Oracle 9i – oracle text

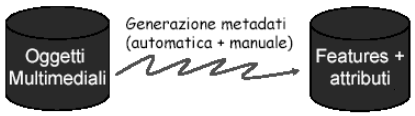
Corso di "Laboratorio di Basi di dati II"

Autori: **Myriam Mapelli, Guido Valente**

Argomenti trattati:

- Rappresentare documenti di testo in un DBMS testuale
- Gestione di testi in Oracle 9i
- Architettura di Oracle text
- Tipi di indice (in particolare CTXXPATH)
- Query sul contenuto di documenti xml (in particlare: funzione contains, scoring, query sulle sezioni, query tematiche)

Rappresentare documenti di testo in un DBMS testuale



Oggetti multimediale come documenti eterogenei nella struttura e nei formati

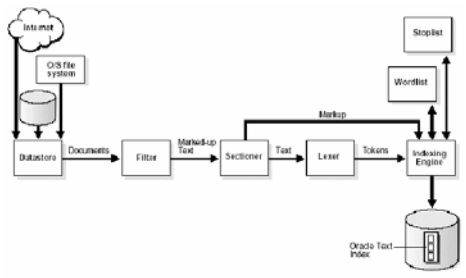
Rappresentare documenti di testo in un DBMS testuale

- Nel caso di documenti testuali le feature sono i termini (che vengono utilizzati come indici)
- Gli indici possono essere:
  - Una parola chiave o un insieme di parole chiave
  - Un insieme di concetti che caratterizzano il contenuto informativo del documento
- Sia i documenti che le interrogazioni vengono rappresentati in termini di feature

Gestione di testi in Oracle 9i

- Oracle 9i Text permette di indicizzare testi e documenti memorizzati in Oracle 9i, in file del sistema operativo o URL, in maniera integrata con i dati relazionali tradizionali con possibilità di interrogazioni basate sul contenuto
- Possibilità di utilizzare un thesaurus
- Classificare documenti in base al loro contenuto

Architettura di Oracle text



Architettura di Oracle text

DataStore

Testi indicizzabili da Oracle Text possono essere memorizzati:

- *Nel Database*: Documento contenuto in un campo di tipo CHAR, VARCHAR, VARCHAR2, LONG,LONG RAW, BLOB, CLOB
- *Nel File system*: Testi disponibili nel file system del server
- *Nel Web*: Pagine Web accessibili in un URL sia nella intranet che in Internet
- *User defined:* Output di procedure PLSQL

Architettura di Oracle text

Filter

- Prende i documenti restituiti dal datastore e li trasforma in una rappresentazione testuale
- Filtra piu' di 150 tipi di documenti e ritorna un file HTML.
  - Tiene traccia del titolo e dell'header del documento. Utile per fare ricerche su specifiche parti di testo
  - Non serve per file tipo testo, XML e HTML
  - Il metodo di filtraggio specificato da Oracle puo' essere sostituito da un sistema personalizzato di filtering

Architettura di Oracle text

Sectioner Object

- Divide l'informazione strutturale (markup) dal contenuto
- Determina le varie parti di un documento (dove possibile).
  - Solitamente utilizza standard HTML o XML markup language.
  - Permette la ricerca tramite WITHIN
  - Esempio: <TITLE> XML Handbook </TITLE>

Architettura di Oracle text

Lexer Object

- Suddivide il plain text in token (parole) e crea token tematici (in relazione alle preferenze specificate). Esempio:
  "Aha! It's the 5:15 train, coming here soon!"
- Toglie segni di punteggiatura simboli speciali
- Rimuove stop-words. Esempio:
- *"Aha * * * 5 15 train, coming * soon"*
- Stop words sono comunque utili in caso di ricerca. Esempio:
  Ricerca "Kicking the ball"…
  "kicking a ball" – Ok search
  "kicking ball" – Fail search

Architettura di Oracle text

Indexing Engine

- crea l'*inverted index* eliminando stopword,contenute in STOPLIST e crea indici utilizzando informazioni in WORDLIST
- Mantiene la lista di parole "di riferimento", dove ogni parola mantiene la lista di documenti che la contengono
- Chiamata *Inversa* perche' considera i testi in maniera inversa a quello che avviene normalmente (dato un testo, indico quali parole vengono specificate)

Architettura di Oracle text

Termine	*N. Doc*	*Tot. Freq*
Medical	2	2
Doctor	1	3

Doc	*Tot. Freq*
1	1
2	3
1	3

Indice inverso

- Solitamente gli indici inversi vengono estesi per mantenere per ogni termine non solo i documenti che lo contengono ma anche il numero di occorrenze di tale termine nei vari documenti

Tipi di indice

- **CONTEXT (standard index)** si tratta dell'indice "standard" in quanto è utilizzato per l'indicizzazione generica di documenti di testo. In particolare è consigliabile l'utilizzo di CONTEXT per documenti non strutturati ed eterogenei nei formati. Come vedremo è possibile utilizzare CONTEXT per indicizzare documeti strutturati come xml, ma non è consigliabile.
- **CTXXPATH(Xpath Index)** preferibile per l'indicizzazione di documenti xml, soprattutto per il minor numero di istruzioni che il programmatore deve implementare in confronto all'utilizzo dell'indice CONTEXT. Vedremo tutto ciò in dettaglio nelle slides successive.

CTXXPATH(Xpath Index)

- CTXXPATH è l'indice creato da oracle per migliorare le prestazioni nella ricerca interna ai documenti xml, ovvero nelle query che utlizzano la funzione existsNode() nella ricerca di nodi elementi o attributi.
- L'implementazione di un indice CTXXPATH su una tabella che contiene documenti xml crea, al momento dell'esecuzione di una query che utilizza existsNode(), un sottoinsieme di tuple tra le quali la funzione existsNode() cerca utilizzando un xpath.
- L'indice CONTEXT permette le ricerche nei documenti xml con l'ausilio della funzione contains(), ma l'utente deve implemetare una struttura PATH_SECTION_GROUP .
- L'indice CONTEXT NON viene aggiornato automaticamente dopo ogni operazione DML
  - indicizzare un singolo documento richiede molto tempo

Esempio di PATH_SECTION_GROUP

- Documento:
  <A>rat</A><A>ox</A> <B>tiger rabbit</B>
  <C>dragon<C>snake</C></C>
- Creazione section group:
  ctx_ddl.create_section_group('mygroup', 'xml_section_group');
- Aggiunta sezioni di tipo ZONE:
  ctx_ddl.add_zone_section('mygroup', 'asec', 'a');
  ctx_ddl.add_zone_section('mygroup', 'bsec', 'b');
  ctx_ddl.add_zone_section('mygroup', 'csec', 'c');
- Tale procedimento per ogni operazione DML.

Esempio di creazione indice CTXXPATH

- Creazione dell'indice:
  CREATE INDEX xml_idx ON xml_tab(col_xml) indextype is ctxsys.CTXXPATH;
- Creazione dell'indice con paramentri:
  CREATE INDEX xml_idx ON xml_tab(col_xml) indextype is ctxsys.CTXXPATH PARAMETERS('storage my_storage memory 40M');

Esempio di query su tabella indicizzata con CTXXPATH

- Seleziona l'identificativo dei documenti xml contenenti un'attributo title con valore XML. Tale attributo deve apartenere all'elemento chapter il quale è figlio dell'elemento book.

```
SELECT xml_id FROM xml_tab x
    WHERE
x.col_xml.existsNode('/book/chapter[@title="XML"]')
>0;
```

Svantaggi dell'indice CTXXPATH

- Esistono dei costrutti XPATH che l'indice CTXXPATH non supporta, in questi casi la stima dei costi e la selettività per la funzione existsNode() non è ottimale:
  - Funzioni XPATH;
  - Operatori aritmetici e di range numerici;
  - Operatore d'unione "|";
  - Esistenza di attributi
  - Attributi preceduti da * o //
  - . o * come fine di una path expression
- Per ottimizzare il lavoro di stima e selezione bisogna utilizzare il comando ANALYZE sull'indice o sulla tabella:

```
ANALYZE INDEX myPathIndex COMPUTE STATISTICS;
ANALYZE TABLE XMLTAB COMPUTE STATISTICS;
```

Quando usare l'indice CONTEXT

- La creazione di un indice di tipo CONTEXT permette di eseguire interrogazioni sul contenuto dei documenti, più adatto a documenti di testo non strutturati come i formati txt o rtf.
- Esempio:

```
Create table docs
(id number primary key,text varchar2(80));

Insert into docs values (1,'first document');
Insert into docs values (2, 'second document');

Create index doc_index on docs(text) indextype is
ctxsys.context;

Select id from docs where contains(text, 'first') > 0;
```

La funzione CONTAINS e SCORE

- Esempio:

```
select id, score(1) from texttab
where contains(textcol, 'query', 1) > 0
      order by score(1) desc;
```

- CONTAINS ha come parametri la colonna contenente i documenti, il testo da cercare nei documenti e solo in caso di score un numero a scelta che deve riferire al numero usato nella proiezione.
- SCORE è una funzione che attribuisce un peso alle tuple tornate per ordinarle in base alle occorenze del testo cercato.

Diversi usi della funzione CONTAINS

- interrogazioni semplici

```
contains(text, 'dog') > 0
```

- si possono interrogare frasi:

```
contains(text,'dog my cat') > 0
```

- le stopword sono trattate come wildcard (matchano qualsiasi parola)

```
contains(text, 'dog the cat') > 0
```

  - restituisce 'dog my cat', 'dog your cat', 'dog the cat'

- le stopword da sole vengono eliminate dalla query

Operatori di SCORE

- WEIGHT (*) moltiplica il punteggio di un termine di ricerca per renderlo più o meno importante nella query (peso tra .1 e 10):

```
contains(text,'(dog*2) AND cat') > 0
```

- THRESHOLD (>) elimina i documenti sotto una certa soglia:

```
contains(text,'(dog*2) AND cat') > 50
```

Operatore WITHIN

- Interrogare documenti strutturati con l'indice CONTEXT:

```
<A>rat</A><A>ox</A> <B>tiger rabbit</B>
   <C>dragon<C>snake</C></C>

ctx_ddl.create_section_group('mygroup', 'xml_section_group');

      ctx_ddl.add_zone_section('mygroup', 'asec', 'a');
      ctx_ddl.add_zone_section('mygroup', 'bsec', 'b');
      ctx_ddl.add_zone_section('mygroup', 'csec', 'c');

contains(text, 'rat within asec') > 0
                restituisce il documento

contains(text, 'tiger within asec') > 0
                non restituisce il documento
```

Operatori di espansione delle parole

- WILDCARD (%_) per pattern matching (come in LIKE di SQL)
- FUZZY (?) trova parole simili (usa wordlist)
- STEM ($) trova parole con radice comune (usa wordlist)
- SOUNDEX (!) trova parole con stesso suono (usa una specifica espansione fuzzy)
- EQUIV (=) permette di indicare esplicitamente varie forme della stessa parola

Operatori MINUS e ACCUM

- MINUS (-) sottrae il punteggio dell'operando destro a quello del sinistro:

  contains(text, 'tiger MINUS rabbit') > 0

  tiger e preferibilmente non rabbit, sottrae i punteggi.

- ACCUM (,) raggruppa più parole o frasi e ne accumula i punteggi:

  contains(text, 'tiger ACCUM rabbit)') > 0

  tiger e preferibilmente rabbit, somma i punteggi.

Conclusioni

Partendo dal definire una base di dati testuale, abbiamo visto la proposta di oracle intermedia nella versione 9, osservandone l'architettura e le funzionalità offerte per gestire in particolare documenti xml.

Il concetto importante più volte richiamato in queste slides è il vantaggio offerto dell'indicizzazione tramite CTXXPATH, al di là dell'implementazioni degli indici e delle query è importante capire quando usare certi operatori piuttosto che altri in base alle esigenze e in funzione dell'ottimizzazione delle performance (per l'utente) e della struttura della base di dati (per il programmatore).

Bibliografia

- Documentazione di Oracle text 9i, sul sito di oracle:

  http://www.oracle.com/global/it/index.html;

- "Searching XML data with Oracle Text", all'indirizzo:

  http://www.lc.leidenuniv.nl/awcourse/oracle/appdev.920/a96620/xdb11sea.htm;

- Per approfondire gli argomenti esposti si consiglia di frequentare il corso di "Basi di dati multimediali"