, il contenuto dei documenti XML è mappato su schemi di database progettati specificamente per quel contenuto. Questo, in principio, mette a disposizione tutte le capacità di modellazione disponibili con un database tradizionale per rappresentare efficientemente e adeguatamente il contenuto del documento. Esiste un grande numero di strumenti e formalismi per la specificazione del mapping tra un formato XML ed uno schema di database. Per esempio, IBM DB2 XML Extender e Microsoft SQLXML forniscono le definizioni di accesso al documento (DAD, Document Access Definitions) e gli schemi annotati, rispettivamente, per questo scopo.

Tuttavia, tali strumenti e formalismi sono stati tralasciati per l'analisi. Considerando che MPEG-7 consente la definizione di schemi di descrizione multimediali superiori a quelli predefiniti con lo standard, lo sforzo necessario per far fronte ad una descrizione multimediale susseguente ad un precedente schema di descrizione sconosciuto sarebbe proibitivo.

Recentemente, c'è stata una notevole ricerca riguardante la derivazione automatica degli schemi di database relazionali da definizioni di schema per documenti XML ed il mapping automatico tra loro. Siccome questi approcci sono basati su Document Type Definitions (DTD) e non su definizioni di schema scritte nel più complesso MPEG-7 DDL, essi non si qualificano per le descrizioni multimediali MPEG-7. Comunque, nell'ultima release dell'estensione database Oracle XML DB, Oracle ha esteso l'approccio base per l'XML Schema per fornire un'opzione addizionale di immagazzinamento per i documenti XML oltre all'immagazzinamento non strutturato. Oracle chiama la sua opzione di immagazzinamento *Structured Mapping*. Giacché l'XML Schema costituisce un largo sottoinsieme del MPEG-7 DDL, si è tenuto in conto anche di Oracle XML DB/Structured Mapping per l'analisi.

2.3 Analisi

Data la sua vasta applicabilità, la sua innata estensibilità e la ricchezza di schemi predefiniti di descrizione dei media, ci si può aspettare che MPEG-7 possa essere utilizzato diffusamente in una grande varietà di applicazioni multimediali future. Presto ci sarà bisogno di soluzioni adeguate per la gestione delle *media descriptions* MPEG-7 che sono documenti XML che aderiscono alle definizioni dello schema scritte in MPEG-7 DDL. Tenendo conto della mancanza di un modello concettuale indipendente dal XML per le media descriptions MPEG-7 e delle difficoltà di ricavare un tal modello, il problema della loro gestione costituisce in generale un problema di controllo dei documenti XML. Quindi, l'idea di impiegare soluzioni database XML è a portata di mano.

In questa sezione, viene analizzato dettagliatamente in che misura le soluzioni database XML introdotte sono adatte alla gestione delle media descriptions MPEG-7. Per questo scopo, vengono valutate le soluzioni database in relazione a certi requisiti considerati importanti riguardo alla gestione delle media descriptions MPEG-7 e, di conseguenza, viene esaminata la soddisfazione degli stessi riguardo alla rappresentazione, all'accesso e agli schemi delle media descriptions, all'estensibilità ed alle funzionalità classiche DBMS.

La sezione viene conclusa con un sommario dei risultati ottenuti. Per un orientamento migliore durante le discussioni seguenti, si possono osservare i risultati dell'analisi nella figura 6.

2.3.1 Rappresentazione delle media descriptions

Le caratteristiche di base delle media descriptions MPEG-7 influenzano considerevolmente il modo in cui esse dovrebbero essere rappresentate in una soluzione database XML. La natura di MPEG-7 richiede specialmente la rappresentazione a grana fine (*fine-grained representation*) della struttura della media description così come la rappresentazione dei diversi tipi di dato (*typed representation*) del suo contenuto di base. Inoltre, potrebbe essere desiderabile per alcune applicazioni che la rappresentazione delle media description in un database di XML consideri la gestione delle versioni (*versioning)*.

- ***Rappresentazione a grana fine***

Le soluzioni database XML native rappresentano tipicamente i documenti XML, e quindi anche le media descriptions MPEG-7, in una maniera a grana fine. Per l'immagazzinamento, decompongono internamente un documento XML nei diversi nodi di cui è costituito. eXcelon XIS, Infonyte-DB, PDOM, X-Hive/DB, Xindice, dbXML ed eXist rappresentano questi nodi secondo il DOM Structure Model. GoXML DB impiega XQuery 1.0 e XPath 2.0 Data Model. I prototipi di ricerca Lore, Natix e TIMBER definiscono loro propri modelli non standard basati su alberi etichettati sul bordo per la rappresentazione interna dei documenti XML.

Di contro, fondamentalmente le soluzioni native Tamino e TEXTML memorizzano i documenti XML nella loro interezza nel loro formato testuale originale, applicando solo tecniche di compressione per ridurre il consumo di spazio di immagazzinamento. Ciò è paragonabile alle estensioni database IBM DB2 XML Extender, Microsoft SQLXML e Oracle XML DB, che pure memorizzano i documenti XML nei CLOB in maniera non strutturata. Tamino allevia un po' gli svantaggi di tale rappresentazione di immagazzinamento a grana grezza controllando informazioni strutturali a grana fine supplementari: una struttura indice mantiene le informazioni del percorso per i diversi elementi e i valori di attributo presenti in un documento.

Poiché le estensioni database XML che seguono l'approccio di immagazzinamento strutturato applicano le primitive di modellazione del DBMS sottostante per la definizione di metamodelli dettagliati per i documenti XML, viene offerta una rappresentazione a grana fine di immagazzinamento delle MPEG-7 media descriptions per costruzione. L'estensione database orientata agli oggetti ozone/XML è una libreria di classi persistenti che costituisce un'implementazione 1:1 del DOM Structure Model e dell'API; le estensioni relazionali Shimura e XML Cartridge rappresentano i documenti di XML attraverso alberi etichettati sul bordo che essi memorizzano in una tabella centrale; Monet XML genera dinamicamente una tabella per ogni percorso distinto da cui un elemento o un valore di attributo, presente in un documento XML contenuto nel database, può essere raggiunto dalla radice del documento rispettivo. Ogni elemento e valore di attributo di un documento è quindi memorizzato nella tabella generata per il percorso corrispondente.

Anche l'estensione database Oracle XML DB/Structured Mapping in teoria facilita la memorizzazione a grana fine dei documenti XML. Lo schema di mapping del sistema genera essenzialmente un tipo di oggetto SQL per ogni tipo complesso dichiarato in una definizione di schema scritta in XML Schema. Un tale tipo di oggetto SQL ha un campo per ogni tipo di elemento ed attributo del quale è costituito il corrispondente tipo complesso. Nel caso in cui un campo corrisponda ad un tipo di elemento o ad un attributo con contenuto semplice, viene assegnato il tipo di dati atomico SQL che è più vicino al tipo semplice che definisce il contenuto rispettivo. Nel caso in cui un campo corrisponda ad un tipo di elemento con contenuto complesso, viene assegnato il tipo di oggetto di SQL generato per il tipo complesso che definisce il contenuto di quel tipo di elemento. Le occorrenze multiple dei tipi di elemento sono trattate dai campi di tipo *collection*. Come punto di entrata a questa struttura, viene generata una tabella per ogni tipo di elemento radice con una singola colonna che è assegnata al tipo SQL di dati o di oggetto adatti per il contenuto del tipo di elemento radice. Per immagazzinare un documento XML che aderisce alla data definizione dello schema usando questa rappresentazione di immagazzinamento, Oracle XML DB/Structured mappa i contenuti di quel documento in una maniera a grana fine nelle colonne e nei campi corrispondenti delle tabelle e nei tipi di oggetto SQL generati.

Quando passa all'immagazzinamento delle media descriptions MPEG-7, tuttavia, questo schema di mapping rivela considerevoli limitazioni: gli elementi con contenuto misto e gli elementi con contenuto arbitrario non possono essere rappresentati, così come gli elementi che usano l'attributo di *xsi:type* per segnalare il polimorfismo di tipo complesso.

Oracle XML DB/Structured Mapping aggira queste limitazioni aumentando i tipi di oggetto SQL con campi CLOB supplementari che servono come depositi di sovrabbondanza (*overflow stores*) che mantengono quelle porzioni di documenti XML che non possono essere rappresentati testualmente in maniera non strutturata in altro modo. Ma poiché MPEG-7 fa un uso considerevole dei costrutti problematici menzionati in precedenza (e particolarmente del polimorfismo di tipo complesso), ci si può aspettare che parti significative delle tipiche media descriptions MPEG-7 siano mantenute in tali depositi di sovrabbondanza. Ciò riduce considerevolmente la granularità della rappresentazione di immagazzinamento per le media descriptions MPEG-7.

- ***Rappresentazione di diversi tipi di dato (typed)***

Le soluzioni database XML studiate hanno forti mancanze riguardo alla rappresentazione dell'informazione non testuale che contribuisce significativamente al contenuto delle media descriptions MPEG-7. La maggior parte di loro memorizza il contenuto base di un documento XML, cioè il semplice contenuto di elemento ed il contenuto dei valori di attributo, come testo senza riguardo al particolare tipo di contenuto.

Infatti, soltanto alcune soluzioni trattano anche la questione della rappresentazione typed dei contenuti di base del documento. La soluzione database nativa GoXML DB, poiché impiega XQuery 1.0 e XPath 2.0 Data Model per la memorizzazione dei documenti XML, potrebbe in teoria sfruttare le possibilità del modello per quanto riguarda la rappresentazione del contenuto semplice di elemento e del contenuto dei valori di attributo. Gli esperimenti con la versione 2.0.2 hanno rivelato, tuttavia, che questa caratteristica del modello non è stata apparentemente implementata: tale contenuto viene sempre immagazzinato e trattato come testo. La soluzione database nativa Lore così come l'estensione database XML Cartridge ha implementato rappresentazioni typed fino ad un certo punto: per la rappresentazione del contenuto dei valori di attributo, Lore fornisce un insieme di tipi di dati atomici compresi integer, real e string. Il contenuto semplice di elemento, tuttavia, è rappresentato sempre come testo. XML Cartridge mantiene i contenuti di base di un documento in tabelle foglia specifiche che usano i tipi di dati atomici di SQL forniti dal DBMS Oracle sottostante per memorizzare i valori, con una tabella foglia per ogni tipo di dato supportato.

Se questo supporto per le rappresentazioni typed conduce ad un adeguato immagazzinamento dei contenuti di base delle media descriptions MPEG-7, tuttavia, è perlomeno discutibile.

Indipendentemente dal fatto che i tipi di dati supportati da Lore e da XML Cartridge costituiscano soltanto un sottoinsieme limitato dei tipi semplici che sono predefiniti con MPEG-7 DDL, tutti i sistemi non supportano i metodi di derivazione del tipo semplice forniti con MPEG-7 DDL. Tra le altre cose, non possono dunque rappresentare adeguatamente le liste e le matrici che sono presenti comunemente nelle media descriptions MPEG-7. Inoltre, rimane poco chiaro come Lore e XML Cartridge siano ritenuti realmente capaci di generare rappresentazioni di diversi tipi di dato, poiché non prendono in considerazione le definizioni dello schema per la memorizzazione dei documenti XML che contengono le informazioni necessarie per quello scopo.

Tra le soluzioni database studiate, Oracle XML DB/ Structured Mapping è l'unico che fornisce finora il maggior supporto per la rappresentazione typed dei contenuti di base di una MPEG-7 media description. Secondo lo schema di mapping spiegato sopra, il contenuto semplice di elemento ed il contenuto dei valori di attributo di un documento XML sono mantenuti nei campi dei tipi dell'oggetto di SQL che usano il tipo di dato atomico di SQL che meglio soddisfa il particolare tipo di contenuto fornito tramite la definizione dello schema associata al documento.

Tuttavia, ci sono ancora limitazioni considerevoli. Da un lato, lo schema di mapping non supporta molti metodi di derivazione di tipo semplice forniti con MPEG-7 DDL: tra le altre cose, le liste sono mappate semplicemente nei campi di testo e non è supportato affatto il mapping delle matrici poiché Oracle XML DB/Structured Mapping non può fare fronte alle estensioni di MPEG-7 DDL all'XML Schema. Dall'altro lato, la rappresentazione del contenuto semplice di elemento o del contenuto di un valore di attributo dipende a seconda che il rispettivo elemento o valore di attributo venga mantenuto in un deposito di sovrabbondanza testuale oppure no, perché si presenta in un elemento con il contenuto misto o arbitrario o in un elemento che impiega il polimorfismo del tipo complesso attraverso l'attributo xsi:type. È probabile che questi fattori risultino nella rappresentazione testuale di grandi parti dei dati non testuali trasportati dalle media descriptions MPEG-7.

- ***Gestione delle versioni (versioning)***

Fra le soluzioni database XML trattate dall'analisi, soltanto X-Hive/DB, Oracle XML DB e Oracle XML DB/Structured Mapping mettono in evidenza meccanismi pronti all'uso che possono essere direttamente impiegati dalle applicazioni per costituire una gestione delle versioni per le media descriptions MPEG-7. Con la soluzione database nativa X-Hive/DB, un documento XML memorizzato nel database può essere contrassegnato come modificabile (*versionable*). Una volta che è stato fatto questo, viene applicato un meccanismo di verifica per i documenti XML. Se un documento XML precedentemente verificato è controllato di nuovo dentro al database, viene generata una nuova versione del documento. Siccome con il meccanismo di verifica nuove versioni di un documento XML possono essere derivate dalle più vecchie versioni del documento e non soltanto da quella più nuova, diventa possibile instaurare alberi di versione. Di conseguenza, X-Hive/DB fornisce un modello di versioning ramificato (*versioning with branching*). X-Hive/DB fornisce una API con cui le applicazioni possono navigare fra le differenti versioni di un documento.

Le estensioni database relazionali Oracle XML DB e Oracle XML DB/Structured Mapping hanno un componente supplementare denominato *Oracle XML DB Repository* che permette di organizzare i documenti XML gerarchicamente in una struttura a cartella. Similmente a X-Hive/DB, Oracle XML DB Repository permette che un documento XML venga contrassegnato come versionable e fornisce un meccanismo di verifica con cui possono essere generate nuove versioni di un documento. Proprio come X-Hive/DB, Oracle XML DB Repository supporta il versioning ramificato. Inoltre ha parecchie funzioni per attraversare le relazioni di predecessore e successore fra le differenti versioni dall'interno delle query SQL.

2.3.2 Accesso alle media descriptions

Un servizio fondamentale di una soluzione database XML usata per immagazzinare MPEG-7 media descriptions è quello di fornire alle applicazioni i mezzi sufficienti per l'accesso a tali descrizioni. Una soluzione adatta dovrebbe permettere l'accesso a grana fine (*fine-grained access*) alla struttura di una media description, l'accesso (*typed access*) al contenuto di base trasportato dalla descrizione e aggiornamenti a grana fine (*fine-grained updates*) delle media descriptions. Inoltre, viene considerato un requisito importante la disponibilità di sofisticate strutture di indice che permettano accesso efficiente anche alle ampie raccolte di media descriptions.

Viene specialmente richiesto che una soluzione database XML adatta debba fornire potenti strutture di indice di valore (*value index structures*), strutture di indice del testo (*text index structures*) e strutture di indice di percorso (*path index structures*).

- ***Accesso a grana fine***

Con l'eccezione di TEXTML che permette soltanto il recupero di documenti XML completi, tutte le soluzioni database XML trattate dall'analisi forniscono mezzi che danno alle applicazioni l'accesso a grana fine ai componenti delle media descriptions MPEG-7. Molte soluzioni offrono interfacce di programmazione di navigazione che possono essere usate per programmare attraversamenti a grana fine della struttura delle media descriptions. Le soluzioni native eXcelon XIS, Infonyte-DB, PDOM e X-Hive/DB così come le estensioni database Oracle XML DB, ozone/XML e Oracle XML DB/Structured Mapping offrono tutte implementazioni del DOM API. Natix offre un file system driver che dà una vista a file system sulla struttura gerarchica dei documenti XML: il contenuto del documento può così essere attraversato mediante le operazioni di file.

Con le estensioni database XML che seguono l'immagazzinamento strutturato e gli approcci di mapping, le applicazioni possono usare direttamente i meccanismi di query disponibili con il DBMS sottostante l'estensione, per accedere al contenuto delle media descriptions in una maniera a grana fine. Queste estensioni rappresentano esplicitamente il contenuto dei documenti XML con le primitive di modellazione del DBMS. Per esempio, le tabelle a grana fine rappresentanti il contenuto del documento XML con le estensioni database relazionali Monet XML, Shimura, XML Cartridge e Oracle XML DB/Structured Mapping possono essere direttamente interrogate con SQL.

La maggior parte delle soluzioni database esaminate supportano inoltre alcune forme di linguaggio dichiarativo di query XML o di algebra di query che facilitano l'accesso a grana fine alle media descriptions MPEG-7. Le soluzioni native eXcelon XIS, Infonyte-DB, X-Hive/DB, Xindice, dbXML ed eXist sono capaci di valutare le espressioni XPath. Tamino implementa una variante proprietaria, leggermente estesa delle espressioni XPath denominata *X-Query*, che non dovrebbe essere confusa con il lavoro di standardizzazione XQuery, mentre Infonyte-DB e PDOM supportano XQL. Lore supporta il linguaggio espressivo di query Lorel per i dati semistrutturati. GoXML DB implementa un sottoinsieme della bozza di lavoro corrente dello standard XQuery.

Natix e TIMBER definiscono rispettivamente le algebre di query NPA e TAX, abbastanza potenti per rappresentare le espressioni XPath come pure la maggior parte di query esprimibili con XQuery e per fornire traduttori adatti da XPath e da XQuery.

Le estensioni database relazionali IBM DB2 XML Extender, Microsoft SQLXML e Oracle XML DB, che seguono l'approccio di immagazzinamento non strutturato, sono capaci di valutare le varianti delle espressioni XPath sui documenti XML contenuti in un CLOB. A questo fine, offrono funzioni dedicate per l'uso all'interno delle query SQL che prendono come argomenti un CLOB che contiene un documento XML ed un'espressione XPath. Una volta invocate, queste funzioni caricano internamente l'intero documento XML dal CLOB nella memoria centrale, lo analizzano, valutano l'espressione e restituiscono il risultato della valutazione come stringa.

Siccome l'interrogazione dei documenti XML memorizzati con le estensioni database relazionali che seguono gli approcci di immagazzinamento e mapping strutturato per mezzo di SQL può essere complessa ed gravosa, le estensioni database XML Cartridge, Shimura ed Oracle XML DB/Structured Mapping offrono funzioni aggiuntive che prendono le espressioni XPath, le query XQL e le espressioni XPath, rispettivamente, come argomenti e le riscrivono nelle equivalenti dichiarazioni SQL.

- ***Accesso appropriato al particolare tipo di dato (typed)***

È già stato precisato che le soluzioni database XML studiate sono deboli nel rappresentare i contenuti di base delle media descriptions MPEG-7 in maniera adeguata ai diversi tipi di dato. Quindi, non deve sorprendere se le applicazioni abbiano difficoltà nell'accedere con questi sistemi a tali contenuti in maniera adatta al particolare tipo di contenuto. In realtà, quelle soluzioni database XML che immagazzinano il contenuto di base di un documento XML testualmente danno alle applicazioni soltanto l'accesso testuale ai contenuti di base del documento. Per accedere e processare contenuto non testuale in modo ragionevole, le applicazioni devono trasformare manualmente il contenuto in rappresentazioni typed ogni volta che si accede al contenuto, per esempio, effettuando esplicitamente le conversioni di tipo o sfruttando le regole di coercizione di tipo implicito associate con gli operatori di un linguaggio di query quale XPath o XQuery.

Alcune soluzioni database XML che immagazzinano testualmente il contenuto di base di un documento facilitano una forma di accesso appropriata al tipo di dato a tutti quei contenuti per cui sono stati definiti gli indici di valore. Per la definizione di un indice di valore sul contenuto degli elementi o dei valori di attributo, eXcelon XIS, Tamino, TEXTML e Infonyte-DB permettono di specificare se il contenuto debba essere interpretato come stringhe, numeri o date per lo scopo di indicizzazione. Quando un elemento o un valore di attributo indicizzato in tale maniera è raggiunto in una query, esso viene trattato in accordo con il tipo di dato specificato. Questa forma di accesso typed, tuttavia, incontra rapidamente delle limitazioni con MPEG-7: l'insieme dei tipi di dati disponibili per l'indicizzazione di valore non si avvicina neanche alla vasta varietà di tipi semplici predefiniti e metodi di derivazioni di tipo semplice disponibili con MPEG-7 DDL. Per esempio, non è possibile l'accesso typed alle liste ed alle matrici che sono frequentemente presenti nelle media descriptions. Inoltre, il contenuto indicizzato viene ancora recuperato come testo, tipicamente costringendo le applicazioni a trasformare il contenuto non testuale recuperato in rappresentazioni typed più adatte per un ulteriore trattamento interno. Infine, il diffuso accesso typed ai contenuti di base di una media description richiede la definizione di molti indici di valore, che è probabile rallentino le prestazioni di aggiornamento in maniera proibitiva.

Quelle soluzioni database XML che mantengono i contenuti di base di un documento XML nella rappresentazione typed, generalmente possono anche dare alle applicazioni l'accesso typed a questi contenuti. Comunque le limitazioni di queste soluzioni riguardo alla rappresentazione colpiscono direttamente attraverso l'accesso: proprio come è lecito chiedersi come Lore e XML Cartridge possano ottenere le rappresentazioni typed senza impiegare l'informazione del tipo contenuta nelle definizioni dello schema, rimane poco chiaro come entrambi i sistemi possano in pratica attuare l'accesso typed al contenuto di base dei documenti di XML. Similmente, a causa delle mancanze già accennate dello schema di mapping applicato, non sarà sempre possibile con Oracle XML DB/Structured Mapping accedere ai contenuti di base delle MPEG-7 media descriptions in maniera typed da dentro le query SQL (o le espressioni XPath che sono state riscritte in query SQL), a seconda che il contenuto particolare sia di un tipo che è sempre rappresentato testualmente quale una lista oppure che il contenuto sia mantenuto in un deposito testuale di sovrabbondanza dovuto al polimorfismo di tipo complesso, al contenuto misto, ecc.

- ***Aggiornamenti a grana fine***

Molte delle soluzioni database XML studiate nell'analisi non solo forniscono meccanismi che danno alle applicazioni l'accesso di lettura a grana fine alle media descriptions MPEG-7 con loro immagazzinate; in parecchi casi, essi consentono anche aggiornamenti a grana fine delle media descriptions. Strumenti comuni offerti dai sistemi esaminati a questo scopo sono le API di navigazione che permettono di programmare gli aggiornamenti a grana fine dei documenti XML. Le soluzioni database native eXcelon XIS, Infonyte-DB, PDOM e X-Hive/DB così come le estensioni database ozone/XML e Oracle XML DB/Structured Mapping implementano tutte i metodi relativi all'aggiornamento del DOM API; l'interfaccia a file system offerta da Natix permette aggiornamenti a grana fine anche per mezzo di operazioni di file.

Poiché le estensioni database che seguono gli approcci di immagazzinamento e mapping strutturato rappresentano dettagliatamente il contenuto delle media descriptions MPEG-7 con il modello di dati del DBMS esteso, essi possono conseguentemente impiegare i meccanismi di aggiornamento disponibili con il DBMS per realizzare modifiche a grana fine delle media descriptions: le estensioni database relazionali Monet XML, Shimura, XML Cartridge e Oracle XML DB/Structured Mapping permettono aggiornamenti a grana fine dei documenti XML usando dichiarazioni SQL UPDATE adatte.

Oltre a questi meccanismi di aggiornamento, molte soluzioni database native permettono aggiornamenti a grana fine delle media descriptions attraverso linguaggi di aggiornamento XML dedicati. eXcelon XIS supporta il proprietario Updategrams; GoXML DB estende XQuery con le operazioni di aggiornamento. Le soluzioni database Xindice e dbXML offrono implementazioni del linguaggio XUpdate. Lorel, il linguaggio di query di Lore, e TAX, l'algebra di query di TIMBER, includono entrambi operazioni per effettuare aggiornamenti a grana fine. Siccome gli aggiornamenti dei documenti XML possono provocare complesse dichiarazioni SQL UPDATE, l'estensione database relazionale Oracle XML DB/Structured Mapping offre una funzione che riscrive le richieste di aggiornamento specificate in un semplice linguaggio di aggiornamento XML basato sulle espressioni XPath per le dichiarazioni semanticamente equivalenti di UPDATE. Di contro, le soluzioni database native Tamino, TEXTML ed eXist non permettono gli aggiornamenti a grana fine delle media descriptions MPEG-7. Con questi sistemi, può essere effettuato un aggiornamento di una piccola frazione di un documento XML solo sostituendo il documento completo con una versione aggiornata.

Ciò è anche il caso delle estensioni database IBM DB2 XML Extender, Microsoft SQLXML e Oracle XML DB che memorizzano in maniera non strutturata i documenti XML nei CLOB.

- ***Strutture di indice di valore***

Il supporto per l'indicizzazione di valore offerto dalle soluzioni database XML trattate dall'analisi non è in generale sufficiente per l'indicizzazione ragionevole delle media descriptions MPEG-7. Le soluzioni database XML native esaminate presentano tipicamente strutture di indice di valore ordinate, basate su BTree e unidimensionali che possono essere impiegate per l'indicizzazione del contenuto semplice di elemento e del contenuto dei valori di attributo dei documenti XML. Eccezioni a questo sono PDOM, eXist e Natix, che non forniscono affatto strutture di indice di valore. Per l'indicizzazione efficace delle media descriptions MPEG-7, tuttavia, le possibilità delle soluzioni database native sono limitate: nessuna delle soluzioni native esaminate offre strutture multidimensionali di indice quali RTree che sono certamente desiderabili per l'indicizzazione di dati multimediali complessi contenuti frequentemente nelle media descriptions.

Le estensioni database XML possono contare sulle strutture di indice di valore già fornite dal DBMS esteso. Estensioni database che implementano gli approcci di immagazzinamento e di mapping strutturato per la rappresentazione dei documenti possono usare direttamente queste strutture di indice, perché rappresentano esplicitamente i documenti XML ed i loro contenuti di base con le primitive di modellazione del DBMS sottostante. Fra le estensioni database studiate, questo non si applica solamente a ozone/XML poiché il DBMS ozone non offre alcuna struttura di indice di valore.

Per quanto riguarda le estensioni database che seguono l'approccio di immagazzinamento non strutturato per la rappresentazione di documenti XML, sorge la questione su come le strutture di indice di valore del DBMS esteso possano essere applicate alle media descriptions MPEG-7 che si trovano all'interno dei CLOB. Infatti, questo non è affatto possibile con Microsoft SQLXML. IBM DB2 XML Extender permette di mappare il contenuto di un elemento o di un valore di attributo di un documento XML contenuto in un CLOB nelle colonne della *side table* sincronizzata, che possono poi essere indicizzate con le strutture di indice di valore di DB2 proprio come qualunque altra colonna.

Oracle XML DB supporta l'indicizzazione dei contenuti di base del documento mantenuti in un CLOB per mezzo di indici funzionali basati sulle funzioni per la valutazione di XPath offerte dal sistema.

Sebbene la maggior parte delle estensioni database XML permette l'indicizzazione unidimensionale di valore dei contenuti di base delle media descriptions MPEG-7, esse non raggiungono lo scopo di un'indicizzazione ragionevole dei dati multidimensionali complessi trasportati comunemente da tali descrizioni, proprio come le soluzioni database XML native esaminate. Anche se parecchi dei sottostanti DBMS offrono strutture multidimensionali di indice di valore quali le strutture di indice di griglia e di indice R-Tree che potrebbero in linea di principio essere sfruttate per quello scopo, è lecito chiedersi se le strutture di indice e le estensioni database multidimensionali si adattino insieme. Per esempio, le strutture di indice di griglia e di indice R-Tree trasportate con IBM DB2 e Oracle possono indicizzare soltanto le colonne di tabella alle quali sono assegnate tipi di dati geometrici specifici per la rappresentazione delle forme bidimensionali. Non è chiaro come le istanze di questi tipi di dati geometrici possano essere ottenute dai dati multimediali complessi quali ad esempio gli intervalli di melodia contenuti nelle media descriptions MPEG-7, immagazzinate con una delle estensioni database XML basate su IBM DB2 ed Oracle, così che i dati siano indicizzati efficacemente.

Allo stesso modo, PostgreSQL offre una struttura di indice R-Tree che può indicizzare colonne di qualunque tipo di dati finché il tipo di dati effettua un insieme specifico di operazioni. Ma l'estensione database XML Shimura che è basata su PostgreSQL non può trarre beneficio da questa struttura di indice: Shimura immagazzina tutto il contenuto semplice di elemento ed il contenuto dei valori di attributo come testo in una singola colonna di stringa. Rimane in dubbio se l'insieme di operazioni richieste dalla struttura di indice R-Tree possa essere implementato per le stringhe in modo tale che il contenuto di tutti gli elementi e i valori di attributo sia ragionevolmente indicizzabile.

- ***Strutture di indice di testo***

Parecchie soluzioni database XML offrono strutture di indice di testo dedicate che possono essere sfruttate dalle applicazioni per la realizzazione del recupero efficiente del testo sulle media descriptions MPEG-7.

Fra le soluzioni database native, X-Hive/DB e Natix forniscono una struttura di indice che permette l'indicizzazione *full-text* di interi documenti XML mentre eXcelon XIS, GoXML DB, Tamino, TEXTML, eXist, Lore e TIMBER possiedono strutture di indice che permettono di indicizzare testualmente il contenuto dei singoli elementi e dei valori di attributo ad ogni livello in un documento XML. Le soluzioni native Infonyte-DB, PDOM, Xindice e dbXML, diversamente, non offrono alcuna struttura speciale per l'indicizzazione del testo.

Come con le strutture di indice di valore, le estensioni database XML possono trarre beneficio dalle esistenti strutture di indice del testo del DBMS sottostante per la realizzazione del recupero efficiente del testo sulle media descriptions MPEG-7. Poiché ci sono strutture di indice del testo disponibili per i DBMS relazionali IBM DB2, Microsoft SQL Server e Oracle che possono essere applicate ai CLOB ed alle colonne del carattere, le estensioni database XML IBM DB2 XML Extender, Microsoft SQLXML e Oracle XML DB, che mantengono i documenti XML nel loro formato testuale originale nei CLOB, possono impiegare direttamente queste strutture per l'indicizzazione full-text di interi documenti XML. Anche le estensioni database XML Cartridge e Oracle XML DB/Structured Mapping fondati sul DBMS Oracle possono sfruttare le strutture di indice di testo disponibili: tale struttura può essere applicata alle rispettive colonne della tabella foglia e ai campi di tipo oggetto SQL in cui entrambe le estensioni memorizzano il contenuto semplice di elemento e i valori di attributo testuali in modalità a grana fine. Le estensioni database ozone/XML, Monet XML e Shimura, tuttavia, non supportano il recupero del testo con strutture di indice dedicate: esse non implementano strutture di indice del testo e neanche i sottostanti DBMS ozone, Monet e PostgreSQL forniscono alcuna struttura speciale di indice del testo.

❖ *Strutture di indice di percorso*

Parecchie delle soluzioni database XML studiate offrono strutture di indice che possono accelerare in vari modi l'attraversamento dei percorsi di accesso all'interno delle media descriptions MPEG-7. Alcune, quali la struttura indice offerta da eXcelon XIS, pre-elabora e mantiene i risultati di valutazione di una data espressione di percorso su tutti i documenti XML in un database. Ogni volta che un'applicazione deve valutare l'espressione di percorso indicizzata, può direttamente contare su questi risultati pre-elaborati invece di dover valutare l'espressione stessa.

Un altro gruppo di strutture di indice di percorso offerte da parecchie soluzioni database mantiene i percorsi di accesso dai quali i diversi elementi e i valori di attributo contenuti nei documenti XML di un database possono essere raggiunti dalle rispettive radici del documento. Questo genere di informazioni di percorso può essere usato per eliminare i documenti o le parti dei documenti che certamente saranno tralasciati da una data traversata di percorso. Questa è l'intenzione della struttura di indice offerta dalla soluzione database nativa Tamino che aiuta a scartare i documenti XML irrilevanti durante la valutazione di un'espressione X-Query. La struttura di indice di firma supportata da Infonyte-DB e da PDOM può persino effettuare questo genere di operazione per parti di documenti XML.

Le informazioni del percorso di accesso per i diversi elementi e i valori di attributo dei documenti XML possono anche essere sfruttate per accelerare il calcolo dei risultati di attraversamento. I fautori di questo metodo sono le estensioni database relazionali Shimura, XML Cartridge e Monet XML. Questi sistemi mantengono tabelle di percorso aggiuntive che aumentano gli elementi ed i valori di attributo mantenuti nelle tabelle *edge tables* con una rappresentazione testuale dei percorsi di accesso da cui essi possono essere raggiunti dalle loro corrispondenti radici del documento. Quindi, persino attraversamenti complessi di percorso possono essere ricondotti a relativamente semplici operazioni di corrispondenza di stringhe sulla tabella di percorso. Monet XML supporta intrinsecamente l'indicizzazione di percorso per costruzione perché immagazzina, come spiegato prima, elementi e valori di attributo in tabelle specifiche generate per il percorso di accesso, da cui essi possono essere raggiunti dalle rispettive radici del documento. Così l'attraversamento di un percorso di accesso consiste essenzialmente nel selezionare le righe della tabella che rappresenta quel percorso.

Ancora un altro approccio all'indicizzazione di percorso supportato da alcune delle soluzioni database XML studiate è basato sul fatto di mantenere sommari schematici dei contenuti dei documenti XML in un database. Tali sommari schematici possono essere così semplici come un indice di nomi di tipo di elemento che è disponibile con le soluzioni database native X-Hive/DB e TIMBER. Dato il nome di un tipo d'elemento, un indice di nomi di tipo di elemento raccoglie tutti quegli elementi che sono presenti nel database i cui tipi portano quel nome. Un'applicazione interessata agli elementi di certo tipo può sfruttare direttamente un tale indice di nomi di tipo di elemento per ottenere tutti gli elementi di interesse invece di dover accedere ed attraversare tutti i documenti per cercare questi stessi elementi. DataGuides, offerto dalla soluzione database nativa Lore, costituisce uno sommario schematico più complesso dei contenuti del documento.

DataGuide è un grafico che ricapitola tutti i percorsi di accesso che sono possibili nei documenti XML contenuti in un database e che aiuta non soltanto a decidere se un attraversamento di percorso può produrre un risultato per ognuno dei documenti contenuti in un database. Inoltre, poiché DataGuide fa riferimento a tutti gli elementi o valori di attributo che possono essere raggiunti attraversando un dato percorso, gli attraversamenti di percorso possono essere effettuati in larga misura sul DataGuide stesso evitando la necessità di accedere ai documenti nel database.

Infine, alcune delle soluzioni database esaminate rappresentano strutture di indice full-text che possono anche svolgere il ruolo di strutture di indice di percorso. Questo è il caso di Natix, la cui la struttura di indice full-text in combinazione alle estese Access Support Relations può accelerare gli attraversamenti di percorso, così come di Oracle XML, la cui struttura di indice full-text può contribuire a filtrare fuori i documenti XML che sono irrilevanti per la valutazione di un'espressione XPath.

2.3.3 Schemi delle media descriptions

I media description schemes che specificano le forme permesse delle media description sono un concetto centrale dello standard MPEG-7. Una soluzione database XML che gestisce le MPEG-7 media description dovrebbe quindi fornire un catalogo di schema compatibile con MPEG-7-DDL (*MPEG-7-DDL-compliant schema catalog*) per l'immagazzinamento degli schemi delle media descriptions. Sulla base di questo catalogo, una soluzione adatta dovrebbe sfruttare gli schemi di descrizione per la convalida delle media descriptions (*validation of media descriptions*), per l'inferenza delle rappresentazioni dei diversi tipi di dato (*inference of typed representations*) e per l'ottimizzazione dell'accesso (*access optimization*).

- ***Catalogo di schema compatibile con MPEG-7-DDL***

Nessuna delle soluzioni database XML considerate nell'analisi fornisce un catalogo di schema che è completamente compatibile con MPEG-7 DDL. Infatti, la maggior parte delle soluzioni studiate, vale a dire Infonyte-DB, PDOM, TEXTML, Xindice, dbXML, eXist, Lore, Natix, TIMBER, Microsoft SQLXML, ozone/XML, Monet XML, Shimura e XML Cartridge, non mantengono affatto cataloghi di schema. Essi non usano le informazioni disponibili dello schema per la memorizzazione dei documenti XML.

Altre soluzioni database, quali XHive/DB ed IBM DB2 XML Extender, mantengono cataloghi di schema basati su DTD. Tuttavia, i cataloghi di schema basati su DTD non possono in generale essere ritenuti adatti per la gestione degli schemi delle media descriptions MPEG-7.

Siccome XML Schema costituisce un grande sottoinsieme di MPEG-7 DDL, le soluzioni database che implementano cataloghi dei schema compatibili con XML-Schema promettono almeno il sostegno parziale per la gestione dei media descriptions schemes MPEG-7. eXcelon XIS, Oracle XML e Oracle XML DB/Structured Mapping forniscono cataloghi di schema che più o meno supportano lo standard XML Schema e quindi ci si aspetta che facciano fronte a molti media description schemes MPEG-7. Contrariamente, i cataloghi di schema di Tamino e di GoXML DB implementano sottoinsiemi molto limitati del XML Schema che non vanno oltre i DTD, i quali limitano significativamente la loro capacità di controllare tipici media description schemes MPEG-7.

- ***Convalida delle descrizioni di media***

Tutte le soluzioni database esaminate che mantengono un catalogo di schema per i documenti XML contenuti in un database impiegano le informazioni dello schema disponibili con il catalogo per la convalida del documento. Fino al grado di MPEG-7 DDL/XML Schema supportato, le soluzioni database eXcelon XIS, GoXML DB, Tamino, Oracle XML DB ed Oracle XML DB/Structured Mapping possono così convalidare le media descriptions MPEG-7.

Solitamente queste soluzioni eseguono automaticamente la convalida dei documenti XML rispetto la loro definizione di schema durante l'importazione del documento. Soltanto le media descriptions MPEG-7 che sono consistenti con i loro media description schemes possono così essere inserite in un database. Poiché Oracle XML DB e Tamino supportano solo aggiornamenti a grana grossa dei documenti XML e, pertanto, l'aggiornamento di un documento è equivalente all'importazione di un nuovo documento nel database con questi sistemi, entrambe le soluzioni convalidano automaticamente anche i documenti dopo un aggiornamento. Ciò garantisce che nessun aggiornamento di una media description MPEG-7 possa violare il suo media description scheme. eXcelon XIS, GoXML DB e Oracle XMLDB/Structured Mapping, di contro, non convalidano automaticamente i documenti dopo gli aggiornamenti. Quindi, le operazioni di aggiornamento possono provocare media descriptions MPEG-7 inconsistenti con il loro description scheme.

Per evitare ciò, le applicazioni sono vincolate ad iniziare esplicitamente la convalida delle media descriptions dopo gli aggiornamenti con questi sistemi.

- ***Inferenza delle rappresentazioni dei diversi tipi di dato***

Fuori da tutte le soluzioni database studiate, soltanto Oracle XML DB/Structured Mapping utilizza fino ad un certo punto le informazioni di tipo disponibili nel catalogo di schema per generare rappresentazioni typed dei contenuti di base delle media descriptions MPEG-7. Come è stato spiegato prima, lo schema di mapping di Oracle XML DB/Structured Mapping traduce una definizione dello schema scritta nel XML Schema in uno schema di database relazionale che consiste di una serie di tipi di oggetto SQL che hanno campi typed appositamente per immagazzinare il contenuto semplice di elemento ed il contenuto dei valori di attributo. Nel mappare un documento XML in uno schema relazionale generato in un tal modo, i contenuti di base del documento sono portati automaticamente dalla rappresentazione testuale a quella typed, una volta che è stato assegnato il campo typed preparato specificamente per il rispettivo contenuto. Tuttavia, l'abilità di Oracle XML DB/Structured Mapping di produrre rappresentazioni typed per le media descriptions MPEG-7 è limitata: si ricordi, per esempio, che le liste sono assegnate generalmente ai campi testuali e che gli elementi con contenuto misto ed arbitrario così come gli elementi che usano il polimorfismo di tipo complesso sono mantenuti in depositi di sovrabbondanza testuali. Le altre soluzioni database XML che fino ad un certo punto supportano la rappresentazione typed del contenuto semplice di elemento e del contenuto dei valori di attributo, vale a dire, Lore e XML Cartridge, non mantengono cataloghi di schema. Poiché quindi non hanno a loro disposizione le informazioni di tipo necessarie, rimane poco chiaro come essi possano inferire i contenuti di base del documento delle rappresentazioni typed.

- ***Ottimizzazione dell'accesso***

Le soluzioni database XML analizzate, soluzioni database native così come le estensioni database, sono generalmente deboli quando occorre sfruttare le informazioni dello schema disponibili all'interno dei media description schemes per l'ottimizzazione degli accessi dell'applicazione alle media descriptions MPEG-7 contenute in un database. Anche se molte soluzioni database presentano strutture di indice che possono considerevolmente accelerare l'accesso, le applicazioni sono solitamente costrette ad usare esplicitamente gli indici esistenti.

Solo per nominare un esempio rappresentativo, il database XML nativo eXcelon XIS si basa sulle applicazioni per formulare le espressioni di XPath in modo tale che sia più che evidente applicare un indice esistente per la valutazione della query. Raramente, i processori della query possono riscrivere query equivalenti basate sulla definizione dello schema del documento XML tali che indici rilevanti siano automaticamente trovati ed applicati.

Inoltre, la maggior parte delle soluzioni database non impiegano definizioni dello schema per ridurre la complessità delle query identificando e tagliando sottoespressioni che sono ridondanti o che non possono mai produrre un risultato. Alcuni sistemi come Infonyte-DB, PDOM e TIMBER riscrivono le query applicando delle regole di trasformazione universalmente valide in base alle informazioni statistiche ed euristiche nella speranza che le query risultanti siano più semplici da valutare e la quantità di risultati intermedi sia ridotta.

Tra le soluzioni esaminate, soltanto Tamino e Oracle XML DB/Structured Mapping offrono tecniche più sofisticate per l'ottimizzazione dell'accesso basate sulla definizione dello schema di un documento XML. Tamino impiega le informazioni dello schema per portare una query X-Query in una forma canonica. Tra le altre, le espressioni di percorso che sono presenti in una query sono sostituite nella forma canonica dalla disgiunzione di tutti quei percorsi che si qualificano per queste espressioni secondo la definizione dello schema. Sulla base di questa forma canonica, è facile per il processore della query decidere quali indici di valore o di testo sono disponibili per la valutazione della query ed applicare la struttura indice per scartare i documenti che non possono contribuire al risultato della query.

Oracle XML DB/Structured Mapping impiega la definizione dello schema per riscrivere un'espressione di XPath che deve essere valutata su un documento XML, rispetto ad una query equivalente di SQL che opera sulle tabelle e sui tipi di oggetto di SQL degli schemi relazionali specificamente generati per la definizione dello schema. La query SQL viene poi passata al processore di query del DBMS Oracle sottostante, che può quindi applicare tutte le tecniche relazionali specializzate di ottimizzazione della query disponibili con il sistema.

2.3.4 Estensibilità

L'estensibilità viene considerata una proprietà desiderabile di una soluzione database XML che gestisce media description MPEG-7. Una soluzione database ideale dovrebbe offrire l'estensibilità con funzionalità (*extensibility with functionality*) ed estensibilità con le strutture di indice (*extensibility with index structures*).

- ***Estensibilità con funzionalità***

Le categorie esaminate di soluzioni database XML, soluzioni native ed estensioni database, differiscono considerevolmente riguardo alla loro estensibilità con funzionalità dedicate ad un'applicazione particolare (*application-specific*). Molte soluzioni database native, cioè, GoXML DB, Infonyte-DB, PDOM, X-Hive/DB, eXist, Lore, Natix e TIMBER, non forniscono mezzi con cui nuove funzioni e procedure personalizzate possano essere integrate in questi sistemi.

Le soluzioni native che possono estendersi con funzioni application-specific sono eXcelon XIS, Xindice, dbXML e Tamino. Una tale estensione funzionale è denominata *DXE Servlet* con eXcelon XIS, *XMLObject* con Xindice e dbXML e *query function* con Tamino. DXE Servlets, XMLObjects e le query function sono realizzate come classi Java; facoltativamente, una funzione di query di Tamino può anche essere implementata come classe COM. Benché molto simili, i concetti di DXE Servlets, XML Objects e query functions differiscono nella maniera in cui esse possono essere applicate per il trattamento dei documenti XML contenuti in un database. Le query function possono essere usate nel linguaggio X-Query di Tamino proprio come ogni funzione predefinita mentre sia DXE Servlets come pure XMLObjects non sono integrati con il linguaggio di query del rispettivo sistema e devono essere invocate esplicitamente attraverso API dedicate.

Le estensioni database XML studiate, in opposizione, possono tutte essere accresciute con funzioni e procedure personalizzate usando i meccanismi per l'estensione funzionale disponibili con il DBMS sottostante. Il DBMS orientato agli oggetti ozone sottostante l'estensione database ozone/XML può essere aumentato con classi arbitrarie che realizzano la funzionalità application-specific; i DBMS relazionali sottostanti le altre estensioni database XML esaminate supportano tutte la definizione di procedure e di funzioni custom immagazzinate. Queste procedure e funzioni personalizzate si adattano perfettamente alle estensioni database XML al livello di SQL dove

possono essere usate nelle query per i documenti XML proprio come funzioni predefinite. Tuttavia, non possono essere usate tipicamente all'interno dei linguaggi di query XML più spesso supportati dalle estensioni database relazionali. Per esempio, le funzioni immagazzinate application-specific non possono essere chiamate all'interno delle espressioni di XPath che sono state eseguite con la funzione di valutazione di XPath sul documento di XML memorizzato nei CLOB usando Oracle XML DB.

- ***Estensibilità con strutture di indice***

La maggior parte delle soluzioni database XML considerate per l'analisi non offre interfacce dedicate con cui nuove strutture di indice possano essere integrate, se desiderato da un'applicazione. Invece, le soluzioni database si presentano solitamente agli sviluppatori di applicazione come sistemi monolitici. Tra le soluzioni database XML native studiate, soltanto X-Hive/DB offre un'interfaccia aperta per l'integrazione di nuove strutture di indice nel sistema. Tuttavia, questa interfaccia è limitata soltanto all'integrazione di strutture di indice full-text.

Per le estensioni database XML, la questione di integrare una nuova struttura di indice si riduce alla questione se il DBMS sottostante sia estendibile con le nuove strutture di indice. Fra i DBMS IBM DB2, Microsoft SQL Server, Oracle, ozone, Monet e PostgreSQL che costituiscono la base delle soluzioni database XML esaminate, solo Oracle e Monet possono essere estesi con le strutture di indice senza la necessità di modificare il kernel del sistema. Oracle può integrare nuove strutture di indice nel sistema attraverso l'Extensible Indexing API; Monet rappresenta un microkernel che è stato specificamente progettato per essere estendibile con i cosiddetti moduli di estensione che possono, fra le altre cose, contenere le strutture di indice application-specific. Di conseguenza, le soluzioni database XML studiate basate su Oracle e Monet, cioè, Oracle XML DB, Monet XML, XML Cartridge e Oracle XML DB/Structured Mapping, possono pure estendersi con nuove strutture di indice.

2.3.5 Funzionalità classiche DBMS

Ci sono alcune funzionalità DBMS classiche ma pur sempre importanti che al giorno d'oggi la maggior parte delle applicazioni database tengono per certo. Fa parte dello stato dell'arte che i DBMS diano supporto per le transazioni (*transactions*) per garantire un accesso atomico, costante, isolato e durevole al contenuto del database, che impieghino il controllo della concorrenza a grana fine (*fine-grained concurrency control*) per ottenere un'alta concorrenza fra le transazioni, che esercitino un controllo di accesso a grana fine (*fine-grained access control*) per concedere e fare rispettare i diritti di accesso degli utenti ai dati contenuti in un database e che forniscano mezzi sicuri per il salvataggio ed il recupero dei dati (*backup and recovery*) per proteggere le applicazioni dalle transazioni abortite, dagli arresti di sistema e dai guasti dei media. Poiché tali caratteristiche sono desiderabili anche per le applicazioni che funzionano con un database di media descriptions MPEG-7, si richiede che una soluzione database XML adatta fornisca pure le funzionalità classiche DBMS.

Molte delle soluzioni database XML native esaminate hanno mancanze notevoli riguardo alle funzionalità classiche DBMS. Benché le transazioni siano un concetto tradizionale dei DBMS, un sorprendente numero di soluzioni native non offre supporto alla transazione: Infonyte-DB e PDOM evitano la necessità delle transazioni permettendo che un database sia raggiunto soltanto da un singolo processo, quindi realizzando una vista isolata sui contenuti del database nel senso più vero della parola; TEXTML implementa un semplice meccanismo di check-in/check-out per i documenti XML anziché le transazioni; Xindice, dbXML ed eXist non forniscono neppure alcun mezzo per proteggere le applicazioni concorrenti dalle altre. Quelle soluzioni database XML native che implementano il supporto alla transazione, cioè, eXcelon XIS, GoXML DB, Tamino, X-Hive/DB, Lore, Natix e TIMBER, non impiegano necessariamente il controllo di concorrenza a grana fine fra le differenti transazioni che accedono simultaneamente alle media descriptions MPEG-7 immagazzinate. Il meccanismo di controllo di concorrenza di Tamino è basato sui blocchi del documento a grana grossa mentre X-Hive/DB blocca l'intero database durante gli aggiornamenti.

Riguardo al controllo di accesso a grana fine, la situazione è ancora peggiore. Infonyte-DB, PDOM, Xindice, dbXML, eXist, Lore, Natix e TIMBER non implementano alcun genere di controllo di accesso. Di conseguenza, ogni utente può accedere a qualunque media description MPEG-7 immagazzinata con uno di questi sistemi. eXcelon XIS, TEXTML ed XHive/DB realizzano almeno l'autenticazione dell'utente quando un collegamento al database è stato stabilito.

Una volta che l'autenticazione dell'utente ha avuto successo, tuttavia, viene concesso l'accesso completo al database. GoXML DB va un piccolo passo più avanti e permette agli utenti di definire i diritti di accesso per i diversi documenti XML. L'unica soluzione database XML nativa coperta dall'analisi che offre un meccanismo a grana fine di controllo di accesso consentendo la regolazione dell'accesso ai diversi elementi delle media descriptions MPEG-7 è Tamino. Per ogni tipo di elemento mantenuto all'interno del catalogo dello schema di Tamino, i diritti di accesso possono essere definiti controllando l'accesso agli elementi di questo tipo che avvengono all'interno dei documenti XML.

Le soluzioni database XML native analizzate inoltre difettano di maturità riguardo al salvataggio ed al recupero dei dati. Soltanto eXcelon XIS, Tamino, X-Hive/DB, Natix, TIMBER ed eXist supportano il backup e mantengono log a scrittura anticipata (*write-ahead*) per riportare un database andato in crash ad uno stato consistente recente. Ma fuori da queste soluzioni, soltanto X-Hive/DB supporta backup incrementali. Pure GoXML DB, Infonyte-DB, PDOM e Lore offrono mezzi per effettuare backup ma non mantengono log write-ahead per il recupero di uno stato consistente recente. In caso d'emergenza, uno deve tornare di nuovo allo stato dell'ultimo backup. Xindice e dbXML non forniscono affatto alcun supporto al backup e al recovery ma contano su un amministratore del database per effettuare backup manuali, off-line e consistenti del database attraverso la copia di file.

Contrariamente alle soluzioni database XML native, le estensioni database XML studiate sono sviluppate in cima a DBMS tradizionali che offrono tutti implementazioni mature delle funzionalità classiche DBMS. Poiché estendono questi DBMS tradizionali, le estensioni database XML possono direttamente trarre beneficio da queste funzionalità, particolarmente riguardo al supporto di transazione ed al backup e al recovery. Se il controllo di concorrenza a grana fine ed il controllo di accesso a grana fine sono disponibili per l'accesso alle media descriptions MPEG-7 immagazzinate con un'estensione database, tuttavia, dipende dal particolare schema di immagazzinamento per i documenti XML. Siccome IBM DB2 XML Extender, Microsoft SQLXML ed Oracle XML DB immagazzinano i documenti in maniera non strutturata nei CLOB, essi non possono né impiegare il controllo di concorrenza a grana fine né il controllo di accesso a grana fine: i DBMS relazionali sottostanti queste estensioni non possono bloccare o definire i diritti di accesso per le parti dei CLOB.

In confronto, l'estensione database XML ozone che segue l'approccio di immagazzinamento strutturato consente sia il controllo di concorrenza a grana fine che il controllo di accesso a grana fine: essa rappresenta gli elementi ed i valori di attributo di un documento XML come diversi oggetti ed i meccanismi di controllo di concorrenza e di controllo di accesso del DBMS ozone sottostante operano a livello dell'oggetto.

Gli schemi di immagazzinamento per i documenti XML impiegati dalle estensioni database Shimura e XML Cartridge che pure seguono l'approccio di immagazzinamento strutturato rappresentano gli elementi ed i valori di attributo di un documento XML come diverse righe nelle edge tables centrali condivise fra tutti i documenti XML in un database. Poiché i DBMS relazionali sottostanti entrambe le estensioni supportano il bloccaggio a livello di riga ma permettono la definizione dei diritti di accesso soltanto per intere tabelle, è possibile il controllo di concorrenza a grana fine per l'accesso alle media descriptions MPEG-7 immagazzinate con Shimura e XML Cartridge ma nessun controllo di accesso a grana fine.

La rimanente estensione database XML Monet che segue l'approccio di immagazzinamento strutturato non può controllare l'accesso alle media descriptions MPEG-7 che esso memorizza. Inoltre, il controllo di concorrenza realizzato da Monet XML è piuttosto a grana grossa. Ciò è dovuto al fatto che il DBMS Monet sottostante non fornisce alcun mezzo per il controllo di accesso ed è soltanto capace di bloccare tabelle intere del database. Dato lo schema di immagazzinamento applicato da Monet XML, un blocco su una tabella corrisponde ad un blocco su tutti gli elementi o valori di attributo contenuti in ogni documento XML memorizzato nel database che può essere raggiunto dal percorso di accesso che la tabella rappresenta.

Infine, lo schema di immagazzinamento applicato da Oracle XML DB/Structured Mapping seguendo l'approccio di mapping mappa i contenuti dei documenti XML nelle istanze dei tipi di oggetto di SQL che rimodellano dettagliatamente le definizioni dello schema a cui i documenti aderiscono. Poiché il DBMS sottostante Oracle può bloccare in maniera a grana fine e definire i diritti di accesso sugli oggetti di SQL, il controllo di concorrenza a grana fine ed il controllo di accesso possono essere realizzati per l'accesso ai documenti XML memorizzati con questa estensione. Per quanto riguarda l'immagazzinamento delle media descriptions MPEG-7 tipiche, tuttavia, la granularità sia di controllo di concorrenza che di controllo di accesso sarà in pratica considerevolmente ridotta perché parti significative di queste descrizioni potranno essere mantenute in maniera non strutturata nei CLOB che fungono da depositi di sovrabbondanza.

		Native database solutions											**Database extensions**							
		eXcelon XIS	GoXML DB	Infonyte-DB/ PDOM	Tamino	TEXTML	X-Hive/DB	Xindice/ dbXML	eXist	Lore	Natix	TIMBER	IBM DB2 XML Extender	Microsoft SQLXML	Oracle XML DB	ozone/XML	Monet XML	Shimura et al.	XML Cartridge	Oracle XML DB/ Structured Mapping
Representation of media descriptions	Fine-grained representation	■	■	■	□	–	■	■	■	■	■	■	–	–	–	■	■	■	■	□
	Typed representation	–	–	–	–	–	–	–	–	□	–	–	–	–	–	–	–	–	□	□
	Versioning	–	–	–	–	–	■	–	–	–	–	–	–	–	■	–	–	–	–	■
Access to media descriptions	Fine-grained access	■	■	■	■	–	■	■	■	■	■	■	■	■	■	■	■	■	■	■
	Typed access	□	–	–	□	□	–	–	–	□	–	–	–	–	–	–	–	–	□	□
	Fine-grained updates	■	■	■	–	–	■	■	–	■	■	■	–	–	–	■	■	■	■	■
	Value index structures	□	□	–	□	□	□	□	–	□	–	□	□	–	□	–	□	□	□	□
	Text index structures	■	■	–	■	■	■	–	■	■	■	■	■	■	■	–	–	–	■	■
	Path index structures	■	–	■	■	–	■	–	–	■	■	■	–	–	■	–	■	■	■	–
Media description schemes	MPEG-7-DDL-compliant schema catalog	□	□	–	□	–	–	–	–	–	–	–	–	–	□	–	–	–	–	□
	Validation of media description	□	□	–	□	–	–	–	–	–	–	–	–	–	□	–	–	–	–	□
	Inference of typed representations	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	□
	Access optimization	–	–	–	■	–	–	–	–	–	–	–	–	–	–	–	–	–	–	■
Extensibility	Extensibility with functionality	■	–	–	■	–	–	■	–	–	–	–	■	■	■	■	■	■	■	■
	Extensibility with index structure	–	–	–	–	–	□	–	–	–	–	–	–	–	■	–	■	–	■	■
Classic DBMS functionality	Transactions	■	■	–	■	–	■	–	–	■	■	■	■	■	■	■	■	■	■	■
	Fine-grained concurrency control	■	■	–	–	–	–	–	–	■	■	■	–	–	–	■	□	■	■	□
	Fine-grained access control	–	–	–	■	–	–	–	–	–	–	–	–	–	–	■	–	–	–	□
	Backup and recovery	■	□	□	■	□	■	–	■	□	■	■	■	■	■	■	■	■	■	■

Figura 6 - Risultati di analisi (■ supporto, □ supporto parziale, - nessun supporto)

2.4 Conclusioni dell'analisi

Attraverso l'analisi sono state rivelate le mancanze critiche delle soluzioni database XML esistenti per la gestione delle media descriptions MPEG-7. Il punto centrale è che queste soluzioni trascurano in gran parte valide informazioni dello schema e del tipo disponibili con gli schemi di descrizione di media scritti in MPEG-7 DDL per la gestione delle media descriptions MPEG-7. Ciò ha un numero considerevole di conseguenze negative per quanto riguarda la capacità di questi sistemi di convalidare le media descriptions rispetto ai loro schemi di descrizione, per rappresentare il contenuto non-testuale che contribuisce in maniera significativa alle tipiche media descriptions MPEG-7 in un modo adatto al tipo di contenuto, per dare alle applicazioni un ragionevole accesso scritto a tali contenuti non-testuali e per effettuare ottimizzazioni di accesso più specializzate.

Per la gestione delle media descriptions MPEG-7 vi è quindi l'esigenza di una nuova generazione di soluzioni database XML che riconoscano l'importanza centrale dello sfruttamento delle informazioni del tipo contenute nelle definizioni dello schema per la gestione adeguata dei documenti XML. Allo stesso tempo, queste soluzioni non dovrebbero trascurare altre questioni importanti quali le strutture (multidimensionali) specializzate di indice di valore, di testo e di percorso, la profonda estensibilità con funzionalità e strutture di indice personalizzate e le funzionalità classiche DBMS quali transazioni, controllo di accesso e di concorrenza a grana fine e mezzi sicuri per backup ed il recovery.

Sembra comunque che la necessità di usare le definizioni dello schema per realizzare un'adeguata gestione dei documenti XML stia divenendo maggiormente riconosciuta. Come una più nuova soluzione database XML, Oracle XML DB/Structured Mapping già in parte fa uso delle definizioni dello schema scritte in XML Schema per la convalida del documento e per la scrittura dei contenuti di base del documento, così come per l'ottimizzazione della query. Almeno per la gestione delle media descriptions MPEG-7, tuttavia, il sistema deve essere ulteriormente sviluppato per superare le relative limitazioni riguardo ai costrutti più complicati di MPEG-7 DDL/XML Schema per essere considerato più di un portatore di una nuova generazione di soluzioni database XML schema-aware.

3 QUERYING DI DATI MPEG-7

3.1 Classi generiche di query

L'interrogazione dei documenti audiovisivi è una questione che è stata argomento di molti studi di ricerca. Per recuperare dati multimediali, descritti da un modello di metadati, un sistema database deve fornire un linguaggio di query multimediale. Questo linguaggio deve avere la capacità di trattare le query specificando complesse relazioni spaziali e temporali, parole chiave e contenuto oggettivo e soggettivo degli oggetti multimediali. In questo contesto, Gudivada [3] ha definito un insieme di classi generiche di query. Queste classi includono:

1) *Retrieval by browsing* (Recupero attraverso la navigazione): impiegato dagli utenti che non sono pratici della struttura e del contenuto dei MMDBMS. La navigazione è anche usata spesso per l'interrogazione esplorativa (exploratory querying).
2) *Querying objective and subjective attributes* (Interrogazione di attributi oggettivi e soggettivi): una query di attributo oggettiva è un modo strutturato di interrogazione simile alle query su tipi di dati predefiniti usati in contesti in cui la forma sintattica della query può essere specificata precisamente. Ciò risulta in contrasto con le situazioni in cui l'utente presenta una formulazione confusa di query. Una query di attributo soggettiva si compone di attributi la cui interpretazione varia considerevolmente da un utente ad un altro. Per esempio, una domanda quale "Dammi tutte le facce che hanno un bel naso" è soggettiva ed a che cosa "un bel naso" assomiglia dipende dal gusto dell'utente.
3) *CBR (Content-Based Retrieval) queries*: permettono di interrogare le caratteristiche di basso livello dei media, per esempio, query di colore, forma e texture per i media visivi.
4) *Correlation queries* (Domande di correlazione): usate per recuperare media che sono simili all'oggetto di domanda in termini di:
    - Relazione spaziale: relazioni direzionali e topologiche fra gli accoppiamenti di oggetti corrispondenti nella query e nel database.
    - Relazione temporale: relazioni tra le istanze di tempo in cui l'oggetto di query e gli oggetti nel database appaiono o scompaiono.
    - Relazione spazio-temporale: un fenomeno di dominio che varia nello spazio e nel tempo.

Il recupero attraverso la navigazione consiste in un'interfaccia facile da usare per il recupero delle informazioni dai database multimediali impiegando tecniche dai linguaggi multimediali. Tipicamente, un browser viene usato quando l'utente è molto vago circa i suoi bisogni di recupero o quando l'utente non è pratico della struttura e dei tipi di informazioni disponibili nel database. La funzionalità di un browser può variare dal fornire poco aiuto all'utente nel guidare il processo di ricerca al fornire comandi di filtrazione specializzati per vincolare efficacemente lo spazio di ricerca. Occorre notare che i browser avanzati sono integrati solitamente con altri tipi di schemi di recupero per vincolare lo spazio di ricerca. In questo senso, la navigazione può anche essere considerata come una tecnica di implementazione per la realizzazione di altri tipi di schemi di recupero.

Per quanto riguarda le restanti tre tipologie di query, in generale, la letteratura le raggruppa in due categorie principali di metodi d'interrogazione audiovisivi: *feature based querying* (interrogazione basata su feature) e *semantic querying* (interrogazione semantica).

3.2 Feature-based querying

La feature-based querying si riferisce alle tecniche che si concentrano sulle caratteristiche audiovisive di basso livello (colore, forma, ecc.) quali la query-by-example e la query-by-sketch.

Il tradizionale metodo di recupero video ha dovuto adottare un metodo query-by-example basato sul contenuto, per cui i dati video vengono misurati secondo caratteristiche rilevabili (quali colore, struttura o movimento) e vengono confrontati ad una query clip sottomessa (o un keyframe) per formare una metrica di similarità, che viene poi usata per effettuare il ranking. Purtroppo però l'efficacia di questo metodo è limitata a causa del gap semantico che esiste fra l'utente ed il sistema. Mentre l'utente ha un fabbisogno informativo inerente espresso nelle semantiche, o nei concetti di alto livello, il sistema opera in accordo con le caratteristiche di basso livello. L'utente deve quindi fare la traduzione del contenuto semantico oppure deve trovare una clip video adatta (o keyframe) per rappresentare la query.

L'adozione di questo metodo inoltre fa l'assunzione che le clip video semanticamente simili abbiano una piccola distanza fra loro nello spazio della caratteristica, cosa non sempre vera. Chiaramente c'è la necessità di adottare un approccio basato sulla semantica, per cui la traduzione del contenuto semantico è fatta dal sistema e la query è espressa nelle parole chiavi semantiche.

Un tale sistema coinvolgerebbe l'estrazione automatica delle semantiche (o la generazione automatica dell'annotazione video), l'immagazzinamento di queste semantiche in una certo formato di metadati arbitrario adatto, e infine, di realizzare il recupero basato su parole chiave utilizzando metadati.

Per questo motivo, noi prenderemo in considerazione per la nostra analisi maggiormente l'interrogazione semantica.

3.3 Semantic querying

La semantic querying si riferisce all'interrogazione basata sulle semantiche di livello più alto che sono più vicine alle interpretazioni dell'utente ed ai contesti d'uso.

Vari linguaggi di query semantici sono stati proposti a questo scopo. I metodi più semplici usano soltanto il recupero tradizionale basato sui dati, cioè usano parole chiave per descrivere il contenuto. Questa modalità di interrogazione è tuttavia molto limitata, poiché non permette una specifica dettagliata del contenuto e del tipo di risultati voluti. Un insieme di linguaggi di query semantici è stato proposto nel presente stato dell'arte ed è basato su un'estensione dei linguaggi di query dei database classici quali SQL ed OQL. Il problema più importante di questi linguaggi è che nessuno di loro è stato definito basandosi su uno studio delle esigenze degli utenti. Ciascuno di questi linguaggi si focalizza soltanto su un sottoinsieme delle descrizioni basate sul contenuto che gli utenti utilizzano per poter interrogare il contenuto audiovisivo.

L'arrivo dello standard MPEG-7 ha portato un importante sviluppo nella modellazione e nella rappresentazione dei dati multimediali. Tuttavia tali descrizioni sono utili soltanto quando possono essere recuperate correttamente e facilmente attraverso un linguaggio di query adatto. Una nuova sfida per l'interrogazione multimediale è quindi quella di fornire una linguaggio di query capace di recuperare il contenuto audiovisivo descritto da MPEG-7 e basato sui requisiti di alto livello degli utenti in campi di applicazione differenti.

3.3.1 Interrogazione mediante XQuery

Attualmente sono state proposte alcune tecniche di interrogazione allo scopo di recuperare il contenuto audiovisivo descritto da MPEG-7. Si possono classificare questi metodi in due gruppi principali. Il primo gruppo consiste nell'usare direttamente XQuery, uno standard definito da W3C, come linguaggio di ricerca. La scelta di XQuery è motivata dal fatto che esso è un linguaggio di query flessibile per estrarre i dati dai documenti basati su XML. Tuttavia, questi metodi presentano lo svantaggio di richiedere una conoscenza avanzata dei dettagli MPEG-7 per esprimere una query precisa. Inoltre, non considerano le differenti interpretazioni possibili del contenuto che sono collegate con i vari requisiti degli utenti.

In [4] viene proposta una soluzione completa per lo sviluppo di un VDMS (Video Data Management System) basato sul contenuto che utilizza XQuery come linguaggio di query per interrogare il video secondo i requisiti degli utenti. Nei loro studi sono stati confrontati i tre linguaggi di query XML più recenti proposti da W3C: XML-QL, XQL e XQuery e si è scoperto che XQuery è superiore se confrontato ai linguaggi più vecchi, perché supporta i requisiti degli utenti, così come propone alcune importanti caratteristiche aggiuntive che si possono usare per le query complesse.

Vengono quindi mostrate alcuni esempi di tipologie di query classiche, per mostrare come si possono recuperare sommari video, usando le caratteristiche di XQuery:

- **Query 1. Un esempio semplice.**

La query di esempio più semplice è quella che permette di accedere ad un elemento particolare. Per esempio, gli utenti vogliono visualizzare i segmenti più significativi da tutti i video. Si può formulare la domanda usando l'espressione di percorso relativo per accedere agli elementi:

/*SportVidLib/Video/HierarchicalSummary/HighlightSummary/HighlightSegment*

Tuttavia, si può usare alternativamente l'espressione di percorso assoluto.

Il percorso assoluto riduce la lunghezza della query; è meno propenso all'errore e non occorre conoscere il percorso per accedere all'elemento di cui si ha bisogno. Inoltre, un percorso assoluto non viene influenzato se deve essere cambiato il percorso attuale all'elemento:

//HighlightSegment

La proiezione può anche essere fatta usando l'espressione FLWR (che assomiglia di più a SQL), tuttavia, l'espressione di percorso è più facile e più compatta se confrontata all'espressione FLWR:

For $h in //HighlightSegment
Return $h

- **Query 2. Un esempio più specifico.**

Usando una tecnica simile (espressione di percorso), si può formulare una domanda più specifica come quella che segue: "Visualizzare tutti i segmenti chiave delle partite di calcio, in cui il Manchester United ospita la partita e segni Van Nistelrooy":

//video[@genre = "sport"] [@type = "soccer"]
[.//homeTeam = "Manchester United soccer team"]
/hierarchicalSummary/highlightSummary/
highlightSegment[.//player = "Van Nistelrooy"]

- **Query 3. Un esempio complesso.**

Occorre notare che nonostante i motori attuali di XQuery non abbiano ancora supportato questa query, XQuery ha proposto le caratteristiche per permettere questo tipo di domanda: "Per ogni giocatore, visualizza tutti i suoi segmenti chiave (dove compaiono) e conta il numero dei segmenti chiave che appartengono a quel giocatore all'interno della biblioteca video":

FOR $p IN DISTINCT-VALUE(//Who/data( ))
LET $c := COUNT (//Video/hierarchicalSummary

```
highlightSummary/highlightSegment[.//Who/data( ) =$p] )
RETURN
<result>
<player>{$p}</player>
<number of goals>$c</number of goals>
{ FOR $v in //Video, $p2
$v/HierarchicalSummary/HighlightSummary
/HighlightSegment/HighlightSegmentAnnotation/Who/data( )
WHERE $p = $p2
RETURN $v/HierarchicalSummary/
HighlightSummary/highlightSegment }
</result>
```

XQuery rappresenta dunque un linguaggio di query flessibile per l'estrazione dei dati dai documenti MPEG-7. Tuttavia, usare XQuery direttamente presenta lo svantaggio di far perdere l'utente nella descrizione dei dettagli di MPEG-7 piuttosto che focalizzarlo sull'essenza dei suoi requisiti di recupero. In effetti, per ricercare documenti MPEG-7, si ha bisogno di un modello astratto che rifletta i bisogni ed i comportamenti di recupero dell'utente. Un linguaggio di query adeguato per recuperare MPEG-7 dovrebbe da un lato riflettere un tale modello astratto di alto livello e dall'altro lato essere adatto per il recupero efficace delle descrizioni di contenuto MPEG-7 nel formato di XML.

Si è sviluppato dunque un secondo gruppo di tecniche d'interrogazione che definisce nuovi linguaggi di query o adatta quelli esistenti per supportare il recupero nelle descrizioni MPEG-7. Gli aspetti semantici non sono direttamente esprimibili in XQuery e quindi ognuno di questi approcci propone il suo proprio linguaggio di query specializzato. Questi linguaggi di query rendono possibile formulare query più complesse.

3.3.2 Interrogazione di informazioni implicite

Per poter sfruttare completamente la varietà delle descrizioni MPEG-7, viene richiesto un sistema che può ragionare sulle stesse descrizioni. Il formalismo MPEG-7 non permette alcun meccanismo di ragionamento perché è definito solamente nel XML Schema. XML Schema è adatto ad esprimere i vincoli di sintassi, strutturali, di cardinalità e di tipo richiesti da MPEG-7. Tuttavia, esso mostra parecchi limiti quando dobbiamo estrarre informazioni implicite. Ciò è dovuto al fatto che non ci sono né semantiche formali né capacità di inferenza associate con gli elementi di un documento XML.

In [6] vengono identificate tre questioni cruciali di specifica delle query nel recupero di documenti XML MPEG-7 che considerano le informazioni implicite da estrarre dalle descrizioni MPEG-7:

- Specifica di dati e relazioni intensionali: i dati e le relazioni estensionali sono quei dati e quelle relazioni che sono esplicitamente immagazzinati in un documento XML. I dati e le relazioni intensionali sono invece quelli calcolati o dedotti dai dati e dalle relazioni estensionali nei documenti XML. Molte relazioni di oggetti multimediali nei documenti MPEG-7 sono derivate dalle descrizioni di contenuto memorizzate basate sui tipi di dato di elemento o sugli schemi DS, piuttosto che da relazioni gerarchiche di elemento XML. Così, la capacità di esprimere le relazioni nei costrutti del linguaggio di query è cruciale per le specifiche di query MPEG-7. In aggiunta, molti dati spaziali e temporali sono rappresentati in una maniera implicita dentro i documenti XML MPEG-7 a differenza dei dati nei database relazionali. Per esempio, un istanza dell'elemento MediaTime in MPEG-7 significa un intervallo di tempo. È importante esprimere questi impliciti MediaTimePoints in quell'intervallo nel linguaggio di query poiché l'identificazione di oggetti multimediali può dipendere da un particolare MediaTimePoint.
- Specifiche di indirizzo di documenti completi: i documenti XML MPEG-7 contengono spesso strutture irregolari di documento. Per esempio, un tag Segment può anche essere all'interno di un altro tag Segment. Le strutture di contenuto MPEG-7 sono basate sui loro propri tipi di dati e schemi di descrizione (DS) piuttosto che sulla gerarchia di elemento XML. I documenti di XML MPEG-7 normalmente non sono documenti data-centered che sono collezioni di strutture quasi identiche. Un costrutto di query di indirizzo di documento completo deve specificare precisamente le ubicazioni desiderate dei documenti nelle strutture XML ricorsive o contestuali per il recupero delle informazioni.

- Specifiche di vincoli di co-occorrenza: in genere le descrizioni di oggetti multimediali hanno vincoli di sincronizzazione temporali e spaziali. Le strutture previste sono inerentemente dentro i documenti multimediali. Così gli elementi del documento XML MPEG-7 hanno normalmente vincoli di co-occorrenza. Per esempio, se un elemento XML per una descrizione di oggetto multimediale ha l'attributo A in una determinata posizione spaziale, esso deve avere lo stesso attributo A in un'altra posizione nella sincronizzazione spaziale. Un altro esempio è costituito da due oggetti multimediali che appaiono dentro la stessa regione spaziale allo stesso tempo nella sincronizzazione temporale.

3.3.2.1 Specifica mediante un formalismo logico

Nel rispondere alle questioni di specifica suddette è stato progettato il linguaggio di query XML sperimentale MMDOC-QL. Questo linguaggio racchiude un formalismo logico, chiamato *Path Predicate Calculus*, per specificare le query. Le formule nel Path Predicate Calculus sono forme ristrette del predicato di primo ordine. Per queste interrogazioni e manipolazioni basate sulla logica, sono stati progettati due importanti predicati: *element predicate* e *path predicate* per dare affermazioni logiche vere riguardo gli elementi del documento in un albero di documento. Questo Path Predicate Calculus può adeguatamente supportare i vincoli di co-occorrenza e le specifiche di indirizzo del documento per interrogare documenti XML.

Per supportare specifiche di dati e relazioni intensionali in questo formalismo logico, sono incorporati in questo linguaggio di query certi operatori logici stereotipali per asserire relazioni di oggetti multimediali. Esempi di operatori logici multimediali sono OVERLAP(element1: RegionLocatorType, element2: RegionLocatorType), TRAJECTORY(element1: MovingRegionType, element2: MediaTimePoint), etc. Un altro operatore logico MEMBERP è pure incluso per asserire dati intensionali quali MediaTimePoint nei costrutti del linguaggio.

Nel seguito viene illustrato MMDOC-QL per specificare query su documenti MPEG-7 XML. Un esempio di query è nella forma di "trovare tutti gli id di oggetti video ed il tempo su un'area particolare.

```
GENERATE   <List>
              <Videoobject>%objectid</videoobject>
              <ShowUpTime>%t</ShowUpTime>
           </List>
PATTERN     {"MR"[0-9][0-9][0-9]/%objectid}
           (<region> ... </region>/%focusarea}
FROM        mpeg7video.xml
CONTEXT     ((<Segment> WITH xsi:type="MovingRegionType"
                           id=%objectid AT %movingregion )
             CONTAINING
             (<SpatioTemporalLocator> DIRECTLY CONTAINING
                 (<MediaTime> AT %x)))
       AND MEMBERP(%t %x)
       AND OVERLAP(TRAJECTORY(%movingregion %t) %focusarea))
```

In MMDOC-QL, ci sono quattro clausole:

- OPERATION che è usata per descrivere le conclusioni logiche sotto forma di element predicate e path predicate ammissibili e che può essere GENERATE per le operazioni di retrieval, INSERT, DELETE o UPDATE;
- PATTERN che è usata per descrivere i vincoli di dominio delle variabili logiche che includono tag, attributo, contenuto, indirizzo e tipo di dato usando espressioni regolari;
- FROM che è usata per descrivere i documenti sorgenti per l'interrogazione;
- CONTEXT che è usata per descrivere le affermazioni logiche riguardo gli elementi di documento in formule logiche ammissibili nel Path Predicate Calculus.

FROM e CONTEXT sono accoppiate insieme e potrebbero esserci coppie multiple per descrivere sorgenti multiple. Le variabili logiche sono indicate da "%" come "%objectid". Le query in MMDOC-QL sono equivalenti a trovare tutte le prove alla chiusura esistenziale delle affermazioni logiche.

In questo esempio la formula logica di percorso ((WITH xsi:type = "MovingRegionType" ... <MediaTime> AT %x))) nella clausola CONTEXT afferma che l'elemento "Segment" con id uguale a %objectid contiene l'elemento "SpatioTemporalLocator" del quale gli oggetti video sono collocati durante il MediaTime %x.

In generale, (<%t> WITH attribute1=%x1, ..., attributen=%xn AT %a CONTAINING %c) è una notazione simile all'inglese per l'element predicate E(x1, x2, ..., xn, c, t, a) che sta per un'asserzione logica che l'elemento "t" all'indirizzo "a" contiene "c" con gli attributi x1, x2, …, xn in un albero di documento.

Per brevità, si possono anche usare versioni ridotte con solo variabili necessarie nelle interrogazioni logiche come (<%t> WITH attribute1=%x1, ... attributen=%xn), (<%t> CONTAINING %c), ecc., se una versione completa può essere impiegata chiaramente nel contesto.

Una formula logica di percorso è una composizione di predicati di elemento da Xpath "operatori d'asse". Sono esempi:

- operatori della relazione *parent/child* (padre/figlio) come INSIDE, DIRECTLY INSIDE, CONTAINING, DIRECTLY CONTAINING, ecc.
- operatori della relazione *sibling* (fratelli) come BEFORE, IMMEDIATELY BEFORE, AFTER, IMMEDIATELY AFTER, SIBLING, IMMEDIATELY SIBLING, ecc.

Si noti che qui viene usata una forma logica dei concetti di asse definiti in XPath poiché le formule di percorso nel Path Predicate Calculus sono affermazioni logiche per asserire verità logiche.
Un esempio del path predicate è (<bibref> INSIDE (<gcapaper> CONTAINING (<fname> CONTAINING "Peiya") AND (<surname> CONTAINING "Liu">))) per specificare tutti i bibref (riferimenti bibliografici) all'interno della pubblicazione di Peiya Liu.

Il dominio della variabile logica %objectid è limitato ad essere stringhe inizianti con "MR" seguite da cifre nella clausola PATTERN. La variabile logica "%t" è usata per legare MediaTimePoint in questo intervallo MediaTime durante la computazione logica. L'operatore TRAJECTORY è usato per asserire la regione di traiettoria da una regione movente %movingregion al MediaTimePoint %t e OVERLAP è un operatore logico spaziale per asserire ulteriormente che la regione desiderata dell'oggetto è anche coincidente con l'area focale.

3.3.2.2 Specifica mediante rete inferenziale

In [5] vengono messi a punto, per la ricerca delle descrizioni video MPEG-7, metodi di accesso basati su un modello di rete inferenziale che permette la deduzione di informazioni semantiche ed implicite.

Viene adottato il modello della rete inferenziale per modellare l'evidenza contenuta all'interno dei documenti MPEG-7, e specificamente, la caratteristica strutturale e concettuale dell'inerenza di evidenza,. L'*Inference Network* (IN) ha la capacità di effettuare un ranking date molte fonti di evidenza realizzando una combinazione di evidenza. Il modello IN è fondamentalmente una Bayesian Network (rete bayesiana) usata per modellare i documenti, i contenuti del documento e la query. L'IN consiste di due sottoreti: la Document Network (DN) prodotta durante l'indicizzazione e quindi statica durante il recupero e la Query Network (QN) prodotta dal testo della query durante il recupero.

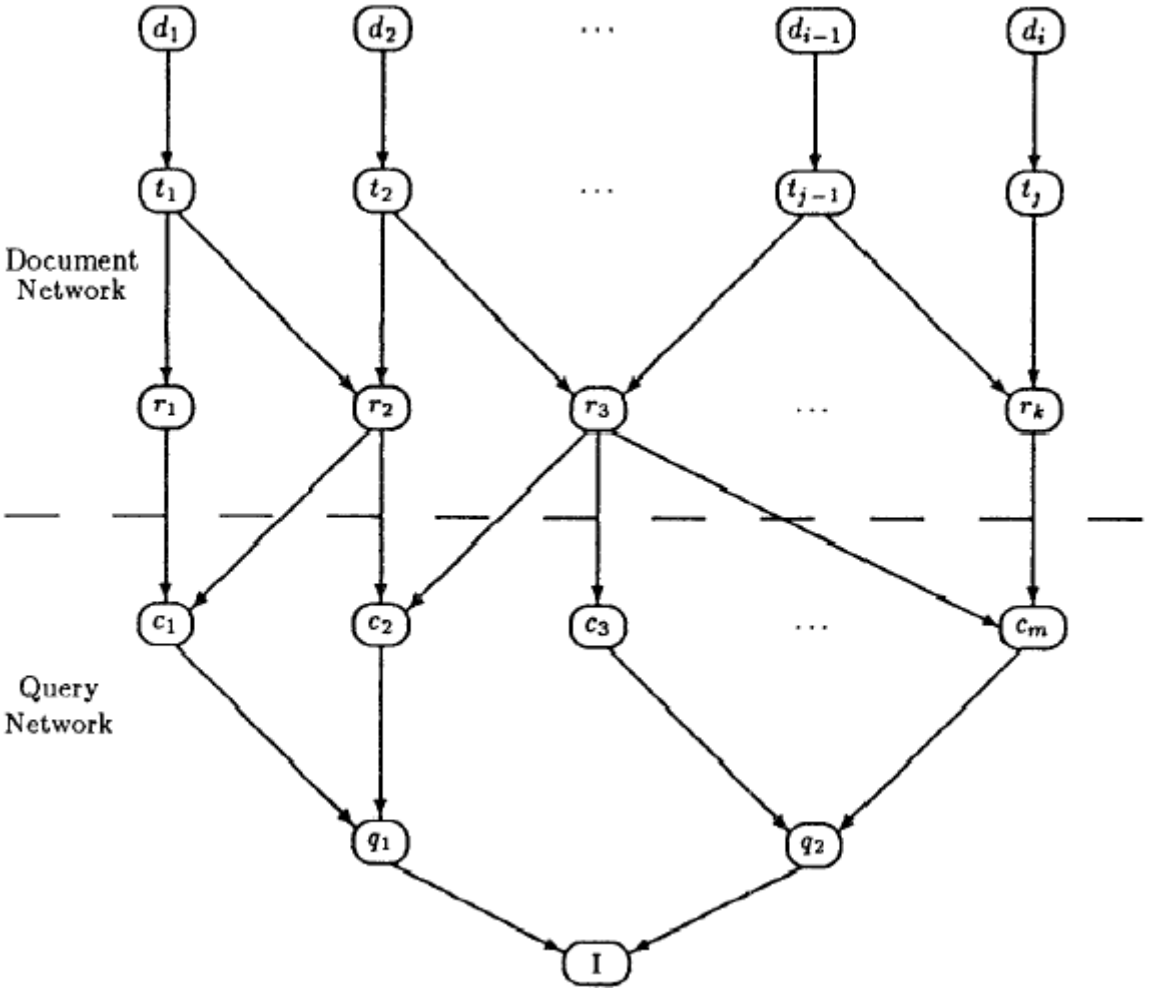


Figura 7 - Una Inference Network

La DN rappresenta la collezione del documento e consiste di nodi per ogni documento (denominati *document nodes*) e di nodi per ogni concetto nella collezione (*document concept nodes*). I nodi del documento rappresentano le unità recuperabili all'interno della rete, cioè, quegli elementi che si desidera vedere nel ranking risultante. Un collegamento causale (rappresentato come →) fra il nodo del documento ed il nodo del concetto del documento indica che il contenuto del documento è rappresentato dal concetto.
Ogni collegamento contiene una probabilità condizionale, o peso, per indicare la forza della relazione. La valutazione di un nodo è fatta usando il valore dei nodi genitori e delle probabilità condizionali.

La QN rappresenta la query sottomessa e consiste di una struttura di nodi che rappresentano i concetti richiesti (*query concept nodes*) e gli operatori (*query operator nodes*), collegati in una struttura ad inverted tree. La QN è costruita con un nodo foglia finale I che rappresenta l'*Information Need* (il fabbisogno informativo) dell'utente. La struttura consente operatori statistici e approssimazioni statistiche degli operatori booleani, alcuni dei quali sono dati in figura 8.

#and	AND the terms
#or	OR the terms
#not	Negate the term (incoming belief)
#sum	Sum of the incoming beliefs
#wsum	Weighted sum of the incoming beliefs
#max	Maximum of the incoming beliefs

Figura 8 - Operatori supportati dal modello IN

La base per il questo modello è quindi il modello di *Inference Network (IN)* poiché esso ha la capacità di modellare gli aspetti strutturali, concettuali e contestuali disponibili in MPEG-7.

- ***Document Network***

La DN cattura gli elementi strutturali dei documenti, i concetti trovati nella collezione e le relazioni (e le forze) fra ogni documento e concetto.

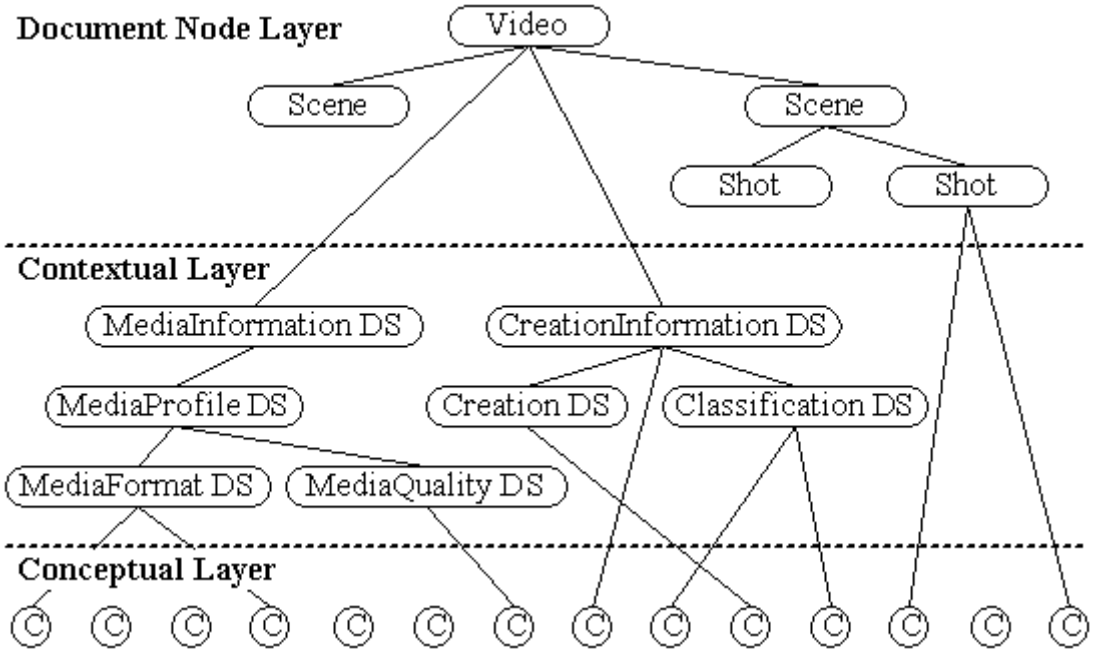


Figura 9 - La Document Network a tre strati

Essa consiste di tre strati come segue:

- Lo strato di nodo del documento, che contiene i nodi che rappresentano le unità recuperabili nella collezione.
- Lo strato contestuale, che contiene i nodi che rappresentano l'informazioni contestuale riguardo al documento → collegamenti concettuali.
- Lo strato concettuale, che contiene i nodi che rappresentano tutti i concetti nella collezione.

Nello strato di nodo del documento vi è una struttura gerarchica dei nodi del documento per rappresentare la struttura del video come delineata nel file MPEG-7. Ogni video è rappresentato così da un nodo Video, ognuno dei quali può contenere a sua volta i nodi Scene per rappresentare le scene, ognuno dei quali può contenere infine i nodi Shot per rappresentare gli shot. Questa struttura può essere estratta dai componenti strutturali MPEG-7.

I collegamenti Video → Scene e Scene → Shot rappresentano la decomposizione del video in parti più piccole e le probabilità condizionali su questi collegamenti indicano l'importanza dell'elemento figlio rispetto al padre (cioè quanto una scena particolare contribuisce al contenuto del video).

Lo strato concettuale contiene i nodi di concetto che rappresentano i concetti identificati all'interno della collezione del documento. I concetti sono identificati durante un processo di estrazione dove ogni documento viene analizzato ed i concetti riconosciuti.

La presenza di un concetto all'interno di un documento è rappresentata come una relazione (diretta o indiretta) fra il nodo del documento ed il nodo di concetto. In realtà, un tal collegamento esiste per ogni relazione documento → concetto e la non-presenza di un concetto è indicata con una probabilità condizionale di 0.5 (così, durante la valutazione, il valore del padre non influenzerà il valore del figlio).

Lo strato contestuale contiene i nodi di contesto. Lo strato tenta di sfruttare la ricca struttura che è disponibile all'interno dei documenti MPEG-7, vale a dire la posizione del concetto all'interno della gerarchia del DS. Queste informazioni strutturali possono essere usate sia per:

- Permettere una valutazione più precisa di quanto un concetto rappresenta un documento che considera il contesto (per esempio: i contesti possono essere di qualità variabile siccome alcuni possono essere un migliore indicatore del contenuto che altri). I nodi di concetto sono inclusi nella DN per tutti gli elementi MPEG-7 che si desidera controllare, risultando in relazioni Document → Context(s) → Concept;
- Permettere l'identificazione dei concetti che si presentano soltanto all'interno di un contesto di particolare (per esempio: la query riguarda il concetto "bbc" che si presenta soltanto all'interno del contesto "Creation DS"). Ciò viene sfruttato nella rete di query usando un concetto vincolato.

❖ *Query Network*

La QN consiste di nodi di concetto di query e di nodi di operatore di query standard, come precedentemente descritto. Gli operatori supportati sono quelli in figura 8. La QN include inoltre i nodi di contesto di query, che sono associati ad un nodo di concetto di query per formare un vincolo. I vincoli specificano il desiderato contesto del concetto all'interno dei documenti corrispondenti aspettati (per esempio: si vuole che il concetto "bbc" si presenti sotto il contesto "Creation DS"). I vincoli sono di due tipi, normale o complesso.

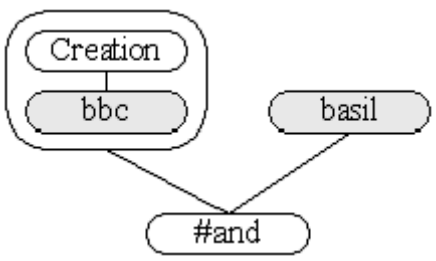


Figura 10 - Un vincolo normale

La QN in figura 10 contiene due nodi di concetto di query (questi sono grigi), un singolo nodo di operatore di query (comincia con #) e un nodo di vincolo di query. Il nodo di vincolo di query è collegato al nodo di concetto "bbc" per formare un vincolo normale. Ciò è denotato schematicamente accludendo il nodo di concetto di query al nodo di vincolo di query.
Questa struttura di query specifica che il nodo "bbc" di concetto di query dovrebbe essere collegato soltanto ai nodi di concetto del documento "bbc" che soddisfano il vincolo, cioè, quei nodi che si presentano sotto il contesto "Creation".

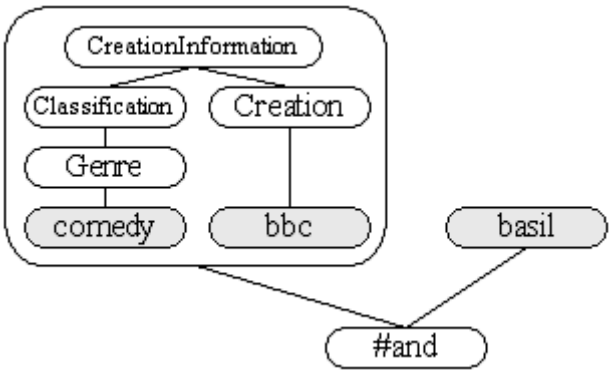


Figura 11 - Un vincolo complesso

La QN in figura 11 mostra un esempio di un vincolo complesso. In questo caso, viene specificata nella query una struttura che si desidera che si presenti nei documenti corrispondenti. La struttura indicata consiste di due concetti e della loro relazione attraverso un certo numero di contesti. La query richiede quindi quei documenti che corrispondono a questa struttura di subquery.

Vengono proposte due nuove parole chiavi che permettono la costruzione di vincoli normali e complessi: *#constraint* per porre un vincolo normale su un concetto; e *#tree* per creare un vincolo complesso. Sotto sono forniti due esempi che rappresentano le QN mostrate nelle figure 10 e 11.

#and(#constraint(Creation, "BBC") "Basil")

#and(#tree(CreationInformation,
#constraint(Classification/Genre,"Comedy")
#constraint(Creation,"BBC") "Basil")

3.3.2.3 Specifica mediante alberi di concetti

Anche in [7] viene presentata una struttura per il recupero semantico dei dati multimediali. La strategia di ricerca è basata sull'algoritmo del tree embedding esteso con un ontologia di dominio per ragionare riguardo alla descrizione dei dati multimediali, rappresentata per mezzo del formalismo MPEG-7. L'algoritmo proposto (Tree Embedding Approximation, *TEA*) trova le corrispondenze approssimative dei nodi dell'albero di query e restituisce un valore booleano. Questo valore booleano è settato a vero se c'è una corrispondenza fra l'albero di query e l'albero di dati, altrimenti è falso.

Vediamo ora un esempio motivante dell'idea di usare un meccanismo di ragionamento su ontologie per la ricerca semantica di dati multimediali. Si assuma di avere una collezione di descrizioni MPEG-7 che immagazzinano metadati riguardo a immagini di insetto. La figura 12 mostra un sottoinsieme di una tale collezione. La query per recuperare tutte le immagini riguardanti "*flying insect*" corrisponde a trovare tutte le immagini che contengono l'annotazione testuale nella descrizione. Al contrario, la ricerca semantica della query permetterà di trovare tutte le immagini che contengono un differente tipo di insetto volante.

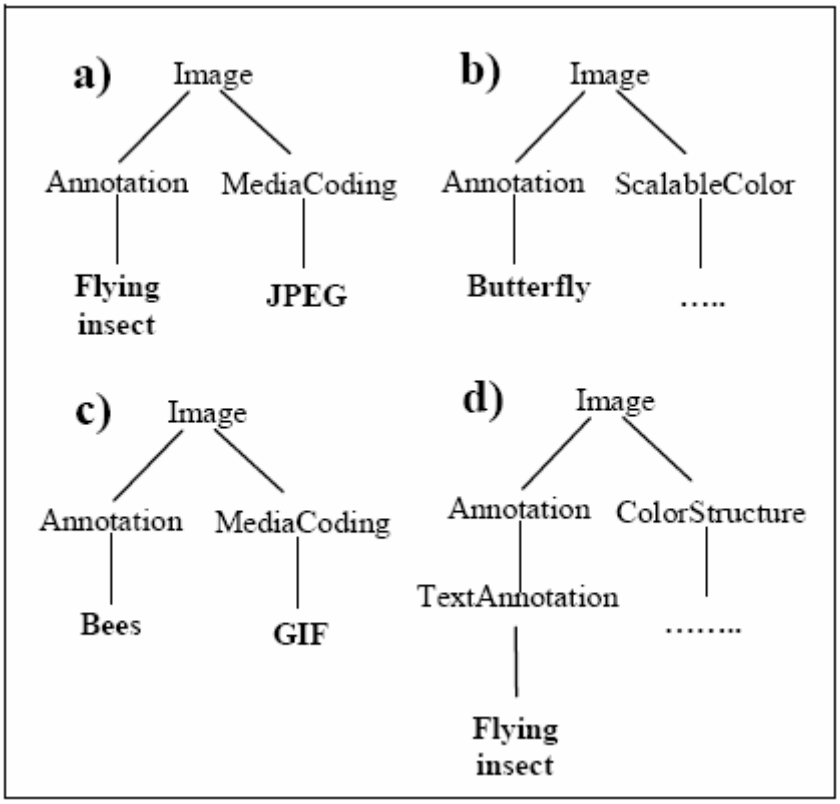


Figura 12 - Estratti di descrizioni MPEG-7 di immagini

L'unica risposta alla query è l'immagine a. Le immagini b e c non soddisfano la query perché contengono istanze di insetti volanti che non corrispondono esattamente ai requisiti della query. Tuttavia, se viene specificata un'ontologia che rappresenta i concetti del dominio dell'insetto come un relazione ordinata, come rappresentato in figura 13, allora le immagini b e c possono essere rinviate come risposte alla query, a condizione che si abbia un meccanismo di inferenza appropriato, che permetta trarre una tale deduzione. I concetti inclusi (*bees* e *butterfly*) sono risposte alla query (*flying insect*), quindi, si ottiene una risposta specifica ad una domanda generale.
Nella struttura proposta, i meccanismi di ragionamento e di tree embedding sono usati in una maniera complementare.

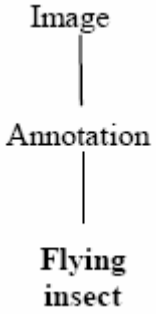


Figura 13 - Descrizione MPEG-7 della query

Una descrizione MPEG-7 è un'istanza di uno schema di descrizione, così esso rappresenta un documento XML. Vari modelli sono stati proposti per rappresentare i documenti XML. Il W3C ha proposto un modello generico chiamato Document Object Model (DOM). In questo modello, un documento XML può essere rappresentato come un albero etichettato ordinato.

Definizione 1. (*Rooted tree*).
Un albero radicato (o albero con radice) è una struttura $T = (N, E, root(T))$ che consiste di un insieme finito di nodi N, un insieme finito di bordi E ed un nodo $root(T) \in N$ che forma il nodo radice di T. E deve soddisfare le seguenti condizioni:
1. la radice non ha genitori;
2. ogni nodo dell'albero, tranne la radice, ha esattamente un genitore;
3. tutti i nodi sono raggiungibili dalla radice.

Definizione 2. (*Labelled rooted tree*).
Un albero radicato etichettato $T = (N, E, root(T))$ è un albero radicato in cui viene definita un'etichetta per ogni nodo $u \in N$.

Definizione 3. (*Path*).
Si supponga che $T = (N, E, root(T))$ sia un albero radicato. Se $n_1, n_2, .., n_m$ è una sequenza di nodi in T dove n_1 è il genitore di n_2, n_2 è il genitore di n_3, ecc., allora $n_1, n_2, .., n_m$ è un percorso di lunghezza $m-1$.

Definizione 4. (*Child node, Leaf, Internal node, Descendant, Ancestor*).
Si supponga che $T = (N, E, root(T))$ sia un albero radicato etichettato. I nodi di T con un genitore comune sono figli del loro genitore, denotati da: $children(u) = \{v \in N \mid (u, v) \in E\}$. Un nodo senza figli è una foglia. I nodi che non sono foglie sono nodi interni. Le foglie a volte sono denominati nodi esterni. Un nodo *n* è un antenato di un nodo *u* (alternativamente, *u* è un discendente di *n*) se c'è un percorso da *n* ad *u*.

Definizione 5. (*Ordered Tree*).
Un albero ordinato è un albero in cui sia la relazione di antenato (padre-figlio) che l'ordinamento da sinistra a destra fra fratelli sono significativi.

❖ *Descrizione MPEG-7 basata su albero*

Una descrizione MPEG-7 è un documento XML. Essa può essere rappresentata come albero etichettato ordinato $T = (N, E, root(T))$ con tre tipi di nodi:

- N_e: nodi di elemento che rappresentano i nodi non-foglia con un'etichetta, il nome;
- N_t: nodi di testo che rappresentano i nodi foglia con un'etichetta, il valore;
- N_a: nodi di attributo che rappresentano i nodi foglia con due etichette, il nome ed il valore.

root(T) rappresenta i dati MPEG-7 descritti. Per esempio, se la descrizione MPEG-7 si occupa di dati video, allora l'etichetta di root(T) è *video*.

Gli elementi dell'albero di descrizione MPEG-7 sono:

- $N = N_e \cup N_t \cup N_a$: insieme dei nodi;
- *E*: insieme degli archi dove ogni coppia appartenente ad E rappresenta la relazione padre-figlio tra i due nodi;

- le etichette dei nodi intermedi rappresentano i nomi di elemento che corrispondono ai metadati MPEG-7, mentre le foglie dell'albero presentano il valore delle caratteristiche estratte come il valore dell'istogramma del colore o i concetti di annotazione usati nel processo di ricerca.

Per illustrare questa rappresentazione formale, la figura 14 mostra un estratto della rappresentazione di albero MPEG-7.

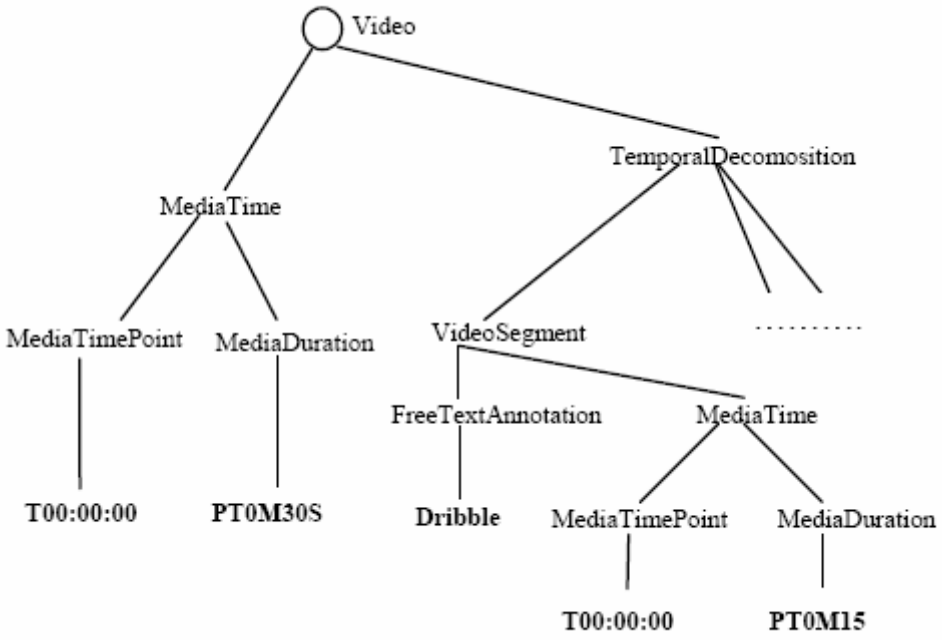


Figura 14 - Esempio dell'albero di descrizione MPEG-7

❖ *Interpretazione come albero di query MPEG-7*

La formulazione di una query dovrebbe essere semplice anche per utenti occasionali e dovrebbe richiedere soltanto una conoscenza parziale della struttura e del contenuto dei dati multimediali. Un processo di recupero dovrebbe recuperare non soltanto i risultati che corrispondono esattamente alla domanda, ma anche rispondere con una struttura ed un contenuto simile alle condizioni specificate dalla query. Si suppongano ricerche da parte dell'utente di dati multimediali sulla base della loro struttura e contenuto.

Un esempio di una tale query è: “trova tutti i video composti da due segmenti ed annotati con le parole ‘dribble goal’ ”. La query sarà poi tradotta nel formalismo MPEG-7 e sarà quindi rappresentata come albero ordinato etichettato (si veda in figura 15).

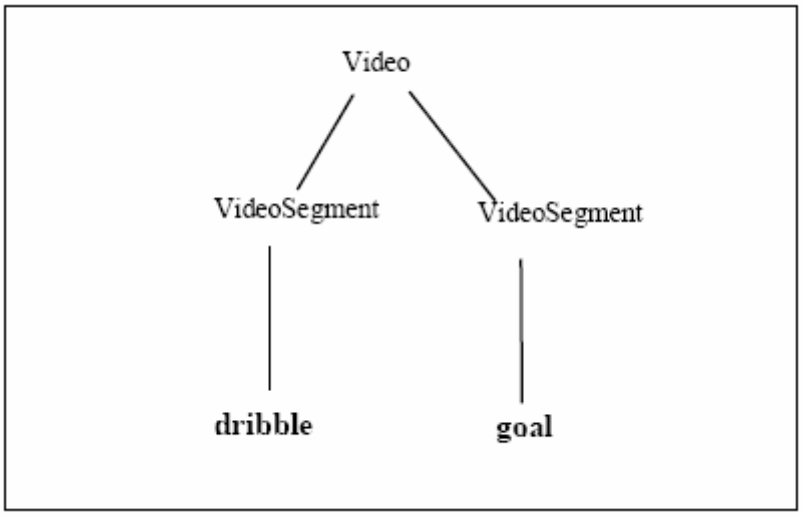


Figura 15 - Esempio di un albero di query MPEG-7

3.3.2.4 Specifica mediante viste semantiche

In [7] è stato proposto un adattamento di XQuery per la ricerca di documenti MPEG-7 denominato SVQL (Semantic Views Query Language). SVQL è un linguaggio di query di livello elevato che permette agli utenti di esprimere i loro requisiti seguendo il modello semantico di vista (PhysicalView, ProductionView, ThematicView, VisualView ed AudioView) in una maniera concisa, astratta e precisa. A tal fine sono state studiate varie query proposte da utenti professionisti e non-professionisti di notizie TV. Analizzando le differenti query viene mostrato che gli utenti adottano cinque viste differenti per esprimere la loro richiesta: *PhysicalView*, *ProductionView*, *ThematicView*, *VisualView* ed *AudioView*. Il seguente esempio mostra una tale query e analizza come l’utente esprima la query attraverso differenti viste.

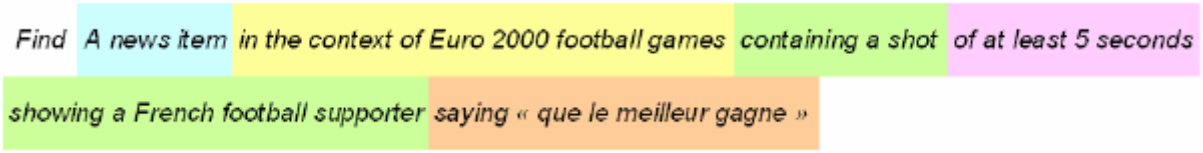


Figura 16 - Un esempio di query analizzato secondo differenti viste

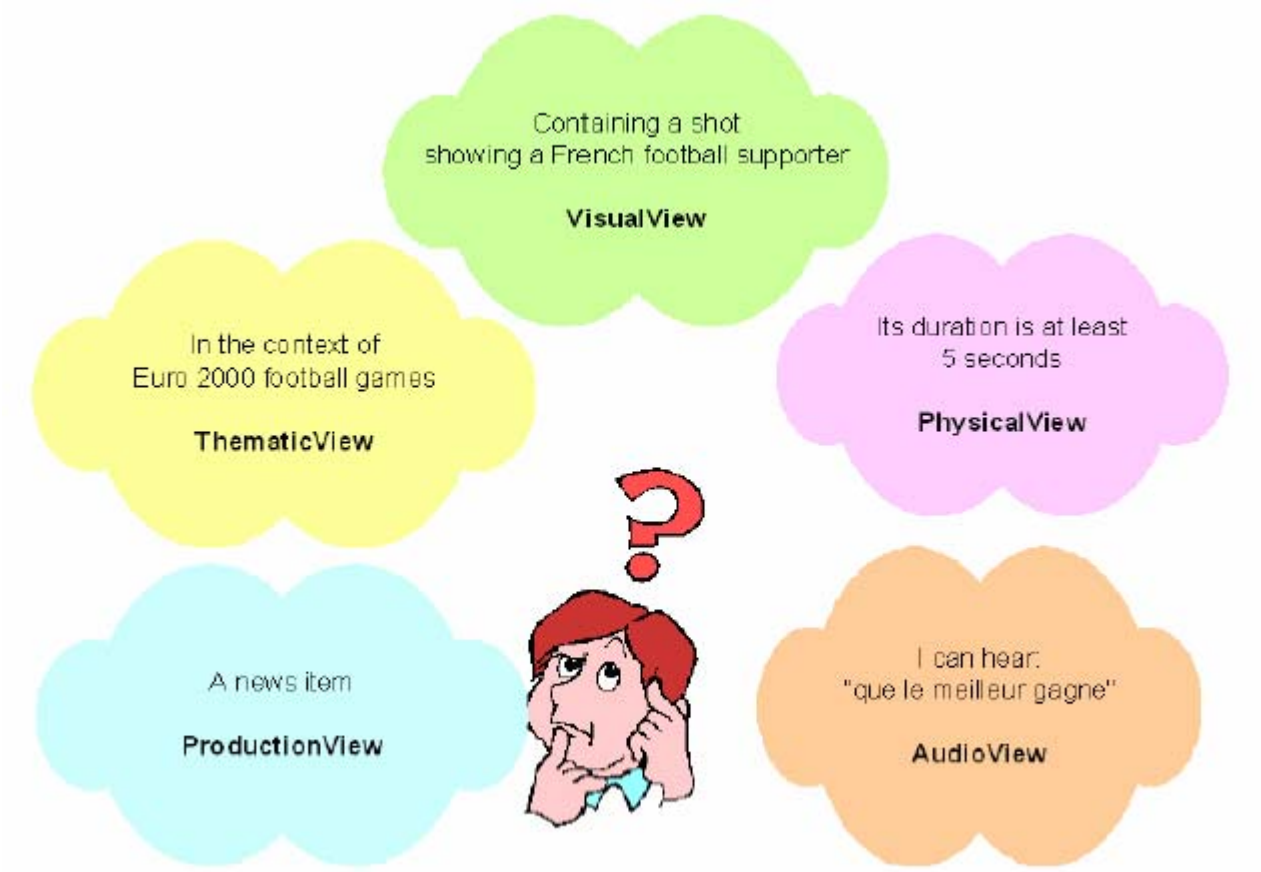


Figura 17 - Le cinque viste

L'utente in effetti sta cercando un segmento video, provando a descrivere questo segmento video dai differenti punti di vista come appare in figura 17.

Nel *Semantic Views Model* ogni vista viene descritta usando cinque elementi: *BasicViewEntity*, *ViewDescriptions*, *IntraViewRelations*, *InterViewRelations* e *ViewOperators*. *BasicViewEntity* è l'unità atomica di descrizione in ogni vista. ViewDescriptions esprime differenti caratteristiche del *BasicViewEntity* in ogni vista. *InterViewRelations* esprime la relazione di corrispondenza fra le *BasicViewEntities* che appartengono a viste differenti. *IntraViewRelations* esprime le relazioni fra le *BasicViewEntities* che appartengono alla stessa vista.

SVQL è un linguaggio di query di alto livello che permette che gli utenti esprimano i loro requisiti seguendo il *Semantic Views Model* (Modello di viste semantico) in un senso conciso, astratto e preciso. Inoltre, SVQL è stato progettato specialmente per rendere possibile il recupero dei dati audiovisivi descritti dallo standard MPEG-7.

SVQL permette che gli utenti formulino query di alto livello in cima alle descrizioni MPEG-7 senza intromettersi nei dettagli di implementazione di MPEG-7. Confrontato ai linguaggi di query dei database tradizionali quali SQL, OQL e XQuery, SVQL presenta il vantaggio di essere stato progettato specialmente per il recupero dei dati audiovisivi basati su un modello semantico delle esigenze degli utenti. I linguaggi di query quali SQL, OQL e XQuery sono stati progettati per adattarsi a strutture specifiche dei dati. Tuttavia questi linguaggi non sono adatti per esprimere il livello astratto del *Semantic Views Model*. Usare XQuery e XML Schema è un metodo conveniente per l'"implementazione" del *Semantic Views Model* e della *Semantic Views Query Language* in cima alle istanze MPEG-7. Tuttavia, da un punto di vista concettuale, XQuery non permette che l'utente esprima direttamente i suoi requisiti astratti seguendo il *Semantic Views Model*.

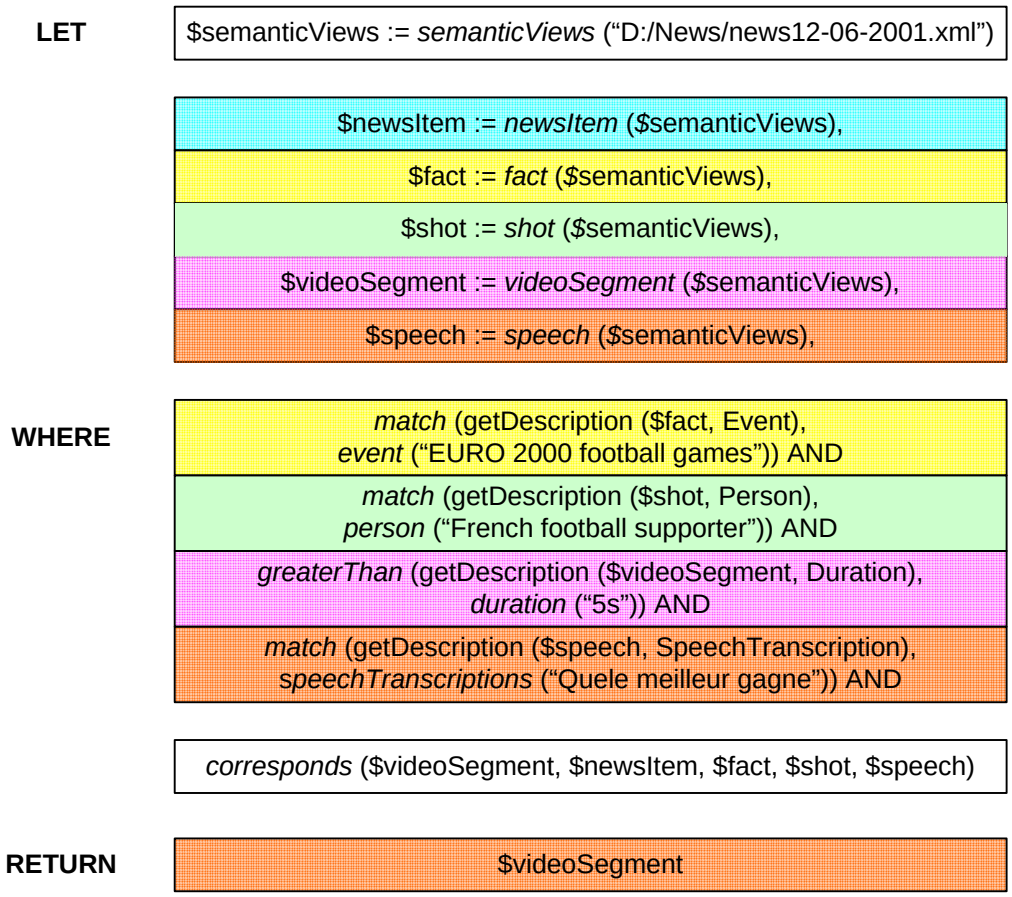


Figura 18 - Un esempio di query SVQL

Il linguaggio può essere usato sia dagli utenti finali che dai programmatori di applicazione. Ai primi viene fornito un linguaggio di query di alto livello che è relativamente facile da usare ed astratto ed ai secondi uno strumento utile che può facilitare ed accelerare lo sviluppo delle applicazioni basate sul *Semantic Views Model*. La figura 18 mostra un esempio di una query in SVQL.

Come si può osservare nell'esempio, la query è basata su una struttura "**LET-WHERE-RETURN**", molto vicina alla struttura FLWR ("**FOR-LET-WHERE-RETURN**") di XQuery, con la piccola differenza che non viene usata la parola chiave FOR.

Questo tipo di sintassi di query, a cui ci si riferisce come "sintassi orientata alla parola chiave", viene usata nei più famosi linguaggi di query, quali SQL, OQL e XQuery ed è un modo familiare dell'espressione di query per gli utenti finali specializzati ed i programmatori di applicazione.

Come si può vedere nella suddetta query, la clausola LET contiene due tipi di espressioni:

- Nella prima espressione, la funzione SemanticViews() è chiamata con il nome del file che contiene le istanze MPEG-7 da recuperare. Questa funzione genera il *Semantic Views Model* che corrisponde al file MPEG-7 al volo.
- Nella serie seguente di espressioni, un insieme di funzioni sono chiamate per ottenere differenti *BasicViewEntities* richieste del *Semantic Views Document* e per assegnare a loro un insieme di variabili. Ogni nome di funzione indica il tipo di BasicViewEntity, per esempio NewsItem, Fact, Shot, VideoSegment e Speech.

Anche la clausola WHERE contiene due differenti tipi di espressioni:

- Le prime quattro espressioni rappresentano un insieme di condizioni che dovrebbero essere tenute sulle *BasicViewEntities* di differenti *Views*. Queste condizioni sono espresse attraverso un insieme di ViewOperators: qui vengono usati *match()* e *greaterThan()*
- L'ultima espressione rappresenta la condizione riguardo alla *InterViewRelation* di BasicViewEntities attraverso l'operatore *corresponds()*. Esso determina quali delle BasicViewEntites citate corrispondono l'una all'altra. Questa espressione è usata se più di una vista è usata nella query.

Infine, la clausola RETURN identifica una variabile che contiene una BasicViewEntity da rinviare all'utente. Nella suddetta query la variabile che contiene il VideoSegment viene restituita.

4 PROCESSING DI QUERY

Una volta stabilite le tipologie di query formulabili attraverso i vari linguaggi proposti, vediamo ora quali modelli e strutture vengono utilizzati per processarli.

4.1 Motore di XQuery

In [9] è stato proposto un motore di XQuery, rappresentato in figura 19, che può essere usato come un modulo di processing XQuery in un sistema di libreria digitale che supporta i documenti XML. Vediamo l'architettura generica del sistema di libreria digitale. Essa consiste di quattro moduli: un'interfaccia di utente (*User Interface*), un motore di XQuery (*XQuery Engine*), un motore di reperimento delle informazioni (*Information Retrieval Engine*) e un deposito XML (*XML Repository*). Il modulo di interfaccia di utente fornisce un modo facile per cercare i documenti XML e trasforma una data query di utente in una XQuery equivalente. Il modulo del motore di XQuery prende un XQuery come input e fornisce un piano di query per il modulo di reperimento delle informazioni come output. Il motore di reperimento delle informazioni esegue un piano di query comunicando con il deposito di XML, che memorizza i documenti XML.

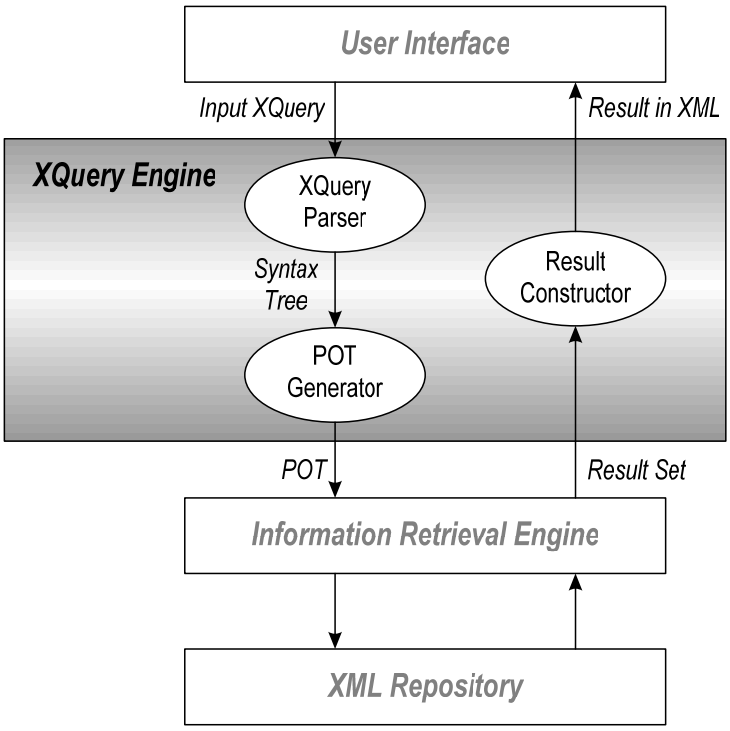


Figura 19 - XQuery Engine

Il motore di XQuery analizza un input XQuery e costruisce un albero di sintassi (*Syntax Tree*) per la query. Quindi, trasforma l'albero di sintassi in un piano di query, denominato *Primitive Operation Tree* (POT). Ogni nodo di un POT rappresenta un'operazione atomica nei termini del motore di reperimento delle informazioni e può essere interpretata e processata dallo stesso. L'insieme dei risultati è restituito al motore di XQuery, che a sua volta trasforma il risultato in un documento XML della forma che è richiesta dall'interfaccia di utente. Il risultato finale in XML viene restituito di nuovo all'interfaccia di utente.

Questo metodo presenta i seguenti aspetti utili. In primo luogo, ogni interfaccia di utente che genera XQuery può accedere a qualunque sistema di libreria digitale compreso questo motore di XQuery. In secondo luogo, viene definito un insieme di operazioni primitive per i POT in modo da poterli transformare in un'interfaccia standard fra un motore di XQuery e un motore di reperimento delle informazioni per un sistema di librerie digitali generico che supporta i documenti XML.
In terzo luogo, alcune ottimizzazioni di query sopra i POT possono essere fatte nel XQuery Engine in modo da avere prestazioni di ricerca migliori. Attualmente si sta sviluppando un prototipo del motore di XQuery. Esso sarà installato all'interno di un Digital Library System basato su MPEG-7 che supporta la ricerca basata sul contenuto delle immagini. La specifica di XQuery è una continua bozza di lavoro e non è ancora stata completata. Poiché la versione corrente della specifica di XQuery non definisce funzioni complete per il reperimento delle informazioni, occorre estendere la sintassi di XQuery aggiungendo alcune funzioni quali ad esempio *rankby()*.

4.2 Reti inferenziali

Per quanto riguarda il modello di rete inferenziale [5] due ulteriori processi sono eseguiti per realizzare il recupero: il processo di collegamento (*attachment process*), per cui la QN viene collegata alla DN per formare la IN completa e ciò viene effettuato dove i concetti sono gli stessi in entrambe le reti; il processo di valutazione (*evaluation process*), in base al quale la IN completa viene valutata per ogni nodo del documento per formare la probabilità di rilevanza alla query. La valutazione è inizializzata regolando l'output di un nodo del documento a 1 e tutti i altri nodi del documento a 0. Questo viene fatto per ogni nodo del documento a turno e la rete viene valutata. La probabilità di rilevanza del documento viene presa dal nodo finale I ed è usata per produrre il ranking.

❖ *Collegamento (attachment)*

Il processo di collegamento viene eseguito durante il recupero una volta che la QN è stata costruita e richiede il collegamento alla DN. Il risultato del processo è un certo numero di collegamenti DN → QN dove i concetti sono considerati essere gli stessi e dove tutti i vincoli specificati sono combinati. Questo collegamento contiene un peso, o la probabilità condizionale, che si può usare per indicare la forza del collegamento (cioè quanto vicino i concetti ed i vincoli siano combinati). Il processo di collegamento esegue i seguenti compiti:

1. Creare i collegamenti candidati (collegamenti DN → QN) confrontando i concetti nella QN con i concetti nella DN. Nell'implementazione viene confrontato il testo dei nodi di concetto di QN con il testo dei nodi di concetto di DN e viene generato un collegamento quando questo è lo stesso. A questo punto si potrebbero considerare gli errori di pronuncia, la localizzazione, i thesauri e lo stemming per raggruppare in maniera intelligente i concetti simili. Ciò viene denominata fusione concettuale (*conceptual fusion*).
2. Creare i collegamenti secondo i vincoli.
    a) Per i collegamenti candidati non vincolati viene creato un collegamento fisso (lo stesso per ogni nodo del documento).
    b) Per i collegamenti candidati vincolati vengono analizzati i vincoli. Ciò viene effettuato individualmente per ogni nodo del documento siccome soltanto i nodi che si presentano sotto il nodo del documento sono considerati facenti parte della struttura della DN candidata.

       Viene calcolata la Edit Distance (ED) fra la struttura della DN candidata ed il vincolo di concetto della QN contando il numero di inserzioni, di cancellazioni e di sostituzioni che sarebbero necessarie per trasformare la prima nel secondo. Ciò provoca una ED metrica per ogni query vincolata (normale o complessa) per ogni nodo del documento. Questa metrica misura la prossimità dell'adattamento.
3. Per ogni nodo del documento vengono poi creati i collegamenti specifici del documento dai collegamenti candidati usando o una soglia o un metodo di collegamento pesato. Nel primo caso viene creato il collegamento usando un peso di 1 (la probabilità condizionale sul collegamento DN → QN) se l'ED è sotto una soglia specificata. Nel secondo caso viene creato il collegamento e viene usato l'ED per formare il peso usando la formula:

$$P(\text{QN node} \mid \text{DN node}) = 1 / (ED + 1)$$

La probabilità condizionale rappresenta dunque la forza del collegamento considerando quanto le strutture si somiglino (mentre l'ED tende a 0, il peso tende verso 1). A questo punto, si possono anche usare le informazioni sul processo di fusione concettuale (uso dei thesauri, della localizzazione, ecc…) nel calcolare la probabilità condizionale sul collegamento DN → QN. Un vantaggio dell'usare il metodo di collegamento pesato è che esso produce una lista di risultati più grande.

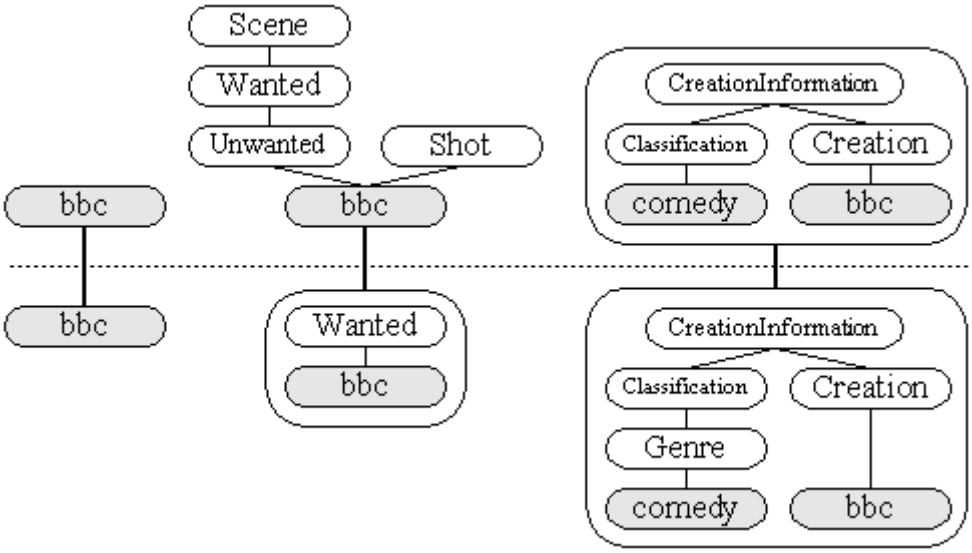


Figura 20 - (a)(b)(c) Tre esempi di collegamento

Il risultato del processo di collegamento è un insieme di collegamenti fissi e un insieme di collegamenti specifici di documento per ogni nodo del documento. Ciò è noto come rete completa (*complete network*). La figura 20 dimostra tre esempi di collegamento in cui le strutture sopra la linea punteggiata sono nella DN e quelle sotto sono nella QN. Questi esempi sono descritti sotto:

a) Mostra un esempio di un concetto non vincolato di query e di un nodo di concetto di documento corrispondente, per cui viene creato un collegamento fisso con P(QN|DN)=1.
b) Mostra un esempio di un concetto di query vincolato normalmente. Il nodo di documento "Creation" soddisfa il vincolo "Wanted" con un'ED = 1 se viene cancellato il nodo "Unwanted". Allora viene creato un collegamento specifico di documento. Usando il metodo della soglia viene creato il collegamento con P(QN|DN)=0 o 1 a seconda che l'ED sia sotto la soglia oppure no. Usando il metodo di collegamento pesato viene creato il collegamento con P(QN|DN) = 1/(1+1) = 0.5.
c) Mostra un esempio di un vincolo complesso. La struttura candidata della DN mostrata soddisfa il vincolo poiché il nodo di contesto della radice "CreationInformation" ed i due nodi di concetto "comedy" e "bbc" cono tutti presenti.

❖ *Valutazione (evaluation)*

Il processo di valutazione viene eseguito per ogni nodo del documento usando la rete completa. La rete viene valutata per ogni nodo di documento per mezzo di collegamenti specifici di documento per quel documento. Il risultato della valutazione della rete viene preso dal nodo del fabbisogno informativo nella QN. Oltre al metodo standard di valutazione della IN completa, vengono considerati due metodi che sfruttano la struttura all'interno della DN:

- Eredità di collegamento *(Link Inheritance, LI*), è la situazione che si presenta quando un nodo figlio può ereditare i nodi di contesto dai nodi genitori.

- Taglio del percorso (*Path Cropping, PC*), è la situazione dove un contributo dei genitori ad un nodo di concetto della DN viene tagliato (rimosso) poiché non soddisfa il vincolo sul nodo di concetto di query della QN collegato.

La LI è illustrata in figura 19. Il nodo Video ha due nodi figli di documento che indicano la struttura, i nodi Scene e Shot. Il nodo Video contiene inoltre un contesto "Creation" (che può contenere i collegamenti ai nodi di concetto). L'idea dell'eredità di collegamento è che questo contesto si applica anche ai figli del nodo Video, i nodi Scene e Shot. Per esempio, il contesto può contenere i nodi di concetto che descrivono quando il video è stato generato, informazione che è applicabile alle scene ed agli shot all'interno del video. I nodi figli di documento ereditano quindi il collegamento Video → Creation dal loro genitore durante la valutazione.

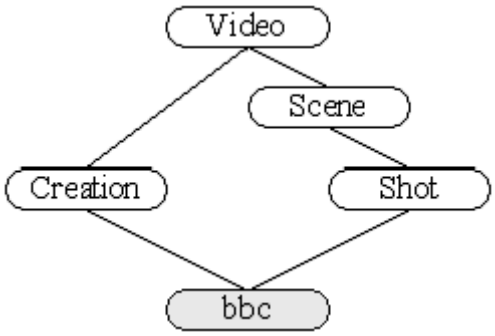


Figura 19 - Link Inheritance

Anche se il contesto ereditato è applicabile ai nodi figli, diventa meno influente come il numero di generazioni fra il figlio ed il contesto aumentano. Nell'esempio, il contesto "Creation" è soprattutto applicabile al nodo Video, poi al nodo Scene ed applicabile minimamente al nodo Shot. Per catturare questo fatto, viene degradata la probabilità condizionale sul collegamento Video → Creation ereditato. Ci si riferisce a ciò come eredità di collegamento con degradazione (*Link Inheritance with Degradation, LID*). La dimensione della degradazione è calcolata secondo la dimensione del gap di generazione (la distanza di nodo fra il genitore reale del contesto ed il nodo che eredita il contesto). La degradazione potrebbe anche considerare il rapporto di durata di frame.

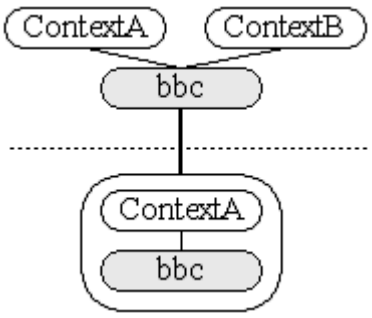


Figura 20 - Path Cropping

Il PC è illustrato in figura 20. La figura mostra un nodo di concetto di documento "bbc" con due contesti parentali, collegati con successo ad un nodo di concetto di domanda vincolato normalmente. Durante la valutazione del nodo di concetto di documento, si può scegliere di ignorare l'influenza dei genitori "ContextB" poiché questo concetto particolare non è specificato all'interno del vincolo. Cioè, l'influenza di questo genitore non è di interesse per la query e quindi è tagliata; il nodo di contesto di documento nella figura viene valutato come se avesse soltanto un genitore.

❖ Stima della probabilità

La parte finale del modello riguarda lo specificare come sono valutate le probabilità condizionali (o pesi) fra i nodi. Un'alta probabilità condizionale fra due nodi indicherebbe che essi sono strettamente accoppiati ed il valore del figlio dovrebbe riflettere strettamente il valore del genitore. I pesi devono soltanto essere valutati per la DN mentre la QN usa un comportamento fisso secondo gli operatori.

Due tipi di probabilità condizionali devono essere valutate:

- Context → Context. Fra un contesto dei genitori e un contesto del figlio. Due sottotipi:
  - Strutturale (per esempio: Video → Scene)
  - Contestuale (per esempio: Video → CreationInformation)
- Context → Concept. Fra un contesto ed i concetti che sono associati con esso.

Le probabilità strutturali vengono valutate usando le informazioni di rapporto di durata (*duration-of-parent/duration-of-child*), poiché ogni nodo di documento ha una durata di frame codificata in MPEG-7. Le probabilità contestuali sono valutate usando le informazioni di fratello di contesto (*1/total-number-of-siblings*). Alternativamente, si potrebbe considerare la dimensione del contesto, la frequenza, o la qualità percepita del contesto del figlio in questa stima.

La probabilità di Context → Concept viene valutata secondo due fattori: le proprietà statistiche del concetto (per esempio: frequenza di termine, frequenza inversa di documento); e le informazioni su come il concetto è stato estratto.

*Weight = tf(t,d) * idf(t)*
*Pweight = 0.5 + (0.5 * weight)*

Le proprietà statistiche del concetto forniscono le informazioni sulla qualità del termine come un fattore discriminante e le informazioni su quanto un documento (o contesto in questo caso) è rappresentato dal termine.

Viene usata la suddetta formula Weight per calcolare la probabilità condizionale dalle statistiche del termine e viene usata la suddetta formula Pweight dall'inizio alla fine della rete per accertare che il peso sia superiore a 0.5 e si abbia quindi un'influenza positiva sul risultato.

<StructuredAnnotation>
<Who>Basil,Sybil,Andre</Who>
</StructuredAnnotation>

Il file MPEG-7 stesso può fornire gli indizi circa la qualità di un termine che si può analizzare durante il processo di estrazione.

Nel suddetto pezzo MPEG-7 di esempio, tre concetti (Basil, Sybil ed Andre) sono contenuti all'interno di un singolo descrittore MPEG-7 usando una virgola come delimitazione. Ogni concetto non è quindi così rappresentativo come un singolo concetto poiché ha due fratelli di descrittore. Vengono usate queste informazioni per raffinare le probabilità usando (*1/total-number-of-siblings*).

Viene ora mostrato un esempio su come vengono valutate le probabilità all'interno della rete di documento.

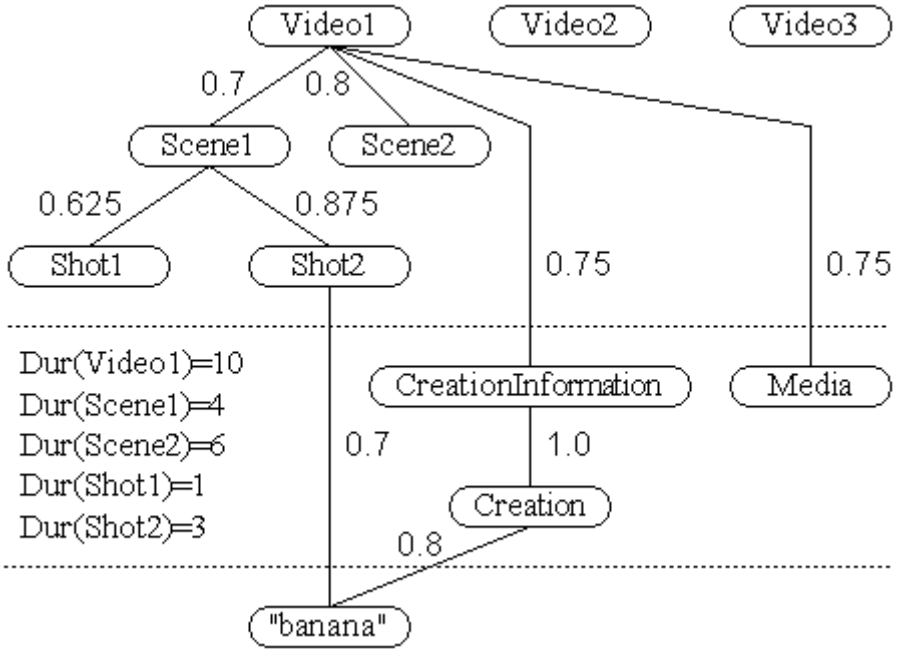


Figura 21 - Una DN di esempio

La DN di esempio come appare in figura 21 contiene 3 Video, 2 Scene e 2 Shot ed anche due occorrenze del concetto "banana" all'interno di due contesti differenti:

Video1->CreationInformation->Creation->Concept
Video1->Scene1->Shot2->Concept

Data la struttura in figura 21, dove Dur(Element) afferma la durata del nodo di documento, le probabilità strutturali di Context → Context sono valutate come segue: (usando il rapporto di durata e la formula di Pweight).

P(Scene1|Video1) Duration ratio = 4/10 = 0.4

*PWeight = 0.5 + (0.5 * 0.4) = 0.7*
P(Scene2|Video1) Duration ratio = 6/10 = 0.6
*PWeight = 0.5 + (0.5 * 0.6) = 0.8*
P(Shot1|Scene1) Duration ratio = 1/4 = 0.25
*PWeight = 0.5 + (0.5 * 0.25) = 0.625*
P(Shot2|Scene1) Duration ratio = 3/4 = 0.75
*PWeight = 0.5 + (0.5 * 0.75) = 0.875*

Le probabilità contestuali di Context → Context sono valutate come segue: (usando il total-number-of-siblings e la formula di Pweight)

P(CreationInformation|Video1) and P(Media|Video1)
Total number of siblings = 2
Sibling weight = 1 / 2 = 0.5
*PWeight = 0.5 + (0.5 * 0.5) = 0.75*
P(Creation|CreationInformation)
Total number of siblings = 1
Sibling weight = 1 / 1 = 1.0
*PWeight = 0.5 + (0.5 * 1.0) = 1.0*

Nel modello le probabilità di Context → Concept sono valutate usando le statistiche di termine. Tuttavia, nella DN di esempio si hanno troppo pochi termini per effettuare questo calcolo così vengono assegnati i valori:

P(banana|Creation) = 0.8
P(banana|Shot2) = 0.7

La stima della probabilità è condotta durante la fase di indicizzazione che risulta nella DN mostrata sopra nella figura 21. La DN consiste di un certo numero di nodi e di collegamenti probabilistici tra di loro. Tutte le probabilità condizionali sono positive (cioè >0.5, a causa della formula di Pweight), che significa che quelle unità che influenzano i genitori del concetto “banana” (cioè Creation e Shot2) saranno posizionati più su.

4.3 Alberi

In [7], viene fornito uno schema della struttura per la ricerca dei dati multimediali basata sulla loro descrizione MPEG-7. L'architettura di questa struttura è descritta in figura 22. In questa struttura, viene definita un'ontologia specifica di dominio che viene integrata nel database MPEG-7 durante il processo di indicizzazione. Gli indicizzatori (persone responsabili di fornire i metadati per l'indicizzazione del contenuto multimediale) usano termini e relazioni con la precisa semantica che l'ontologia contiene per l'indicizzazione.

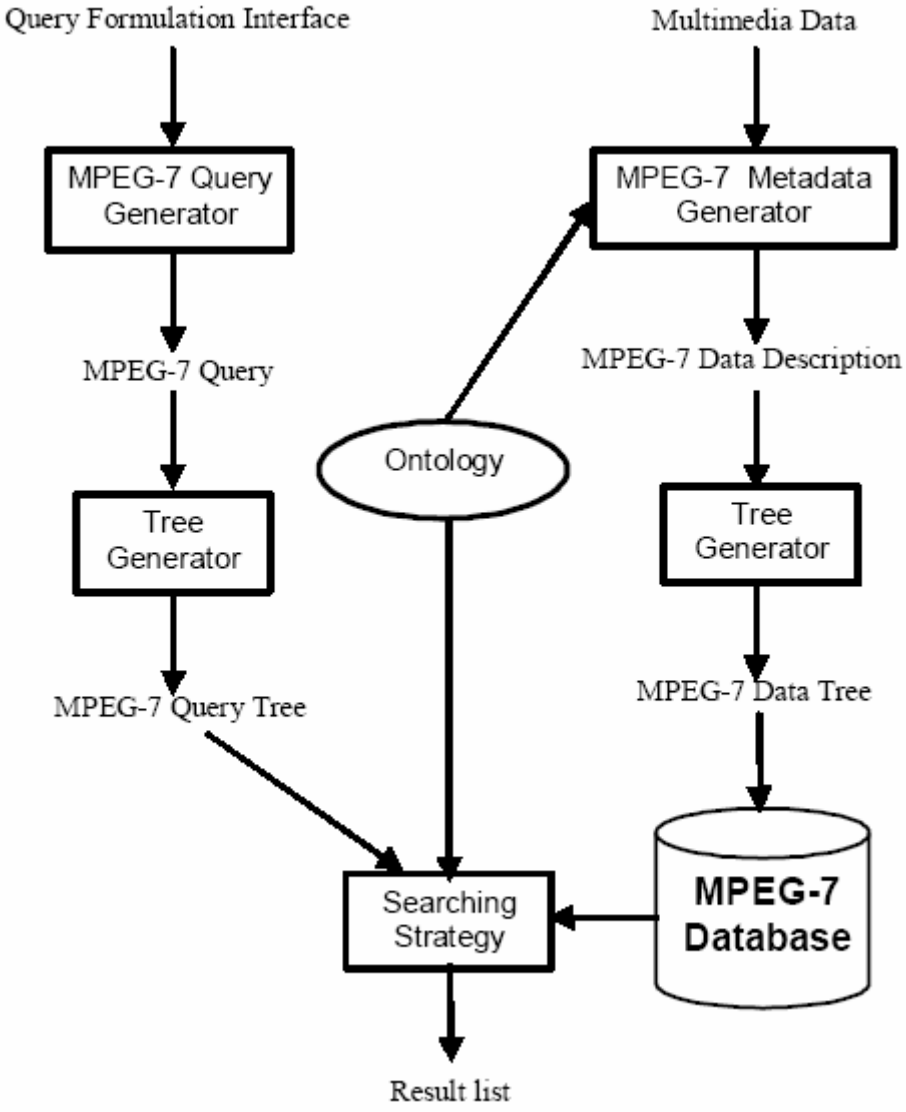


Figura 22 - Struttura del processo di recupero

Ci sono quattro componenti differenti:

- *MPEG-7 Metadata Generator*: questo componente è responsabile del processo di creazione dei metadati che è guidato dall'appropriata ontologia. Riceve come input un dato multimediale e l'annotazione del testo e produce come output la corrispondente descrizione di metadati MPEG-7. I metadati possono essere estratti automaticamente o manualmente dai dati multimediali. Gli algoritmi di estrazione automatici si adattano bene per le caratteristiche di basso livello quali i colori o la dimensione del file.
  Per le più alte caratteristiche semantiche quali gli oggetti e le loro proprietà, è richiesta l'annotazione manuale;

- *MPEG-7 Query Generator*: l'interfaccia di formulazione di query (vedere la figura 23) fornisce agli utenti un'interfaccia interattiva, facile da usare e grafica. La query dell'utente è tradotta nella sua forma corrispondente in MPEG-7 dal componente generatore di query MPEG-7. La query risultante è rappresentata come descrizione MPEG-7;

- *Tree Generator*: modella la descrizione MPEG-7 come albero etichettato ordinato. I particolari di questo processo sono stati forniti nella sezione precedente;

- *Searching Strategy*: questo componente è basato sul formalismo del tree embedding. L'albero di query MPEG-7 è inserito approssimativamente negli alberi di dati MPEG-7, immagazzinati nel database MPEG-7, per trovare i dati multimediali che rappresentano risposte alla query dell'utente. L'algoritmo di embedding approssimato usa l'ontologia per ragionare riguardo alle descrizioni MPEG-7 e per dedurre informazioni implicite.

❖ Esempio di ontologia

È stata definita un'ontologia per la descrizione delle partite di pallacanestro. La definizione dell'ontologia è basata sulle regole della FIBA (l'organizzazione internazionale responsabile della definizione delle regole della pallacanestro). Sono state quindi separate cinque aree in una partita di pallacanestro: periodi di tempo (primo tempo, secondo tempo, timeout, periodo di penalty, ecc.), posti (stadio), ruoli degli attori (giocatore, arbitro, allenatore, ecc.), squadre (squadra di casa, squadra ospite) ed eventi (fallo, canestro, passaggio, tiro, ecc.). Infine viene introdotta una frazione dell'ontologia che si occupa dei ruoli degli attori in una partita di pallacanestro (si veda la figura 23).

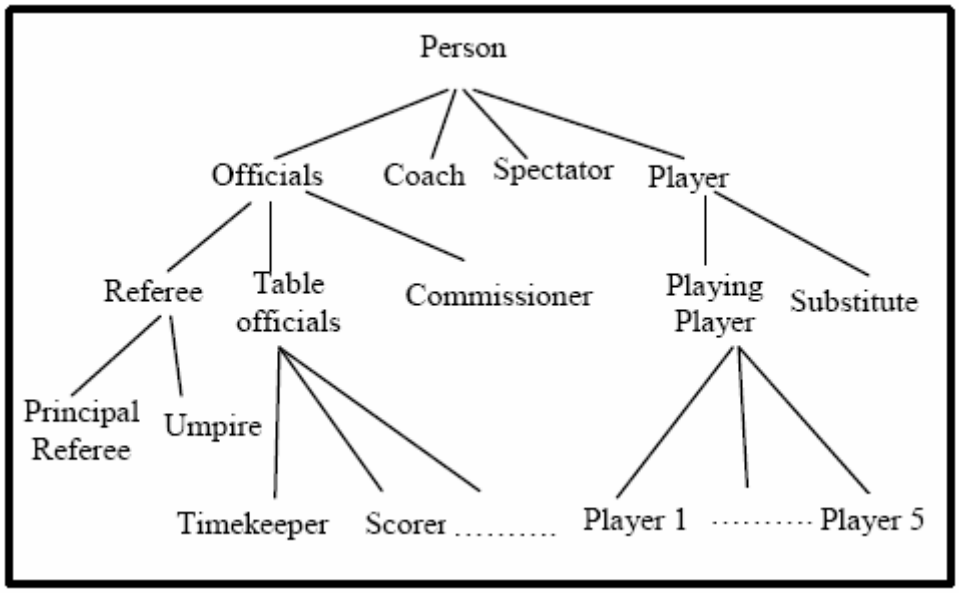


Figura 23 - Concetti dell'ontologia per i ruoli di persona in una partita di pallacanestro

Secondo la figura 23, i ruoli in un gioco di pallacanestro sono: Officials, Coach, Spectator e Player. Un'istanza di Official può essere un Referee, un Tabel Officials o un Commissioner. Un Referee può essere un Principal Referee o un Umpire. Un'istanza di Player può essere un Playing Player o un Substitute.

Viene ora presentato un modo di interpretare sia i dati che le query multimediali come alberi identificati ordinati. Con questa interpretazione, il problema di rispondere ad una query può essere visto come il problema di includere un query tree in un data tree riguardo all'ontologia di dominio.

❖ Il problema dell'approssimazione del tree embedding

L'obiettivo nel progettare la struttura corrente è di includere l'albero di query MPEG-7 approssimativamente nell'albero di descrizione MPEG-7 in modo tale che le etichette e le relazioni padre-figlio dei nodi vengano conservate. Quindi, viene permesso che l'albero di descrizione MPEG-7 abbia nodi intermedi tra quelli trovati nell'albero di query MPEG-7, ma la loro relazione predecessore-successore deve ancora essere mantenuta nell'albero. Le etichette sono conservate se sono uguali oppure se l'etichetta del nodo di dati è inclusa dall'etichetta del nodo di query. Un esempio di inclusione consentita è indicata in figura 24. L'albero P è incluso approssimativamente nell'albero T.

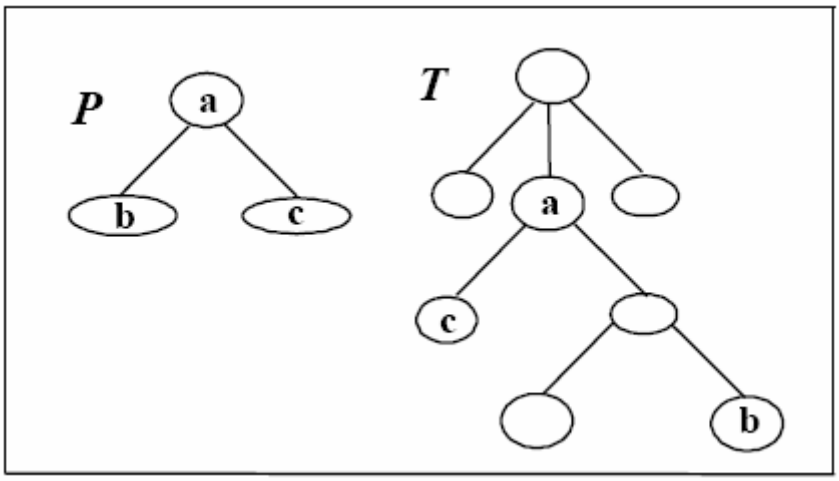


Figura 24 - Esempio di inclusione consentita

Definizione 6. (*Unordered tree inclusion*).

I problemi dell'inclusione di albero sono definiti attraverso *embedding* fra gli alberi. Siano *P=(V, E, root(P))* e *T=(W, F, root(T))* due alberi. Una funzione iniettiva *f* dai nodi di *T* è un embedding di *P* in *T* se conserva etichette e relazioni genitore-figlio. Cioè, per tutti i nodi *u* e *v* di *P* viene richiesto che:

1. $f(u) = f(v)$ se e soltanto se $u = v$,
2. $label(u) = label(f(u))$, e
3. *u* è un antenato di *v* in *P* se e soltanto se $f(u)$ è un antenato di $f(v)$ in *T*.

Si dice che P è un *unordered included tree* di *T* se c'è un embedding di *P* in *T*.

Definizione 7. (*Tree Embedding Approximation*).

Dato un albero *q* di query e un albero *d* di dati, un embedding approssimativo di *q* in *d* è una funzione iniettiva parziale f : $node(q) \rightarrow node(d)$ tale che $\forall q_i, q_j \in q$:

1. $sim(label(q_i), label(f(q_i))) > 0$, dove sim è una funzione di similarità che restituisce un valore normalizzato in [0,1] che dichiara la similarità fra due etichette date.
2. q_i è un genitore di $q_j \Rightarrow f(q_i)$ è un antenato di $f(q_j)$.

Definizione 8. (*Postorder traversal*).

Un attraversamento a posteriori è una tecnica per processare ricorsivamente i nodi di un albero in cui vengono visitati prima tutti i sottoalberi e successivamente la radice.

Definizione 9. (*Match*).

Sia *f* un embedding secondo la definizione 7. Sia *u* un nodo di una query. Si può definire il nodo di dati *f*(*u*) un match di *u.*

❖ Algoritmo per la Tree Embedding Approximation *TEA*

Algorithm 1 Searching embedding between query tree in data tree

Input : *Q* - a list of query nodes sorted in postorder
D - a list of data nodes

Output : Boolean to indicate if a match is found

1: *Match* := *True* // *We suppose that there is a match in the beginning*
2: *i* := 1 // *i is an index representing the postorder numeration*
3: **while** (*Match* = *True* and $i \leq |Q|$) **do** { // $|Q|$ *is the number of query node*}
4: $M(q_i) := \{0\}$ // *The match set belonging to* q_i
5: **if** (*children*(q_i) = {0}) **then** {// *Process query leaves*}
6: **for all** $d \in$ *leaves*(*D*) such that (*label*(*d*) = *label*(q_i) or *label*(*d*) $\subseteq$ *label*(q_i)) **do**
{ // Make the subsumption test if *label*(*d*) $\neq$ *label*(q_i) }
7: $M(q_i) := M(q_i) \cup \{d\}$ // *Construct the matching set for the leaves query*
8: **end for**
9: **else** {// *Process internal nodes of the query*}
10: **for all** $d \in D$ such that (*label*(*d*) = *label*(q_i)) **do**
11: $S := \{0\}$ // *The set of data node combinations*
12: **for all** $q_j \in$ *children*(q_i) **do**
13: $S' := \{0\}$ // *The set of child node combinations*
14: **for all** $d_m \in M(q_j)$ such that (*d* is an ancestor of d_m) **do**
15: $S' := S' \cup \{d_m\}$
16: **end for**
17: $S := S \cup S'$
18: **end for**
19: **if** $S \neq \{0\}$ **then**
20: $M(q_i) := M(q_i) \cup \{S \cup \{d\}\}$
21: **end if**
22: **end for**
23: **end if**
24: **if** ($M(q_i) = \{0\}$) **then**
25: *Match* := *False* // *There is a query node that does not match* $\Rightarrow$ *stop the processing*

```
26:   else
27:     i := i + 1                    // Process the next query node in postorder
28:   end if
29: end while
30: return Match;
```

Per ogni nodo di query, i nodi di dati presi sono quelli che sono membri di un embedding potenziale dell'albero di query nell'albero di dati. L'albero di query Q è processato in senso ascendente seguendo la lista a posteriori. Ciò assicura che per ogni nodo figlio del nodo di query corrente, l'insieme di corrispondenza di quel nodo figlio venga calcolato.

Le foglie di query e gli altri nodi di query sono processati differentemente. Per ogni foglia di query q_i (linee 5-8), l'algoritmo calcola il relativo insieme di corrispondenza $M(q_i)$. Questo insieme include le foglie di dati che hanno la stessa etichetta di q_i e quelle per cui l'etichetta corrispondente è inclusa dall'etichetta di q_i (condizione nella linea 6). Un algoritmo di inclusione è usato per controllare l'inclusione fra i concetti (etichette). Collegando gli altri nodi (nodi interni e la radice), l'algoritmo *TEA* costruisce incrementalmente un embedding del nodo di query corrente. Si assuma che q_i sia un nodo interno di query corrente.

L'algoritmo prende tutti i nodi di dati che hanno la stessa etichetta di q_i (linea 10), poi prova ad inserire il sottoalbero di query radicato in q_i in tutto il sottoalbero di dati radicato in d (linee 12-18). Quindi, per lo stesso nodo di query, si possono ottenere vari embedding nell'albero di dati.
Poiché il problema del tree embedding è NP-hard e per limitare la quantità di nodi di dati da considerare nel processo di ricerca, l'algoritmo *TEA* inizia dalla parte inferiore, cercando corrispondenze di similarità fra le foglie di query e di dati. Se non c'è corrispondenza di nessuna foglia o nodo interno di query (linea 25), allora l'albero di dati non soddisfa la query ed il processo di ricerca su questo albero di dati viene interrotto (la condizione del ciclo di *While* è *false*). Tuttavia, se tutti i nodi di query sono processati ($i = |Q|$ e $Match = True$), allora viene trovata una corrispondenza. L'albero di dati soddisfa l'albero di query ed i dati multimediali che corrispondono a questo albero di dati vengono restituiti.

❖ Illustrazione dell'algoritmo

L'ontologia di dominio è introdotta nella struttura a due livelli. Il primo è il processo di indicizzazione. Il secondo è durante la strategia di ricerca rappresentata dall'algoritmo di tree embedding. Per illustrare come i processi di algoritmo ottengano risposta ad un dato albero di query MPEG-7, usando la conoscenza di base supplementare rappresentata nell'ontologia, viene introdotto un esempio (si veda la figura 25).

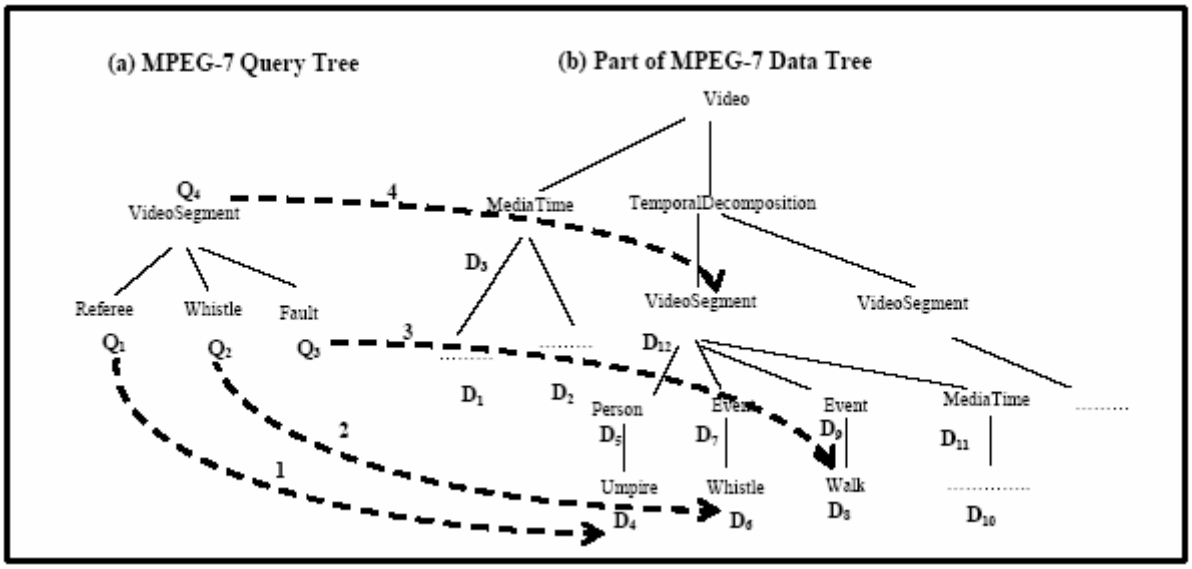


Figura 25 - Embedding di un query tree MPEG-7 in un data tree MPEG-7

In questo esempio, vengono richiesti estratti video di pallacanestro in cui l'arbitro fischia un fallo. La rappresentazione dell'albero di una tale query è illustrata in figura 26 (a). La figura 26 (b) è una parte di una descrizione MPEG-7 immagazzinata nel database MPEG-7. Q_1, Q_2 e Q_3 sono la lista a posteriori dei nodi di query. D_1, D_2, D_3, ecc. sono la lista a posteriori dei nodi di descrizione MPEG-7. Le frecce tratteggiate rappresentano la funzione di matching fra un nodo di query ed il nodo di dati appropriato. Essa conserva le relazioni ancestor-child ed anche il significato del concetto. L'algoritmo calcola il matching delle foglie di query. Il corrispondente di Q_1, noto come $M(q_1)$, è l'insieme $\{D_4\}$ anche se *label*(Q_1) è differente da *label*(D_4). In questo caso, viene controllato se l'etichetta del nodo di dati è inclusa dall'etichetta del nodo di query (cioè se le due etichette hanno lo stesso significato). Questa verifica viene fatta usando l'ontologia di figura 24. Il processo di embedding continua fino a processare la radice della query. Si ha lo stesso caso quando si processa il nodo di query Q_3. L'etichetta del nodo di dati D_8 è inclusa dall'etichetta del nodo di query rispetto all'ontologia che definisce gli eventi in una partita di pallacanestro (*Walk* (passi) è un fallo). Nell'esempio, $M(Q_4) = \{D_4, D_6, D_8, D_{12}\}$ non è vuoto, così la query radicata in Q_4 è inclusa nel

sottoalbero di dati radicato in D_{12} e questo sottoalbero costituisce una risposta alla query. L'embedding è approssimativo poiché l'algoritmo conserva il significato dei concetti della query e le relazioni ancestor-child quindi non si ha un embedding esatto.

4.4 Viste

Vediamo infine come avviene il processing delle query nel linguaggio SVQL.

Come appare in figura 27, SVQL permette la formulazione delle query che seguono il *Semantic Views Model* il quale sarà mappato attraverso strati differenti nelle istanze MPEG-7.

La sintassi di SVQL è una specializzazione della sintassi di XQuery. Questa caratteristica di progetto di SVQL presenta due vantaggi principali. In primo luogo, questo tipo di sintassi permette un modo familiare dell'espressione di query per gli utenti finali specializzati ed i programmatori di applicazione. Secondariamente, permette un'implementazione diretta di SVQL basata su XQuery. La specializzazione di XQuery in SVQL viene realizzata definendo un insieme di funzioni e di operatori, che rappresentano le funzionalità di base del *Semantic Views Model* ad un livello astratto. Il trattamento delle query di SVQL viene realizzato da un processor di XQuery che si prende cura delle funzioni e degli operatori di SVQL. Questi sono rappresentati come chiamate di funzioni interne di XQuery.

Come si può vedere in figura 26, quattro classi di funzioni sono coinvolti nel processing di una query SVQL:

- **Funzioni API MPEG-7**: un insieme di funzioni che facilitano l'accesso alle istanze MPEG-7 e che sono chiamate dalle tre classi di funzioni citate sotto. Queste funzioni che sono implementate in XQuery e/o in JDOM possono essere riutilizzate in ogni applicazione che richiede l'accesso e la manipolazione delle descrizioni MPEG-7.

- **Semantic Views Document Creation** (Creazione del Documento di Viste Semantico): la creazione al volo del *Semantic Views Document* attraverso la funzione di *SemanticViews()*.

Un *Semantic Views Document* è effettivamente un documento XML che contiene la descrizione di differenti viste che rappresentano un dato documento audiovisivo.

- **BasicViewEntity Location** (Posizione della BasicViewEntity): un insieme delle funzioni (quali *NewsProgram()*, *Shot()*, ecc.) che individuano le BasicViewEntities (di tipo *NewsProgramType*, *ShotType*, ecc.) all'interno del *Semantic Views Document* il cui percorso viene passato come parametro della funzione.

- **Semantic Views Operators** (Operatori Semantici di Viste): tutti i ViewOperators, quali *match()*, *corresponds()*, ecc.

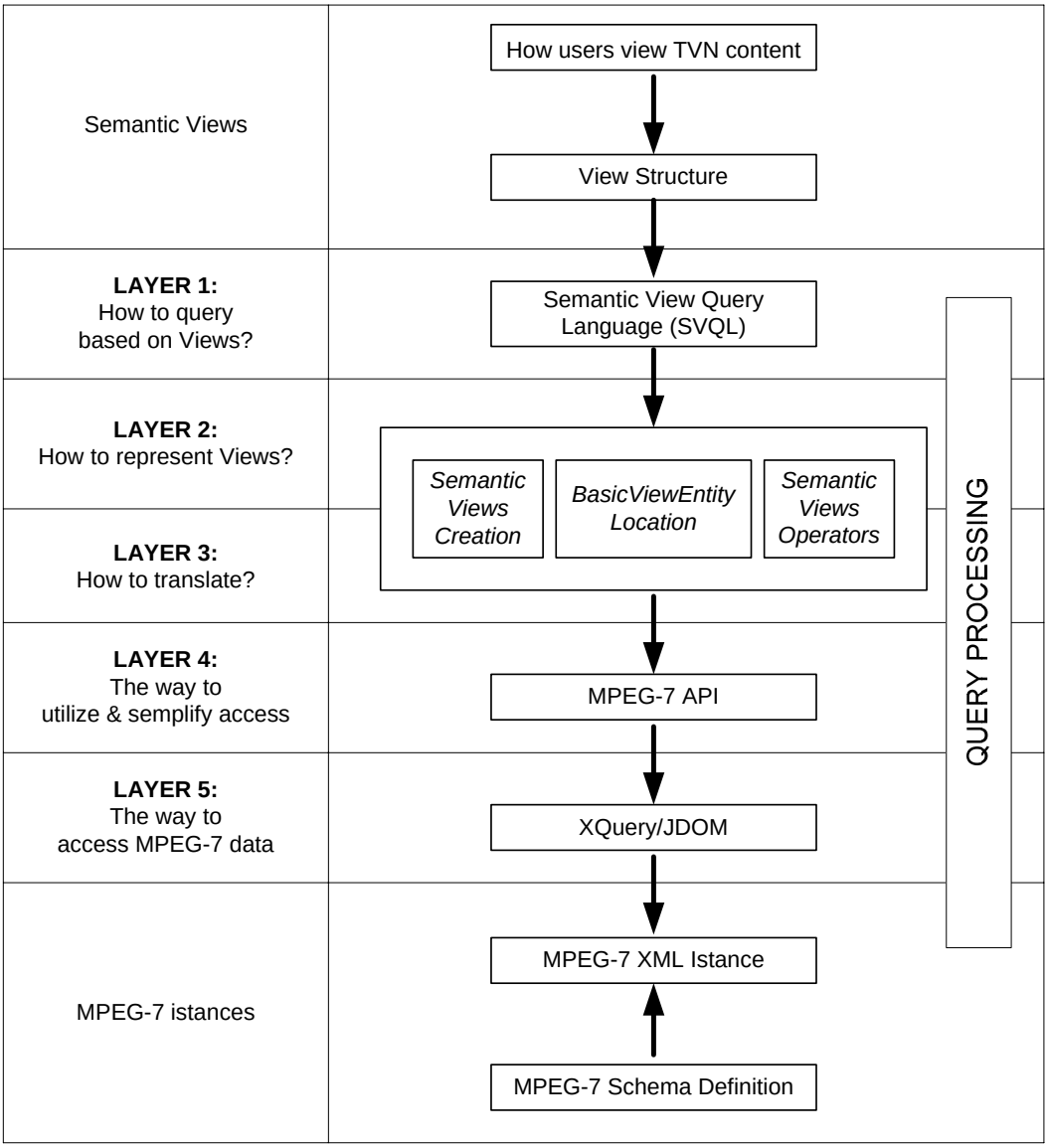


Figura 26 - Architettura a strati di recupero basata sull'uso di SVQL

L'architettura di implementazione descritta precedentemente fornisce la possibilità di effettuare direttamente le query SVQL sulle istanze MPEG-7. Questa caratteristica permette l'interrogazione MPEG-7 su fonti differenti (database, stream, ecc.); consente di interrogare direttamente le descrizioni parziali MPEG-7 durante la fase di produzione senza avere bisogno di tradurle ed immagazzinarle basandosi sul *Semantic Views Document Model*; e per concludere, permette un'estensione più facile del *Semantic Views Model*, poiché il mapping fra MPEG-7 ed il *Semantic Views Model* viene rappresentato attraverso un insieme di funzioni XQuery. L'unica modifica da fare è allora sul nucleo di queste funzioni, sui dati immagazzinati restanti intatti. Tuttavia, questo metodo può anche avere un insieme di svantaggi. La traduzione al volo delle istanze MPEG-7 in istanze semantiche di viste può rendere più onerosa l'interrogazione in termini di tempo e di memoria. Il compromesso fra queste differenti misure (flessibilità ed estensibilità da un lato, memoria e tempo dall'altro lato) può determinare se sia meglio realizzare la creazione del documento semantico di viste al volo o in anticipo. Un'interfaccia di interrogazione di notizie TV basata su SVQL è fornita dal progetto sperimentale COALA (Content Oriented Audiovisual Library Access) che fornisce vari strumenti sia per l'indicizzazione che per il recupero dei programmi audiovisivi nell'azienda di broadcasting TSR. Lo strumento di indicizzazione di COALA chiamato LogCreator crea la descrizione MPEG-7 dei programmi di notizie TV. Queste descrizioni sono immagazzinate in un database nativo XML, TAMINO. Lo strumento di interrogazione delle viste semantiche fornisce un'interfaccia adattata per l'espressione delle query SVQL (figura 27). Le query sono risolte usando Quip, il processore di XQuery fornito da TAMINO.

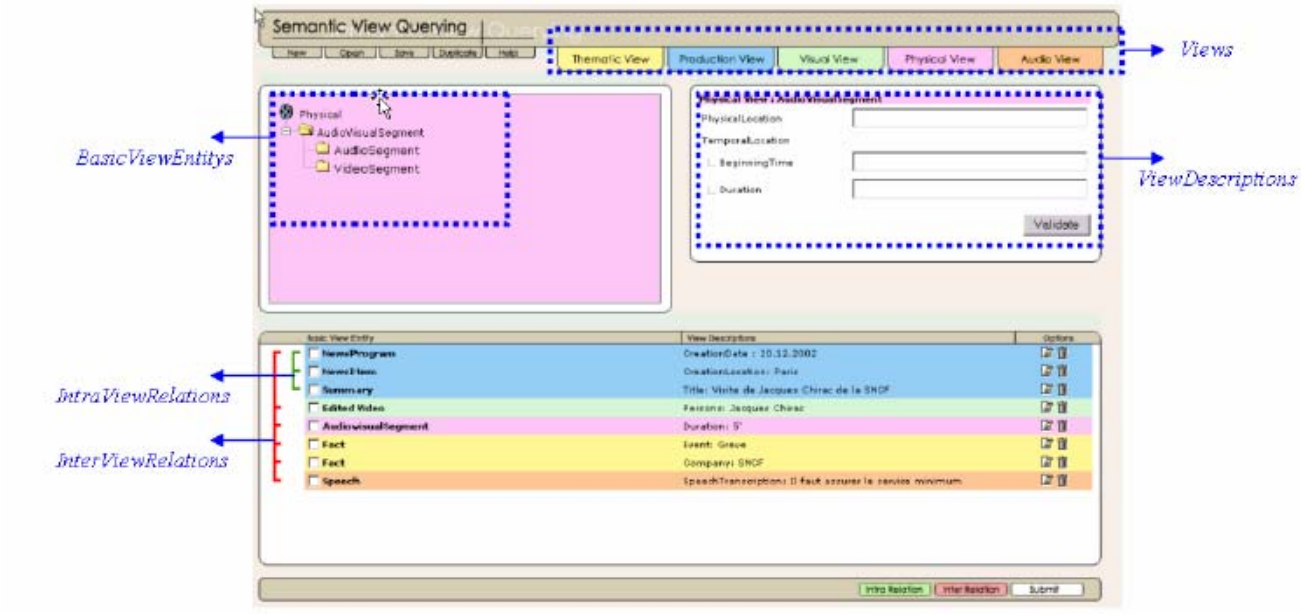


Figura 27 - Interfaccia di interrogazione SQVL in COALA

Parte II

Tecniche per l'esecuzione di interrogazioni sul contenuto in collezioni MPEG-7

5 IL SISTEMA XSITER

In questo capitolo si descriverà il sistema XSiter nel suo insieme, analizzando nel dettaglio la sua architettura, in previsione di dover effettuare al suo interno delle modifiche strutturali che si renderanno necessarie per realizzare il progetto. È fondamentale quindi eseguire uno studio approfondito sulle sue funzionalità: le sue strutture e i suoi singoli algoritmi dovranno essere conosciuti alla perfezione per poterli adattare alle nuove richieste di esecuzione di interrogazioni sul contenuto. Nel capitolo seguente ci si concentrerà invece sui concetti per realizzare dal punto di vista operativo quella che è la funzione alla base di tutto il progetto: la ricerca di similarità. Dalla unione di queste conoscenze sarà possibile eseguire sperimentalmente interrogazioni in collezioni MPEG-7 in maniera efficiente.

5.1 Generalità

Il sistema XSiter (XML SIgnaTure Enhanced Retrieval) è un query processor XML nativo ed estendibile che fornisce prestazioni molto alte in scenari di interrogazione XML generali. È un sistema capace di immagazzinare, gestire ed interrogare collezioni di documenti XML di qualsiasi numero e dimensione, è stato implementato sulla piattaforma Java ed ha un'architettura flessibile capace di sfruttare diversi tipi di signature (standard o estesa) e di fornire diversi algoritmi per poter effettuare diversi tipi di ricerca.

5.2 Architettura

La figura 28 mostra l'architettura astratta di XSiter. Il sistema è composto essenzialmente da tre sottosistemi, che rispettivamente, dalla parte superiore alla parte inferiore, gestiscono l'interazione con l'utente (*GUI System*), il processo di importazione dei documenti e delle query e l'elaborazione delle stesse (*Core System*) ed infine la persistenza dei documenti gestiti (*Store System*).

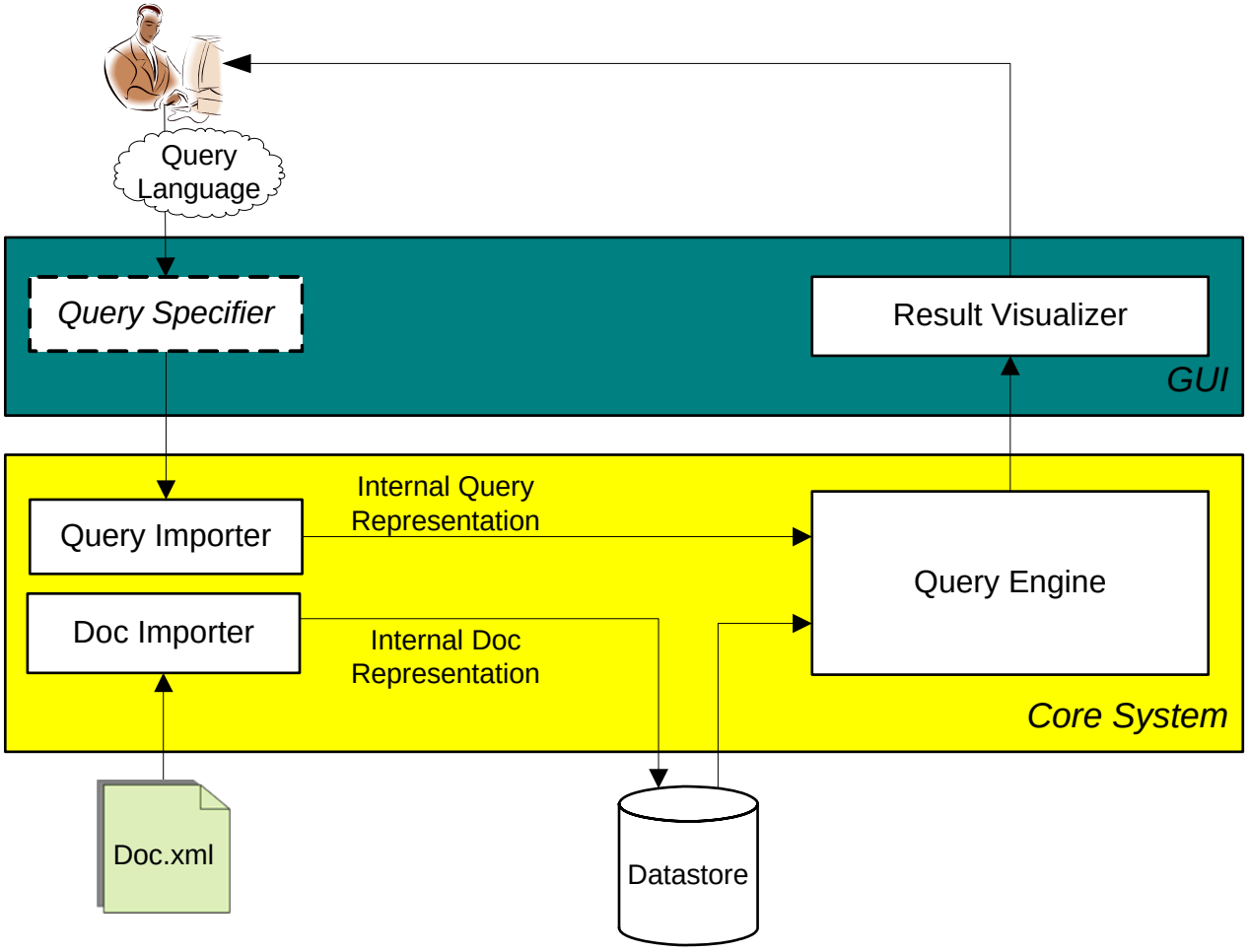


Figura 28 – Architettura astratta di XSiter

Nel GUI System l'utente interagisce con il sistema sottomettendo delle query e, dopo che il sistema le ha processate, esaminando i loro risultati. Il componente *Query Specifier* dovrebbe aiutare l'utente a specificare le query, cosa che viene ottenuta scrivendo il suo codice in uno specifico linguaggio di query. Dato che attualmente il sistema non supporta un linguaggio di query, quale ad esempio XQuery, le query vengono fornite come documenti che rappresentano la risposta richiesta e specificando quale tipo di ricerca (ordinata o non ordinata) deve essere effettuata. Una volta che la query è stata processata, i suoi risultati vengono presentati all'utente attraverso il componente *Result Visualizer* insieme ad un breve riassunto dei passi della computazione. Nell'implementazione corrente i risultati vengono rappresentati come una lista di elementi di albero che rappresentano le occorrenze del pattern richiesto nel database. Il sottosistema core calcola tutti i risultati per la query specificata nel database corrente ma, per limitare lo sforzo del componete visualizzatore, solo un numero limitato di risultati viene effettivamente presentato all'utente.

Prima che l'utente possa sottomettere query, egli ha bisogno di settare il sistema specificando alcuni parametri (come per esempio il massimo numero di risultati che devono essere visualizzati) e di aprire un insieme di documenti appena o precedentemente creati (chiamato *Datastore*) su cui verrano valutate le query che saranno sottomesse.

Lo strato centrale, chiamato Core System, riguarda il query processing. Il principale componente del Core System è sicuramente il *Query Engine* che processa una singola query su un Datastore specificato. Questo motore necessita di una query e di un datastore come input, tuttavia non gestisce direttamente documenti e query nella loro forma originale (che si assume essere un semplice file di testo o uno stream di caratteri). I documenti e le query per essere usati dal motore devono essere prima convertiti in una rappresentazione interna; questa è la ragione per cui il Core System include due componenti, chiamati *Query Importer* e *Doc Importer*, che trasformano rispettivamente query e documenti nella loro rappresentazione interna.

Infine vi è lo Store System, sistema che gestisce la persistenza dei datastore. Concettualmente esso mantiene una collezione di datastore cioè la collezione dei datastore precedentemente creati e salvati. Ogni datastore salvato potrebbe essere usato per un processamento di query o potrebbe essere aggiornato aggiungendo nuovi documenti. Poiché il processo di trasformazione da un documento alla sua rappresentazione interna potrebbe impiegare molto tempo (il tempo è proporzionale alla dimensione del documento e dipende da quali e quanti indici l'utente ha deciso di costruire sul documento) è conveniente effettuarlo una volta sola, immagazzinare il suo risultato e poi usarlo tante volte quante ne ha bisogno l'utente. Inoltre è anche possibile creare datastore volatili. Il processo di importazione dei documenti avviene quindi offline in quanto viene assunto per il query processing che i documenti sui quali la query deve essere processata siano sempre disponibili in una rappresentazione interna. Ovviamente l'importazione delle query è un processo online poiché non si conoscono le query fino a che l'utente non le sottomette.

In figura 29 possiamo ora vedere più in dettaglio il Query Engine. Prima di tutto vengono selezionati i documenti che devono essere processati, infatti poiché il datastore può immagazzinare documenti eterogenei, alcuni potrebbero essere inutili per la query. Poi per ogni documento viene applicato l'appropriato algoritmo di pattern matching. Quindi il processore di query scansiona il range della signature del documento di input stabilito dal Range Filter. Infine durante il matching il processore può trarre vantaggio dagli indici di contenuto disponibili per evitare di scansionare regioni vuote della signature del documento di input.

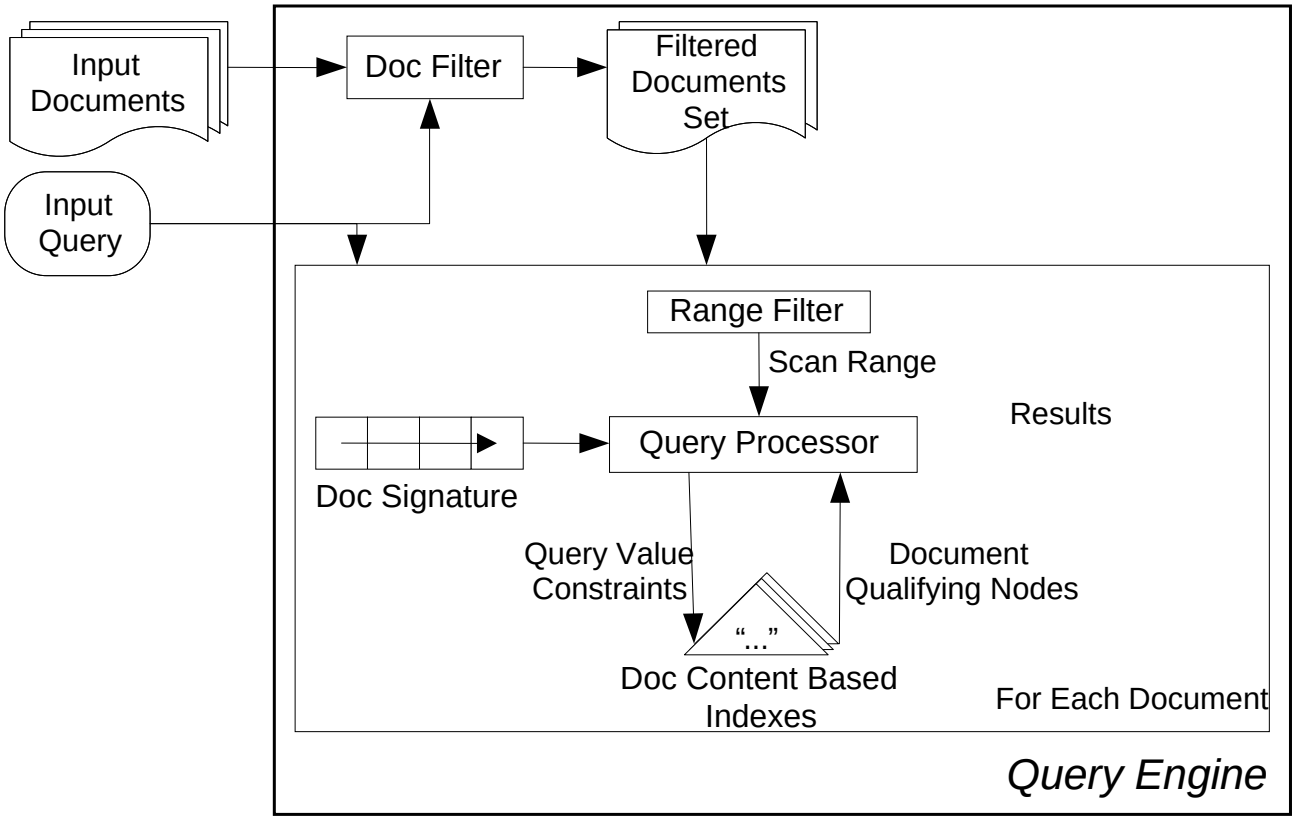


Figura 29 - Architettura del Query Engine

5.3 Strutture datastore e rappresentazioni interne

Essenzialmente, un *datastore* è una collezione usata per il raggruppamento dei documenti concettualmente collegati e per mantenere le loro rappresentazioni persistenti fra differenti sessioni di query. Per poter essere processati dal motore di query, i documenti e le query vengono trasformate in una *rappresentazione interna*. Tale rappresentazione è la chiave dell'efficienza del sistema ed è sfruttata da algoritmi ad-hoc di query processing.

Le query ed i documenti sono trasformati in una rappresentazione quasi omogenea. La rappresentazione interna scelta si rivolge alle questioni principali necessarie per l'interrogazione dei documenti XML, che consistono fondamentalmente nell'individuazione delle porzioni di documenti che obbediscono ai vincoli strutturali e di contenuto ed è costituita di quattro parti principali (si veda la parte destra di figura 30):

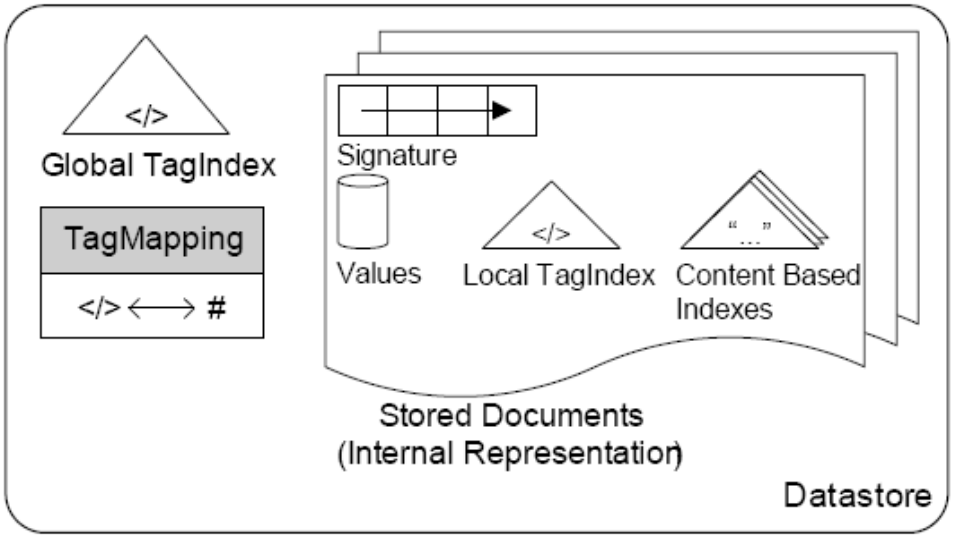


Figura 30 - Struttura e contenuto di un datastore

- una rappresentazione compatta (*Signature*) basata su uno schema di numerazione che è usato per risolvere il tree pattern matching (*vincoli strutturali*);
- un sommario semplice di documento (*Local TagIndex*) viene usato come un filtro per limitare lo spazio di ricerca. In particolare, per ogni tag che è presente in un documento, vengono mantenute la prima e l'ultima posizione del documento;
- la rappresentazione compatta non include *valori* che sono immagazzinati separatamente e sono valutati soltanto per necessità (*vincoli di valore*);
- infine, i valori, i contenuti degli elementi o i valori di attributo, possono essere indicizzati (*Content Based Indexes*) per accelerare il processo di ricerca. Tale parte è facoltativa ed è generata secondo i bisogni dell'utente.

Per concludere, insieme alle rappresentazioni interne del documento, in un datastore vengono mantenute anche due strutture globali condivise (si veda la parte sinistra di figura 30), chiamate *Global TagIndex* e *TagMapping*. Global TagIndex tiene traccia di quale tag è presente in ogni documento gestito e viene usato eliminare rapidamente quei documenti che non possono contenere corrispondenze per una particolare query. TagMapping, invece, fornisce il mapping fra i tag testuali ed i numeri unici corrispondenti (id_s), che sono usati per immagazzinare i tag nella maniera più compatta possibile.

5.4 Algoritmi specializzati, indici e caratteristiche di memorizzazione

Ora analizziamo alcune caratteristiche del sistema più dettagliatamente. Come signature del documento viene sfruttata la struttura *tree signature*, che permette di effettuare una piccola ma sufficiente rappresentazione delle strutture di albero in grado di decidere e risolvere i vincoli strutturali delle query. Come schema di codificazione vengono usati i rank di *pre-ordine* e di *post-ordine*, che permettono di verificare e scoprire facilmente le differenti relazioni strutturali fra gli elementi, quali quelle di ancestor, descendant, preceding e following.

Sfruttando le proprietà delle tree signature ed in particolare la loro natura sequenziale, si possono costruire e sfruttare una serie di algoritmi ad-hoc con prestazioni ottimali di I/O.
XSiter fornisce differenti algoritmi che risolvono differenti problemi di pattern matching XML: uno che riguarda il percorso e due che riguardano alberi. Il *path matching* è il tipo più semplice di pattern matching, nel quale la query viene espressa come una serie di elementi disposti in una semplice struttura lineare. Query più complesse sono gli alberi etichettati ordinati (*ordered tree pattern matching*) nel quale è importante l'ordine fra gli elementi e gli alberi etichettati non ordinati (*unordered tree pattern matching*). Il supporto di tutti i diversi tipi di pattern matching è essenziale, specialmente per le interrogazioni multimediali. In effetti, mentre ci sono certamente situazioni dove l'ordered tree pattern matching riflette perfettamente i fabbisogni informativi degli utenti, ce ne sono molte altre che preferiscono considerare gli alberi di query come non ordinati.

Oltre al supporto dei vincoli strutturali, XSiter supporta completamente la valutazione di query con vincoli di valore, che è un requisito della massima importanza. Il database valori mantiene semplicemente per ogni elemento/attributo il valore corrispondente. D'altra parte, i valori possono anche essere organizzati in Content Based Indexes per accelerare il processo di recupero. Un Content Based Index è costruito su un tag specifico per un tipo specifico di ricerca (per esempio indici per un match esatto o parziale) e la sua interfaccia astratta permette agli algoritmi di ricerca di recuperare attraverso di lui tutti gli elementi il cui contenuto soddisfa la condizione specificata. Anche se gli indici basati sul contenuto non sono obbligatori, è fortemente raccomandato costruirli per gli elementi o gli attributi che sono interrogati più frequentemente. Il sistema include implementazioni efficienti delle strutture di indice più efficaci disponibili per le ricerche esatte (*exact match*), quali gli inverted index per i dati testuali. Le versioni standard degli algoritmi di matching forniscono il supporto al valore anche quando non sono disponibili i dati specifici di indice sugli elementi interrogati.

D'altra parte, il sistema fornisce anche le versioni specializzate al valore di algoritmi differenti, che traggono massimo vantaggio dagli indici basati sul contenuto disponibili, sviluppati sul contenuto dei nodi del documento. Essi migliorano gli algoritmi standard in due modi: riconoscendo gli elementi più inutili e evitando l'esame di regioni che non contengono alcun match valido. Questo aumenta ulteriormente le prestazioni del sistema per risolvere interrogazioni che trattano valori.

In generale, tutti gli algoritmi di XSiter processano efficientemente le strutture dati supportate in maniera sequenziale, saltando per quanto possibile le zone ovvie di nessuna risposta alla query. La chiave alla loro efficienza è saltare tanti dati sottostanti quanto possibile ed allo stesso tempo non tornare mai indietro nella sequenza processata. I depositi che coinvolgono tantissimi documenti sono efficientemente gestiti ed interrogati, anche grazie ai filtri di documento che sfruttano Global TagIndex Data. Inoltre è pure disponibile un filtro basato sui Local TagIndexes, che viene usato per limitare le parti di un documento che devono essere esplorate per risolvere una particolare query. Queste caratteristiche, insieme all'utilizzo minimo di memoria degli algoritmi, rendono il sistema adatto ad interrogare ed a gestire documenti molto grandi.

XSiter è attualmente implementato come sistema general purpose, che significa che attualmente sono state applicate piccole ottimizzazioni speciali di dominio ma l'architettura è stata sviluppata per essere estendibile in maniera semplice. In particolare, gli algoritmi correnti usano un livello di astrazione molto elevato di accesso di indice di contenuto che permette sostanzialmente di usare differenti tipi di indici senza modificare gli algoritmi di ricerca. In aggiunta, le strutture specializzate di indice possono essere integrate e sfruttate facilmente nel sistema per rispondere meglio ai diversi bisogni di applicazione.

6 RICERCHE DI SIMILARITÀ IN SPAZI METRICI: M-TREE

Per poter realizzare il recupero basato sul contenuto e quindi soddisfare i fabbisogni informativi degli utenti, è di vitale importanza supportare efficacemente ed efficientemente il processo di recupero per determinare quali porzioni del database sono rilevanti alle richieste degli utenti. Vi è quindi il bisogno di tecniche di indicizzazione in grado di supportare l'esecuzione di *query di similarità*. Poiché le applicazioni multimediali richiedono tipicamente *funzioni di distanza* complesse per misurare le similarità di caratteristiche multidimensionali, quali forma, tessitura, colore, modelli di immagine, suono, testo, valori fuzzy, valori d'insieme, dati di sequenza, ecc., sono stati considerati *metodi di accesso multidimensionali spaziali* (*SAM*), quali l'R-tree e le relative varianti, per poter indicizzare tali dati. Tuttavia, l'applicabilità dei SAM è limitata dalle seguenti assunzioni su cui contano tali strutture:

1. gli oggetti devono, per gli scopi di indicizzazione, essere rappresentati per mezzo dei valori di feature in uno spazio vettoriale multidimensionale;
2. la (dis)similarità di due oggetti qualsiasi deve essere basata su una funzione di distanza che non introduce alcuna correlazione (o "cross-talk") fra i valori di feature. Più precisamente, deve essere usata una metrica L_p, quale la distanza euclidea.

Inoltre, da un punto di vista delle prestazioni, i SAM suppongono che il confronto delle chiavi (valori di feature) sia un'operazione insignificante riguardo al costo di accesso ad una pagina del disco, cosa che non avviene sempre nelle applicazioni multimediali. Di conseguenza, nel progetto di queste strutture non è stato fatto alcun tentativo di ridurre il numero dei calcoli di distanza.

6.1 Alberi e spazi metrici

Un approccio più generale al problema di "indicizzazione di similarità" ha guadagnato una certa popolarità negli ultimi anni, conducendo allo sviluppo di cosiddetti alberi metrici (*metric trees*). Gli alberi metrici considerano soltanto le distanze relative degli oggetti (piuttosto che le loro posizioni assolute in uno spazio multidimensionale) per organizzare e partizionare lo spazio di ricerca e richiedono solo che la funzione usata per misurare la distanza (dissimilarità) fra gli oggetti sia una *metrica* di modo che si possa applicare la proprietà della disuguaglianza triangolare e possa essere usata restringere lo spazio di ricerca.

Anche se l'efficacia degli alberi metrici è stata dimostrata chiaramente, i progetti correnti soffrono il fatto di essere intrinsecamente statici, che limita la loro applicabilità negli ambienti database dinamici. Al contrario dei SAM, gli alberi metrici conosciuti hanno provato solamente a ridurre il numero di calcoli di distanza richiesti per rispondere ad una query, non prestando alcuna attenzione ai costi di I/O.

Indicizzare uno *spazio metrico* significa fornire un supporto efficiente per rispondere a *query di similarità*, cioè query il cui scopo è recuperare oggetti database che sono "simili" ad un oggetto di riferimento (query) e dove la (dis)similarità fra gli oggetti è misurata da una specifica funzione di distanza metrica d. Formalmente uno spazio metrico è una coppia, $\mathcal{M} = (\mathcal{D}, d)$, dove $\mathcal{D}$ è un dominio dii valori di feature (le chiavi di indicizzazione) e d è una funzione di distanza con le seguenti proprietà:

1. $d(O_x, O_y) = d(O_y, O_x)$ (*simmetria*)
2. $d(O_x, O_y) > 0$, $(O_x \neq O_y)$ e $d(O_x, O_x) = 0$ (*non negatività*)
3. $d(O_x, O_y) < d(O_x, O_z) + d(O_z, O_y)$ (*disuguaglianza triangolare*)

In linea di principio, ci sono due tipi di base di query di similarità: la *range query* e la *k nearest neighbors query*.

Definizione range query.
Dato un oggetto di query $Q \in \mathcal{D}$ e una massima distanza di ricerca $r(Q)$, la range query **range**(Q, $r(Q)$) seleziona tutti gli oggetti indicizzati O_j tali che $d(O_j, Q) \leq r(Q)$.

Definizione k nearest neighbors (k-NN) query.
Dato un oggetto di query $Q \in \mathcal{D}$ e un numero intero $k \geq 1$, la k-NN query **NN**(Q, k) seleziona i k oggetti indicizzati che hanno la più corta distanza da Q.

Ci sono già stati alcuni tentativi di affrontare il difficile problema di indicizzazione dello spazio metrico. L'algoritmo FastMap trasforma una matrice di distanze accoppiate in un insieme di punti basso-dimensionali, che possono poi essere indicizzati da un SAM. Tuttavia, FastMap presuppone un insieme di dati statico ed introduce errori di approssimazione nel processo di mapping. Il *Vantage Point* (VP) tree partiziona un insieme di dati a seconda delle distanze che hanno gli oggetti riguardo ad un punto di riferimento (di vantaggio).

Il valore medio di tali distanze viene usato come separatore per suddividere gli oggetti in due sottoinsiemi bilanciati, a cui può essere applicata ricorsivamente la stessa procedura. L'MVP-tree estende questa idea usando multipli punti di vantaggio e sfrutta le distanze pre-calcolate per ridurre il numero di calcoli di distanza a tempo di query. Il progetto GNAT applica uno stile di partizionamento differente chiamato iperpiano generalizzato (*generalized hyperplane*). Nel caso generale, vengono scelti due oggetti di riferimento e ciascuno degli oggetti rimanenti viene assegnato all'oggetto di riferimento più vicino. I sottoinsiemi così ottenuti possono essere divisi di nuovo in maniera ricorsiva, se necessario.

Poiché tutte le organizzazioni suddette costruiscono gli alberi per mezzo di un processo ricorsivo dall'alto in basso, non vi è garanzia che questi alberi rimanghino equilibrati in caso di inserimenti e cancellazioni e richiedono riorganizzazioni costose per impedire la degradazione delle prestazioni.

6.2 L'M-tree

L'M-tree è un albero bilanciato, esplicitamente progettato per essere integrato con altri metodi di accesso nei sistemi database, e in grado di occuparsi di file di dati dinamici (come tale non richiede riorganizzazioni periodiche). L'M-tree può indicizzare oggetti che usano caratteristiche confrontate con le funzioni di distanza che non entrano in uno spazio vettoriale nè usano una metrica L_p; in questo modo estende considerevolmente i casi per cui è possibile un efficiente processamento di query. Poiché il progetto dell'M-tree è ispirato da entrambi i principi degli alberi metrici e dei metodi di accesso database, l'ottimizzazione delle prestazioni riguarda sia i costi di CPU (calcoli della distanza) che di I/O.

La sfida di ricerca che ha condotto al progetto dell'M-tree doveva unire i vantaggi dei SAM bilanciati e dinamici, con le capacità degli alberi metrici statici per indicizzare gli oggetti usando le caratteristiche e le funzioni di distanza che non entrano in uno spazio vettoriale e sono vincolati soltanto dai postulati metrici. L'M-tree partiziona gli oggetti sulla base delle loro distanze relative, misurate da una funzione di distanza specifica *d*, e memorizza questi oggetti in nodi di dimensione fissata, che corrispondono alle regioni vincolate dello spazio metrico. L'M-tree è completamente parametrico sulla funzione di distanza *d*, in modo che l'implementazione della funzione è una *black-box* per l'M-tree.

6.2.1 La struttura dei nodi

I nodi foglia di ogni M-tree memorizzano tutti gli oggetti indicizzati (database), rappresentati dalle loro chiavi o caratteristiche, mentre i nodi interni memorizzano i cosiddetti oggetti di percorso (*routing object*). Un routing object è un oggetto database a cui viene assegnato un ruolo di percorso da un'algoritmo specifico di promozione. Per ogni routing object O_r vi è un puntatore associato, chiamato $ptr(T(O_r))$, che punta alla radice di un sotto-albero, $T(O_r)$, denominato *covering tree* di O_r. Tutti gli oggetti nel covering tree di O_r sono all'interno della distanza $r(O_r)$ da O_r, che è denominata *covering radius* di O_r e costituisce una parte dell'entrata O_r in un particolare nodo dell'M-tree. Infine, un routing object O_r è associato con una distanza da $P(O_r)$, il suo oggetto genitore, che è il routing object che riferisce il nodo dove l'entrata O_r è immagazzinata. Ovviamente, questa distanza non è definita per le entrate nella radice dell'M-tree. Le informazioni generali per un'entrata routing object sono ricapitolate nella seguente tabella.

O_r	(valore del) routing object
$ptr(T(O_r))$	puntatore alla radice of $T(O_r)$
$r(O_r)$	covering radius of O_r
$d(O_r, P(O_r))$	distanza di O_r dai suoi genitori

Tabella 1 - Entrata routing object

Un'entrata per un oggetto database O_j in una foglia è abbastanza simile a quella di un routing object, ma non è necessario nessun covering radius ed il campo puntatore memorizza l'identificatore corrente dell'oggetto (oid), che è usato per fornire l'accesso all'intero oggetto che risiede possibilmente in un file dati separato. Ricapitolando, le entrate nei nodi foglia sono strutturate come si vede in seguito.

O_j	(valore del) DB object
$oid(O_j)$	identificatore dell'oggetto
$d(O_j, P(O_j))$	distanza di O_j dai suoi genitori

Tabella 2 - Entrata DB object

6.2.2 Processing delle query di similarità

Prima di presentare gli algoritmi specifici per la costruzione dell'M-Tree, mostriamo come le informazioni memorizzate nei nodi sono usate per processare le query di similarità. Anche se le prestazioni degli algoritmi di ricerca sono in gran parte influenzate dalla costruzione dell'M-Tree, la correttezza e la logica della ricerca sono indipendenti da tali aspetti. In tutti gli algoritmi, l'obiettivo è quello di ridurre, oltre al numero di nodi raggiunti, anche il numero di calcoli della distanza necessari per eseguire le query. Ciò è particolarmente rilevante quando la ricerca risulta essere maggiormente legata alla CPU piuttosto che all'I/O, che potrebbe essere il caso delle funzioni di distanza computazionalmente onerose. A tal fine vengono usate tutte le informazioni riguardo alle distanze (pre-calcolate) immagazzinate nei nodi dell'M-Tree, cioè $d(O_i, P(O_i))$ e $r(O_i)$, per applicare efficacemente la disuguaglianza triangolare.

6.2.2.1 Range Query

La range query $(Q, r(Q))$ seleziona tutti gli oggetti DB tali che $d(Oj, Q) \leq r(Q)$. L'algoritmo RS comincia dal nodo radice ed in maniera ricorsiva attraversa tutti i percorsi che non possono essere esclusi dal condurre agli oggetti che soddisfano la suddetta disuguaglianza.

RS(*N*:node, *Q*:query_object, *r*(*Q*):search_radius)
{ let O_p be the parent object of node *N*;
  if *N* is not a leaf
  then { $\forall$ O_r in *N* do:
        if $|d(O_p,Q) - d(O_r,O_p)| \leq r(Q) + r(O_r)$
        then { Compute $d(O_r,Q)$;
              if $d(O_r,Q) \leq r(Q) + r(O_r)$
              then RS(*$ptr(T(O_r))$,Q,$r(Q)$); }}
  else { $\forall$ O_j in *N* do:
        if $|d(O_p,Q) - d(O_j,O_p)| \leq r(Q)$
        then { Compute $d(O_j,Q)$;
              if $d(O_j,Q) \leq r(Q)$
              then add $oid(O_j)$ to the result;}}}

Poiché quando si accede al nodo N, è già stata calcolata la distanza fra Q e O_p, l'oggetto genitore di N, è possibile eliminare un sotto-albero senza calcolare affatto alcuna nuova distanza. La condizione applicata per l'eliminazione è la seguente.

Lemma 1. Se $d(O_r,Q) > r(Q) + r(O_r)$, allora, per ogni oggetto O_j in $T(O_r)$, si ha che $d(O_j,Q) > r(Q)$. Così $T(O_r)$ può essere eliminato in modo sicuro dalla ricerca.

Infatti poiché $d(O_j,Q) \geq d(O_r,Q) - d(O_j,O_r)$ (per la disuguaglianza triangolare) e $d(O_j,O_r) \leq r(O_r)$ (per la definizione di covering radius), si ha $d(O_j,Q) \geq d(O_r,Q) - r(O_r)$. Poiché per ipotesi si ha $d(O_r,Q) - r(O_r) > r(Q)$, il risultato viene dimostrato.

Per applicare il lemma precedente, la distanza $d(O_r,Q)$ deve essere calcolata. Questo può essere evitato approfittando del seguente lemma.

Lemma 2. Se$|d(O_p,Q) - d(O_r,O_p)| > r(Q) + r(O_r)$, allora $d(O_r,Q) > r(Q) + r(O_r)$.

Questa è una diretta conseguenza della disuguaglianza triangolare che garantisce sia che $d(O_r,Q) \geq d(O_p,Q) - d(O_r,O_p)$ (figura 32a) che $d(O_r,Q) \geq d(O_r,O_p) - d(O_p,Q)$ (figura 32b). Lo stesso principio di ottimizzazione è applicato pure ai nodi foglia. L'unico caso in cui le distanze da calcolare sono necessarie si ha quando ci si occupa del nodo della radice, per cui O_p non è definito.

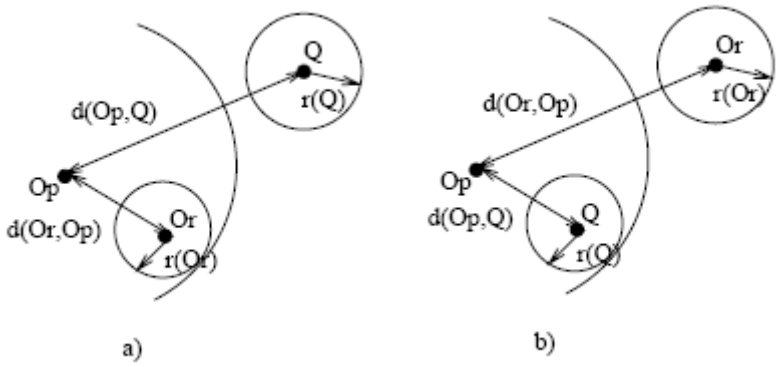


Figura 31 - Applicazione del lemma 2 per evitare calcoli di distanza

6.2.2.2 Nearest Neighbor Query

Se si assume che almeno k oggetti siano indicizzati dall'M-Tree, l'algoritmo k-NN Search recupera i k vicini più prossimi di un oggetto di query Q. Viene usata una tecnica abbastanza simile a quella progettata per altre strutture quali gli R-Tree, che utilizza due strutture globali: una coda di priorità, PR, e un array di k elementi, NN, che conterrà il risultato alla conclusione dell'esecuzione. PR è una coda di puntatori ai sotto-alberi attivi, cioè ai sotto-alberi in cui possono essere trovati oggetti qualificanti. Con il puntatore alla radice del sotto-albero $T(O_r)$, viene inoltre mantenuto un limite più basso, $d_{min}(T(O_r))$, sulla distanza di ogni oggetto da Q in $T(O_r)$. Il limite più basso è dato da:

$$d_{min}(T(O_r)) = \max\{d(O_r, Q) - r(O_r), 0\}$$

poiché nessun oggetto in $T(O_r)$ può avere una distanza da Q minore di $d(O_r, Q) - r(O_r)$. Questi limiti sono usati dalla funzione ChooseNode per estrarre da PR il prossimo nodo da esaminare. Poiché il criterio di eliminazione dei sottoalberi del k-NN Search è dinamico (il raggio di ricerca è la distanza fra Q ed i suoi correnti k vicini più prossimi), l'ordine in cui i nodi sono visitati può influenzare le prestazioni. Il criterio euristico implementato dalla funzione ChooseNode è quello di selezionare il nodo per cui il limite più basso d_{min} è minimo. Secondo osservazioni sperimentali, altri criteri non portano a migliorare le prestazioni.

ChooseNode(PR:priorità_queue): node
{ let $d_{min}(T(O_r^*)) = \min\{d_{min}(T(O_r))\}$,
considering all the entries in PR;
Remove entry [$ptr(T(O_r^*))$, $d_{min}(T(O_r^*))$] from PR;
return * $ptr(T(O_r^*))$; }

Alla fine dell'esecuzione, l'i-esima entrata dell'array NN avrà valore NN[i] = [oid(O_j), $d(O_j, Q)$], con O_j che è l'i-esimo vicino più prossimo di Q. Il valore di distanza nell'entrata i-esima viene denotato come d_i, di modo che d_k sia il più grande valore di distanza in NN. Chiaramente, d_k svolge il ruolo di un raggio di ricerca dinamico, poiché ogni sotto-albero per cui si ha $d_{min}(T(O_r)) > d_k$ può essere eliminato in modo sicuro. Le entrate dell'array NN sono inizialmente settate a NN[i] = [-,∞] (i = 1, . . . , k), cioè gli oid non sono definiti e $d_i = \infty$.

Come la ricerca comincia ed i nodi (interni) sono raggiunti, l'idea è quella di calcolare, per ogni sotto-albero $T(O_r)$, un limite superiore, $d_{max}(T(O_r))$, sulla distanza di ogni oggetto da Q in $T(O_r)$. Il limite superiore è settato a:

$$d_{max}(T(O_r)) = d(O_r, Q) + r(O_r)$$

Si consideri il caso più semplice in cui si hanno $k = 1$, due sotto-alberi, $T(O_{r1})$ e $T(O_{r2})$ e si supponga che $d_{max}(T(O_{r1})) = 5$ e $d_{min}(T(O_{r2})) = 7$. Poiché $d_{max}(T(O_{r1}))$ garantisce che un oggetto, la cui distanza da Q è al massimo 5, esiste in $T(O_{r1})$, $T(O_{r2})$ può essere eliminato dalla ricerca. I limiti d_{max} sono inseriti nelle appropriate posizioni nell'array NN, lasciando il campo oid non definito. L'algoritmo k-NN_Search è data sotto.

k-NN Search(*T*:root_node, *Q*:query_object, *k*:integer)
{ PR =[T,-];
  for i = 1 to *k* do: NN[i] =[-,∞];
  while PR ≠ 0 do:
  { Next_Node = ChooseNode(PR);
    k-NN_NodeSearch(Next Node, *Q*, *k*); }}

Il metodo k-NN_NodeSearch effettua la maggior parte della logica di ricerca. Su un nodo interno, in primo luogo determina i sotto-alberi attivi e li inserisce nella coda PR. Quindi, se necessario, chiama la funzione NN_Update per ottenere un inserimento ordinato nell'array NN e riceve indietro un valore di d_k (possibilmente nuovo). Questo viene poi usato per rimuovere da PR tutti i sotto-alberi per cui il limite più basso d_{min} eccede d_k. Azioni simili sono realizzate nei nodi foglia. In entrambi i casi viene inoltre applicata ottimizzazione per ridurre il numero di calcoli di distanza per mezzo di distanze pre-calcolate dall'oggetto genitore.

k-NN NodeSearch(*N*:node, *Q*:query_object, *k*:integer)
{ let O_p be the parent object of node *N*;
  if *N* is not a leaf then
  { ∀ O_r in *N* do:
    if $|d(O_p, Q) - d(O_r, O_p)| \leq d_k + r(O_r)$ then
    { Compute $d(O_r, Q)$;
      if $d_{min}(T(O_r)) \leq d_k$ then

{ add [$ptr(T(O_r))$, $d_{min}(T(O_r))$] to PR;
if $d_{max}(T(O_r)) < d_k$ then
{ d_k = NN_Update([-, $d_{max}(T(O_r))$]);
Remove from PR all entries
for which $d_{min}(T(O_r)) > d_k$; }}}}
else /* N is a leaf */
{ $\forall$ O_j in N do:
if $|d(O_p, Q) - d(O_j, O_p)| \leq d_k$ then
{ Compute $d(O_j, Q)$;
if $d(O_j, Q) \leq d_k$ then
{ d_k =NN_Update([$oid(O_j)$, $d(O_j, Q)$]);
Remove from PR all entries
for which $d_{min}(T(O_r)) > d_k$; }}}}

6.2.3 La costruzione dell'M-tree

Gli algoritmi per la costruzione dell'M-tree specificano come gli oggetti sono inseriti e cancellati e come vengono gestiti gli overflow ed underflow dei nodi. L'algoritmo Insert discende ricorrentemente l'M-tree per individuare il nodo foglia più adatto per inserire un nuovo oggetto, O_n, che può innescare una divisione (*split*) se la foglia è piena. Il criterio razionale usato per determinare il nodo foglia più adatto è discendere, ad ogni livello dell'albero, lungo un sotto-albero, $T(O_r)$, per cui non è necessario alcun allargamento del covering radius, cioè $d(O_r, O_n) \leq r(O_r)$. Se esistono più sotto-alberi con questa proprietà, viene scelto quello per cui l'oggetto O_n è più vicino a O_r. Questa euristica prova ad ottenere sotto-alberi ben raggruppati, fatto che ha un effetto benefico sulle prestazioni. Se non esiste nessun routing object per cui $d(O_r, O_n) \leq r(O_r)$, la scelta sarà quella di minimizzare l'aumento del covering radius, $d(O_r, O_n) - r(O_r)$. Ciò è strettamente collegato con il criterio euristico che suggerisce di minimizzare il volume totale coperto dai routing object nel nodo corrente.

Insert(N:node, entry(O_n):M-tree_entry)
{ let $\mathcal{N}$ be the set of entries in node N;
if N is not a leaf then

{ let $\mathcal{N}_{in}$ = entries such that $d(O_r, O_n) \leq r(O_r)$;
if $\mathcal{N}_{in} \neq 0$
then let entry(O_r^*) $\in \mathcal{N}_{in}$: $d(O_r^*, O_n)$ is minimum;
else { let entry(O_r^*) $\in \mathcal{N}$: $d(O_r^*, O_n)$ - $r(O_r^*)$ is minimum;
let $r(O_r^*) = d(O_r^*, O_n)$; }
Insert(*ptr($T(O_r^*)$), entry(O_n)); }
else /* N is a leaf */
{ if N is not full
then store entry(O_n) in N
else Split(N, entry(O_n)); }}

La determinazione dell'insieme $\mathcal{N}_{in}$ (routing object per i quali non è necessario alcun allargamento del covering radius) può essere ottimizzato conservando i calcoli di distanza. Dal lemma 2 visto in precedenza, sostituendo O_n con Q e settando $r(O_n) \equiv r(Q) = 0$, deriva che:

$$\text{Se } |d(O_p, O_n) - d(O_r, O_p)| > r(O_r) \text{ allora } d(O_r, O_n) > (O_r)$$

da cui segue che $O_r \notin \mathcal{N}_{in}$. Si noti che questa ottimizzazione non può essere applicata al nodo radice.

6.2.4 La gestione degli split

Come qualunque altro albero bilanciato dinamico, l'M-tree si sviluppa dal basso verso l'alto. L'overflow di un nodo N viene gestito allocando un nuovo nodo, N', allo stesso livello di N, partizionando le entrate fra questi due nodi ed inviando (promuovendo) al nodo genitore, N_p, due routing object per riferire i due nodi. Quando la radice si divide, viene generata una nuova radice e l'M-tree cresce di un livello.

Split(N:node; E:M-tree_entry)
{ let $\mathcal{N}$= entries of node $N \cup$ {E};
if N is not the root then
let O_p be the parent of N, stored in N_p node;
Allocate a new node N';

Promote($\mathcal{N}$, O_{p1}, O_{p2});
Partition($\mathcal{N}$, O_{p1}, O_{p2}, $\mathcal{N}_1$, $\mathcal{N}_2$);
Store $\mathcal{N}_1$'s entries in N and $\mathcal{N}_2$'s entries in N';
if N is the current root
then { Allocate a new root node, N_p;
    Store entry(O_{p1}) and entry(O_{p2}) in N_p; }
else { Replace entry(O_p) with entry(O_{p1}) in N_p;
    if node N_p is full
    then Split(N_p, entry(O_{p2}))
    else store entry(O_{p2}) in N_p; }}

Il metodo Promote sceglie, secondo un certo criterio specifico, due routing object, O_{p1} ed O_{p2}, da inserire nel nodo genitore, N_p. Il metodo Partition divide le entrate del nodo che ha subito l'overflow (l'insieme $\mathcal{N}$) in due sottoinsiemi disgiunti, $\mathcal{N}_1$ e $\mathcal{N}_2$, che vengono poi immagazzinati rispettivamente nei nodi N e N'. Un'implementazione specifica dei metodi Promote e Partition definisce una cosiddetta *split policy*. Diversamente da altri progetti di alberi metrici (statici), ciascuno dei quali conta su uno specifico criterio per organizzare gli oggetti, l'M-tree offre la possibilità di implementare politiche di split alternative, per adattare le prestazioni ai bisogni della specifica applicazione. Senza riguardo alla politica di split specifica, deve essere conservata la semantica dei covering radii. Se il nodo da separare è una foglia, allora il covering radius di un oggetto promosso, detto O_{p1}, è settato a:

$$r(O_{p1}) = max\ \{d(O_j, O_{p1}) \mid O_j \in \mathcal{N}_1\}$$

mentre se l'overflow si presenta in un nodo interno:

$$r(O_{p1}) = max\ \{\ d(O_r, O_{p1}) + r(O_r) \mid O_r \in \mathcal{N}_1\}$$

che garantisce che $d(O_j, O_{p1}) \leq r(O_{p1})$ sia vera per ogni oggetto in $T(O_{p1})$.

6.2.5 Le politiche di split

La politica di divisione "ideale" dovrebbe promuovere gli oggetti O_{p1} ed O_{p2} e partizionare gli altri oggetti in modo tale che le due le regioni così ottenute abbiano il minimo "volume" e la minima "sovrapposizione". Entrambe le cose mirano a migliorare l'efficacia degli algoritmi di ricerca, poiché avere piccole regioni (basso volume) porta ad avere alberi ben raggruppati e riduce la quantità di *dead space* (spazio dove non c'è nessun oggetto presente) indicizzato, ed avere una piccola sovrapposizione (possibilmente nulla) fra le regioni riduce il numero di percorsi da attraversare per rispondere ad una query. Il criterio del volume minimo conduce ad escogitare politiche di split che cercano di minimizzare i valori dei covering radii, mentre quello della minima sovrapposizione suggerisce che, per valori fissi dei covering radii, la distanza fra gli oggetti di riferimento scelti dovrebbe essere massimizzata. Oltre ai requisiti suddetti, che sono abbastanza "standard" anche per i SAMs, dovrebbe anche essere considerato il possibile alto costo di CPU del calcolo delle distanze. Ciò suggerisce che vale la pena considerare persino le politiche native (per esempio una scelta casuale dei routing object), che comunque eseguono pochi calcoli di distanza.

6.2.5.1 La scelta dei routing object

Il metodo Promote determina, dato un insieme di entrate, $\mathcal{N}$, due oggetti da promuovere e memorizzare nel nodo genitore. Gli algoritmi specifici possono essere in primo luogo classificati a secondo che "confermino" l'oggetto genitore originale nel proprio ruolo oppure no.

Definizione. Una politica di split confermata sceglie come uno degli oggetti promossi, detto O_{p1}, l'oggetto O_p, cioè l'oggetto genitore del nodo diviso.

In altri termini, una politica di split confermata "estrae" solo una regione, centrata sul secondo routing object, O_{p2}, dalla regione che rimarrà ancora centrata in O_p. In generale, questo facilita l'esecuzione dello split e riduce il numero dei calcoli di distanza.

Alcune alternative per implementare Promote sono:

- **m_RAD:** L'algoritmo "minima somma dei raggi" è il più complesso in termini di calcoli di distanza.

Esso considera tutte le possibili coppie di oggetti e, dopo aver partizionato l'insieme delle entrate, promuove la coppia di oggetti per i quali è minima la somma dei covering radii, $r(O_{p1}) + r(O_{p2})$.

- **mM_RAD:** Questo algoritmo è simile a m_RAD ma minimizza il massimo dei due raggi.

- **M_LB_DIST:** Questa politica si differenzia dalle precedenti in quanto usa soltanto le distanze precalcolate memorizzate. Nella versione confermata, dove $O_{p1} \equiv O_p$, l'algoritmo determina O_{p2} come l'oggetto più lontano da O_p, che è $d(O_{p2}, O_p) = max_j \{d(O_j, O_p)\}$. L'acronimo sta per "Maximum Lower Bound on DISTance".

- **RANDOM:** Questa variante seleziona in maniera casuale l'oggetto o gli oggetti di riferimento. Sebbene non sia una strategia "compatta", è veloce e le sue prestazioni possono essere usate come riferimento per le altre politiche.

- **SAMPLING:** Questa è la politica RANDOM ma iterata su un campione di oggetti di dimensione $s > 1$. Per ognuna delle $s(s\text{-}1)/2$ coppie di oggetti nel campione, vengono distribuite le entrate e determinati i covering radii potenziali. La coppia per il quale il massimo risultante dei due covering radii è massimo viene poi selezionata. In caso di promozione confermata, sono provate solo s differenti distribuzioni.

6.2.5.2 La distribuzione delle entrate

Dato un insieme di entrate $\mathcal{N}$ e due routing object O_{p1} ed O_{p2}, il problema è come partizionare efficientemente $\mathcal{N}$ in due sottoinsiemi, $\mathcal{N}_1$ e $\mathcal{N}_2$. A questo fine vi sono due strategie di base. La prima è basato sull'idea della decomposizione generalizzata dell'iperpiano (*generalized hyperplane decomposition*) e conduce a divisioni non bilanciate, mentre la seconda ottiene una distribuzione bilanciata.

- **Generalized Hyperplane:** Assegna ogni oggetto $O_j \in \mathcal{N}$ al routing object più vicino; se $d(O_j, O_{p1}) \leq d(O_j, O_{p2})$ allora assegna O_j a $\mathcal{N}_1$, altrimenti assegna O_j a $\mathcal{N}_2$.

- **Balanced:** Calcola $d(O_j, O_{p1})$ e $d(O_j, O_{p2})$ per tutti gli $O_j \in \mathcal{N}$. Quindi ripete finchè $\mathcal{N}$ è vuoto:
    - Assegna a $\mathcal{N}_1$ il vicino più prossimo di O_{p1} in $\mathcal{N}$ e lo rimuove da $\mathcal{N}$;
    - Assegna a $\mathcal{N}_2$ il vicino più prossimo di O_{p2} in $\mathcal{N}$ e lo rimuove da $\mathcal{N}$;

A seconda della distribuzione dei dati e del modo in cui vengono scelti i routing object, una politica di split non bilanciata potrebbe condurre ad un miglior partizionamento degli oggetti, a causa del grado di libertà supplementare che esso ottiene. Infine deve essere sottolineato che, mentre ottenere una split bilanciato con i SAMs forza l'allargamento delle regioni soltanto lungo le dimensioni necessarie, in uno spazio metrico il conseguente aumento del covering radius si propagherebbe in tutte le dimensioni.

7 IL PROGETTO DEL SOFTWARE

A partire dai concetti già descritti di XSiter e di M-tree, l'evoluzione del progetto prevede la scelta di una libreria per ottenere le funzionalità di interrogazione richieste.

7.1 Scelta della libreria

La scelta è ricaduta sulla libreria XXL (eXtensible and fleXible Library) che rispetto ad altre alternative rappresenta una libreria Java di alto livello, facile da usare e indipendente dalla piattaforma utilizzata. XXL supporta l'implementazione di nuove funzionalità di query e fornisce una struttura tale da essere indipendente dai tipi di dati e dalle strutture dati sottostanti. Inoltre permette di aggiungere specifici metodi di accesso ad un sistema database estensibile. Un insieme di strutture indice basate su albero così come molte implementazioni delle stesse sono raccolte nel package xxl.indexStructures: esse includono B-tree, R-tree, X-tree, hB-tree e soprattutto M-tree, utilizzata nel progetto svolto.

XXL rispetto ad esempio al framework GiST (Generalized Search Tree) supporta una più vasta classe di strutture; inoltre fornisce efficienti implementazioni di algoritmi di processing di query generiche e di operazioni bulk, non disponibili in GiST. Occorre anche notare che GiST è fondamentalmente un sistema isolato che si focalizza esclusivamente sull'indicizzazione ma non fornisce né un'infrastruttura di elaborazione delle query né alcuna connettività esterna. Comunque le strutture indice di XXL consegnano i risultati delle query come cursori che permettono un'ulteriore elaborazione delle query usando l'algebra dei cursori.

Pertanto si è provveduto ad includere la possibilità di usufruire di tutte le funzionalità presenti in questo importante tipo di libreria all'interno del sistema XSiter. *Il nostro sistema quindi potrà avvalersi di tutta una serie di metodi per potenziare le proprie caratteristiche.*

7.2 Specializzazione di XSiter per il supporto a MPEG-7 ed al recupero multimediale

Grazie alle caratteristiche discusse nel capitolo 5, XSiter è già capace di fornire supporto efficiente all'interrogazione in biblioteche digitali XML (MPEG-7) molto grandi. Specificamente, l'alta efficienza del sistema con grandi documenti è un requisito particolarmente cruciale per l'interrogazione dei dati delle DLs multimediali, a causa dell'enorme dimensione dei particolari dei metadati associati, particolarmente quelli per i video. Tuttavia, per migliorare l'efficacia e l'efficienza per le applicazioni specifiche MPEG-7 e i metadati multimediali, il sistema è stato migliorato. La gestione del valore è stata specializzata per gestire media descriptors tipici. Effettivamente, grandi parti delle informazioni contenute nelle media descriptions MPEG-7 consistono tipicamente di dati non testuali ed un sistema che li controlla necessita di mantenere il contenuto di base delle media descriptions nella rappresentazione appropriata al tipo di dato e non solo come testo. A questo scopo, verranno incluse le astrazioni necessarie allo strato di applicazione per controllare tipi di dato complessi quali i tipi di dato XML Schema ed le loro estensioni specifiche MPEG-7, compresi gli array e le matrici. Le strutture specializzate di indice che sono adatte a gestire questi tipi supplementari di dati sono state incluse nell'architettura del sistema. Occorre notare che questi tipi supplementari di indici, come quelli multidimensionali per i dati numerici, permetteranno inoltre ad XSiter di andare facilmente oltre il match esatto per i valori, che non è assolutamente sufficiente in un ambiente di metadati multimediali, e di supportare la ricerca di similarità. Verranno sviluppati inoltre sommari supplementari di documento per scartare meglio i documenti o le parti non interessanti.

Per capire meglio perchè questi cambiamenti siano utili facciamo un esempio pratico. Supponiamo di voler controllare una collezione di immagini. Un tipico media descriptor di contenuto per le immagini è l'istogramma colore che definisce una quantizzazione uniforme dello spazio di colore. Per meglio supportare con XSiter un'applicazione esterna che accede a questi dati, l'astrazione degli istogrammi colore dovrebbe essere spinta verso il sistema; in questo modo l'applicazione potrebbe direttamente interagire con il sistema usando oggetti di dominio anziché valori puri. Poiché gli istogrammi di colore sono dati multidimensionali, XSiter dovrebbe essere potenziato con indici multidimensionali e la misura di similarità dovrebbe essere fornita dall'applicazione stessa.

7.3 Modifiche al sistema XSiter

Il primo intervento effettuato sul sistema XSiter ha permesso di modificare la struttura degli indici: da uno unico chiamato *ContentIndex* è stato necessario crearne due, richiamabili mediante la superclasse *ContentIndexes* e denominati InvertedIndex e SimilarityIndex. Il primo permette l'esecuzione di query esatte mentre il secondo consente interrogazioni approssimate.

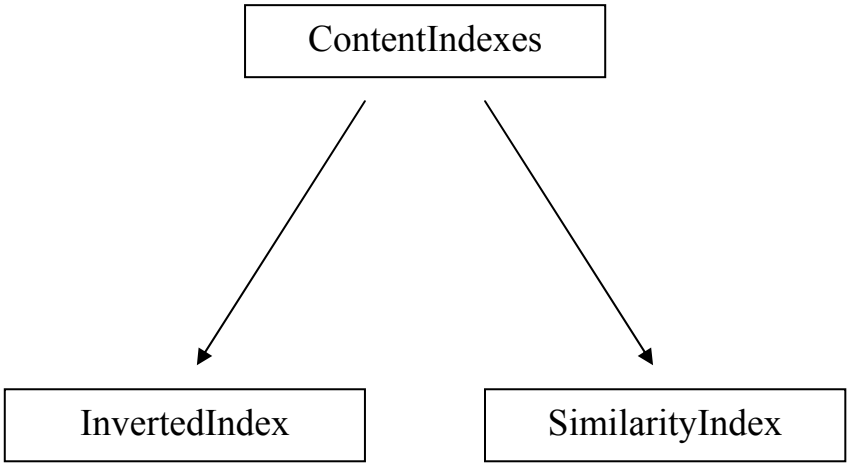


Figura 32 - Classi di XSiter che rappresentano indici

Innanzitutto nella classe XSiter vengono definiti tramite gli array di stringhe INVINDEXEDTAGS e SIMINDEXEDTAGS quali sono i tag XML per ognuno dei quali verrà creata una specifica tipologia di indice di valori. Per il momento non è stata prevista la possibilità di indicizzare stessi tag con diverse strutture indice.

In particolare, nel nostro caso specifico, i tre tags "author", "editor" e "year" rappresentano un esempio di stringhe INVINDEXEDTAGS mentre "coeff" è la stringa contenente la sequenza numerica dei valori che saranno oggetto delle nostre interrogazioni. Quest'ultimo tag, è contenuto in un file XML che rappresenta un'ontologia MPEG-7 di video di Formula 1.

Queste array di stringhe vengono propagati attraverso la gerarchia di classi di XSiter fino ad arrivare alla classe ContentIndexes (XSiter -> XMLInfoHandler -> XMLDoc -> ContentIndexes) che creerà tanti InvertedIndex e SimilarityIndex quanti quanti sono i tag delle diverse tipologie (rispettivamente tre e una nel nostro caso).

Inoltre è stata spostata la struttura Map di tipo Hashtable che conteneva gli indici di contenuto dalla classe XMLDoc alla classe ContentIndexes. L'inserimento in Map verrà ora eseguito in sequenza con associazioni di chiavi univoche e valori che rappresentano rispettivamente i tag indicizzati e le diverse tipologie di indice, trattando prima gli InvertedIndex e successivamente i SimilarityIndex.

Un altro importante aggiornamento riguarda l'inserimento nelle funzioni richiamate *startRead()*, *endReadWrite()*, *update()*, *writeList()* e *getPreList()* della classe ContentIndexes del parametro *curTagID*, l'identificativo del tag corrente, che permette di richiamare a seconda delle esigenze i metodi della classe InvertedIndex oppure quelli di SimilarityIndex. A questo proposito è stato creato un metodo, richiamato da tutte le funzioni citate sopra, che dato il tagID corrente restituisce il tipo di indice associato. L'esecuzione efficiente di questo metodo è garantita dalla lettura della Map.

```
public int whatIndex (int curTagID)  throws IOException
{
  String curTagName = tagMapping.getTagNameFromTagID(curTagID);

  if (map.get(curTagName) instanceof InvertedIndex)
  {
    return 1; // InvertedIndex
  }
  else
    return 2; // SimilarityIndex
}
```

Tutto il codice contenuto nella classe ContentIndex che permetteva l'indicizzazione dei valori nella versione originale del programma è stato poi trasferito nella classe InvertedIndex senza subire modifiche sostanziali. A questo punto l'attività si è focalizzata sulla creazione completa della classe SimilarityIndex che rappresenta l'aspetto cruciale del mio lavoro di tesi.

7.4 Interrogazioni di similarità in XSiter

La teoria indicata nel capitolo 6 ha trovato il suo seguito sperimentale nella generazione della classe SimilarityIndex. Qua vengono ridefiniti i metodi della superclasse ContentIndexes utilizzando la struttura indice M-tree.

7.4.1 Costruzione dell'M-tree

Creare un nuovo M-tree è stata una cosa abbastanza semplice: per ottenere una nuova struttura di indice è stato sufficiente instanziare la classe *MTree*. Essa rappresenta una sottoclasse della classe astratta ORTree (Overlapping Region Tree) che implementa feature utilizzate da alberi che memorizzano oggetti con descrittori che si possono sovrapporre. *ORTree* a sua volta è una sottoclasse della superclasse generica ed altamente flessibile *Tree*.

```
MTree mTree;
mTree = new MTree(LpMetric.MANHATTEN, balanced ? MTree.BALANCED_SPLIT : MTree.HYPERPLANE_SPLIT);
```

Come si può vedere dal codice, il costruttore permette di creare un nuovo M-tree con una specifica distanza per i punti e una specifica modalità di split.

È interessante notare la scelta di MANHATTEN come metrica di distanza rispetto ad altre metriche possibili in XXL quali la EUCLIDEAN o la LEVENSHTEIN. In generale esiste un vasto numero di opzioni utilizzabili per esprimere la funzione di distanza, ma quella più largamente usata è la famiglia delle distanze L_s o distanze di Minkovski espresse dalla seguente formula:

$$d(Q,C) \equiv \sqrt[p]{\sum_{i=1}^{n} (q_i - c_i)^p}$$

$p = 1$ Manhattan (Rectilinear, City Block)

$p = 2$ Euclidean

$p = \infty$ Max (Supremum, "sup")

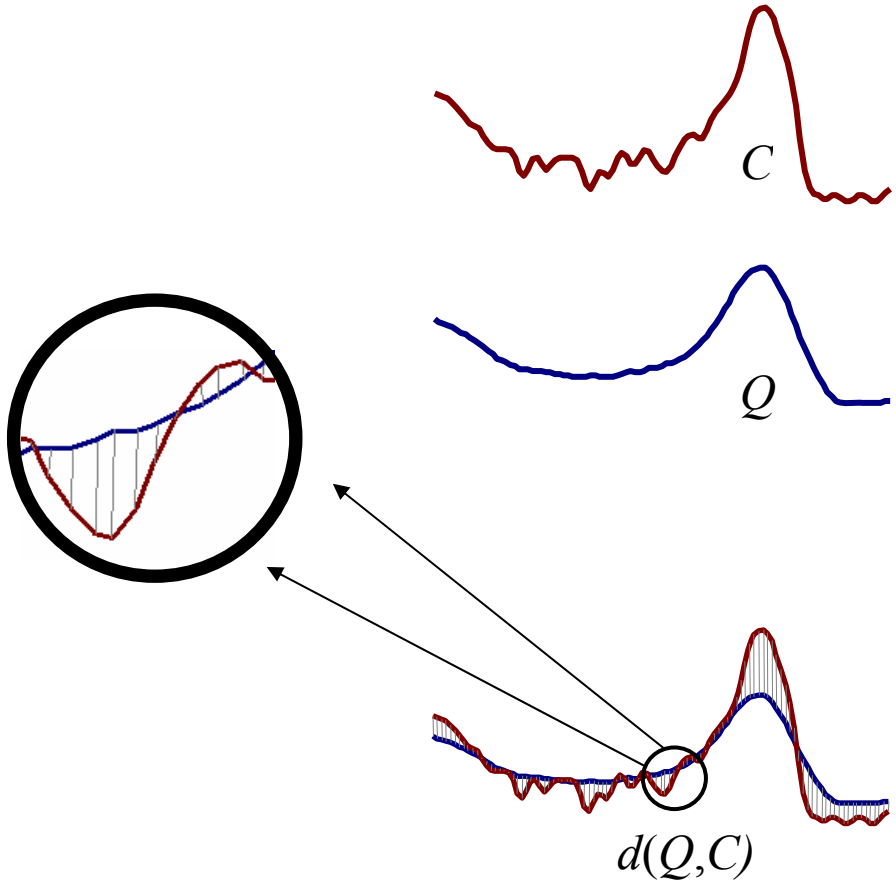
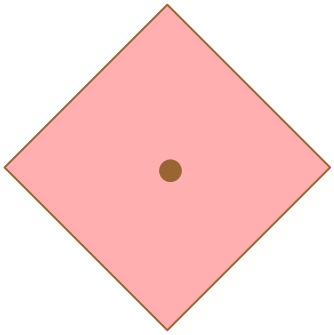


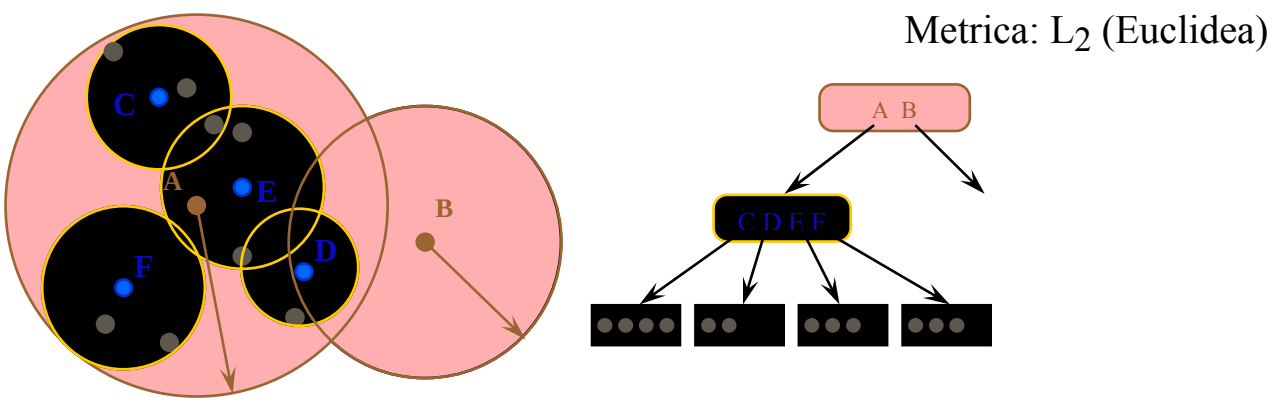
Figura 33 - Esempio di metriche di Minkovski



L_1 (city-block)

Per esempio, la distanza L_1 riporta la somma delle differenze tra le coordinate. Essa è anche chiamata distanza "block" o "Manhattan" perché quando è espressa in due dimensioni corrisponde alla distanza da percorrere tra due punti in una città di blocchi regolari. I punti alla stessa distanza *r* da un punto dato *p* formano una scatola centrata nel punto e ruotata di 45° essendo uguale a 2*r* la distanza tra due angoli opposti della scatola.

La distanza L_2 è meglio conosciuta come distanza euclidea perchè essa corrisponde alla nostra nozione di distanza spaziale. Il punti alla stessa distanza r da un dato punto p formano una sfera di diametro 2r centrata nel punto stesso.



Sulla sinistra della figura 34 viene evidenziato un esempio di una range query su un insieme di punti. Sulla destra appare invece, per differenti distanze di Miskowski, l'insieme di punti alla stessa distanza da un punto centrale.

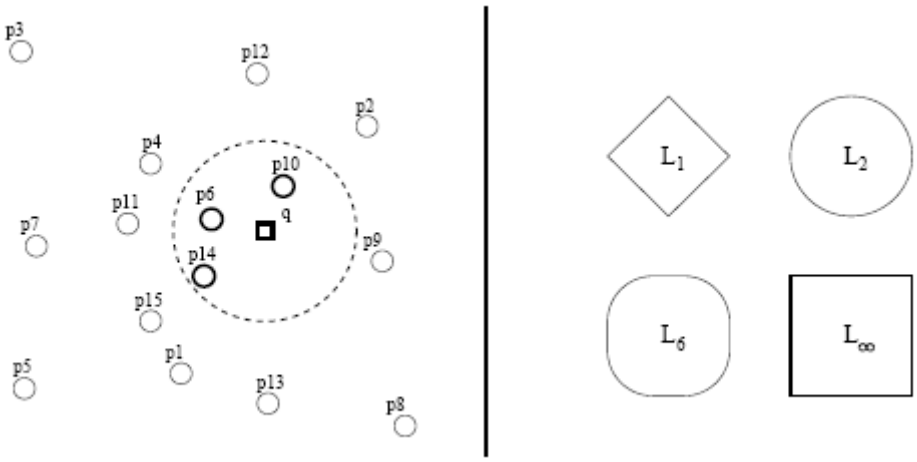


Figura 34 - Distanze di Miskowski

Oltre a specificare la fuzione di distanza, l'istruzione

mTree = new MTree(LpMetric.MANHATTEN, balanced ? MTree.BALANCED_SPLIT : MTree.HYPERPLANE_SPLIT);

attraverso la variabile booleana *balanced* permette di condizionare la scelta verso la strategia di split BALANCED oppure HYPERPLANE, descritte in maniera più approfondita nel paragrafo 6.2.5.2.

Nella costruzione dell'M-tree, il passo successivo introduce il concetto di Container. Le strutture d'indice basate su albero mantengono i loro dati in nodi dove ciascuno di essi si riferisce ad un blocco che è generalmente di dimensione fissa. Questi blocchi sono gestiti da un oggetto che soddisfa l'interfaccia *Container*. Ovviamente, un utente può implementare le proprie classi container.

Lo scenario per l'utilizzo dei containers è illustrato in figura 35. Il container in alto appartiene alla classe *BufferedContainer* in cui presumabilmente un grande numero di nodi della struttura indice viene mantenuto nel buffer. La classe *LRUBuffer* fornisce un buffer con una strategia di spostamento LRU (*least recently used*). Cioè quando si cerca un oggetto da spostare, viene scelto l'oggetto nel buffer che è stato usato meno recentemente. Se un nodo desiderato non è nel buffer, il nodo è richiesto da un *ConverterContainer*. Il solo compito di un *ConverterContainer* è convertire un nodo nel suo blocco e viceversa. Quindi, esso propaga una richiesta ad un oggetto *FileContainer* che legge il blocco desiderato dal disco.

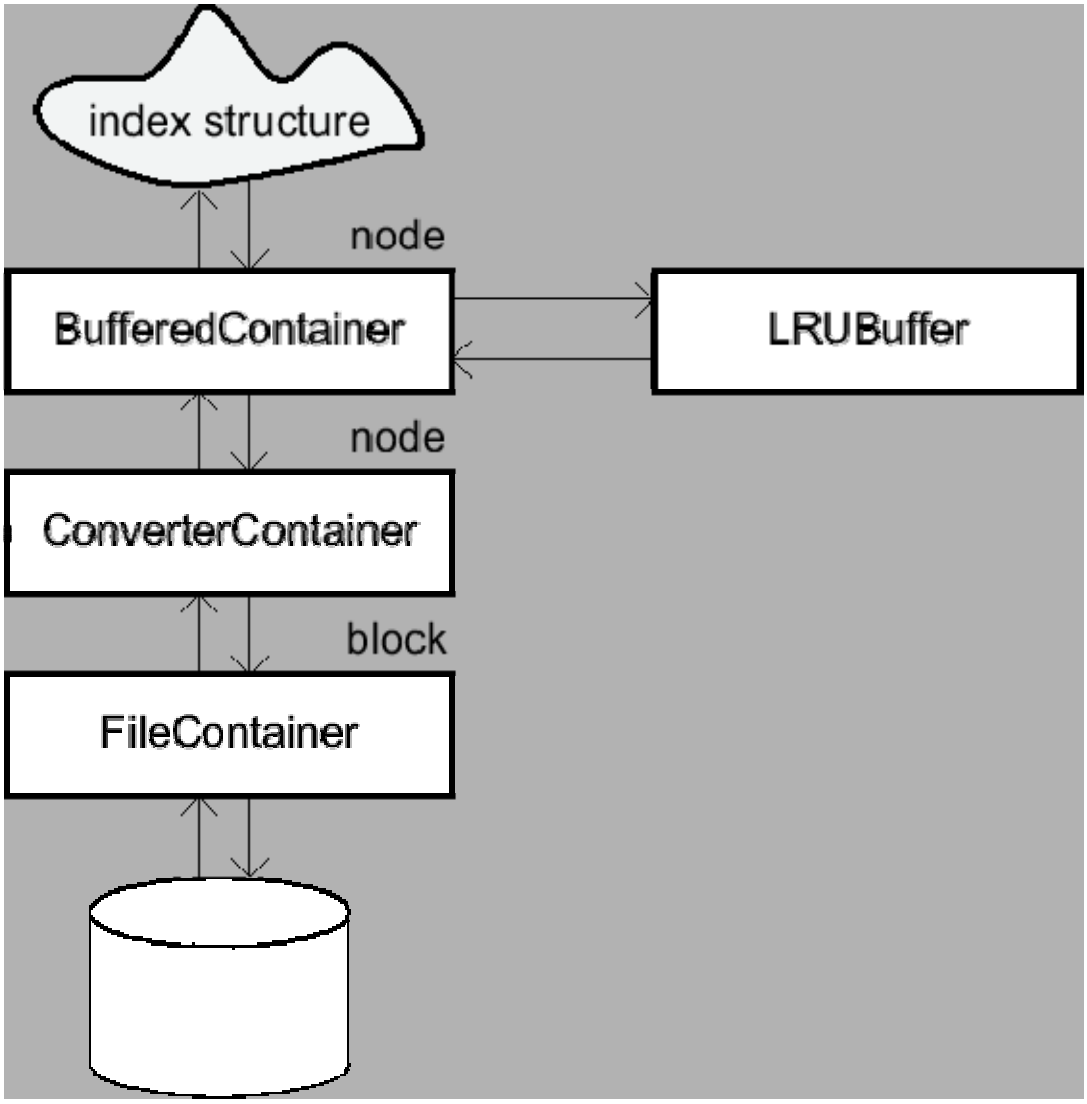


Figura 35 - Un esempio di uso dei containers

Nel progetto tra il BufferedContainer ed il ConverterContainer viene utilizzato il *CounterContainer* per effettuare il conteggio degli accessi all'M-tree: letture, inserimenti, aggiornamenti e cancellazioni.

BlockFileContainer rappresenta il FileContainer che memorizza blocchi di dimensione massima di 40 bytes (i nodi indice hanno grandezza 16+8+8+8 bytes) oltre a ulteriori 4 bytes per il numero del nodo e 2 bytes per l'informazione del livello, il tutto moltiplicato per la massima capacità dei nodi (*maxCapacity*).

```
CounterContainer lowerCounterContainer = new CounterContainer(
  new ConverterContainer(
    new BlockFileContainer("MTree", 2+4+40*maxCapacity),
    mTree.nodeConverter(mTree.leafEntryConverter(centerConverter(2)),
                mTree.indexEntryConverter(descriptorConverter(2)))
  )
);

    BufferedContainer bufferedContainer = new BufferedContainer(lowerCounterContainer,
      new LRUBuffer(bufferSize), true);
    CounterContainer upperCounterContainer = new CounterContainer(bufferedContainer);
    Container container = upperCounterContainer;
```

Al termine di questa fase, il contenuto dell'M-tree è salvato nel container *Container.*

Per completare la costruzione inizializzo l'M-tree con un descrittore, ottenuto attraverso il metodo getDescriptor, con un container per memorizzare i nodi e con la loro capacità minima e massima.

```
mTree.initialize(getDescriptor, container, minCapacity, maxCapacity);
```

7.4.2 Inserimento degli oggetti

Successivamente alla creazione abbiamo la fase di inserimento dei valori nella struttura indice. Questa inizia nel momento in cui viene richiamato il metodo update da parte del parser XML che analizza il file XML e passa come parametro una stringa da dividere nei suoi valori elementari. Questo viene realizzato attraverso il metodo *split()* che crea un array di double con i numeri costituenti la stringa e infine restituisce un oggetto di tipo DoublePoint da inserire all'interno dell'M-tree. Sono possibili tecniche di caricamento a bulk ma più semplicemente è stato scelto di eseguire l'inserimento in modo sequenziale di un oggetto alla volta.

```
public void update (String value, int pre)  throws IOException
 {
  while (value != null) {
  DoublePoint valori = split(value," ");
  System.out.println(valori);
  mTree.insert(valori);
  }
 }
```

In una prima versione semplificata utilizzata per testare il codice al di fuori di XSiter, la stringa su cui effettuare la suddivisione faceva parte di un file testuale: ogni riga di questo file era sottoposto alle stesse operazioni di inserimento nell'indice alla pari dei valori dei tag XML.

7.4.3 Esecuzione delle query

Si possono effettuare ricerche sull'M-tree usando query esatte oppure query di similarità quali la range query o la nearest neighbor query.

Innanzitutto sono state testate le istruzioni relative all'esecuzione delle query esatte: queste hanno dato esito positivo anche dopo le modifiche effettute su XSiter. A questo punto l'introduzione delle query di similarità è stato effettuato tenendo conto degli studi teorici sintetizzati nel capitolo 6.

In sintesi le interrogazioni vengono eseguite mediante il metodo:

```
mTree.query(queryDescriptor, targetLevel)
```

Il metodo query è un implementazione di un efficiente algoritmo di interrogazione. Esso restituisce un Cursor che punta a tutti gli oggetti i cui descrittori si sovrappongono con il queryDescriptor dato. Nel caso di query esatta viene semplicemente usato il query descriptor mentre il livello target è 0 a causa del fatto che le exact match query possono essere eseguite solo sui nodi foglia.

Per effettuare le range query (primo esempio di query di similarità) ricorro ad una rappresentazione geometrica implementando un descrittore come se fosse una sfera, il cui centro è rappresentato dalla query; impostato il raggio e un converter per il centro della sfera, il risultato della query è dato dal numero di elementi che ricadono all'interno della sfera così disegnata.

```
Sphere sfera = new Sphere(valquery, 50, centerConverter(valquery.dimensions()));
```

```
System.out.println("Risultato range query: "+ Cursors.count(mTree.query(sfera, targetLevel)));
```

Per eseguire una nearest neighbour query (secondo esempio di query di similarità) sull'M-tree con le istruzioni seguenti determino le 50 entrate più vicine al livello bersaglio:

```
Cursor cursor = new Taker(
  mTree.query(new DynamicHeap(getDistanceBasedComparator(sfera)), targetLevel),
  50
);

int counter = 0;
while (cursor.hasNext()) {
  Sphere next = (Sphere)((Candidate)cursor.next()).descriptor();
  System.out.println("candidate no. "+(++counter)+": "+next
                + "\t distance to query object: "+sfera.centerDistance(next));
}
cursor.close();
```

L'ordine degli elementi è determinato dal comparator dato alla struttura dynamic heap che realizza una coda di priorità. Il comparator valuta la distanza tra due oggetti candidati dal queryObject specificato (*sfera*) e viene restituito dalla funzione *getDistanceBasedComparator()*.

7.4.4 Chiusura e riapertura dell'albero

La fase di chiusura dell'M-tree consiste nel salvataggio di informazioni fondamentali per permettere la sua successiva riapertura. In particolare è necessario memorizzare tre variabili: l'identificatore del nodo radice, il descrittore e l'altezza dell'albero. Queste tre informazioni possono essere mantenute ovunque si voglia, ad esempio in un file. Successivamente viene dapprima posta a *null* l'istanza dell'M-tree, quindi chiuso il container e rilasciate tutte le sue risorse.

```
Object rootPageId = mTree.rootEntry().id();
Sphere rootDescriptor = (Sphere) mTree.rootDescriptor();
int height = mTree.height();

mTree = null;

container.close();
container = null;
```

A questo punto vengono create le istanze M-tree e Container in maniera identica a come si è proceduto per la costruzione dell'albero. Viene infine inizializzato l'albero con le informazioni memorizzate in precedenza.

```
    ORTree.IndexEntry rootEntry = (ORTree.IndexEntry)
((ORTree.IndexEntry)mTree.createIndexEntry(height)).initialize(rootDescriptor).initialize(rootPageId);
    mTree.initialize(rootEntry, MTreeTest.getDescriptor, container, minCapacity, maxCapacity);
```

7.5 Prove sperimentali

Viene mostrata di seguito una prova sperimentale semplificata eseguita con un computer Amd Athlon 64 X2 Dual Core Processor 3800+ con 1 GB di memoria RAM.

Nell'esempio mostrato in figura 36 sono stati inseriti pochi valori per ottenere una lista di dimensione limitata ma ricca di risultati significativi. Si può osservare che la fase di inserimento nella struttura indice impiega un tempo di 32 ms. Una volta fornito l'oggetto da ricercare viene dapprima effettuata una query esatta: viene riportato il numero di elementi nell'M-tree uguali ad esso ed il tempo impiegato nella sua esecuzione. Quindi viene eseguita una query di similarità (in questo caso una range query) che va a contare il numero di oggetti indicizzati la cui distanza dall'oggetto di query sia minore di un dato raggio. Nel caso specifico vi sono 5 oggetti che rientrano nel raggio impostato a 50. Infine è stata effettuata una k nearest neighbors query con k posto a 10 che restituisce i primi 10 elementi che hanno la minore distanza dall'oggetto di query. Naturalmente per ogni operazione viene riportato il tempo impiegato.

Le ricerche possono essere effettuate su un numero qualsiasi di valori inseriti. Sono state effettuate molteplici prove sperimentali variando i parametri secondo diverse combinazioni possibili: i risultati hanno evidenziato che le interrogazioni vengono eseguite in maniera efficiente. Altri test più complessi non vengono descritti in quanto non portano a nessuno sviluppo significativo rispetto a quanto già dimostrato nei casi più semplici. Rimandiamo pertanto a sviluppi futuri l'approfondimento di queste sperimentazioni.

```
Valori inseriti:
45.0   83.0   -83.0
-2.0   -34.0   0.0
12.0   3.0    12.0
0.0    0.0    0.0
-1.0   99.0   -1.0
34.0   2.0    -66.0
-5.0   -11.0  -94.0
16.0   0.0    -32.0
8.0    8.0    21.0
4.0    82.0   1.0
Tempo impiegato per l'inserimento: 32 ms.
Inserimento completo, height: 1, universe:
center: (45.0   83.0   -83.0  )      radius: 247.0   distance to parent: 0.0

Query: 0.0     0.0     0.0

Eseguo l'exact match query sull'M-Tree:
Risultato exact match query: 1
Tempo impiegato per eseguire la query: 16 ms.

Eseguo la range query sull'M-Tree:
Risultato range query: 5
Tempo impiegato per eseguire la query: 15 ms.

Eseguo la k nearest neighbor query:
Le 10 entrate più vicine alla query data sono:
candidate no. 1: center: (0.0   0.0    0.0    )   radius: 0.0   distance to parent: 211.0   distance to query object: 0.0
candidate no. 2: center: (12.0  3.0    12.0   )   radius: 0.0   distance to parent: 208.0   distance to query object: 27.0
candidate no. 3: center: (-2.0  -34.0  0.0    )   radius: 0.0   distance to parent: 247.0   distance to query object: 36.0
candidate no. 4: center: (8.0   8.0    21.0   )   radius: 0.0   distance to parent: 216.0   distance to query object: 37.0
candidate no. 5: center: (16.0  0.0    -32.0  )   radius: 0.0   distance to parent: 163.0   distance to query object: 48.0
candidate no. 6: center: (4.0   82.0   1.0    )   radius: 0.0   distance to parent: 126.0   distance to query object: 87.0
candidate no. 7: center: (-1.0  99.0   -1.0   )   radius: 0.0   distance to parent: 144.0   distance to query object: 101.0
candidate no. 8: center: (34.0  2.0    -66.0  )   radius: 0.0   distance to parent: 109.0   distance to query object: 102.0
candidate no. 9: center: (-5.0  -11.0  -94.0  )   radius: 0.0   distance to parent: 155.0   distance to query object: 110.0
candidate no. 10: center: (45.0 83.0   -83.0  )   radius: 0.0   distance to parent: 0.0     distance to query object: 211.0
Tempo impiegato per eseguire la query: 47 ms.
```

Figura 36 - Risultati interrogazioni di similarità

Conclusioni e sviluppi futuri

Nell'ambito di questa tesi sono state implementate tecniche per l'interrogazione efficiente di contenuti multimediali. Infatti i search engine delle moderne digital library devono essere in grado di gestire non solo documenti testuali ma anche e soprattutto i sempre più diffusi oggetti multimediali di tipo audio/video. L'interrogazione di tali oggetti passa attraverso l'estrazione automatica delle informazioni in essi contenute tramite i metadati MPEG-7 e l'interrogazione efficiente ed efficace di tali metadati.

In particolare, si è scelto di impiegare il sistema XSiter per accedere efficamente a tali contenuti. Le funzionalità del sistema XSiter possono essere sfruttate con successo per eseguire alcuni possibili compiti relativi all'interrogazione ed alla gestione delle digital library multimediali. XSiter gestisce in maniera semplice query semplici. La sua funzione è quella di recuperare efficientemente le occorrenze di semplici strutture di percorso con valori associati ed effettuare semplici elaborazioni sulle soluzioni restituite. Le relazioni espresse fra gli elementi, di tipo ancestor-descendant, vengono gestite facilmente per mezzo di algoritmi di ricerca di percorso e della sottostante rappresentazione tramite signature dei documenti. Quanto al supporto al valore, in questo caso viene richiesta una ricerca esatta che è efficientemente supportata per mezzo di algoritmi specializzati al valore, che possono andare più di 10 volte più velocemente di quelli standard per i casi in cui sono disponibili le informazioni indicizzate per gli elementi interrogati più di frequente; nel caso specifico, l'opzione migliore è rappresentata dagli inverted index. XSiter supporta inoltre strutture più complesse di query, che coinvolgono twig pattern ordinati e non ordinati per soddisfare fabbisogni più specifici degli utenti.

Poiché però la maggior parte del contenuto multimediale non può essere descritto facilmente soltanto attraverso termini linguistici e parole chiavi, le annotazioni delle librerie digitali della nuova generazione sono basate su ontologie, le quali contengono informazioni di dominio sul soggetto. Esplorando l'ontologia l'utente può recuperare facilmente tutte le informazioni relative ad una particolare classe di contenuti multimediali. Ciò può essere effettuato usando semplici path (o twig) query sulle annotazioni della classe. Possono quindi essere eseguite query di similarità sugli indici delle caratteristiche, che riguardano twig query con vincoli sia strutturali che di valore non esatto. In questo caso, per ottenere una buona efficienza, è fondamentale lo sfruttamento degli indici metrici multidimensionali, insieme ad algoritmi specializzati di XSiter.

La soluzione proposta considera un indice di tipo metrico chiamato M-tree. Questa struttura ad albero ha la caratteristica di poter indicizzare oggetti generici, non necessariamente in forma di punti in uno spazio vettoriale, con l'unico vincolo che tra essi sia possibile definire una metrica di

distanza. Nel caso specifico tale metrica corrisponde alla distanza di Manhattan. L'albero organizza gli oggetti ai diversi livelli in raggruppamenti, in accordo alla loro distanza da elementi dell'insieme scelti come centri del raggruppamento. Questa aggregazione degli elementi, caratterizzati dall'oggetto centrale e dal raggio, rende possibile l'attraversamento dell'albero ed il taglio di sottoalberi dai cammini di ricerca utilizzando la disuguaglianza triangolare. Per la sua applicazione sono infatti richieste unicamente la distanza tra la query ed il centro di un raggruppamento, il raggio del raggruppamento ed il raggio di query. Il risultato di questa soluzione è una considerevole riduzione del numero di calcoli di distanza da eseguire, conservando l'esattezza del risultato.

Questo approccio per la ricerca di similarità efficiente su grandi database è stata integrato nell'architettura di XSiter per poter permettere la risoluzione di query per la ricerca di similarità.

Il mio lavoro di tesi si presta a molti sviluppi futuri, in particolare sarebbe interessante:

- migliorare le strategie d'inserimento implementando tecniche di "bulk insertion" in cui viene inserita una certa quantità di elementi alla volta che vanno oltre l'inserimento classico in cui ogni elemento viene inserito separatamente;
- usare funzioni di distanza approssimata per velocizzare la ricerca e che possano tenere conto di più caratteristiche degli oggetti in esame;
- utilizzare e valutare differenti politiche di split dei nodi per migliorare l'efficacia degli algoritmi di ricerca;
- implementare e testare altri metodi di accesso in memoria secondaria;
- risolvere interrogazioni di similarità ancora più complesse;
- includere alcune estensioni nel sottosistema GUI del sistema XSiter, in particolare aiutare l'utente a specificare le query MPEG-7 e includere ai risultati presentati strumenti audiovisivi;
- condurre ulteriori prove sperimentali;
- applicare le tecniche di similarità ad applicazioni della vita reale, per esempio nell'ambito dei sistemi di videosorveglianza oppure di archivi video sportivi o cinematografici.

Concludendo, col lavoro di tesi si è avuta la possibilità di studiare concetti legati ai sistemi database multimediali e alla sottomissione ed elaborazione di interrogazioni sugli stessi con particolare attenzione alla ricerca di similarità. Si è quindi proceduto ad estendere un software di notevoli dimensioni (XSiter) con nuove funzionalità attraverso una minuziosa opera di reingegnerizzazione approfondendo tra l'altro il linguaggio di programmazione Java, la libreria XXL e gli standard XML e MPEG-7.

Bibliografia

- [1] Harald Kosch, *"Distributed Multimedia Database Technologies Supported by Mpeg7 and Mpeg21"*, CRC Press, 2003

- [2] Utz Westermann and Wolfgang Klas, *"An Analysis of XML Database Solutions for the Management of MPEG-7 Media Descriptions"*, ACM Computing Surveys (CSUR), 2003

- [3] Venkat N. Gudivada, Vijay N. Raghavan e Kanonluk Vanapipat, "*A Unified Approach to Data Modelling and Retrieval for a Class of Image Database Applications"*, IEEE Transactions on Data and Knowledge Engineering, 1994

- [4] Dian Tjondronegoro e Yi-Ping Phoebe Chen, *"Content-Based Indexing and Retrieval Using MPEG-7 and X-Query in Video Data Management Systems"*, World Wide Web: Internet e Web Information Systems, p. 207-227, 2002

- [5] Andrew Graves e Mounia Lalmas, *"Video Retrieval using an MPEG-7 Based Inference Network",* Proceedings of ACM SIGIR '02, Tampere, Finland, 2002

- [6] Peiya Liu e Liang H. Hsu, *"Queries of Digital Content Descriptions in MPEG-7 and MPEG-21 XML documents"*, XML Europe 2002, Barcelona, Spain, 2002

- [7] Samira Hammiche, Salima Benbernou, Mohand-Saïd Hacid e Athena Vakali, *"Semantic Retrieval of Multimedia Data"*, Proceedings of the 2nd ACM International Workshop on Multimedia Databases, Arlington, VA, USA, 2004

- [8] Nastaran Fatemi, Omar Abou Khaled e Giovanni Coray, *"An XQuery Adaptation for MPEG-7 Documents Retrieval"*, XML Conference & Exposition, Philadelphia, PA, USA, 2003

- [9] Ji-Hoon Kang, Chul-Soo Kim e Eun Jeong Ko, *"An XQuery Engine for Digital Library Systems"*, Proceedings of 3rd ACM/IEEE-CS Joint Conference on Digital Libraries, Houston, Texas, 2003

- [10] Giuseppe Amato, Claudio Gennaro, Pasquale Savino, "*Indexing and Searching Audio/Video Documents in Multimedia Digital Libraries*", 7th European Conference on Research and Advanced Technology for Digital Libraries (ECDL), Trondheim, Norway, 2003.

- [11] Giuseppe Amato, Claudio Gennaro, Pasquale Savino, "*Audio/Video Digital Libraries: designing, searching for documents, and generating Metadata*", Joint Conference on Digital Libraries (JCDL), Houston, TX, USA, 2003.

- [12] Pavel Zezula, Federica Mandreoli, Riccardo Martoglia, "*Tree Signatures and Unordered XML Pattern Matching*", Proceedings of 30th International Conference on Current Trends in Theory and Practice of Informatics (SOFSEM '04), 2004

- [13] Federica Mandreoli, Riccardo Martoglia, Pavel Zezula, "*Unordered XML Pattern Matching with Tree Signatures*" Proceeding of Sistemi Evoluti per Basi di Dati, S. Margherita di Pula, Cagliari, Italy, 2004

- [14] Federica Mandreoli, Riccardo Martoglia, Mattia Righini, Pavel Zezula, "*Twig Pattern Matching on Tree Signatures*"

- [15] Federica Mandreoli, Riccardo Martoglia, Mattia Righini, "*Sequential Pattern Matching on Tree Signature*"

- [16] Paolo Ciaccia, Marco Patella, Pavel Zezula, "*Searching in Metric Spaces with User-Defined and Approximate Distances*", ACM Transactions on Database Systems, 2002

- [17] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, José Luis Marroquín, "*Searching in metric spaces*", ACM Computing Surveys (CSUR), New York, USA, 2001

- [18] Paolo Ciaccia, Marco Patella, Pavel Zezula, "*Processing Complex Similarity Queries with Distance-based Access Methods*", Proceeding of 6th International Conference on Extending Database Technology (EDBT '98), Valencia, Spain, 1998

- [19] Marco Patella, "*Similarity Search in Multimedia Databases*" PhD thesis, Dipartimento di Elettronica Informatica e Sistemistica, Università degli Studi di Bologna, Italy, 1999

- [20] Paolo Ciaccia, Marco Patella, Pavel Zezula, "*M-tree: An Efficient Access Method for Similarity Search in Metric Spaces*", Proceedings of the 23rd VLDB, Athens, Greece, 1997

- [21] Paolo Ciaccia, Marco Patella, Fausto Rabitti, Pavel Zezula, "*Indexing metric spaces with M-tree*", Atti del Quinto Convegno Nazionale SEBD, Verona, Italy, 1997

- [22] Federica Mandreoli, Riccardo Martoglia, Mattia Righini, Paolo Tiberio "*A Native Extensible XML Query Processor Towards Efficient and Effective MPEG-7 Querying*", Proceeding of 2nd Italian Research Conference on Digital Library Management Systems, Padova, Italy, 2006

- [23] Jochen van der Bercken, Björn Blohsfeld, Jens-Peter Dittrich, Jürgen Krämer, Tobias Schäfer, Martin Schneider, Bernhard Seeger, "*XXL - A Library Approach to Supporting Efficient Implementations of Advanced Database Queries*", Proceedings of the 23th VLDB, Rome, Italy, 2001

- [24] Michael Cammert, Christoph Heinz, Jürgen Krämer, Martin Schneider, Bernhard Seeger, "*A Status Report on XXL - A Software Infrastructure for Efficient Query Processing*", Data Engineering Bulletin 26, 2003