UNIVERSITÀ DEGLI STUDI
DI MODENA E REGGIO EMILIA

Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Informatica

Approximate Sequence Matching: Implementazione e Analisi Prestazionale Comparata di Tecniche Portabili e Efficienti

Marcello Pietri

Tesi di Laurea

Relatore:
Prof. Riccardo Martoglia

Anno Accademico 2007/2008

RINGRAZIAMENTI

Ringrazio tutti i professori del corso di laurea ed in particolare il prof. Riccardo Martoglia per la continua disponibilità e assistenza.
Un ringraziamento speciale alla mia famiglia ed ai miei amici che mi hanno permesso il raggiungimento di questo importante obiettivo sostenendomi e sopportandomi.

PAROLE CHIAVE

Sequence matching

DBMS

JDBC

Portabilità

Analisi prestazionale

Indice

Elenco delle figure:

(*)

Note sulle figure:

Nel capitolo 2.2 sotto paragrafi 2.2.1, 2.2.2, 2.2.3, 2.2.4 e 2.2.5, le immagini in alto a destra, rappresentanti i loghi dei DBMS relativi sono da ricollegarsi alla Bibliografia, utilizzando come metro di riferimento la dicitura "[#]" riportata di fianco al nome del DBMS, a sua volta riportato di fianco al numero del sotto paragrafo.

Note sulle tabelle:

Le tabelle sono classificate con lo stesso metodo delle figure, ad eccezione della dicitura iniziale che non sarà più "Fig." ma "Tab.". Non si riporta nell'indice il loro elenco in quanto sensate solo in riferimento al contesto del capitolo o del paragrafo ad esse associato. Alcune sono riportate in formato "immagine", ma sono da considerasi comunque, a tutti gli effetti, tabelle. I grafici sono riportati in riferimento (Rif.) alle loro tabelle, e questo compare nel sottotitolo del grafico stesso.

(*) Il metodo di classificazione delle figure riporta il seguente schema: *Fig. #paragrafo[.#spar][.#sspar]-#Lettera - #Nome*

Introduzione

Con questa tesi si vogliono studiare e comparare tecniche di *approximate sequence matching,* ovvero degli algoritmi che ci permettono il riconoscimento di pattern in un testo, a meno di un numero reale o intero di non corrispondenze ammissibili. Questi algoritmi possono essere implementati in modi completamente differenti, ad esempio utilizzando metriche di Edit Distance oppure tecniche basate su Suffix Tree e Suffix Array. Il progetto EXTRA [3], il cui obiettivo è risolvere il problema della ricerca di similarità tra frasi nel contesto della traduzione multilingue, è studiato appunto per implementare alcune di queste tecniche e questa tesi ne tratta lo sviluppo e l'analisi comparata.

Extra è uno strumento EBMT (Example Based Machine Translation) il cui compito principale è quello di aiutare un traduttore umano effettuando una pre-traduzione. Un sistema EBMT, infatti, traduce per analogia, utilizzando le traduzioni passate per tradurre altre frasi dalla lingua sorgente alla lingua destinazione. Al centro del funzionamento di questi sistemi vi sono le metriche per la valutazione del grado di somiglianza tra le frasi, tra cui l'Edit distance. All'interno del nucleo del programma troviamo l'algoritmo stesso di ricerca che è basato su query, a loro volta ottimizzate a livello di performance da opportuni filtri. Queste query, che impiegano una notevole parte di tempo macchina, sopratutto quando interrogano un'enorme quantità di dati, sono anche al centro dello studio dell'analisi prestazionale in quanto necessitano dell'appoggio ad un DataBase Management System. Essendo molto ampia la gamma di DBMS tra cui scegliere ed essendo molto differenti le performance di tali sistemi, entra in gioco la possibilità di un miglioramento delle prestazioni, passando però prima da un discorso di portabilità del software su tali piattaforme.

Il problema della portabilità e quello dell'analisi prestazionale derivano dalla natura stessa dell'informatica e, da poco dopo gli esordi di questa disciplina, sono stati affrontati con numerose tecniche, ma senza una risposta definitiva.

Se consideriamo la crescente complessità dei sistemi operativi, dei linguaggi di programmazione e dei programmi stessi, si può notare come la ramificazione delle problematiche sia anch'essa cresciuta in modo esponenziale, introducendo notevoli difficoltà nella risoluzione della questione.

Lo scopo di questa tesi, partendo appunto da un software discretamente complesso, è quello di renderlo portabile e di poter effettuare analisi prestazionali, sia comparate che non, su differenti piattaforme, al fine di trovare nuove soluzioni che minimizzino i tempi di ricerca.

Se la tecnologia Java, infatti, permette di passare da un sistema operativo ad un altro senza modificare il codice, ad eccezione di caratteri non UTF-8 e problemi similari, questa non permette di passare da un DBMS ad un altro mantenendo sempre lo stesso codice.

Partendo da questi presupposti ed introducendo nuove funzionalità, descritte nelle fasi successive di questo documento, si renderà il software EBMT portabile, sia a livello di sistema operativo che di DBMS utilizzabile, potendo così confrontare sia l'efficienza del DBMS utilizzato che l'efficienza del software rispetto ad altri simili, implementati con altri algoritmi quali Suffix Tree e Suffix Array. Extra, infatti, pur essendo un sistema EBMT, permette comunque la ricerca di sequenze non testuali, ad esempio di tipo genetico.

Per facilitare l'utilizzo del software così creato e dei DBMS supportati, verrà sviluppata un'interfaccia grafica (ed una equivalente linea di comando) che permetterà anche di semplificare la procedura d'installazione.

Riportiamo ora una panoramica dei principali argomenti trattati:

La struttura del documento è divisa in tre parti; la prima riguarda “Il caso di studio”, la seconda riguarda “Progetto, implementazione e analisi sperimentale” ed infine la terza riporta le appendici.

Nella prima parte si analizzano nel dettaglio le tecniche necessarie alla comprensione del lavoro svolto, oltre chiaramente al loro impiego all'interno dei software utilizzati, in particolare:

- Nel capitolo 1 si presenta l'*approximate sequence matching*, le tecniche implementative utilizzate a tale scopo e le implementazioni vere e proprie all'interno dei software impiegati.
- Nel capitolo 2 si presentano le tecnologie utilizzate, quali Java, JDBC, SQLJ ed i numerosi DBMS provati ed analizzati.

Nella seconda parte si lavora principalmente all'implementazione del software e alla sua analisi prestazionale, in particolare:

- Nel capitolo 3 si passa alla stesura dei requisiti individuati utilizzando i principi del *Software Requirement Specification* e del linguaggio UML (*Unified Modelling Language*).
- Nel capitolo 4 si implementano gli aspetti visti nel capitolo 3 giungendo alla versione definitiva del software.
- Nel capitolo 5, ultimo e seguito dalle conclusioni, si svolge la fase di analisi prestazionale comparata, sia a livello di prove rapportate ad altri software in precedenza analizzati, sia a livello di DBMS.

Parte I

Il caso di studio

Capitolo 1

Approximate sequence matching

Definendo come *approximate sequence matching* un algoritmo che ci permette il riconoscimento delle occorrenze tra due stringhe, a meno di un numero reale o intero di non corrispondenze ammissibili, descriviamo in dettaglio gli utilizzi possibili di tale strategia sia in ambito di traduzione assistita che in ambito di ricerca di sequenze genetiche.

1.1 Introduzione all'approximate sequence matching

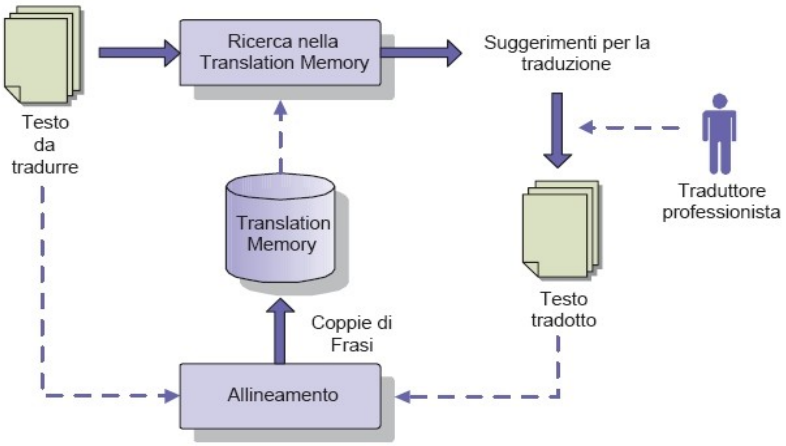
Il problema dell'*approximate sequence matching,* in questo contesto, nasce dalla necessità di cercare sequenze comuni tra due testi di tipo testuale o genetico. Dipendentemente dall'approccio seguito, si otterrà quindi un software maggiormente specifico per l'ambito scelto. Nel caso di Extra i testi sono prevalentemente manuali che, una volta tradotti, dovranno essere vagliati da parte di team di traduttori professionisti, infatti, Extra è un sistema di tipo EBMT.
I sistemi EBMT emulano la traduzione umana, riconoscendo la similarità di una nuova frase nel linguaggio sorgente (testo da tradurre) con una precedentemente tradotta, ed utilizzando quest'ultima per realizzare una "traduzione per analogia".
L'idea di base dell'EBMT fa riferimento a un database di traduzioni passate, in cui compaiono in parallelo le frasi da tradurre e quelle già tradotte: si tratta della cosiddetta *Translation Memory*.

L'accuratezza e la qualità della traduzione dipendono fortemente dalla dimensione e dalla copertura della Translation Memory, ma non è necessario che questa assuma le dimensioni dei database richiesti, ad esempio, dai sistemi di traduzione statistica, poiché non è necessario coprire l'intero vocabolario. I normali sistemi di traduzione automatica producono le traduzioni applicando conoscenza linguistica; essi analizzano linguisticamente il testo nel linguaggio sorgente, associano le strutture linguistiche alle loro controparti nel testo destinazione, e costruiscono una traduzione da zero. Questi sistemi hanno due principali svantaggi: sono dipendenti dalla lingua poiché impiegano avanzate tecniche linguistiche che si possono applicare esclusivamente a una lingua specifica. Pertanto sono limitati ai linguaggi che possono gestire. Inoltre, poiché un programma per computer non potrà mai riuscire a capire realmente il linguaggio e il contesto, i sistemi automatici producono traduzioni sterili e spesso inopportune. La maggior parte di questi sistemi hanno dizionari e frasi che possono essere personalizzati, ma difficilmente raggiungono la qualità di una traduzione umana professionale.

I sistemi di tipo *Translation Memory (TM)*, invece, memorizzano le traduzioni passate e le propongono quando, traducendo nuovo materiale, incontrano frasi identiche o simili. I sistemi TM "imparano" e migliorano con l'utilizzo: più vengono utilizzati più diventano accurati e utili.

I sistemi TM non traducono da soli; quando il sistema TM è in procinto di tradurre una nuova frase, svolge una ricerca nella Translation Memory e presenta all'utente le traduzioni di frasi identiche o simili che il traduttore ha fatto in passato. Nonostante in molti casi non proponga di suo una traduzione, lo aiuta ad assicurare la consistenza dello stile e della terminologia permettendogli di consultare con facilità le traduzioni passate. In altre parole, i sistemi basati su Translation Memory non traducono ma forniscono consistenza, permettendo al traduttore un lavoro più rapido e di maggiore qualità.

Per permettere quindi tutto il processo appena descritto è necessario, dopo essere passati da fasi quali l'allineamento e altre similari descritte approfonditamente in [3], utilizzare la tecnica di *approximate sequence matching,* infatti, per soddisfare tutti questi punti, è stato studiato e implementato [3] un procedimento piuttosto innovativo, basato nelle sue linee fondamentali sul concetto di *Edit Distance* e su un'analisi della frase prettamente sintattica. Il procedimento di ricerca vero e proprio è suddiviso in più fasi, ognuna implementata mediante query ad un DBMS. Per migliorare l'efficienza, si utilizzano inoltre particolari strutture dati (le tabelle dei *q-grammi*) ed una serie di *filtri* per ottenere ottime prestazioni in ciascuna di esse. A seconda delle esigenze dell'utente, è possibile ricercare intere frasi simili ma anche sottoparti di queste, permettendo in questo modo un maggiore sfruttamento del potenziale insito nella Translation Memory: spesso due frasi sono diverse nel loro insieme, ma presentano magari una sottoparte del tutto analoga come struttura e, pertanto, utile. Come si vedrà, è anche possibile agire sui principali parametri in modo, ad esempio, da escludere frasi che siano diverse da quella cercata oltre una certa soglia, da aumentare l'accuratezza o piuttosto la velocità della ricerca, e così via.



(Fig. 1.1-a – [3] Funzionamento di un sistema EBMT)

Presentiamo ora le tecniche studiate e utilizzate sia per il software in questione sia per quello relativo l'analisi prestazionale comparata in modo rigoroso e puntuale.

Definiamo con *approximate sequence matching (*)* un algoritmo che ci permette il riconoscimento delle occorrenze tra due stringhe a meno di un numero ***k*** di non corrispondenze ammissibili.

Nello specifico:

- **Σ** è l'alfabeto di tutti i simboli presenti in testo e pattern
- ***T*** (di lunghezza ***n***) è il testo su cui si effettua la ricerca
- ***P*** (di lunghezza ***m***) è il pattern ricercato nel testo
- ***x[i]*** , con $i \in \aleph$, rappresenta l'***i***-esimo simbolo della stringa ***x***
- $k \in \aleph$ è il numero massimo di errori o non corrispondenze
- p_1 e p_2 sono due stringhe formate da simboli di **Σ**
- $d(p_1, p_2)$ è una funzione di distanza che restituisce un valore in $\Re$

Il problema può quindi essere riscritto nel seguente modo:

Determinare l'insieme delle posizioni **j** del testo ***T*** tali che:

$$i \leq j,\ d(T[i] \ldots T[j],\ P) \leq k$$

(per qualche ***i*** minore o uguale a ***j*** la distanza ***d*** tra sottosequenze ***i-j*** di ***T***esto e il ***P***attern ***(T[i]…T[j],P)*** sia minore o uguale a ***k***)

Cioè l'insieme delle posizioni in cui si trova una corrispondenza di ***P*** con al più ***k*** errori.

(*) in italiano "ricerca approssimata di sequenze".

I metodi che ci permettono la realizzazione pratica di questo algoritmo sono numerosi, infatti, per il software ACGT [8] essi sono spiegati nel capitolo 1.2, mentre per il progetto in corso sono stati utilizzati i seguenti:

Edit distance:

(implementazione nel capitolo 1.1.1)

Definiamo l'edit distance $d(p_1, p_2)$ tra due frasi (o stringhe) p_1 e p_2 come il minimo costo della sequenza di operazioni sui simboli che trasformano p_1 in p_2.
Nel nostro caso si considerano come operazioni possibili l'inserimento la cancellazione e la sostituzione ciascuna di costo unitario.

Esempio:

Consideriamo Σ come l'insieme di tutte le parole delle due frasi

Σ = {'Questa','frase','semplice','è','un','altro','esempio','per','l´','la','funzione','di','edit','distance'}

p_2: *Questa frase è un altro esempio per la funzione di edit distance*

p_1: *Questa semplice frase è un esempio per l' edit distance*

Per trasformare p_1 in p_2 sono necessari i seguenti cambiamenti:

Questa frase è un altro esempio per la funzione di edit distance
Questa c frase è un i esempio per s i i edit distance

La distanza tra p_1 e p_2 risulta quindi di 5.

Whole matching:

(implementazione nel capitolo 1.1.2)

Il *whole matching (*)* è la ricerca completa di pattern nel testo, infatti, si selezionano tutti e soli i suggerimenti che differiscono dal pattern per una distanza relativa minore a ***d * lunghezza pattern***.

Esempio:

Dire che la frase "a b c d e f" di lunghezza 6 deve avere distanza ***d*** *= 0.7* significa che solo i suggerimenti che distano meno di *4 (**)* sono considerati validi.

Sub^2 matching:

(implementazione nel capitolo 1.1.3)

La tecnica del Sub^2 matching si basa sull'individuazione di coppie di q-grammi posizionali [5][6][9][10], contenuti all'interno di due frasi che soddisfino i vincoli imposti sulla lunghezza minima e sulla massima distanza consentita.

Come nella tecnica del Whole match (Paragrafo 1.1.2) si utilizzano, anche qui, più filtri posti in cascata, che hanno il compito di eliminare dall'insieme dei possibili risultati tutte le coppie che non soddisfano i requisiti richiesti.

L'algoritmo ottenuto risulta quindi essere particolarmente performante.

(*) in italiano "ricerca completa"
*(**) (round(6*0.7) = round(4.2) = 4)*

1.1.1 Edit distance

Diverse sono le implementazioni possibili di questa funzione, in particolare, per questo progetto si sono utilizzate due versioni differenti, una normale [1][3] e una ottimizzata per il calcolo su diagonali [2][3]:

Il primo, è una versione classica, basata sulla programmazione dinamica:
Immaginando di dover calcolare $d(p_1, p_2)$, si riempie una matrice $C0..|p_1|,0..|p_2|$, dove Ci,j rappresenta il minimo numero di operazioni necessarie per trasformare $p_1(1..i)$ in $p_2(1..j)$.
Il calcolo procede come segue:

se $(p_1(i) = p_2(j))$ $\quad C_{i,0} = i \quad C_{0,j} = j \quad C_{i,j} = C_{i-1,j-1}$
altrimenti $\quad 1 + min(C_{i-1,j}, C_{i,j-1}, C_{i-1,j-1})$

Il risultato finale si legge nella cella in basso a destra: $C|p_1|,|p_2| = d(p_1, p_2)$.

L'algoritmo deve riempire la matrice in modo che le celle in alto, a sinistra e in alto a sinistra rispetto alla cella corrente siano calcolate prima di quella cella; è possibile perciò utilizzare sia un riempimento per righe da sinistra verso destra, sia per colonne dall'altro verso il basso. La complessità di questo algoritmo sarà $O(|p_1||p_2|)$, sia per il caso medio sia per il caso peggiore.

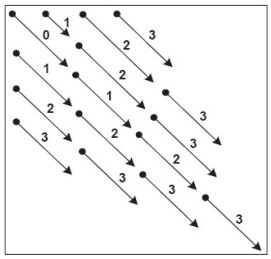
		equal	even	like	different	uniform	disjoined	displaced
	0	1	2	3	4	5	6	7
equal	1	0	1	2	3	4	5	6
even	2	1	0	1	2	3	4	5
like	3	2	1	0	1	2	3	4
unequal	4	3	2	1	1	2	3	4
uniform	5	4	3	2	2	1	2	3
displaced	6	5	4	3	3	2	2	2

(Fig. 1.1.1-a – Tabella esempio di edit distance)

Un'ottimizzazione, l'edit distance per diagonali:

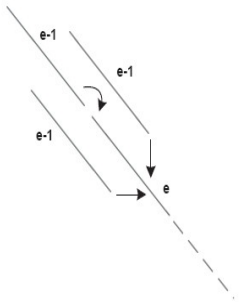
Questo secondo algoritmo [2] è una miglioria di quello precedentemente descritto per quanto riguarda il caso peggiore. È possibile, infatti, sfruttare la proprietà di monotonia crescente delle diagonali della matrice di programmazione dinamica, in particolare:

$C_{i+1,j+1} \in \{C_{i,j}, C_{i,j} + 1\}$.



(Fig. 1.1.1-b – [3] Edit distance diagonale, monotonia crescente)

Come mostrato in figura (1.1.1-b) qui sopra a fianco , l'algoritmo procede calcolando i tratti diagonali; ogni tratto rappresenta un numero di errori ed è una sequenza in cui entrambe le frasi corrispondono. Quando inizia un tratto di ***e*** errori, esso continua finché continuano gli adiacenti tratti ***e-1*** oppure finché continua la corrispondenza delle stringhe (é sufficiente una delle condizioni enunciate) (figura (1.1.1-c) a fianco).



(Fig. 1.1.1-c – [3] Edit distance diagonale, errori)

In questo modo è possibile arrivare alla cella finale (il risultato) calcolando un numero notevolmente inferiore di celle e velocizzando dunque i calcoli:

In figura (1.1.1-d) viene mostrato l'esempio completo del calcolo dell'edit distance con il metodo ottimizzato per diagonali, applicato alle frasi già utilizzate negli esempi precedenti.

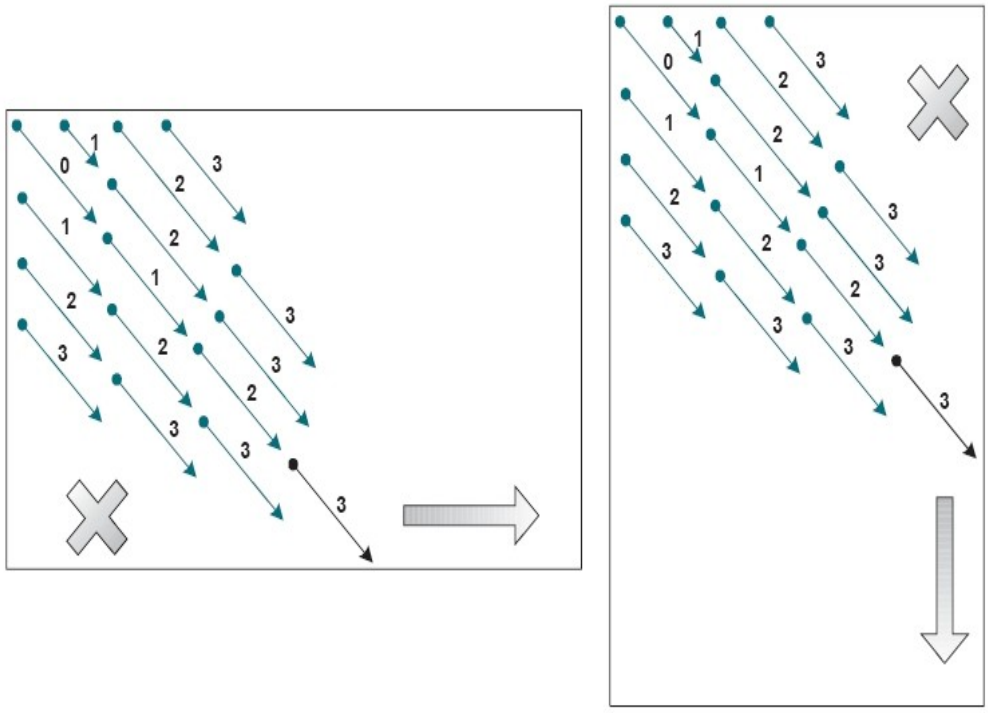
		equal	even	like	different	uniform	disjoined	displaced
	0	1	2	3	4	5	6	7
equal	1	0	1					
even	2	1	0	1				
like	3		1	0	1			
unequal	4			1	1	2		
uniform	5				2	1	2	
displaced	6					2	2	2

(Fig. 1.1.1-d – Tabella esempio di edit distance diagonale)

Nell'implementazione [3], sono stati inoltre apportati i seguenti accorgimenti:

- raggiunta la massima distanza consentita ***k***, l'algoritmo si interrompe comunicando esito negativo;

- durante il calcolo dei tratti diagonali, in caso di raggiungimento degli estremi della matrice il calcolo procede solo più per i tratti utili (quelli cioè che porteranno verso la cella finale) (vedi figura (1.1.1-e) sotto).



(Fig. 1.1.1-e – [3] Edit distance diagonale, algoritmo modificato)

Entrambi gli algoritmi sono riportati nell'appendice A1 sia sotto forma di codice Java che codice C.

1.1.2 Whole matching

Nello specifico:
Siano dati:
un insieme di frasi da pre-tradurre ***Q***
un insieme di frasi della Translation Memory (d'ora in poi ***TM***)
una massima distanza relativa ***d***

Per ogni frase f_q in ***Q*** di lunghezza $|f_q|$ si ricercano tutte le frasi f_{TM} in ***TM*** (i suggerimenti) tali che $d(f_q,f_{TM}) \leq round(d^*|f_q|)$ (ordinate sulla base del risultato dell'edit distance).

[3] La query per la ricerca è una rielaborazione di quanto presentato in [5] e presenta la seguente struttura:

```
INSERT INTO FULLMATCH
SELECT r2.codice AS cod2, r1.codice AS cod1,
wordEditDistanceDiag (r1.frase, r2. frase , <k>)
FROM <tab1> r1, <qtab1> r1q, <tab2> r2, <qtab2> r2q
WHERE r1.codice= r1q.codice
AND r2.codice = r2q.codice
AND r1q.qgram = r2q.qgram
... AND <<filtri>> ...
AND wordEditDistanceDiag (r1.frase, r2.frase, <k>) >= 0
```

Si tratta di una query costruita dinamicamente, in cui i nomi delle tabelle sono variabili e in cui figurano parametri liberamente modificabili (come k). Come si può vedere, essa agisce sulle frasi individuando le coppie simili ed inserendone i dati direttamente nella tabella FullMatch (codici identificativi della coppia di frasi e relativa distanza).
WordEditDistanceDiag() è la stored procedure che implementa l'algoritmo per il calcolo dell'edit distance ottimizzato per diagonali descritto nella sezione 1.1.1; in particolare verifica se le due frasi passate come parametro sono sufficientemente

"simili", cioè con una edit distance inferiore a k. In caso affermativo, restituisce un valore non negativo (la distanza calcolata, che verrà registrata in tabella), altrimenti restituisce -1 e provoca l'esclusione della coppia di frasi candidate.
La sezione identificata dall'etichetta <<filtri>> contiene particolari condizioni che riescono a filtrare preventivamente le possibili coppie di frasi, velocizzando così l'esecuzione della query. Essi si basano sull'utilizzo dei q-grammi precedentemente estratti e memorizzati e verranno ora analizzati in dettaglio.

Un punto di forza dell'approccio presentato è la possibilità di inserire nella query una serie di filtri che riducono al minimo indispensabile il richiamo della stored procedure per il calcolo dell'edit distance; essi, sfruttando i q-grammi memorizzati nelle tabelle ausiliarie, riescono a garantire ottime prestazioni, grazie al basso numero di falsi positivi restituiti, e correttezza, grazie all'assenza di false esclusioni.

Non volendo approfondire ulteriormente l'argomento si rimanda il lettore al capitolo 3.6.1 e successivi di [3]; per completezza si riporta comunque la query finale compresa di filtri (evidenziati per diversi colori, divisi per righe):

```
INSERT INTO FULLMATCH
SELECT r2.codice AS cod2, r1.codice AS cod1, wordEditDistanceDiag (r1.frase, r2. frase ,
<k>)
FROM <tab1> r1, <qtab1> r1q, <tab2> r2, <qtab2> r2q
WHERE r1.codice = r1q.codice
AND r2.codice = r2q.codice
AND r1q.qgram = r2q.qgram

-- filtro di posizione
AND ABS (r1q.pos − r2q.pos) <= ROUND(<k> ∗ r2.wordlen)

-- filtro di lunghezza
AND ABS (r1.wordlen − r2.wordlen) <= ROUND(<k> ∗ r2.wordlen)

-- filtro di conteggio
GROUP BY r2.codice, r1.codice, r1.frase, r2.frase
HAVING      COUNT(∗) >= (r1.wordlen − 1 − (ROUND(<k> ∗ r2.wordlen) − 1) ∗ <q>)
AND         COUNT(∗) >= (r2.wordlen − 1 − (ROUND(<k> ∗ r2.wordlen) − 1) ∗ <q>)

AND         wordEditDistanceDiag (r1.frase, r2.frase, <k>) >= 0
```

1.1.3 Sub^2 matching

Per descrivere l'algoritmo di sub^2 matching è necessario dividere la spiegazione nelle tre parti seguenti, dove si spiegano in modo più rigoroso i passi che portano alla scrittura dell'algoritmo:

1 Q-grammi posizionali

Definizione:

> Un q-gramma posizionale di una sequenza ***s*** è costituito dalla coppia
> ***(i; [i, ..., i+q −1])***
> dove ***[i, ..., i+q−1]*** è il q-gramma di ***s*** avente ***i*** come posizione di partenza all'interno di ***s***.

Indicheremo con ***Gs*** l'insieme di tutti i q-grammi posizionali di ***s*** costituito da tutte le $|s| + q - 1$ coppie costituite dai q-grammi di ***s***.

Per il match eseguito sulle stringhe intere valgono le seguenti proprietà:

Proposizione 1 (Count Filtering):
Date le sequenze s_1 e s_2, se la loro distanza è minore di ***d*** allora la cardinalità di
$Gs_1 \cap Gs_2 \leq max(|s_1|, |s_2|) - 1 - (d - 1) * q$,
indipendentemente dalla posizione delle sequenze stesse.

Proposizione 2 (Position Filtering):
Se le sequenze s_1 e s_2 distano al massimo ***d*** allora il q-gramma della prima non può differire dal q-gramma della seconda per più di ***d***.

Come vedremo di seguito, gli stessi principi possono essere sfruttati per il calcolo di match fra sotto stringhe.

2 Sub²Count Filtering

(Primo passo del sub² match, codice Java reperibile in A2)

Quando ci riferiamo al Sub²Count Filtering implicitamente ci riferiamo al concetto che ne sta alla base [5] e prevede che entrambe le stringhe abbiano un determinato numero minimo di q-grammi, in modo che un q-gramma della prima stringa possa contenere una parte comune ad un q-gramma della seconda. In particolare il numero di q-grammi comuni deve essere determinato in funzione dei soli simboli delle sequenze stesse, senza l'uso di estensioni delle sequenze.

Proposizione:
Siano S_1 e S_2 due sequenze aventi una coppia $(S_1[i_1, ..., j_1, S_2[i_2...j_2])$ tale che $(ji - i_1 + 1) \geq minL$, $(j_2 - i_2 + 1) \geq minL$ e la funzione di edit distance calcolata su $(S_1[i_1...j_1], S_2[i_2...j_2])$ sia $\leq d$, allora la cardinalità di $GS_1 \cap GS_2$ sarà minore o uguale a $minL + 1 - (d + 1) * q$.

Partendo dalla soglia imposta nella Proposizione 1:

$$(max(|S_1|, |S_2|) - 1 - (d-1)*q)$$

Sostituendo $max(|S_1|, |S_2|)$ con la soglia di lunghezza minima minL,

Sottraendo il numero di q-grammi ottenuti espandendo la stringa con l'uso di caratteri jolly, otteniamo la formula:

$$minL-1-(d-1)*q-(q-1)*2$$

che può anche essere espressa come:

$$minL + 1 - (d + 1) * q$$

In questo modo si può imporre un primo vincolo sulla cardinalità dell'insieme composto da tutte le parti comuni dei q-grammi di S_1 e S_2.

3 Sub²Position Filtering

(Secondo e ultimo passo del sub² match, codice Java reperibile in A3)

Come esplicato dal sotto capitolo precendente abbiamo visto che il filtro Sub²Count non tiene conto delle posizioni delle parti comuni all'interno delle sequenze analizzate. Tale approccio, pur essendo particolarmente performante, non si rivela adatto nel caso in cui, ad esempio, si voglia trattare un testo scritto o, più in generale, un insieme di sequenze in cui la posizione relativa dei simboli sia significativa. Nasce quindi la necessità di realizzare un algoritmo capace di tener traccia di tali informazioni.

La tecnica del Sub²Position Filtering [5] prevede quindi di effettuare un'analisi dinamica delle posizioni relative e, in parte, dell'ordine delle parole uguali all'interno delle stringhe da confrontare. Come si può vedere dal codice Java in appendice, la funzione riceve in ingresso la soglia ***minL*** e la distanza relativa ***k*** calcolata con l'algoritmo di edit distance. Tramite una query seleziona poi le due sequenze S_1, S_2 e i loro codici. L'algoritmo implementa quindi due cicli, uno esterno per le parole della sequenza S_2 ed uno interno per le parole di S_1. Ad ogni posizione p_1 di S_1 viene associato un contatore quindi, ogni volta che una parola $S_2[p_2]$, appartenente alla stringa S_2, risulta uguale ad una parola $S_1[p_1]$, appartenente alla stringa S_1, vengono incrementati tutti i contatori da p_1 a p_1+w-1, con $w = minL$ indicante l'ampiezza della finestra del filtro.

Il filtro termina positivamente qualora un generico contatore raggiunge una soglia di conteggio ***c*** fissata e contemporaneamente la parola da esso associata $S_1[p_1]$ risulta uguale alla parola $S_2[p_2]$ del ciclo esterno. In ogni altro caso il filtro darà esito negativo. Le coppie di sequenze così individuate vanno infine sottoposte ad un ultimo passagio all'interno del quale ne viene calcolata, e valutata, la distanza, la quale deve essere minore della soglia di distanza ***k***.

In tal modo ne vengono dunque individuate tutte le coppie di sequenze S_1 e S_2 distanti al massimo ***k*** e lunghe almeno ***minL*** simboli. Si noti come la soglia ***c*** utilizzata in tale filtro sia la stessa del Sub²Count Filtering con lunghezza del q-gramma pari a 1.

1.2 Un approccio differente, Suffix tree e Suffix array

Questo approccio non è inserito nel progetto Extra, ma è visibile nel software ACGT sviluppato e descritto da D. Gelmini [8], ed è utilizzato sostanzialmente come il whole match ed il sub^{2}match, ovvero permette di trovare sequenze e sotto sequenze di un pattern in un testo.
Dato che l'operazione appena descritta deve poter essere svolta sia su sequenze di tipo genetico quali parti del DNA o sequenze di amminoacidi, sia su testi veri e propri, come ad esempio un manuale di istruzioni, il suo funzionamento tramite suffix tree necessita di una tabella dei simboli Σ, che può essere generata a partire dal testo stesso del quale successivamente verrà eseguita la conversione. Il risultato finale consiste in un nuovo file marcato con l'estensione ".sym". Una volta trasformato il testo in una sequenza, si passa al layer sottostante il quale serve alla creazione dell'indice. Esso è costituito da un Suffix Array leggermente modificato per gestire le sequenze, ma comunque riconducibile ad un Suffix Tree.

Dopo aver dato una breve descrizione delle strutture di massima, passiamo all'algoritmo di match vero e proprio ("All-Against-All").

Esso richiede innanzitutto una struttura dati per il salvataggio dei nodi durante l'esplorazione, implementata come un generico stack nel quale salvare la coppia di nodi ed il corner ad essa associato. Di basilare importanza risulta essere inoltre la funzione capace di calcolare il corner sucessivo, il quale non solo fornisce il grado di match ma risulta anche fondamentale per il proseguo dell'esplorazione nei successivi livelli dell'indice.
La simulazione di un Suffix Tree con un Suffix Array richiede un diverso approccio di esplorazione, infatti, mentre nel primo esistono i nodi, con le loro biforcazioni, nel secondo tali nodi vanno individuati in modo dinamico, estrapolandoli a partire dai suffissi memorizzati e dal loro ordinamento.

Si simula quindi un nodo radice dal quale far partire le esplorazioni iniziali. In esso si analizza il primo simbolo di ogni suffisso, effettuando una partizione dell'insieme dei suffissi in base a tale informazione.
Si ottiene così il primo livello dell'albero. Ovviamente, se il primo simbolo risulta essere uguale in tutti i suffissi, si ripete l'analisi sul secondo e così via fino al raggiungimento della condizione cercata. In tal modo è quindi possibile determinare quelli che, nella teoria dei Suffix Tree, vengono indicati con il nome di Prefissi Comuni.
Determinati i nodi del primo livello si reitera il procedimento per ognuno di essi e così via per quelli a seguire fino al raggiungimento dei nodi foglia. In tal modo è quindi possibile risalire alla struttura di un Suffix Tree e l'esplorazione può dunque essere analogamente compiuta come richiesto dall'algoritmo per il match "All-Against-All". Sempre come nel caso precedente di match, anche con questa tecnica è possibile utilizzare dei filtri, infatti, posizionando opportunamente i controlli sui vincoli di minima lunghezza (minL) e di massima distanza (d), già trattati in precedenza, è possibile eliminare a priori interi sotto alberi.

L'esecuzione di queste modifiche all'algoritmo ha portato ad un software, sviluppato e descritto in modo completo da [8], in grado di effettuare una ricerca di occorrenze fra tutte le possibili sequenze, e sotto sequenze, di due testi dati, il quale imponga al tempo stesso un vincolo di minima lunghezza, ed uno di massima distanza, sul set dei risultati così ottenuti.

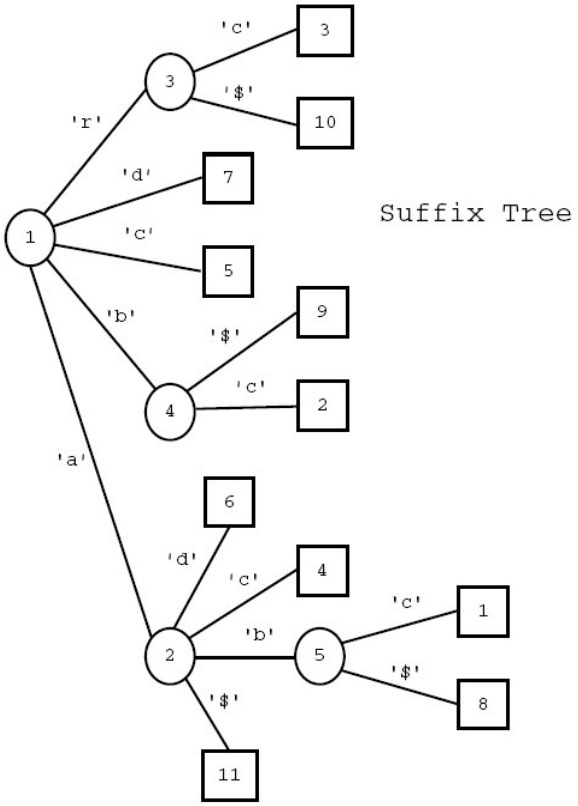
Dato che i risultati tra i due differenti algoritmi (indicati e abbreviati come *Sub^2 match* e *Suffix Array*) a parità di parametri, sono equivalenti, è possibile comparare in modo approfondito le prestazioni sui diversi dataset scelti (*).

Con le immagini successive riguardanti Suffix Tree e Suffix Array, concludiamo la parte di implementazione per l'*approximate sequence matchin*g.

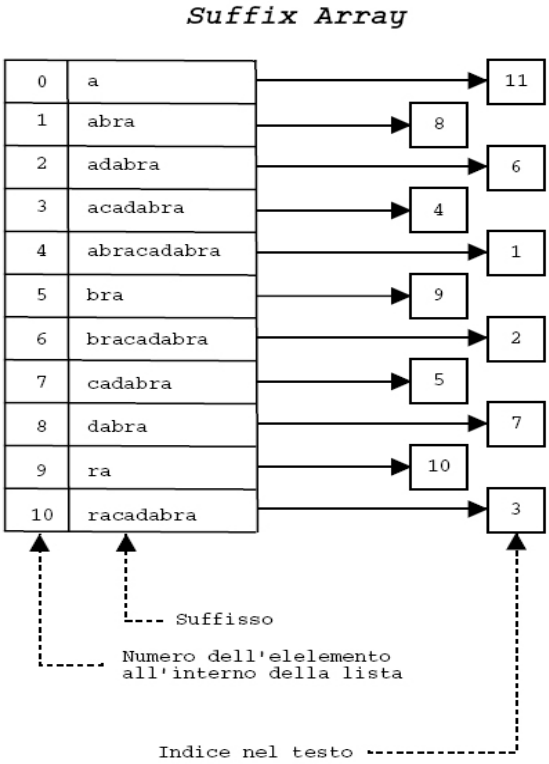
(*) Vedi paragrafo 5.3 "Analisi sperimentale comparata".

*Riportiamo nelle immagini successive (1.2-a e 1.2-b) la struttura di massima di un Suffix Tree per la parola "abracadabra" (***1 a 2 b 3 r 4 a 5 c 6 a 7 d 8 a 9 b 10 r 11 a***) e del Suffix Array riferito alla medesima parola:*

(Fig. 1.2-a – [8] Suffix Tree)



(Fig. 1.2-b – [8] Suffix Array)



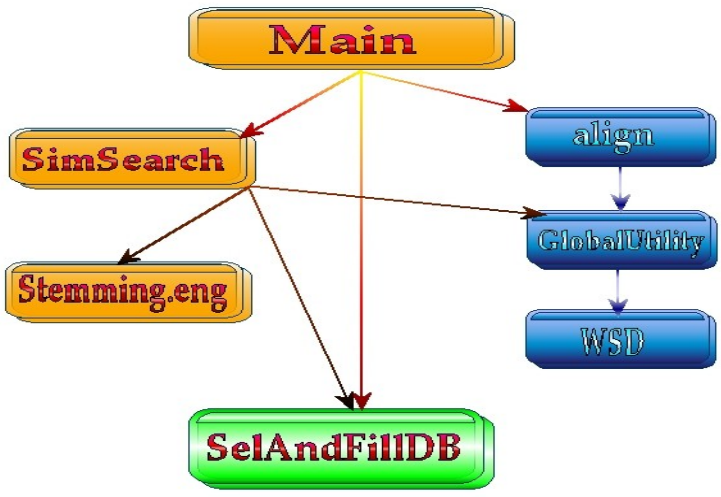
1.3 Extra, un sistema EBMT

Tutti i medoti descritti nel paragrafo 1.1 e nei suoi sotto paragrafi, sono implementati nel programma scritto e creato per *ISgroup* [14] da R. Martoglia [3] in linguaggio Java e SQL chiamato EXTRA (EXample-based TRanslation Assistant), che ha come scopo la ricerca di similarità tra frasi.

A questo progetto sono state aggiunte negli anni varie funzionalità (ad esempio il *word sense disambiguation* [12]) e con questa tesi si vuole continuare questa opera di ampliamento, oltre ad analizzare, anche in modo comparato, le performance del programma.

Entrando nel particolare, alla base dell'ampliamento sta la portabilità su altri DBMS, in quanto il progetto originale era studiato per interfacciarsi unicamente al DBMS Oracle 8i [15], mentre ora, nella versione 2.7.1, si può interfacciare efficientemente, salvo problematiche relative al sistema in uso, a 5 differenti DBMS e a più di 15 versioni diverse degli stessi.

Nel disegno sottostante, una panoramica dei package di Extra:
(In **blu**, a destra i 3 package originali, in **arancione** in alto ed a sinistra i 3 modificati, infine, in **verde** in basso il nuovo package "SelAndFillDB")



(Fig. 1.3-a – I package di Extra)

1.3.1 Elaborazione e stemming

Dopo aver descritto i metodi di implementazione possibili ed utilizzati, passiamo ad altre tecniche, sempre implementate nel software, per migliorare e velocizzare le operazioni di ricerca. Per questo, infatti, è necessario prima elaborare le frasi trattate, in particolare, è possibile scegliere due differenti processi, semplice rimozione della punteggiatura e stemming (o normalizzazione). Togliendo i segni di punteggiatura si lasciano le frasi immutate ma pronte per una corretta individuazione delle parti simili. Lo stemming invece risulta un processo più complicato, infatti, si tratta di eliminare parole insignificanti al fine semantico, come articoli e preposizioni, e di rendere le altre in un formato standard portando ad esempio i nomi al singolare, i verbi all'infinito, etc...

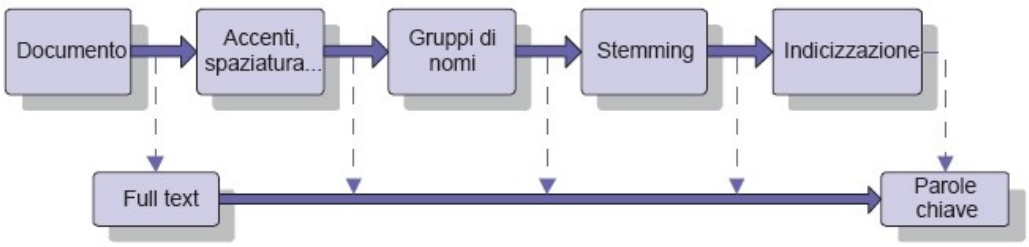
Esempio:

"Before beginning the driver installation , you should exit the X-server"

viene trasformata in:

"before begin driver installation shall exit X-server"

Con questa tecnica la ricerca di frasi simili si distacca dalla forma differente delle parole quindi si può concentrare solo su quelle che danno veramente significato alla frase. Nell'immagine sottostante (Fig. 1.2-a) uno schema esemplificativo.



(Fig. 1.3.1-a – [3] Processo di stemming)

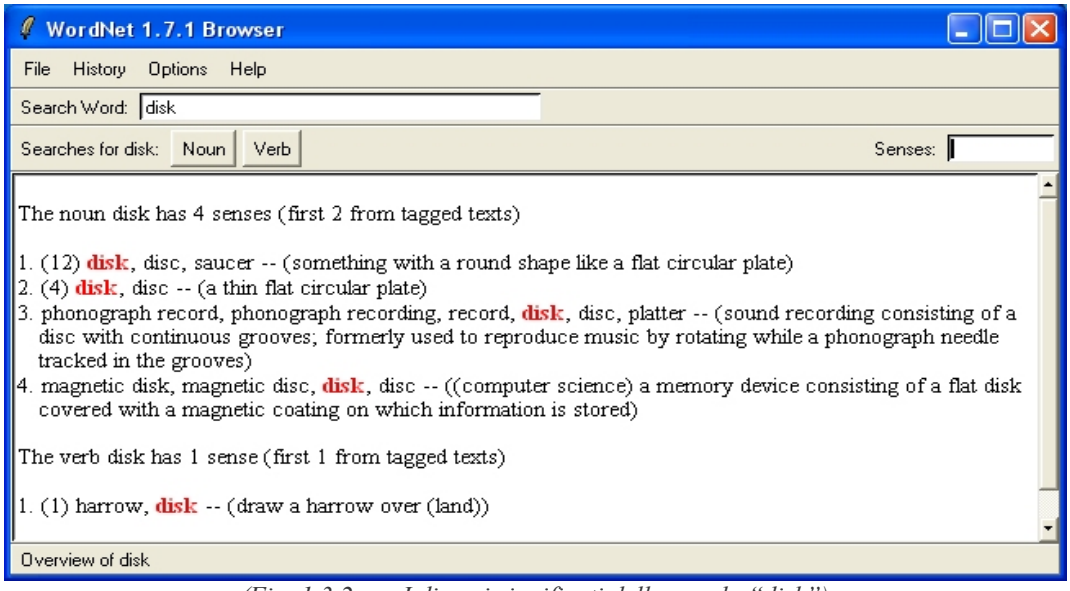
1.3.2 Word Sense Disambiguation

Come per la tecnica dello *stemming* (Paragrafo 1.3.1), il *Word Sense Disambiguation (*)* [13] è un ulteriore algoritmo studiato e inserito nel software, da E. Stefanini [12], utilizzato per fornire risultati migliori e precisi sempre nell'ambito di ricerca di patterns in un testo.

Il problema di determinare in modo automatico il significato più appropriato di una parola in base al contesto, ossia alla frase, in cui si trova, è un problema non semplice da risolvere, ma una volta fatto esso consente di considerare nei suggerimenti delle traduzioni anche il significato delle parole polisemiche.

Partendo dagli studi compiuti sul word sense disambiguation, è stato realizzato un modulo in grado di determinare il significato più appropriato di sostantivi e verbi presenti in una frase. A tal fine, sono stati implementati algoritmi (uno per i nomi e uno per i verbi) in grado di calcolare un valore che rappresenti la "confidenza" con cui si può dire che il significato x di un termine è migliore degli altri in quel determinato contesto.

Utilizzando questi algoritmi insieme al modulo originale, si ottengono suggerimenti per la traduzione che tengono conto del significato dei termini.



(Fig. 1.3.2-a – I diversi significati della parola "disk")

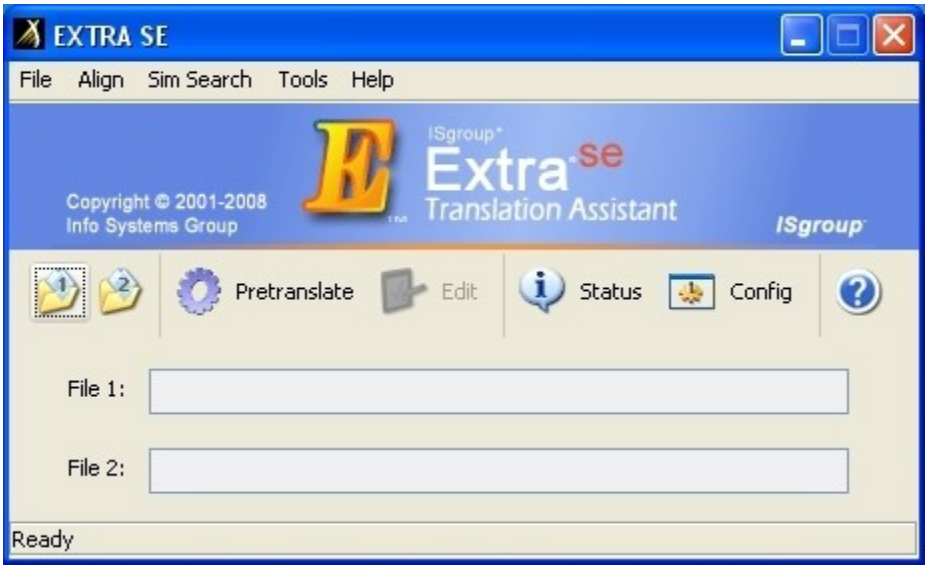
(*) In italiano "Risoluzione di ambiguità semantiche".

1.3.3 Come utilizzare Extra

Supponendo di avere a disposizione un DBMS già pronto all'uso *(*),* un sistema compatibile con una versione di Java (JRE-JDK) superiore alla 1.5, il programma compilato per la versione in uso e altre caratteristiche che non nominiamo, descriviamo brevemente, riportando anche alcuni *screenshoot* del programma, la procedura per il funzionamento di Extra:

1) Lanciare il programma linkando ad esso le librerie necessarie:

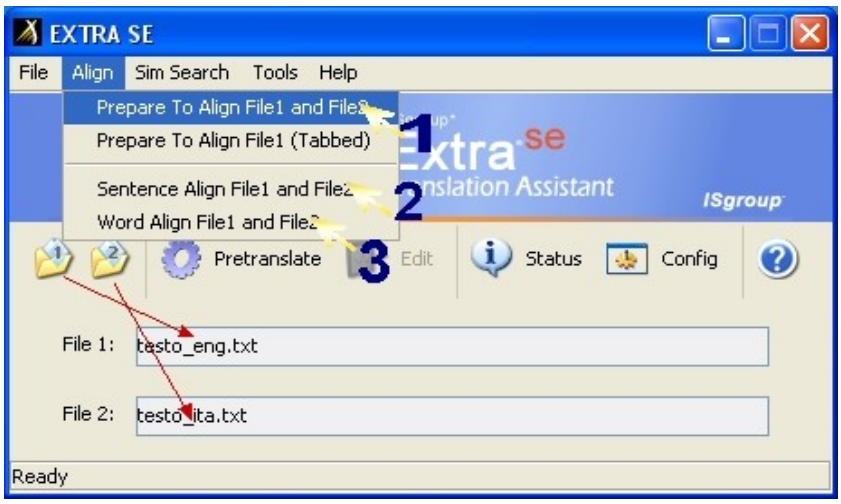
   es. ***c:\Programmi\java\jdk1.6.0_03\bin\java -cp ..\lib\ojdbc6.jar; ..\lib\colt.jar; ..\lib\commons-logging.jar; ..\lib\jbcl.jar; ..\lib\jbccclient.jar; ..\lib\jwnl.jar; ..\lib\lp.jar; ..\lib\monetdb-1.8-jdbc.jar; ..\lib\mysql-connector-java-5.1.6-bin.jar; ..\lib\postgresql-8.2-508.jdbc4.jar; ..\lib\qtag.jar; ..\lib\jaybird-full-2.1.6.jar; main.Extra***



(Fig. 1.3.3-a – Extra, main)

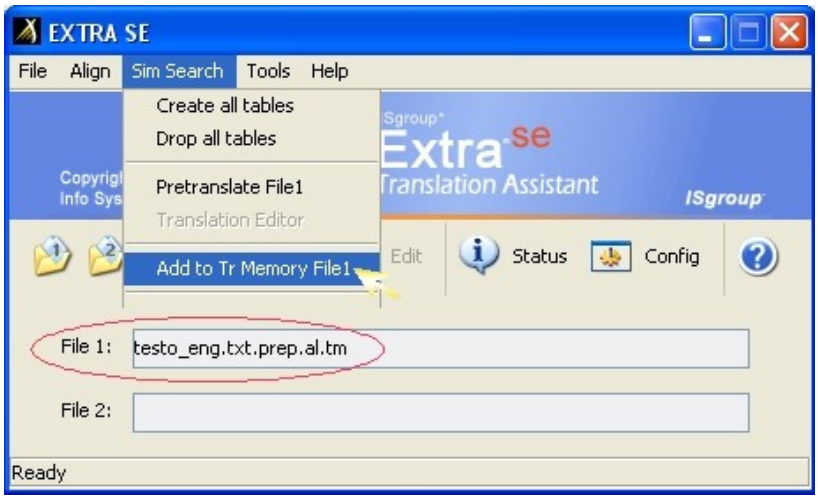
2) Aprire i due file, File1 come testo in lingua originale (inglese) e File2 come testo in lingua tradotta (in realtà qualsiasi, in figura 1.3.3-b italiano), e dal menu *"aling"* seguire in ordine i passaggi *"Prepare to align File1 and File2", "Sentence Aling File1 and File2"* ed infine *"Word Aling File1 and File2".*

(*) Riferimenti ai DBMS nel paragrafo 2.2, dettagli d'installazione nel paragrafo 4.3.1



(Fig. 1.3.3-b – Align and create TM file)

3) Ora il file per la TM generato (*testo_eng.txt.prep.al.tm*) è inserito automaticamente in *"File1"*, ma è necessario inserirlo nella tabella del DBMS tramite il menu *"SimSearch"* opzione *"Add to Tr Memory File1"*, ma prima di ciò è necessario avere creato tutte le tabelle (*"Create all tables"*).



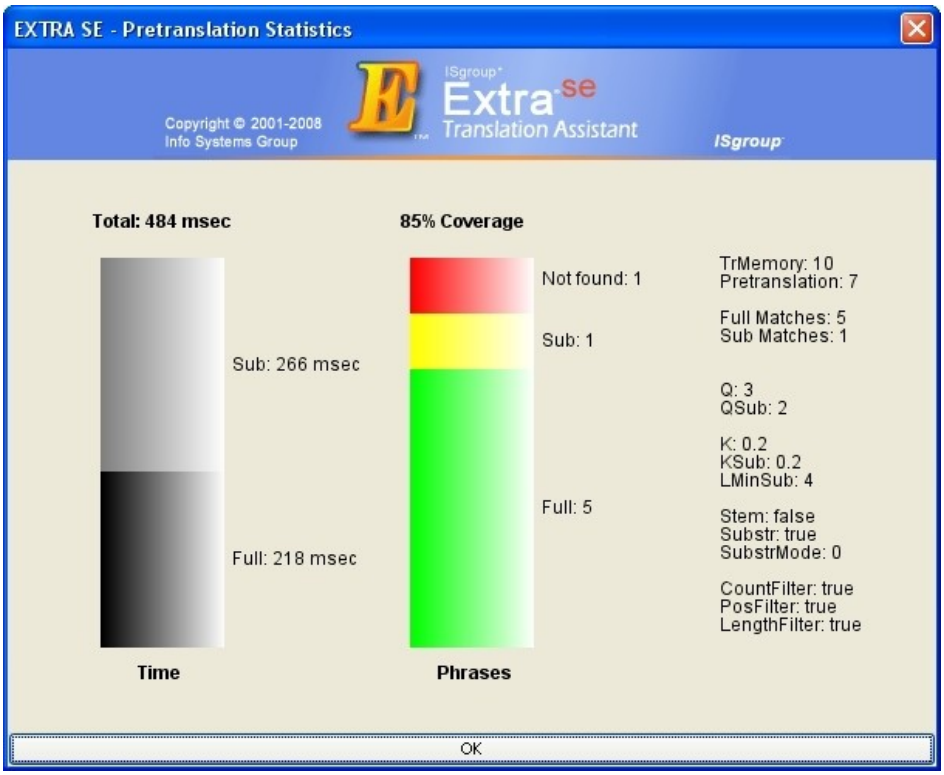
(Fig. 1.3.3 -c – Add to TM)

4) Dopo aver "riempito" la TM ed aver visualizzato i messaggi di conferma, selezionare il File1 come pattern ed eseguire la ricerca *"pretranslate"* dopo aver eventualmente settato i parametri dal bottone *"config"*.

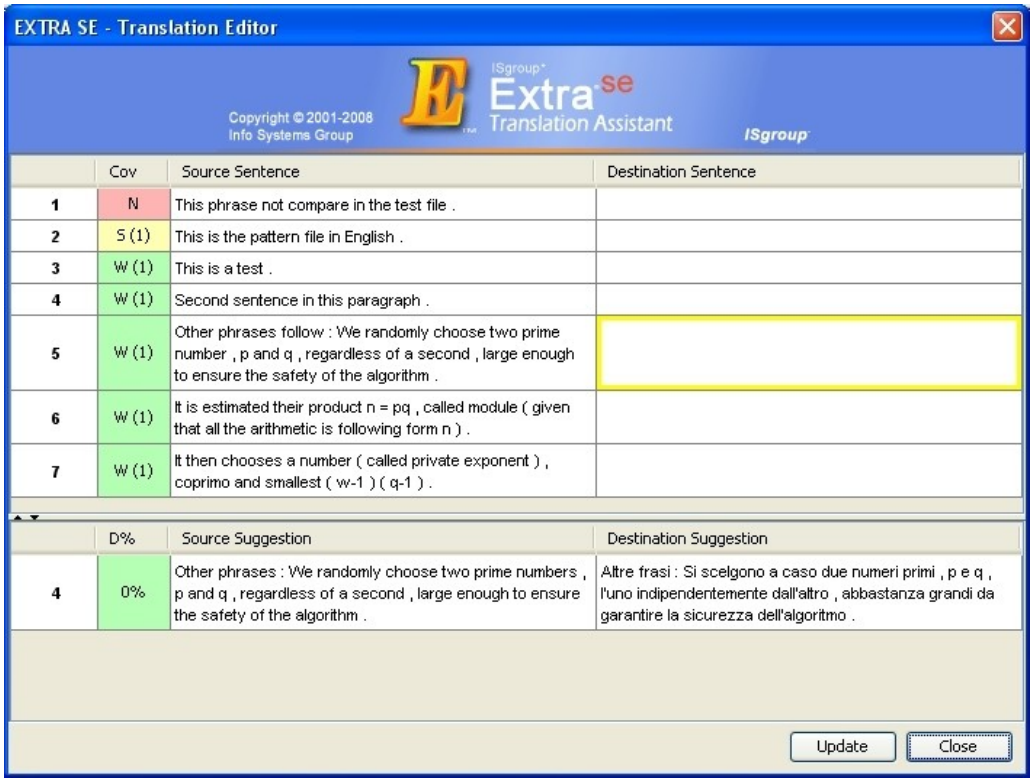
(Fig. 1.3.3 -d – Pretranslate)

Considerando che il processo di utilizzo, in modo più avanzato, è più complesso ed utilizzato sopratutto a fini di ricerca, sopratutto la parte d'impostazione dei parametri di configurazione, lasciamo ad un eventuale utente la possibilità di testare il software nella maniera che preferisce.

Riportiamo nelle immagini seguenti (1.3.3-e e 1.3.3-f) il risultato della traduzione appena effettuata. Possiamo vedere come, in un caso preparato chiaramente ad hoc, si riescano a trovare 5 frasi complete, 1 parziale ed 1 no, sull'insieme totale proposto. Il suggerimento fornito è una traduzione effettuata in precedenza, da far correggere ad uno specialista, che comunque impiegherà un tempo nettamente inferiore a quello di una traduzione normale "da capo". I testi dei tre files sono riportati in appendice (A5).



(Fig. 1.5-e - Pretranslate Statistics)



(Fig. 1.5-f – Translation Editor)

Capitolo 2

Le tecnologie utilizzate

In questo capitolo elenchiamo e descriviamo le tecnologie utilizzate, intendendo come tecnologie i linguaggi di programmazione, i DBMS e qualsiasi supporto sviluppato da terzi fortemente utilizzato per lo sviluppo del software. In particolare, descriveremo le tecniche Java utilizzate per la connessione ai vari DBMS, che, a loro volta, approfondiremo nel capitolo 2.2.

2.1 La programmazione in Java: connessione a un DBMS

Il linguaggio di programmazione e l'ambiente Java sono stati sviluppati da un gruppo di ricercatori della Sun Microsystems che iniziarono a studiare la possibilità di creare una tecnologia in grado di integrare con le allora attuali conoscenze nel campo del software con l'elettronica di consumo. Avendo subito focalizzato il problema sulla necessità di avere un linguaggio indipendente dalla piattaforma il gruppo iniziò i lavori nel tentativo di creare un linguaggio che estendesse il C++ e da questo ottennero risultati eccellenti, primi fra tutti la semplicità e la portabilità. Java è un linguaggio di programmazione orientato agli oggetti, dove tutte le istruzioni sono contenute nelle classi, le quali, una volta compilate, costituiscono ognuna un modulo a sé stante, inoltre, in Java non vi è la possibilità di accedere direttamente alla memoria, non sono previsti né puntatori, né funzioni di allocazione e deallocazione. Anche se quest'ultima caratteristica rende meno flessibile il linguaggio, allo stesso tempo lo rende molto più sicuro ed affidabile in quanto è il "garbage collector" a pensare a tutto ciò.
Infine, attraverso i metodi nativi, un programma Java può includere programmi scritti in C e C++, dando la possibilità di usare software sviluppato con questi linguaggi. Dopo questa breve panoramica, passiamo ai punti cardine dell'argomento ovvero la spiegazione delle tecnologie che hanno permesso la portabilità del sistema su altri DBMS.

2.1.1 SQLJ

SQLJ è uno standard, nato per inserire istruzioni SQL direttamente all'interno di codice java, riducendo la quantità di codice necessario all'esecuzione delle istruzioni SQL è nato dalla collaborazione dei maggiori produttori di database mondiali (Compaq, IBM, Informix, Oracle, Sybase e Sun).

Si tratta di istruzioni statiche, cioè definite in fase di scrittura del codice, che quindi non possono mutare in fase di esecuzione.

SQLJ è composto da un traduttore (una sorta di precompilatore) e da una libreria di runtime. Scopo del traduttore è quello di andare a convertire le istruzioni SQL in istruzioni java che vengono passate al runtime che si occupa dell'esecuzione del codice, normalmente attraverso l'uso di driver JDBC. Più in dettaglio il traduttore si occupa di verificare la correttezza sintattica delle istruzioni, quella semantica andando a verificare l'esistenza di oggetti come le tabelle, accedendo quindi al database e della consistenza fra i tipi di dati definiti nel codice java e quelli restituiti dalle istruzioni SQL. Infine trasforma il codice SQLJ in codice java contenente chiamate JDBC.

Le istruzioni SQLJ devono essere salvate in un file con estensione .SQLJ Questo file è convertito dal traduttore in un file *.java*, contenente le chiamate JDBC ed infine passato al compilatore java che crea il file di bytecode *.class*. Infine il runtime si occupa di generare le opportune chiamate al driver JDBC.

Conclusioni:

I vantaggi principali derivanti dall'utilizzo di SQLJ piuttosto che JDBC sono:

- Maggiore semplicità: dovuta all'interfaccia ad alto livello, che consente di scrivere direttamente istruzioni SQL, ottenendo codice più compatto.
- I programmi SQLJ tendono ad essere più brevi di quelli JDBC.
- La sintassi SQL può essere controllata in fase di compilazione.

Gli svantaggi principali sono:

- La difficile portabilità del codice su differenti DBMS.
- L'impossibilità di introdurre istruzioni dinamiche.
- SQLJ richiede un preprocesso.
- Molte IDE non hanno un supporto specifico per SQLJ.
- Non c'è supporto SQLJ in alcune piattaforme middleware, come Hibernate.
- Il codice SQLJ è trasformato comunque in codice Java che contiente chiamate JDBC.

2.1.2 JDBC
(Java DataBase Connectivity)

Come si può intuire dal nome, esso è un connettore per database che consente l'accesso ai vari database da qualsiasi programma scritto con il linguaggio di programmazione Java, indipendentemente dal tipo di DBMS utilizzato.
È costituito da una API, raggruppata nel package *java.sql*, che serve ai client per connettersi a un database. Fornisce metodi per interrogare e modificare i dati.
È orientato ai database relazionali ed è Object Oriented; è implementato in codice nativo e non in Java. JDBC, a livello di architettura, prevede l'utilizzo di un "driver manager", che espone alle applicazioni un insieme di interfacce standard e si occupa di caricare a "run-time" tutti i driver opportuni per governare gli specifici DBMS. Le applicazioni Java utilizzano le "JDBC API" per dialogare con il JDBC driver manager, mentre il driver manager usa le JDBC driver API per dialogare con i singoli driver che governano i DBMS specifici.

JDBC ammette che esistano diverse implementazioni e vengano utilizzate dalla stessa applicazione. L'API fornisce un meccanismo che carica dinamicamente i driver appropriati e li registra nel JDBC Driver Manager. Esso funge da creatore di connessioni. Le connessioni JDBC supportano la creazione e l'esecuzione delle istruzioni. Esse possono essere comandi SQL come INSERT, UPDATE, DELETE, interrogazioni come SELECT o chiamate a stored procedure.

I tipi di istruzioni supportati sono:

- Statement - l'istruzione viene inviata al database di volta in volta;
- Prepared Statement - l'istruzione viene compilata una sola volta, in modo che le chiamate successive siano più efficienti;
- Callable Statement - usati per chiamare le stored procedure.

I comandi di scrittura come INSERT, UPDATE e DELETE restituiscono un valore che indica quante righe sono state affette (inserite, modificate, cancellate) dall'istruzione. Essi non restituiscono altre informazioni.

Le interrogazioni (query) restituiscono un *result set*, ovvero un insieme di risultati identici ad una normale tabella che si ottiene da query in ambiente client del DBMS. È possibile spostarsi nel result set riga per riga (tramite il metodo next()). Si può accedere alle colonne di ogni singola riga chiamandole per nome o per numero. Il result set può essere costituito da un numero qualsiasi di righe. Esso comprende dei metadati che indicano il nome, il tipo e le dimensioni delle colonne.

Conclusioni:

Gli svantaggi principali derivanti dall'utilizzo di JDBC piuttosto che SQLJ sono:

- Maggiore difficoltà nella scrittura dell'SQL.
- I programmi JDBC tendono ad essere più lunghi di quelli SQLJ.
- La sintassi SQL non può essere controllata in fase di compilazione.

I vantaggi principali sono:

- I comandi sono scritti in SQL standard.
- Le procedure restano immutate nel cambiare DBMS.
- È l'unico che supporta istruzioni dinamiche.
- Non richiede preprocessi.
- Quasi completa portabilità tra differenti DBMS.
- L'aggiornamento dei driver non richiede la riscrittura del codice.
- Si possono usare differenti versioni di un DBMS con lo stesso driver.
- L'aggiornamento di un driver consiste nella semplice copia di un file.

Essendo che uno degli scopi di questa tesi riguarda la portabilità del sistema su differenti DBMS, la soluzione migliore è sicuramente l'utilizzo di JDBC.

2.2 DataBase Management System

Un Database Management System (abbreviato in DBMS) è un sistema software progettato che consente la creazione e la manipolazione efficiente di database (ovvero di collezioni di dati strutturati) solitamente da parte di più utenti. I DBMS svolgono un ruolo fondamentale in numerose applicazioni informatiche, dalla contabilità, la gestione delle risorse umane e la finanza fino a contesti tecnici come la gestione di rete o la telefonia.

Se in passato i DBMS erano diffusi principalmente presso le grandi aziende e istituzioni (che potevano permettersi l'acquisto delle grandi infrastrutture hardware necessarie per realizzare un sistema di database efficiente), oggi il loro utilizzo è diffuso praticamente in ogni contesto. L'espressione applicazione enterprise, che nel gergo informatico si riferisce ad applicazioni legate al business delle aziende che le utilizzano, implica quasi "per definizione" la presenza di una o più basi di dati amministrate da uno o più DBMS.

Un DBMS è differente dal concetto generale di applicazione sui database, in quanto è progettato per sistemi multi-utente. A tale scopo, i DBMS si appoggiano a kernel che supportano nativamente il multitasking e il collegamento in rete. Una tipica applicazione per la gestione dei database non includerebbe, infatti, tali funzionalità, ma si appoggerebbe al sistema operativo per consentire all'utente di fruirne dei vantaggi.

Un DBMS può essere costituito da un insieme assai complesso di programmi software che controllano l'organizzazione, la memorizzazione e il reperimento dei dati (campi, record e archivi) in un database. Un DBMS controlla anche la sicurezza e l'integrità del database. Il DBMS accetta richieste di dati da parte del programma applicativo e "istruisce" il sistema operativo per il trasferimento dei dati appropriati.

Vi sono diversi prodotti software (complessi) disponibili sul mercato e si dividono in due grandi categorie:

- Proprietari: *Access, DB2, Oracle, Informix, SQLServer, ...*
- Open source: *MySQL, PostgreSQL, FireBird, MonetDB, ...*

I motivi della scelta dell'utilizzo di DBMS sono:

- Indipendenza dei dati e accesso efficiente
- Tempo ridotto di sviluppo dell'applicazione
- Integrità dei dati e sicurezza
- Amministrazione dei dati uniforme
- Accesso concorrente, ripristino da crash
- Prestazioni migliori rispetto ad altre soluzioni

Nei sotto paragrafi successivi, proseguiamo elencando e descrivendo i DBMS installati, testati ed utilizzati con il software di traduzione. Accanto alla numerazione del paragrafo troviamo il logo ufficiale del DBMS relativo; in caso di richieste specifiche si faccia riferimento alla bibliografia ed alla pagina finale di questo documento.

2.2.1 Oracle [15][16]

Oracle ha coperto e copre tuttora un ruolo fondamentale nel campo dei DBMS, è il pioniere per eccellenza nello studio delle basi di dati, infatti è il primo, in termini di tempi di sviluppo, ad implementare nuove tecnologie:

- 1979, il primo DBMS commerciale basato su SQL
- 1983, il primo che supporta sistemi SMP *(*)*
- 1986, il primo che supporta una base di dati distribuita
- 1993, il primo ad essere testato con il linguaggio standard ANSI SQL
- 1995, il primo ad implementare la tecnologia a 64-bit.
- 1998, il primo che ha implementato ed in modo nativo la JRE *(**)*
- 1999, il primo DBMS commerciale disponibile per sistemi Gnu/Linux
- 1999, il primo a supportare XML *(***)*
- 2001, il primo DBMS a poter girare su sistemi cluster, implementando una tecnologia, chiamata RAC (Real Application Clusters) la quale permette al DBMS di girare anche in ambienti di Grid-Computing, che permette, infine, di ridurre i costi dell' hardware.

A differenza di altri DBMS, Oracle non è open source. Questo significa che i sorgenti non sono di pubblico accesso, il che comporta l'impossibilità di installare i DBMS di Oracle sui sistemi i cui binari non siano stati pre-compilati e distribuiti dalla società stessa e significa anche che non vi sarà modo di verificare le procedure che il software svolgerà sulle macchine su cui gira.

(*) Sistemi dotati di MultiProcessore Simmetrico cioè un architettura hardware dotata di più processori dove questi possono accedere equamente a tutta la memoria RAM del sistema.
(**) JRE (Java Runtime Environment) - Sun Microsystems - www.java.com
(***) XML (eXtensible Markup Language) metalinguaggio creato e gestito dal W3C.

L' unica edizione di facile accesso è la Express Edition, reperibile dal web, ma i cui binari sono disponibili solo per i sistemi Windows e Gnu/Linux basati su architettura x86; per quelle a pagamento invece la gamma di sistemi compatibili è molto più vasta.

Facendo riferimento alla versione utilizzata (9i , est. 9.2.0.1) si riportano alcuni dati del DBMS sul sistema in uso:

Sistema: Windows XP (HE x86, 32Bit) – Disco: NTFS

Occupazione in RAM:	90 – 130 Mb
Occupazione su disco:	3500 Mb
Tempi d'installazione:	40 - 120 Minuti

Si noti che, a differenza di altri DBMS cui sono state testate le versioni più recenti, con Oracle non è stato possibile, in quanto dalla versione 10 non è permesso un funzionamento adeguato sulla macchina utilizzata, sia per quanto riguarda le performance, che per quanto riguarda i requisiti. Evidenziamo quindi che, in un ipotetico test su una macchina dove sia permessa la funzionalità di tali sistemi, le versioni successive alla 10 di Oracle, secondo la maggior parte delle guide disponibili, dovrebbero incrementare le performance di un 12% rispetto alla versione 8.2 di PostgreSQL e di una percentuale leggermente minore rispetto la versione da noi utilizzata.

2.2.2 MySQL [17]

La società che detiene i diritti di MySQL, la MySQL AB ha dichiarato nel 2006 di essere costituita da 300 dipendenti dislocati in 25 Paesi e di contare circa 30.000 download del DBMS al giorno e suppone che ci siano state circa 5 milioni di installazioni del server in tutto il mondo.

Installazione libera ed alta Compatibilità, infatti, MySQL è rilasciato con licenza GPL e questo significa che i sorgenti sono di pubblico accesso, ovvero, ela possibilità di compilare il software su qualsiasi sistema operativo ricavando così i binari, pronti ad essere utilizzati sul sistema che si preferisce.

MySQL è scritto in C e C++, la documentazione ufficiale offre anche un aiuto per chi avesse bisogno di provare a compilare il software su qualsiasi sistema operativo discretamente diffuso ed utilizzando i tools *automake*, *autoconf* e *libtools* viene garantita anche un elevata compatibilità.

Il linguaggio SQL di MySQL comprende anche numerose estensioni che sono tipiche di altri DBMS, quali PostgreSQL ed Oracle. In questo modo le query non standard scritte per altri DBMS saranno interpretate e salvo alcuni casi particolari, funzioneranno senza problemi.

Ci sono alcuni sistemi operativi volti a semplificare notevolmente il lavoro ai propri utenti, e questo è il caso del sistema Mac OS X. Per aiutare dunque i propri utenti ed evitare loro di dover compilare un DBMS, Apple ha scelto di pre-installare MySQL sui suoi prodotti.

Nonostante sia evidenziato spesso nell'ambito della cultura open source, l'importanza di compilare sulla propria macchina il software prima di utilizzarlo, MySQL fornisce anche i binari (software pre-compilato) per moltissimi sistemi, ad esempio: Windows, Gnu/Linux, BSD, Solaris, Mac OS X, Hp-UX, IBM AIX,... per architetture sia a 32 che a 64bit.

Facendo riferimento alle versione utilizzate (4.1.22, 5.0.22, 5.1 e 6.0.5-alfa) si riportano alcuni dati del DBMS sul sistema in uso:

Versione 4.1.22
Sistema: Windows XP (HE x86, 32Bit)
Disco di tipo NTFS

Occupazione in RAM:	15 – 35 Mb
Occupazione su disco:	40 Mb
Tempi d'installazione:	5 - 20 Minuti

Versione 5.0.22
Sistema: Windows XP (HE x86, 32Bit)
Disco di tipo NTFS

Occupazione in RAM:	25 – 50 Mb
Occupazione su disco:	80 Mb
Tempi d'installazione:	5 - 20 Minuti

Versione 6.0.5
Sistema: Windows XP (HE x86, 32Bit)
Disco di tipo NTFS

Occupazione in RAM:	40 – 70 Mb
Occupazione su disco:	165 Mb
Tempi d'installazione:	5 - 20 Minuti

Versione 5.1
Sistema: Gnu/Linux (Kubuntu 7.04 AMD-X64, Kernel 2.6.20-17)
Disco di tipo Raiserfs

Occupazione in RAM:	30 – 50 Mb
Occupazione su disco:	180 Mb (pacchetto)
Tempi d'installazione:	5 - 20 Min

2.2.3 MonetDB [18]

MonetDB è un DBMS open-source, sviluppato presso il Consiglio Nazionale delle Ricerche, Istituto per la Matematica e Informatica (CWI; Centrum voor Wiskunde en Informatica) nei Paesi Bassi.
È stato progettato per fornire prestazioni elevate a query complesse nei confronti di banche dati di grandi dimensioni. In quanto tale, MonetDB può essere utilizzato in aree di applicazione che, a causa di problemi di prestazioni non sono indicate per l'utilizzo di database a tecnologia in tempo reale. MonetDB è stato utilizzato con successo e con alte prestazioni per applicazioni di data mining, OLAP, GIS, XML Query, testo e recupero di informazioni multimediali. MonetDB ha introdotto innovazioni a tutti i livelli di un normale DBMS, infatti, utilizza un modello di archiviazione basato sulla frammentazione verticale e una moderna CPU-tuned per vettorializzare l'esecuzione di query, architettura che spesso dà a MonetDB un vantaggio di più di 10 volte sullo stesso algoritmo confrontato con un interprete tipico di un RDBMS. MonetDB è uno dei primi DBMS a concentrare i propri sforzi per indirizzare l'ottimizzazione di query a sfruttare le cache della CPU. MonetDB è automatico per quanto riguarda la regolazione degli indici, di ottimizzazione di query in tempo reale, etc..

È disponibile in versione pre-compilata per le principali piattaforme, e compilabile dai sorgenti per tutte le altre.

Versione 5.4
Sistema: Windows XP (HE x86, 32Bit)
Disco di tipo NTFS

Occupazione in RAM:	30 – 40 Mb
Occupazione su disco:	15 Mb
Tempi d'installazione:	5 - 30 Minuti

NB: Purtroppo, la perdita di informazioni durante il caricamento, ha indotto a pensare che questo DMBS non è affidabile per la memorizzazione dei dati. I test sono stati effettuati su diverse versioni, su diverse macchine con diversi driver JDBC, ma senza miglioramenti effettivi; non volendo condannare l'operato della CWI, ne si sconsiglia l'utilizzo con l'applicazione in questione.

2.2.4 PostgreSQL [19]

PostGreSQL (alias “Postgres”) è un database indipendente al 100%, non è gestito da nessuna azienda, non appartiene a nessuno, non soffre di “politiche aziendali” o problemi di mercato; è nato come un esperimento, all'università della California con sede a Berkeley, e continua ad esserlo dopo circa 20 anni. Viene sviluppato da una grande comunità di programmatori ed è finanziato da molte aziende interessate al progetto.

PostGreSQL è un DBMS open source. Ha più di 15 anni di sviluppo attivo ed ha un' architettura testata che gli ha permesso di acquisire una reputazione positiva per affidabilità, integrità dei dati e precisione.

Funziona su tutti i sistemi operativi maggiormente noti e rispetta completamente il modello ACID (Atomicity, Consistency, Isolation, Durability).

Include i tipi di dati di SQL92 e SQL99 ed inoltre supporta anche la memorizzazione di grandi oggetti come binari, immagini, suoni, video, etc...

Ha un interfaccia nativa per programmare in C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, etc... Esso vanta anche funzionalità come “Multi-Version Concurrency Control (MVCC), point in time recovery, table spaces, asynchronous replication, nested transactions (save points), online/hot backups, a sophisticated query planner/optimizer, and write ahead logging for fault tolerance”.

PostGreSQL è molto apprezzato soprattutto da coloro che lo usano o lo sviluppano, sia utenti che aziende.

Secondo alcuni è il miglior DBMS, infatti ha ricevuto sia il “Linux New Media Award” e per tre volte il “The Linux Journal Editors' Choice Award”.

Per quanto si possa considerare “inferiore” questo prodotto rispetto ai suoi concorrenti commerciali, è da evidenziare come ci siano casi in cui PostGreSQL viene usato in sistemi di produzione che gestiscono più di 4 TB di dati. La sua architettura gli permette di avere limiti molto alti, garantendo prestazioni tali da potersi inserire nella stragrande maggioranza dei casi in cui è richiesto l'ausilio di un DBMS efficiente.

Riportiamo alcuni dati su PostgreSQL:

Grandezza max del DB Illimitato
Grandezza max di una tabella 32 TB
Grandezza max di righe 1.6 TB
Grandezza max di un campo 1 GB
Max righe in una tabella Illimitato
Max colonne in una tabella 250 – 1600
Max indici per tabella Illimitato

La versione 8 di PostGreSQL ha contato circa 1.000.000 di download in 7 mesi, cioè circa 4.760 download al giorno; chiaramente anche il fatto che PostGreSQL è rilasciato con la BSD License BSD (acronimo di Berkeley Software Distribution) ha aiutato questo.

PostGreSQL supporta i seguenti metodi di storage: B-tree, R-tree, Hash e GiST (Generalized Search Tree). Quest'ultimo offre la possibilità di specificare cosa immagazzinare, come memorizzarlo e in che modo ricercare al suo interno includendo i metodi dei B-tree, R-tree ed altri.

Per incrementare le performance, PostGreSQL ha implementato due comandi speciali, *listen* e *notify*. Entrambi permettono una semplice comunicazione peer to peer tra client e server in modo da semplificare il monitoraggio sui dati.

Quanto descritto sopra potrebbe portare a credere che PostGreSQL sia una soluzione semplice e completa. Purtroppo non è così. PostGreSQL è ridotto ai minimi termini, è un sistema dalle grandi potenzialità ma necessita di uno studio molto accurato per poterlo sfruttare al massimo.

A dimostrazione delle difficoltà che si potrebbero incontrare usando PostGreSQL, notiamo che esso è disponibile in versione binaria (compilata) solo per utenti che usano Windows o Gnu/Linux e non sarà possibile installarlo, ufficialmente, utilizzando un eseguibile come ad esempio è possibile per MySQL. Nel sito infatti oltre ai binari per Gnu/Linux e Windows, sono disponibili solo i sorgenti.

Per provare PostGreSQL su un prodotto della Apple, o della Sun Microsystems, bisogna scaricare i sorgenti e compilarli sulla macchina.
PostGreSQL però si preoccupa di segnalare all'utente ulteriori soluzioni per procurarsi una versione compilata di PostGreSQL a seconda del sistema che egli sta usando, infatti come repositoring è disponibile per Debian Apt, Ubuntu Apt, BitRock LAPP Stack, Cygwin, Fink, Mammoth, Portage, Ports (FreeBSD), Yum (Fedora, Red Hat), etc...

Facendo riferimento alle versione utilizzate (8.2 e 8.3) si riportano alcuni dati del DBMS sul sistema in uso:

Versione 8.2-8.3
Sistema: Windows XP (HE x86, 32Bit)
Disco di tipo NTFS

Occupazione in RAM:	30 – 40 Mb
Occupazione su disco:	100 Mb
Tempi d'installazione:	5 - 30 Minuti

Versione 8.3
Sistema: Gnu/Linux (Kubuntu 7.04 AMD-X64, Kernel 2.6.20-17)
Disco di tipo Raiserfs

Occupazione in RAM:	30 – 40 Mb
Occupazione su disco:	75 Mb (pacchetto)
Tempi d'installazione:	5 - 25 Min

2.2.5 FireBird [20]

Firebird SQL è un database management system relazionale (RDBMS) opensource distribuito sotto licenza IPL (Interbase Public License). Supporta numerosi sistemi operativi tra i quali: GNU/Linux, Windows, FreeBSD, Mac OS X e alcuni sistemi Unix. Le principali caratteristiche di questo RDBMS sono l'alto livello di conformità con gli standard SQL, la completa integrazione con molti linguaggi di programmazione e la facile installazione e manutenzione del software. Firebird nasce dal codice sorgente di InterBase 6.0 e quindi ne acquista tutte le sue caratteristiche principali. Interbase, già da quando era un prodotto commerciale e chiuso (prima della versione 6.0), era molto stimato e vantava una numerosa utenza, tra cui grossi nomi come NASA, Nokia, Ericsson, Boeing, etc...
La bassa occupazione di memoria, sia RAM che su disco, la facilità d'installazione, d'utilizzo e di gestione erano (e sono) molto competitivi verso altri RDBMS con uguali ed anche inferiori qualità e funzionalità. InterBase fu sviluppato (nell'ultima fase) dalla Borland International. La prima versione di Firebird (1.0) non è altro che un InterBase riveduto e migliorato dopo che la Borland l'aveva reso opensource, infatti, non rappresenta alcuna rivoluzione rispetto ad InterBase 6.0, ma elimina alcuni seri problemi di sicurezza e qualche bug con l'aggiunta di poche migliorie. Firebird 1.5, pur essendo una versione di transizione verso la nuova 2.0, rappresenta un notevole passo avanti rispetto alla versione precedente. L'intero motore del database server è stato riscritto in C++, mentre la versione precedente era scritta in C. Il vantaggio di questa riscrittura è una maggior chiarezza e leggibilità del codice.
Firebird 2.0 conferma tutte le precedenti funzionalità e qualità e ne aggiunge molte altre a tutti i livelli. Le più interessanti sono quelle relative alle estensioni del linguaggio SQL derivate dall'SQL 200X e spianare la strada per Vulcano che sarà il Firebird della prossima generazione. Confrontato con altri più popolari RDBMS come MySQL e Postgres, Firebird si differenzia sostanzialmente perché è un vero RDBMS con caratteristiche professionali native come l'integrità referenziale dei dati, le stored procedure e i trigger.

È conforme allo SQL-92 Entry Level. Si può usare lo standard ANSI SQL per scrivere delle query portabili tra piattaforme diverse, in più ha delle estensioni che anticipano l'SQL3 come le stored procedure e i trigger e anche altre dell'SQL200X. È possibile accedere contemporaneamente ad un database Firebird da più applicazioni permettendo a più client di lavorare con gli stessi dati in conformità al modello client/server.

Implementa le specifiche A.C.I.D.; cioè il concetto di atomicità, consistenza, isolamento e durabilità. Supporta le transazioni. Una funzionalità indispensabile per garantire la correttezza e il buon esito di operazioni di inserimento, aggiornamento o cancellazione di dati.

Gestisce i lock a livello del singolo record anziché dell'intera pagina. In questo modo gli altri record sono manipolabili liberamente da altri client.

Supporta il protocollo di rete TCP/IP su tutte le piattaforme garantendo l'utilizzo di Firebird come SQL server sia nelle applicazioni client/server sia in quelle web con completa trasparenza.

Utilizza il protocollo XNET usato per accedere ai dati in maniere locale.

Implementa l'ottimizzazione delle query, infatti, quando viene scritto del codice SQL, Firebird, prima di eseguirlo, cerca di ottimizzarlo grazie al suo optimizer interno. Implementa le stored procedure e i trigger. Le prime sono delle applicazioni SQL che vengono memorizzate all'interno del database e vengono eseguite a livello di server. Offrono una grande flessibilità e potenza per svolgere i compiti più impensabili e bilanciare il carico di lavoro tra il client ed il server. I Trigger sono simili alle stored procedure ma non vengono mai eseguiti esplicitamente; svolgono le loro azioni in seguito a modifiche apportate alle tabelle (inserimento, modifica o eliminazione di un dato).

Oltre a queste caratteristiche native di Firebird ce ne sono molte altre realizzate che non elenchiamo; ricordiamo solo che vi sono programmi esterni che permettono a Firebird di avere altre significativi vantaggi ad esempio la ricerca full text, la possibilità di gestire la replica oppure il salvataggio dei database automatica, e il supporto per il clustering.

Facendo riferimento alla versione utilizzata (2.1) si riportano alcuni dati del DBMS sul sistema in uso:

Versione 2.1
Sistema: Windows XP (HE x86, 32Bit)
Disco di tipo NTFS

Occupazione in RAM:	4 – 15 Mb
Occupazione su disco:	30 Mb
Tempi d'installazione:	5 - 20 Minuti

Parte I - *Conclusioni e sviluppi*

Dopo aver analizzato gli aspetti precedenti, si giunge alla conclusione che il software così com'è non è pronto per un'analisi prestazionale comparata su differenti DBMS, oltre a non essere portabile su di essi e non consentire una facile installazione. Si prosegue quindi con l'adattamento e la modifica del codice, che a loro volta devono passare una fase di ulteriore elaborazione, consentita dal linguaggio *Software Requirement Specification*, ma sopratutto dall'UML.
Nella parte successiva (Parte II – Progetto, implementazione e analisi sperimentale) si analizzano, appunto, il progetto con Unified Modelling Language (UML) (Capitolo 3) , l'implementazione (Capitolo 4) ed infine l'analisi sperimentale (Capitolo 5). Seguendo un discorso lineare, che porta inevitabilmente a delle conclusioni (vedi "Conclusioni e sviluppi futuri"), si parte con l'elencazione dei requisiti funzionali rilevati e ad alcuni esempi di casi d'uso. Si passa poi all'implementazione vera e propria, con riferimenti al codice sorgente in appendice (A), ed infine si giunge all'analisi sperimentale, che permette una valutazione oggettiva del lavoro svolto.

Parte II

Progetto, implementazione e analisi sperimentale

Capitolo 3

Il progetto con Unified Modelling Language

La progettazione per l'ampliamento del software è passata attraverso una serie di fasi; per prima cosa sono stati raccolti i requisiti da sviluppare.
Prima di procedere con la scrittura del codice Java e SQL sono state valutate una serie di modifiche da apportare al programma in modo da migliorare alcuni aspetti in fatto di comprensione del codice, gestione degli errori e semplicità di utilizzo.

3.1 Requisiti funzionali

Elenchiamo (in ordine d'importanza decrescente) i requisiti principali che hanno portato alla trasformazione del software:

RF01 - Portabilità sui vari DBMS:

Introduzione:	Il sistema deve interfacciarsi con diversi DBMS, pertanto deve essere possibile effettuare questa operazione con trasparenza assoluta
Input:	L'utente seleziona il database voluto (dal menu relativo RF05 o tramite modifica del file di configurazione prima dell'avvio del software)
Processing:	Verifica del DBMS (utente, password, creazione tabelle, query, etc..); selezione delle impostazioni tramite RF02
Output:	Messaggio di conferma o errore

RF02 - Selezione delle impostazioni:

Introduzione:	Il sistema deve essere in grado di selezionare tutte le impostazioni relative al DBMS selezionato.
Input:	Informazioni sul DBMS selezionato correttamente
Processing:	Caricamento dei driver JDBC relativi; Impostazione delle variabili d'ambiente e dei parametri di selezione delle query SQL;
Output:	Il programma è pronto all'uso sul DBMS scelto

RF03 – Importazione dei dati Stemmer:

Introduzione:	Per poter utilizzare la funzione di stemming è necessario importare sul DBMS in uso i dati contenuti in quattro tabelle, salvate come files .CSV
Input:	L'utente seleziona l'importazione della/e tabella/e voluta/e (dal menu relativo RF05 o da linea di comando)
Processing:	Caricamento dei dati dai files .CSV selezionati al DBMS scelto; Verifica dei dati inseriti tramite RF
Output:	Messaggio di attesa

RF04 – Verifica dei dati Stemmer:

Introduzione:	Per uniformare i risultati dei differenti DBMS è necessario che partano tutti dai medesimi dati, pertanto serve una funzione che li verifichi
Input:	Selezione inserimento dei dati (dal menu apposito RF05 o da linea di comando); selezione della funzione Stemming
Processing:	Verifica dei dati inseriti nel DBMS
Output:	Messaggio di errore o conferma; selezione o meno dello Stemmer

RF05 – Creazione dell'interfaccia grafica:

Introduzione:	Per permettere un facile e trasparente impiego delle funzionalità descritte nei vari Requisiti Funzionali serve un'interfaccia grafica (GUI)
Input:	Interfaccia del software preesistente
Processing:	Sviluppo dell'interfaccia con menu di scelta e messaggi a video
Output:	Interfaccia grafica utente (GUI)

RF06 – Settaggio di User e Password:

Introduzione:	Tutti i DBMS utilizzati necessitano di autenticazione tramite Username e Password, pertanto in RF05 vi è la possibilità di cambiarli e settarli.
Input:	Selezione del settaggio di User e Password da RF05
Processing:	Impostazione delle variabili d'ambiente relative; utilizzo di RF01 per la verifica dei parametri inseriti
Output:	Messaggio di avvenuta impostazione

RF07 – Salvataggio dei parametri nel file di configurazione:

Introduzione:	Per evitare ogni volta la selezione del DBMS preferito si opta per il salvataggio dei dati nel file di configurazione
Input:	Selezione del salvataggio del DBMS o di user e password da RF05
Processing:	Salvataggio su file di configurazione dei dati selezionanti
Output:	Conferma dell'avvenuto salvataggio o meno

RF08 – Caricamento del file di configurazione:

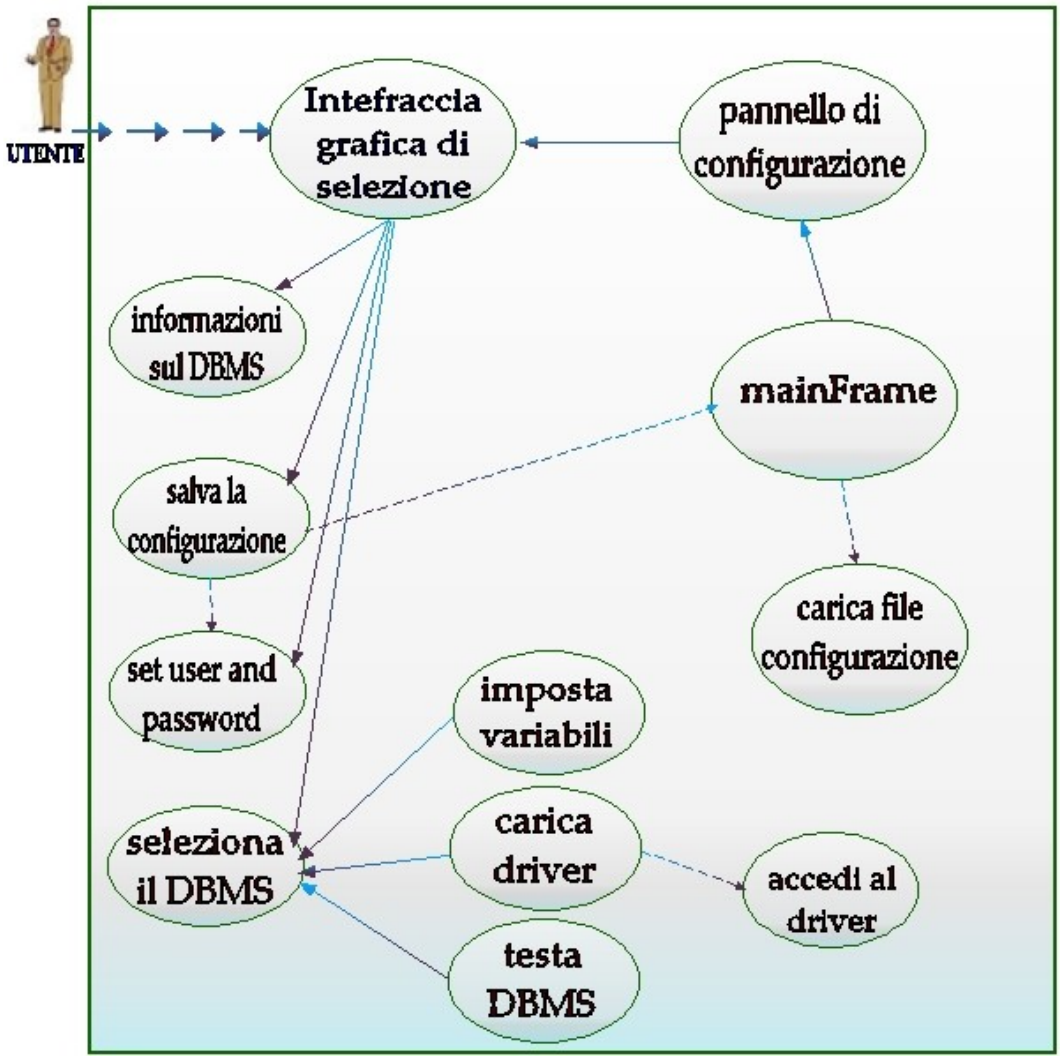
Introduzione:	Si desidera il caricamento delle impostazioni scelte all'avvio del software nel modo più trasparente possibile
Input:	Avvio sel software
Processing:	Lettura del file di configurazione (se esiste); caricamento dei parametri e settaggio delle variabili d'ambiente
Output:	Eventuale messaggio d'errore

RF09 – Informazioni sui vari DBMS:

Introduzione:	Nel caso in cui l'utente non sia esperto di DMBS si diano informazioni
Input:	Selezione di richiesta informazioni tramite RF05
Processing:	Seleziona i dati relativi al DBMS “richiesto”
Output:	Messaggio d'informazione.

3.2 Scenario: selezione di opzioni sui DBMS

Diamo un esempio di scenario possibile, ovvero quello nel caso in cui l'utente del software selezioni un'opzione sul DBMS da utilizzare:



(Fig. 3.2-a – Scenario: selezione di opzioni sui DBMS)

3.3 Casi d'uso

Seleziona un nuovo DBMS
Obiettivo: La funzione si propone la gestione della selezione di un nuovo DBMS.
Attore primario; ambito: Utente generico; software in questione, scenario 3.2.
Evento scatenante: Selezione di un nuovo DBMS dal menu.
Precondizioni: Lo user e la password devono essere impostati correttamente.
Sequenza: Selezione, test del DBMS, responso, impostazione o meno, messaggio.
*Post condizione di successo:*Il DBMS in uso funziona correttamente.
Post condizione di fallimento: Il DBMS in uso non funziona correttamente.

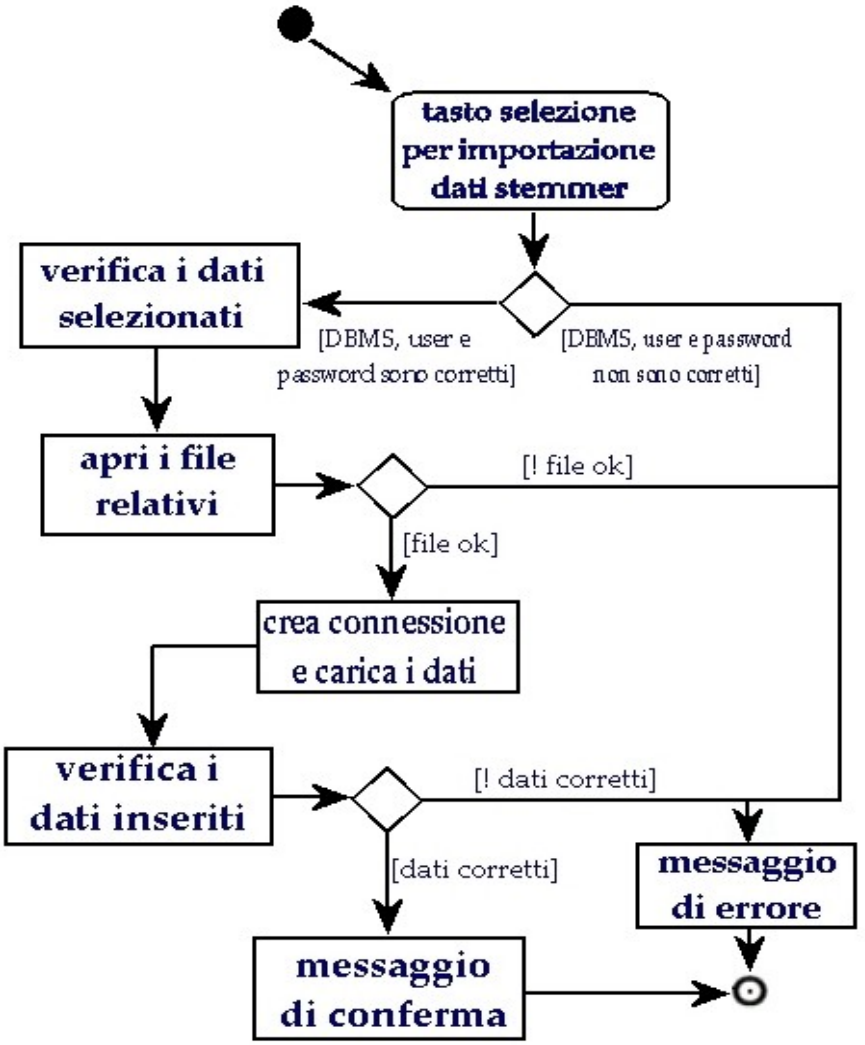
Importa dati Stemmer
Obiettivo: La funzione si propone l'importazione dei dati Stemmer.
Attore primario; ambito: Utente generico; software in questione, scenario 3.2.
*Evento scatenante:*Selezione dell'importazione di dati stemmer dal menu.
Precondizioni: Lo user e la password e il DBMS devono essere impostati correttamente.
Sequenza: Selezione dati, messaggio di attesa, importazione dati, verifica dati.
*Post condizione di successo:*I dati sono stati importati correttamente.
Post condizione di fallimento: I dati non sono stati importati o non sono stati importati correttamente.

Seleziona user e password
Obiettivo: La funzione si propone si settare user e password per la sessione.
Attore primario; ambito: Utente generico; software in questione, scenario 3.2.
Evento scatenante: Selezione del settaggio di user e password dal menu.
Precondizioni: Il DBMS funziona correttamente o non è impostato.
Sequenza: Scrittura di user e password, pressione del bottone, settaggio delle variabili, controllo del DBMS.
Post condizione di successo: User e password sono corretti o il DBMS non è impostato.
Post condizione di fallimento: User e password non sono corretti e il DBMS è impostato.

Salva file di configurazione
Obiettivo: La funzione si propone di salvare i parametri in uso sul file di configurazione.
Attore primario; ambito: Utente generico; software in questione, scenario 3.2.
Evento scatenante: Selezione del salvataggio di parametri sul file di configurazione.
Precondizioni: Il file di configurazione esiste e può essere scritto.
Sequenza: Selezione del salvataggio, salvataggio impostazioni, messaggio.
Post condizione di successo: Il file viene scritto correttamente.
Post condizione di fallimento: Il file non viene scritto correttamente o non esiste.

3.4 Activity Diagram

Questo activity diagram riguarda l'inserimento di una o più tabelle all'interno del DBMS selezionato in precedenza.



(Fig. 3.4-a – Activity, importa dati)

Capitolo 4

Implementazione

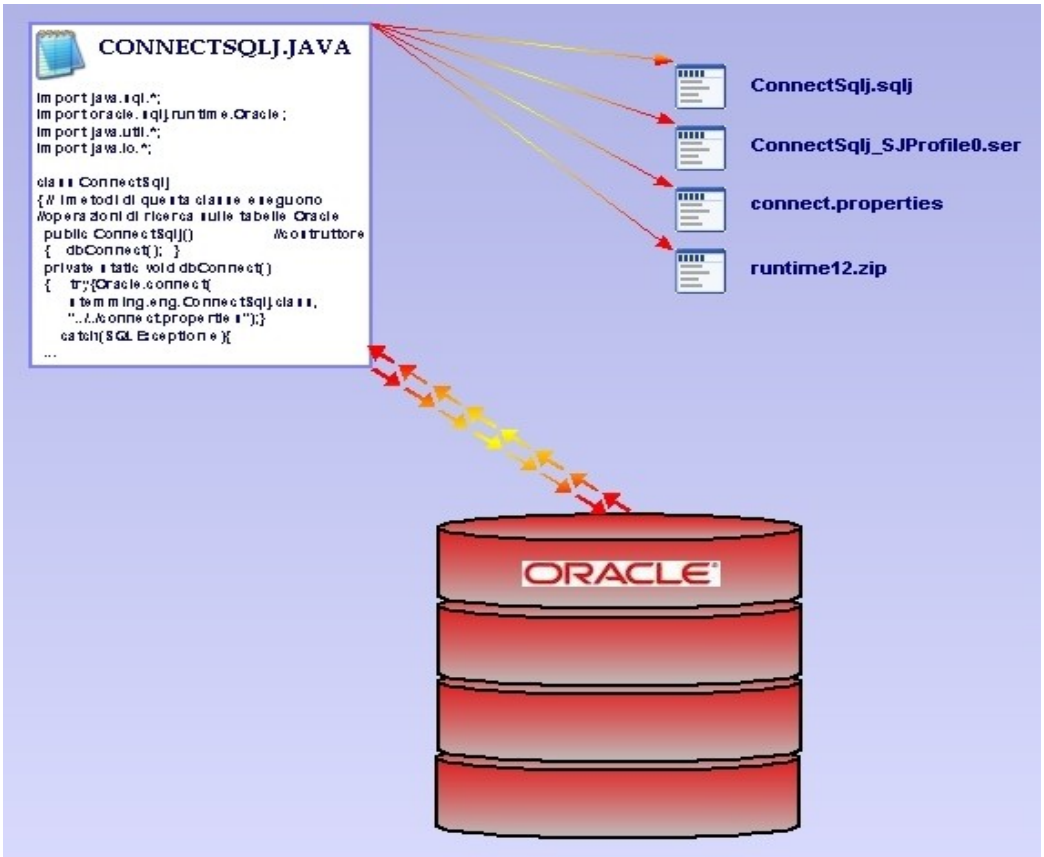
In questo capitolo si vuole analizzare in dettaglio la parte di progetto sviluppata, partendo dalle modifiche di base, per rendere il software portabile, concludendo con la creazione dell'interfaccia grafica che ne permette un semplice utilizzo.

4.1 Da SQLJ a JDBC

Il primo passo per poter creare una portabilità estesa a numerosi DBMS è stato quello di tradurre il codice SQLJ in codice JDBC.
Come già descritto in dettaglio nei punti 2.1.1 e 2.1.2 di questo documento, JDBC si presta in modo più efficiente di SQLJ per l'utilizzo dell'applicazione su diverse piattaforme.
Presentiamo ora, utilizzando riferimenti al codice Java in appendice, il processo seguito e il risultato finale.

4.1.1 La classe originale

La classe originale *(*)* presenta alcuni aspetti statici, tra cui l'utilizzo di file esterni modificabili solo in relazione al DBMS Oracle 7 e 8, che non soddisfano i criteri di portabilità richiesti per questo progetto, pertanto la modifica di questo sistema risulta inevitabile. Nell'immagine sottostante un quadro della situazione.



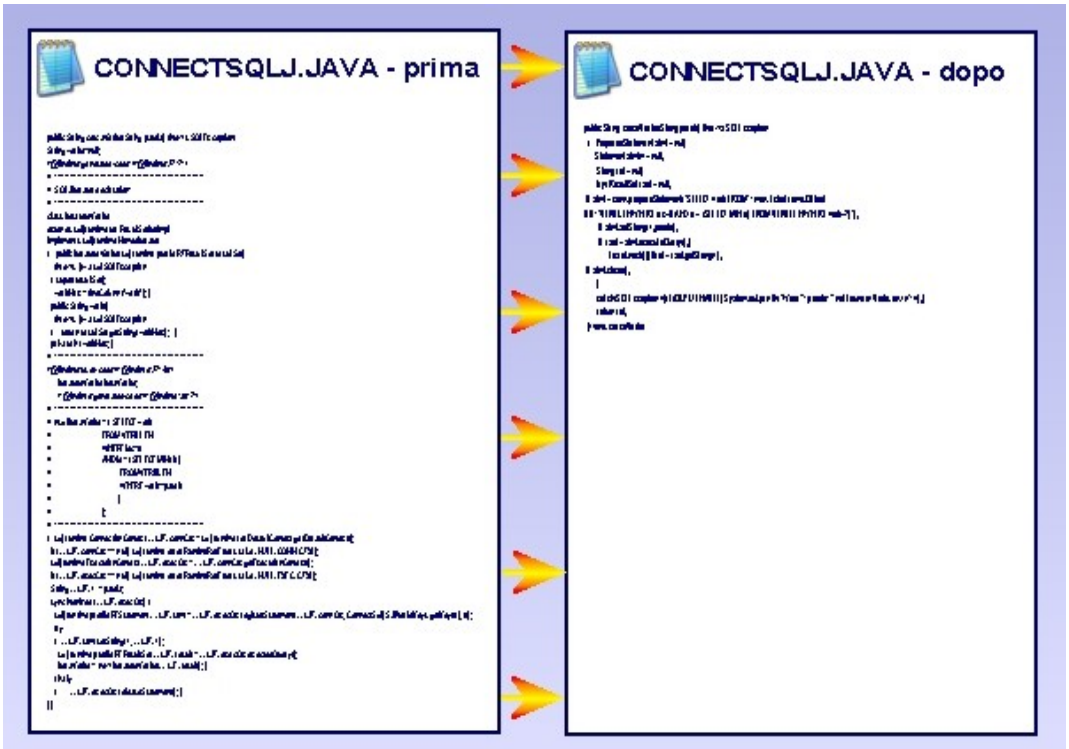
(Fig. 4.1.1-a – La classe connectSQLJ originale)

Si noti in particolare come l'importazione per il funzionamento dell'SQLJ con Oracle *"import oracle.sqlj.runtime.Oracle;"* nella seconda riga di codice non permetta l'utilizzo di altre piattaforme e porti ad errore certo nel qual caso manchino i file dei driver o non sia possibile una connessione al DBMS.

(*) Sviluppata originariamente da F. Gavioli in [4], modificata poi da R. Martoglia in [3], disponibile in appendice A6, è contenuta nel file connectsqlj.java.

4.1.2 Il processo di traduzione

Una volta acquisita dimestichezza con JDBC e avendo a disposizione le query in formato SQL, il processo di traduzione diviene sostanzialmente una riscrittura del codice per il prelievo delle informazioni dal database.
Il primo passo compiuto è stato quello di tradurre la classe per il solo Oracle, eliminando tutte le funzioni SQLJ, per verificare performance e comportamento.
Dopo aver fatto ciò ed averne constatato il perfetto funzionamento, si è proceduto all'introduzione di una nuova classe che permettesse la scelta delle query, delle connessioni, dei driver e desse la possibilità di testare il DBMS scelto.
Al termine di ciò il software risulta utilizzabile su diversi DBMS semplicemente richiedendo alcuni parametri alla nuova classe introdotta.



(Fig. 4.1.2-a – connectSQLJ.java prima e dopo, funzione cercaVerbo)

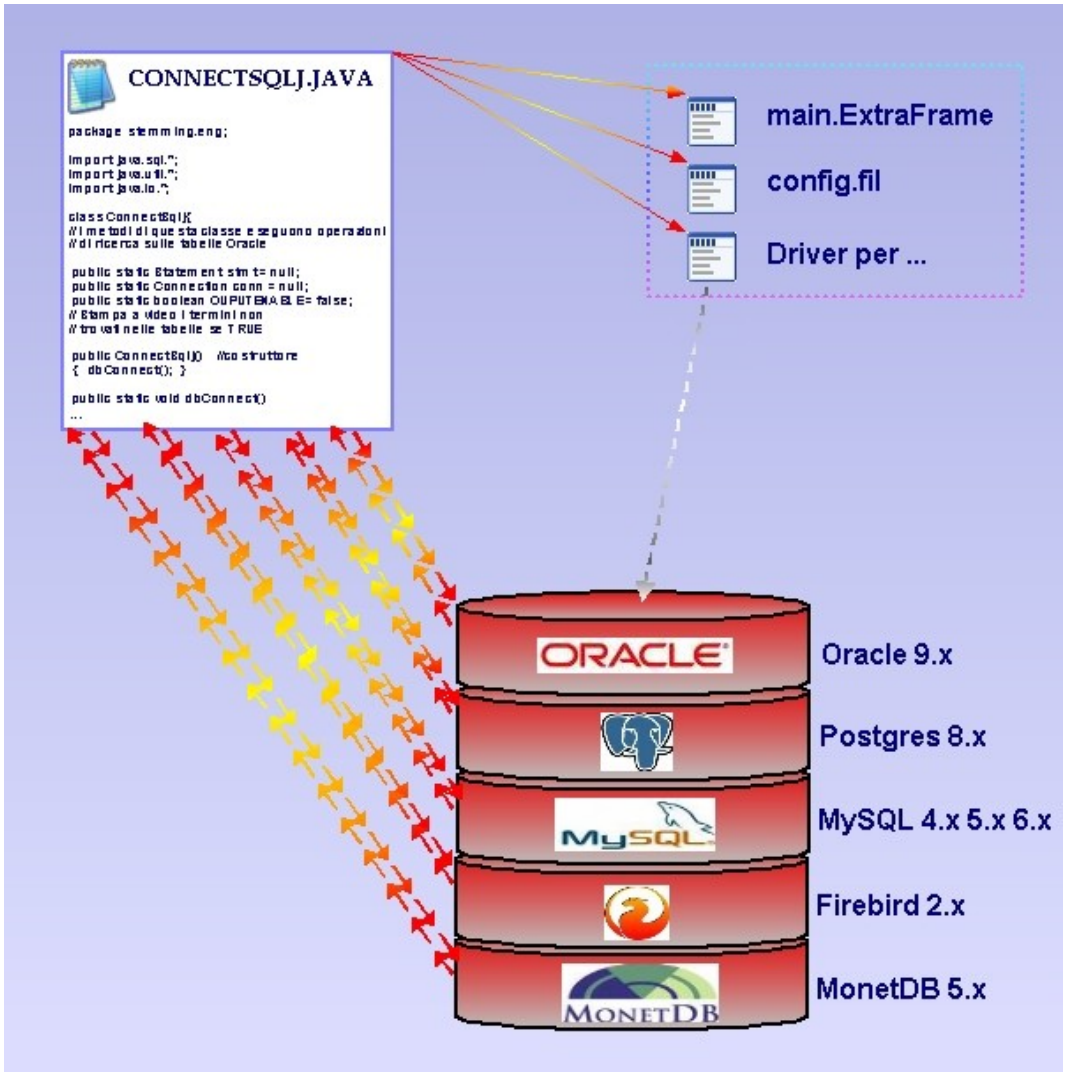
Sottolineiamo che nell'esempio della figura precendente (Fig. 4.1.2-a) la lunghezza di codice effettivo visibile non è quella reale, anche se le righe di codice per la connessione risultano inferiori per il caso JDBC (dopo) al contrario di quanto pronosticato dalla stragrande maggioranza delle guide ufficiali e non, infatti, la versione finale, a parità di comprensibilità del codice dovuta a commenti, risulta nel totale 8121 byte al posto dei 13091 di quella per SQLJ, considerando anche un'aggiunta di più di 40 righe per problemi relativi ad un singolo "nuovo" DBMS *(*)*.

(*) Vedi paragrafo 4.2.1 (MonetDB) per ulteriori dettagli

4.1.3 Il risultato finale

Il risultato finale di questa traduzione, disponibile, sotto forma di codice sorgente in appendice A6, è visibile nell'immagine sottostante. Segnaliamo che in appendice (A6) il codice sorgente è diviso in due colonne per evidenziare i cambiamenti tra prima e dopo l'intervento; i diversi colori e i diversi attributi del carattere evidenziano diverse proprietà del codice, quali commenti originali, commenti aggiunti, codice cambiato, etc...

La nuova classe, che comunque conserva lo stesso nome, si interfaccia ai vari DBMS richiedendo i parametri dalla classe ExtraFrame del package main, che a sua volta viene influenzata dal file di configurazione all'avvio e, se richiesto dall'utente, tramite un menu di configurazione dal pannello *"config"* durante l'esecuzione del programma.



(Fig. 4.1.3-a – La classe connectSQLJ finale)

Mettendo a confronto le figure 4.1.1-a e 4.1.3-a si notano facilmente le differenze tra le due versioni, infatti, oltre all'evidente numero di DBMS supportati dalla seconda rispetto alla prima, cambia radicalmente il modo di configurazione.

4.2 La portabilità

Nel campo dell'informatica, la portabilità di un componente software è un adattamento o una modifica del componente, volto a consentirne l'uso in un ambiente di esecuzione diverso da quello originale.

L'operazione di porting, cioè la creazione di un port, è solitamente richiesta a causa delle differenze tra le CPU, dalle diverse interfacce, dei sistemi operativi, dalla diversità dell'hardware, del linguaggio di programmazione sull'ambiente target, etc... .

Considerando già la portabilità del linguaggio Java utilizzato per l'applicazione, nel nostro caso la portabilità riguarda quindi i diversi DBMS che differiscono nella sintassi SQL e nei parametri di connessione.

Nei sotto paragrafi successivi descriviamo in modo dettagliato il processo che ha portato alla scrittura di classi per la portabilità del software.

4.2.1 Considerazioni sui vari DBMS

Come già descritto nell'introduzione a questo capitolo, il problema principale da affrontare per la portabilità del software su diversi DBMS riguarda le differenze, a volte piccole, ma sostanziali, del codice SQL interpretato dai diversi DBMS, i parametri di connessione ed i linguaggi da essi internamente supportati.

Nei punti successivi elenchiamo le cinque macro-modifiche apportate al software, tutte opportunamente e dettagliatamente descritte tramite codice sorgente in appendice A7 intitolata “Le modifiche” divisa per punti equivalenti ai successivi.

1. *Java Inside (JI) e Java Outside (JO)*

   Nella versione iniziale dell'applicazione il codice era scritto per funzionare esclusivamente sulle versioni 7.x e 8.x di Oracle [15][16], infatti, alcune classi Java del progetto, quali *DBUtility* e *Distance* del package *simSearch*, dovevano essere incluse nel DBMS per permettere il funzionamento del programma *(*)*. La prima modifica apportata riguarda appunto questo problema, infatti, quasi nessun altro DBMS oltre Oracle [15][16] supporta questa funzionalità.

   Da query SQL che includono, all'interno della stringa utilizzata, i nomi delle funzioni da richiamare all'interno del DBMS, si passa all'utilizzo di codice che chiama funzioni direttamente da Java e inserisce nella stringa di connessione per la query SQL i valori restituti da tali funzioni. Questo permette un'indipendenza anche dal punto di vista della versione di Java (JDK-JRE) che non è più legata al DBMS come nel caso iniziale. Infatti, se prima le classi da includere dovevano per forza essere compilate con la versione presente nel DBMS (e in caso questo non avvenisse, il software non funzionava), ora la compilazione è legata a quella di tutto il software, e dipende solo da Java la compatibilità a posteriori, fino ad ora “testata funzionante” dalla versione 1.5.0_01 passando per la 1.5.0_12, la 1.6.0_01, 1.6.0_03, 1.6.0_04, 1.6.0_05 e concludendo con la 1.6.0_07, oltre alla precedente 1.3.0_01 di Oracle.

(*) Riferimenti a “Oracle Java Inside” od “Oracle JI”

2. *Le connessioni*

   Un altro problema per la portabilità riguarda le connessioni JDBC sui diversi DBMS, infatti, ognuno è supportato solo dal suo driver, salvato in un file, da caricare nella fase di connessione, insieme alla sua stringa di connessione.

   Il package SelAndFillDB nasce principalmente a questo scopo. Esso è utilizzato fin dall'inizio dell'esecuzione del programma e, dipendentemente dalla classe richiamata che spesso richiede in ingresso il nome del DBMS sotto forma di stringa, provvede alla restituzione di tutti i parametri necessari al corretto funzionamento del DBMS, quali inserimento di dati per lo stemmer (Expdmp.java – Appendice A8), stringhe per driver di connessione (Appendice A7, dettaglio su "Le connessioni"), etc...

3. *L'SQL*

   Se per i diversi sistemi operativi il problema della portabilità nasce dal codice dell'applicazione, per i DBMS il problema nasce dalle istruzioni SQL, infatti, quasi nessun DBMS è conforme al 100% con gli standard classici, per esempio: Oracle utilizza variabili VARCHAR2, NUMBER(#); non supporta i "punti" nei nomi delle colonne se selezionate (SELECT p2.frase ... restituisce una colonna di nome "frase"). MonetDB richiede una sintassi diversa per la funzione ROUND, usa variabili di tipo INT e VARCHAR, supporta i "punti" nei nomi delle colonne. MySQL utilizza variabili di tipo VARCHAR e INT(#), richiede attenzioni particolari nei costrutti GROUP BY e ORDER BY. PostgreSQL necessita della parola chiave AS per dare nomi alle colonne e non supporta i "punti" come Oracle, etc.., etc...

4. *MonetDB e gli apici*

   Un caso particolare lo riserviamo al DBMS MonetDB [19] che, al momento di richieste tramite *PreparedStatement,* fornisce alcune problematiche se la stringa passata contiene apici singoli *(*).*

5. *Firebird e i ResultSet*

   Il secondo caso particolare lo riserviamo invece a FireBird [18] che fornisce problemi relativi all'apertura simultanea di *ResultSet.*

(*) Per apice singolo s'intende il carattere ' che in codice ASCII è il numero 39.

4.3 Installazione e interfaccia utente

Come già annunciato in precedenza, il programma necessita l'appoggio di almeno un DBMS. Originariamente questo DBMS era la verione 7.x o 8.x di Oracle [15][16], ma dopo l'intervento di portabilità i DBMS sono diventati cinque con una scelta di versioni superiore alle 7 e sottoversioni superiore alle 50 e, appunto per questo, è stata prevista una nuova e semplice interfaccia utente che permetta di importare i dati necessari al software in tutta tranquillità.

Si deve notare però che pur agevolando notevolmente la procedura di configurazione con la scelta di user, password, nomi tabelle, etc..., questa interfaccia non può installare il DBMS e creare il DataBase vero e proprio sullo stesso. Questo limite è intrinseco e invalicabile e, appunto per questo, riportiamo le note d'installazione per i vari DBMS (Cap. 4.3.1).

4.3.1 Il processo di installazione

La versione originale JI (installazione su Oracle [15][16])

I passi per la configurazione di questo DBMS risultano i più lunghi e complessi di tutti quelli riportati, oltre al fatto che potrebbero cambiare da versione a versione.

* INSTALLARE ORACLE e CREARE DB TRANS1

* CREARE UTENTE TRANS1\TRANS1 e assegnarli tutti i privilegi!

* IMPORTARE DATI STEMMER
(da prompt cmd digitare "imp" e seguire il procedimento)

* C:\oracle\ora92\network\admin\sqlnet.ora
* commentare riga: SQLNET.AUTHENTICATION_SERVICES= (NTS)

* Compilare le 2 classi da importare con la vesrione JDK di oracle e
* CARICARE IL CODICE COMPILATO IN ORACLE
* NB: l'area di lavoro deve essere pulita da ogni altra classe Java con lo stesso nome
*loadjava -user trans1/trans1 -resolve DBUtility.class
*loadjava -user trans1/trans1 -resolve Distance.class

* CREARE LE SEGUENTI FUNZIONI IN ORACLE

```
CREATE OR REPLACE FUNCTION wordLen (frase VARCHAR2)
  RETURN NUMBER
AS LANGUAGE JAVA
NAME 'simSearch.DBUtility.wordLen (java.lang.String)
  return int';
/

CREATE OR REPLACE FUNCTION wordSubString (frase VARCHAR2, da NUMBER, a NUMBER)
  RETURN VARCHAR2
AS LANGUAGE JAVA
NAME 'simSearch.DBUtility.wordSubString (java.lang.String, int, int)
  return java.lang.String';
/

CREATE OR REPLACE FUNCTION transPos (frase VARCHAR2, n NUMBER)
  RETURN NUMBER
AS LANGUAGE JAVA
NAME 'simSearch.DBUtility.transPos (java.lang.String, int)
  return int';
/

CREATE OR REPLACE FUNCTION wordEditDistance (stringa1 VARCHAR2, stringa2 VARCHAR2)
  RETURN NUMBER
AS LANGUAGE JAVA
NAME 'simSearch.Distance.wordEditDistance (java.lang.String, java.lang.String)
  return int';
/

CREATE OR REPLACE FUNCTION wordEditDistanceDiag (stringa1 VARCHAR2, stringa2 VARCHAR2, maxd NUMBER)
  RETURN NUMBER
AS LANGUAGE JAVA
NAME 'simSearch.Distance.wordEditDistanceDiag (java.lang.String, java.lang.String, double)
  return int';
/

CREATE OR REPLACE FUNCTION wordEditDistanceSubCheck (s1 VARCHAR2, c1 NUMBER, s2 VARCHAR2, c2 NUMBER, maxd
NUMBER)
  RETURN NUMBER
AS LANGUAGE JAVA
NAME 'simSearch.Distance.wordEditDistanceSubCheck (java.lang.String, int, java.lang.String, int, double)
  return int';
/
```

La versione modificata JO (installazione su Oracle [15][16])

* INSTALLARE ORACLE e CREARE DB TRANS1

* CREARE UN UTENTE e assegnarli tutti i privilegi sul DB trans1

* IMPORTARE DATI STEMMER con (S1) o (S2):
* (S1) da GUI o linea di comando (più comodo di S2)
* (S2) da prompt cmd digitare "imp" e seguire il procedimento (più veloce di S1)

* C:\oracle\ora92\network\admin\sqlnet.ora
* commentare riga: SQLNET.AUTHENTICATION_SERVICES= (NTS)

* SE S'INTENDE UTILIZZARE LA VERSIONE JavaInside PROSEGUIRE, ALTRIMENTI FERMARSI QUI.
* Compilare le 2 classi da importare con la vesrione JDK di oracle e
* CARICARE IL CODICE COMPILATO IN ORACLE
* NB: l'area di lavoro deve essere pulita da ogni altra classe Java con lo stesso nome
*loadjava -user trans1/trans1 -resolve DBUtility.class
*loadjava -user trans1/trans1 -resolve Distance.class

* CREARE LE SEGUENTI FUNZIONI IN ORACLE

```
CREATE OR REPLACE FUNCTION wordLen (frase VARCHAR2)
  RETURN NUMBER
AS LANGUAGE JAVA
NAME 'simSearch.DBUtility.wordLen (java.lang.String)
  return int';
/

CREATE OR REPLACE FUNCTION wordSubString (frase VARCHAR2, da NUMBER, a NUMBER)
  RETURN VARCHAR2
AS LANGUAGE JAVA
NAME 'simSearch.DBUtility.wordSubString (java.lang.String, int, int)
  return java.lang.String';
/

CREATE OR REPLACE FUNCTION transPos (frase VARCHAR2, n NUMBER)
  RETURN NUMBER
AS LANGUAGE JAVA
NAME 'simSearch.DBUtility.transPos (java.lang.String, int)
  return int';
/

CREATE OR REPLACE FUNCTION wordEditDistance (stringa1 VARCHAR2, stringa2 VARCHAR2)
  RETURN NUMBER
AS LANGUAGE JAVA
NAME 'simSearch.Distance.wordEditDistance (java.lang.String, java.lang.String)
  return int';
/

CREATE OR REPLACE FUNCTION wordEditDistanceDiag (stringa1 VARCHAR2, stringa2 VARCHAR2, maxd NUMBER)
  RETURN NUMBER
AS LANGUAGE JAVA
NAME 'simSearch.Distance.wordEditDistanceDiag (java.lang.String, java.lang.String, double)
  return int';
/

CREATE OR REPLACE FUNCTION wordEditDistanceSubCheck (s1 VARCHAR2, c1 NUMBER, s2 VARCHAR2, c2 NUMBER, maxd
NUMBER)
  RETURN NUMBER
AS LANGUAGE JAVA
NAME 'simSearch.Distance.wordEditDistanceSubCheck (java.lang.String, int, java.lang.String, int, double)
  return int';
/
```

La versione modificata (installazione su MySQL [17])

* INSTALLARE MYSQL e CREARE DB TRANS1

* CREARE UN UTENTE e assegnarli tutti i privilegi sul DB trans1

* IMPORTARE DATI STEMMER (da GUI o linea di comando)

La versione modificata (installazione su PostgreSQL [20])

* INSTALLARE POSTGRES

* Il primo utente (di win) a scelta il secondo (admin del DBMS) a scelta

* CREARE un nuovo schema "trans1" (volendo sul DB “trans1”)

* CREARE nuovo UTENTE con tutti i privilegi sullo schema “trans1”

* IMPORTARE DATI STEMMER (da GUI o linea di comando)

La versione modificata (installazione su MonetDB [19])

* INSTALLARE MONETDB

* APRIRE una shell (O prompt dei comandi) sulla directory di installazione e digitare:

shell> MSQLserver --dbname=trans1

NON CHIUDERE LA SHELL!!!
APRIRE UN'ALTRA SHELL come quella di prima e digitare:

shell> mclient -lsql

```
sql>CREATE USER "trans1" WITH PASSWORD 'trans1' NAME 'trans1' SCHEMA "sys";
sql>CREATE SCHEMA "trans1" AUTHORIZATION "trans1";
sql>ALTER USER "trans1" SET SCHEMA "trans1";
sql>\q
```

* IMPORTARE DATI STEMMER (da GUI o linea di comando)

(NB: al termine di ogni riga di comando va digitato “Enter” o “Invio”; lo user e la password possono essere modificati a piacere, mentre lo schema no)

La versione modificata (installazione su FireBird [18])

* INSTALLARE Firebird

* Creare un utente e una password a scelta es. user "trans1" password "trans1"

* Per cambiare la password di SYSDBA (sicurezza) digitare:
* PATHFirebird*bin\gsec -user sysdba -pass masterkey -mo sysdba -pw NUOVA*

* PATHFirebird*bin\gsec -user sysdba -pass NUOVA -add trans1 -pw trans1*

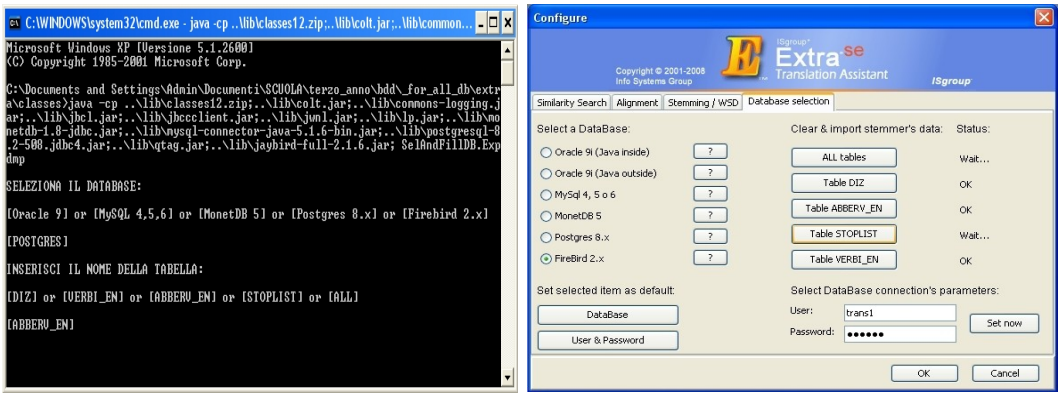
* CREARE un nuovo DB "trans1": PATHFirebird*bin\isql*
* *SQL> create database 'PATHscelto\trans1.fdb' page_size 8192;*
* *SQL> connect trans1 user trans1 password trans1*

* Per connettersi usando "connect trans1" bisogna inserire il percorso
* del file ('PATHscelto\trans1.fdb') nel file PATHFirebird\aliases.conf

* IMPORTARE DATI STEMMER (da GUI o linea di comando)

Le linee guida per l'installazione dei vari DBMS appena descritte sono puramente indicative, anche se perfettamente funzionanti sulle versioni scelte. Per informazioni complete si consiglia l'utilizzo dei manuali del DBMS realtivo.

L'importazione dei dati stemmer diviene necessaria quando il software è utilizzato nella modalità con stemmer acceso. Nella prima versione del programma l'importazione di tali dati era lasciata completamente all'utente, che doveva anche provvedere alla creazione manuale degli indici e delle tabelle. Nella nuova versione, data l'impossibilità di caricare gli stessi dati dump tra un DBMS e l'altro si è provveduto ad una interfaccia (si grafica che da shell) per l'importazione di tali dati. Essa si basa sostanzialmente sulla classe (main) ExpDmp del package SelAndFillDB *(*)*.

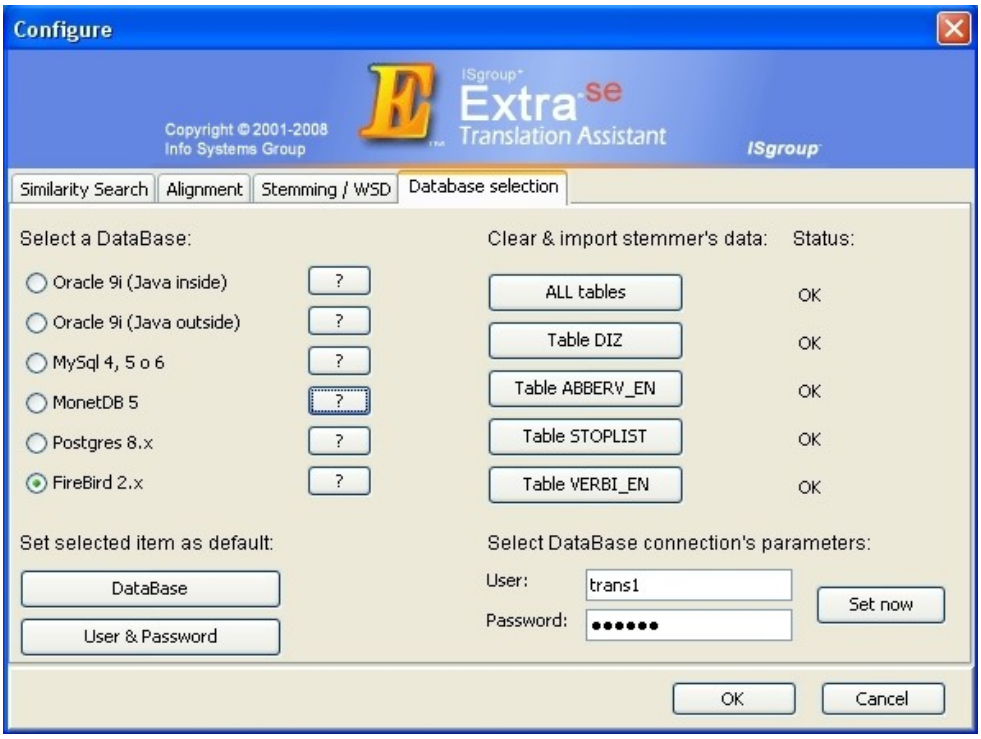


(Fig. 4.3.1-a – ExpDmp, Shell vs GUI)

(*) Codice sorgente disponibile in appendice A8

4.3.2 La nuova interfaccia utente

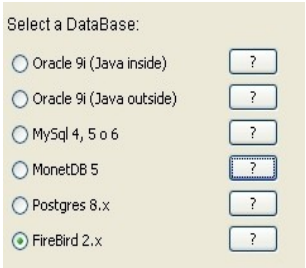
La nuova interfaccia utente, inclusa nel package *main*, nella classe preesistente *ExtraFrame_ConfigureDialog* è stata creata per semplificare notevolmente la procedura di settaggio di differenti DBMS ed agevolare l'importazione dei dati stemmer, prima disponibile solo per Oracle [15][16].



(Fig. 4.3.2-a – Database Selection)

Come si può vedere in figura (Fig. 4.3.2-a), il pannello è composto da quattro macro-aree: in alto a sinistra *"Select a DataBase"*, in basso a sininstra *"Set selected item as default"*, in basso a destra *"Select DataBase connection's parameters"* ed infine in alto a destra *"Clear & import stemmer's data"*.

L'area *"Select a DataBase"* (Fig. 4.3.2-b) si utilizza per selezionare il DBMS voluto, cliccando sull'apposito ed esclusivo bottone a sinistra. Per informazioni empiriche sul DBMS si rimanda l'utente al bottone "?" a fianco del nome e del bottone di sinistra.
(Fig. 4.3.2-f, vedi particolare dell'area).

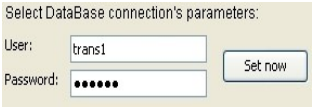


(Fig. 4.3.2-b – selDB area)

L'area *"Set selected item as default"* (Fig. 4.3.2-c) si utilizza per impostare il DBMS o lo user e la password in uso come predefiniti, infatti, una volta premuto il bottone relativo, il file di configurazione viene aggiornato e una volta riavviato il programma caricato all'avvio. (Fig. 4.3.2-f, vedi particolare dell'area).

(Fig. 4.3.2-c – default area)

L'area *"Select DataBase connection's parameters"* (Fig. 4.3.2-d) si utilizza per impostare user e password scelti a fianco del bottone "Set now" nella sessione in corso. (Fig. 4.3.2-f, vedi particolare dell'area).



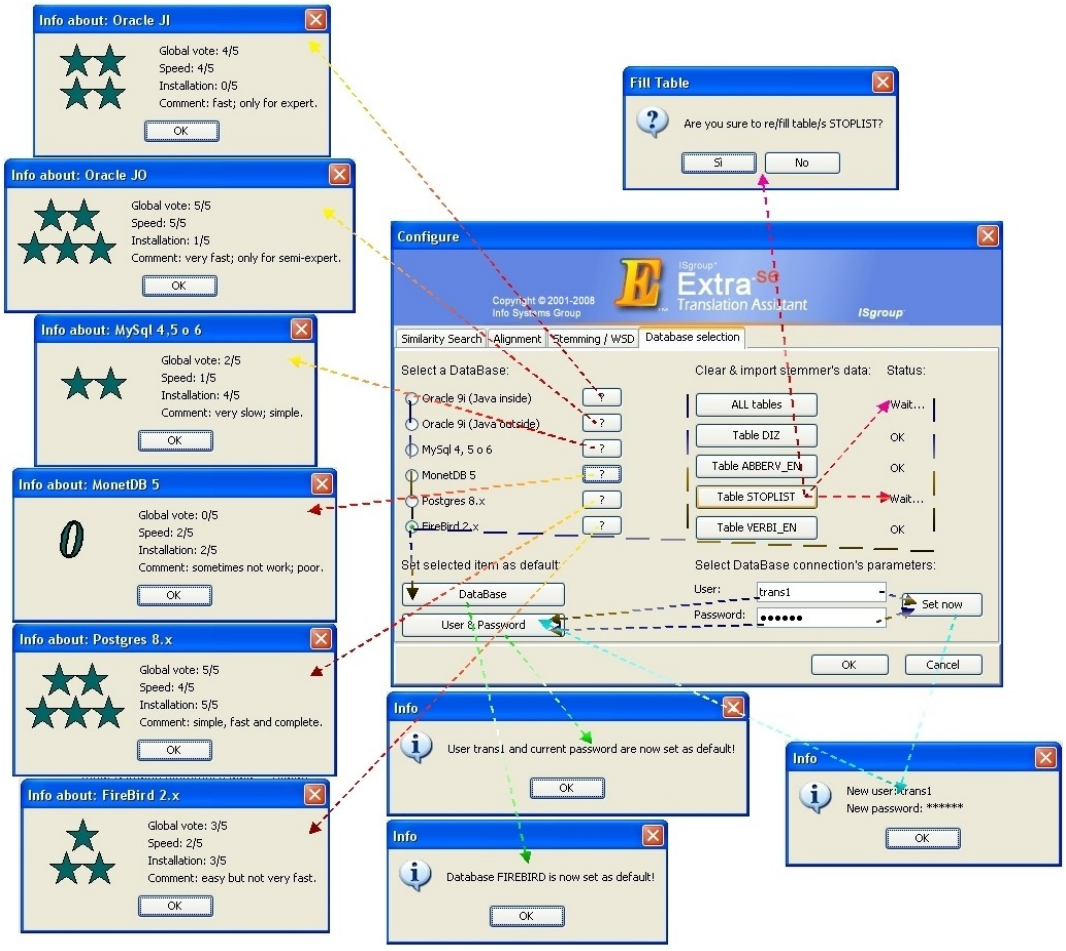
(Fig. 4.3.2-d – param area)

L'area *"Clear & import stemmer's data"* (Fig. 4.3.2-e) si utilizza, come già dettagliatamente spiegato nei capitoli precedenti, per importare i dati stemmer all'interno del DBMS in uso nella sessione.
Lo status a fianco cambia dinamicamente secondo criteri di correttezza, di attesa od errore.
(Fig. 4.3.2-f, vedi particolare dell'area).

(Fig. 4.3.2-e – stem area)

Nella figura della pagina successiva (4.3.2-f - Database Selection dependences) , riportata come dettaglio di particolari per le quattro aree, evidenziamo che tutti i colori diversi delle frecce tratteggiate evidenziano le dipendenze delle diverse aree, mentre le linee e le frecce tratteggiate in ocra e blu tra diverse aree individuano le dipendenze tra esse.

Il codice sorgente rappresentato da questa immagine (4.3.2-f) è disponibile in appendice A9. Si evidenzia che esso non è stato scritto soltanto per questa parte, ma per tutto il pannello di configurazione del software. Le parti che individuano questa immagine (4.3.2-f) sono comunque riconoscibili dai commenti, mentre quelle ininfluenti sono rimpiazzate da simboli quali "***(...)***".



(Fig. 4.3.2-f – Database Selection dependences)

Capitolo 5

Analisi sperimentale

L'analisi sperimentale del software, intesa come scelta dei dataset, dei parametri e dei DBMS su cui effettuare prove sperimentali, è il passo conclusivo e di maggior rilevanza di questo progetto.
Partendo da una base di test effettuati in precedenza e ripercorrendo, ma solo nei primi passi, questa strada, si ci è accorti che i risultati ottenuti dopo le modifiche al programma, oltre alla loro correttezza nella quasi totalità dei casi salvo piccole incongruenze dovute a fattori pressoché irrilevanti al nostro scopo, erano in molti casi migliorati rispetto alle implementazioni precedenti.

5.1 Scelta dei dataset e dei parametri

Il primo passo per intrapredere una corretta analisi prestazionale è quello di scegliere dataset e parametri in modo da uniformare l'area di lavoro ed ottenere risultati perfettamente comparabili.
Dopo questo primo passo, bisogna notare che anche l'ambiente di esecuzione influisce notevolmente sulla buona riuscita dei test, per questo è opportuno effettuare quest'ultimi su piattaforme il più possibili equivalenti. Come si potrà notatre nei test dei capitoli successivi, cambiando piattaforma, ad esempio il sistema operativo, avremo notevoli differenti prestazioni su DBMS all'apparenza equivalenti, come avremo notevoli differenti prestazioni su diversi DBMS sullo stesso sistema, su versioni differenti degli stessi DBMS e chiaramente anche su differenti dataset e parametri sullo stesso DBMS.

5.1.1 Una panoramica sui dataset

Innanzitutto riportiamo la definizione classica di dataset (o data set), ovvero una raccolta od un insieme di dati, dove spesso ogni colonna rappresenta una particolare "variabile" ed ogni riga corrisponde a un determinato membro del insieme di dati in questione. Nel nostro caso invece, definiamo più dettagliatamente un dataset come un insieme di testi, divisi in paragrafi, frasi, parole, etc..., che verranno utilizzati per effettuare test sul software.

I dataset individuati per il nostro obiettivo sono quattro:
Deluxe Paint (abbreviato con DP), WhirlPool (abbreviato con Whirl), NVidia (abbreviato con NV) ed infine DNA.

I primi tre (DP, Whirl e NV) sono versioni diverse di manuali dello stesso oggetto, scritti in lingua inglese, mentre il quarto (DNA) contiene sequenze genetiche, tra cui amminoacidi e sequenze di dna.
Due dataset (Whirl e NV) sono stati utilizzati principalmente per l'analisi prestazionale, mentre DP e DNA sono stati utilizzati in entrambi i casi ed in particolare per l'analisi prestazionale comparata.

Descriviamo ora separatamente e in dettaglio i vari dataset:

- *Deluxe Paint:*
  Questo dataset è composto da tre file, *DP3*, *DP5* e *DP5-100*.
  DP3 è la versione 3 del manuale ed è utilizzato per la Translation Memory, cioè come frasi preesistenti da cui far partire la traduzione di testi successivi (argomento già introdotto nel capitolo 1.4 e sviluppato integralmente in [3]). Contiene 1499 frasi.
  DP5 è la versione 5 del manuale da tradurre e contiene 400 frasi.
  DP5-100 è un sottoinsieme di DP5 che contiene 107 frasi.

Nome	**Tipo**	**Numero Frasi**	**Rapporto con TM**
DP3	testuale	1499	100%
DP5	testuale	400	26,68%
DP5-100	testuale	107	7,14%

- *Whirl:*
  Questo dataset è composto da due file, *whirl.txt1* e *whirl_all*.
  Whirl.txt1 è un file che contiene diverse versioni di manuali ed è utilizzato per la Translation Memory. Contiene 34551 frasi.
  Whirl_all è una versione successiva del manuale da tradurre per i test. Contiene 267 frasi.

Nome	Tipo	Numero Frasi	Rapporto con TM
whirl.txt1	testuale	34551	100%
whirl_all	testuale	267	0,77%

- *NVidia:*
  Questo dataset è composto da cinque file:*1541, 2313, 2880, 2960* e *3123*.
  Sono tutti file "readme" per i driver ("relativi al numero del file") di schede grafiche ("NVIDIA Accelerated Linux Driver Set README & Installation Guide").
  1541 è usato come base per la Translation Memory. Contiene 610 frasi.
  2313 è una versione successiva da tradurre. Contiene 743 frasi.
  2880 è una versione successiva da tradurre. Contiene 846 frasi.
  2960 è una versione successiva da tradurre. Contiene 856 frasi.
  3123 è una versione successiva da tradurre. Contiene 890 frasi.

Nome	Tipo	Numero Frasi	Rapporto con TM
1541	testuale	610	100%
2313	testuale	743	121,80%
2880	testuale	846	138,69%
2960	testuale	856	140,33%
3123	testuale	890	145,90%

- *DNA:*
  Questo dataset è composto da quattro ulteriori classi: AA1-60, AA1-120, DNA3 e DNA4 che contengono a loro volta due file, un testo e un pattern di sequenze genetiche, amminoacidi o dna.
  AA1-60, (Sequenze di amminoacidi lunghe 60 simboli)
  Testo: usato come Translation Memory. Contiene 54 frasi e 3190 parole (o simboli) che rappresentano amminoacidi.
  Pattern: sequenza da ricercare. Contiene 10 frasi e 554 simboli.

Nome	Tipo	Numero Simboli	Rapporto con TM
AA1-60T	genetico	3190	100%
AA1-60P	genetico	554	17,37%

  AA1-120, (Sequenze di amminoacidi lunghe 120 simboli)
  Testo: usato come Translation Memory. Contiene 27 frasi e 3190 parole (o simboli) che rappresentano amminoacidi.
  Pattern: sequenza da ricercare. Contiene 5 frasi e 554 simboli.

Nome	Tipo	Numero Simboli	Rapporto con TM
AA1-120T	genetico	3190	100%
AA1-120P	genetico	554	17,37%

  DNA3
  Testo: usato come Translation Memory. Contiene 3 frasi e 357 parole (o simboli) che rappresentano basi azotate.
  Pattern: sequenza da ricercare. Contiene 3 frasi e 357 simboli.

Nome	Tipo	Numero Simboli	Rapporto con TM
DNA3-T	genetico	357	100%
DNA3-P	genetico	357	100%

  DNA4
  Testo: usato come Translation Memory. Contiene 28 frasi e 3296 parole (o simboli) che rappresentano basi azotate
  Pattern: sequenza da ricercare. Contiene 5 frasi e 574 simboli.

Nome	Tipo	Numero Simboli	Rapporto con TM
DNA4-T	genetico	3296	100%
DNA4-P	genetico	574	17,42%

5.1.2 La scelta dei parametri

Innanzitutto elenchiamo tutti i parametri modificabili utili a questi test, che ci permetteranno di modificare i risultati ottenuti.
In elenco, il primo nome è quello dato dal nostro software, il secondo, tra parentesi, se esiste, il nome o il valore dato dal programma per l'analisi comparata.

LMINSUB (minL): Soglia minima di lunghezza per l'estrazione di una sotto sequenza trovata.

K: Errore massimo ammesso tra due sequenze. L'aumento di ***K*** provoca l'aumento sia dei tempi di esecuzione sia del numero di match trovati (vedi [3] per maggiori dettagli).

KSUB: L'equivalente di ***K*** , ma per le sotto parti.

Q (1): Lunghezza di un ***Q***-gramma (per le frasi intere). es. ***Q***=1 parola o simbolo, ***Q***=3 tri-gramma, etc..

QSUB (1): Lunghezza di un ***Q***-gramma (per le sotto parti). es. ***Q***=1 parola o simbolo, ***Q***=7 epta-gramma, etc..

KINT (d): È equivalente a ***K***, ma col vincolo di essere un numero intero.

La scelta dei valori dei parametri, in generale, si è sempre basata sull'impostazione di valori che consentissero di avere un discreto numero di risultati, ma che al tempo stesso non implicassero tempi di attesa superiori ad alcuni minuti.
I parametri ottimali sono risultati molto spesso quelli impostati di default.
Da rilevare che in alcuni test è stato necessario settare i parametri in modo da poter effettuare confronti incrociati con altri programmi in modo da valutare l'effettiva efficienza del software utilizzato.

5.2 Analisi prestazionale

L'analisi prestazionale, intesa come verifica sperimentale delle prestazioni, è uno dei principali, se non il principale scopo del lavoro svolto.

Passando obbligatoriamente da una fase di porting, ovvero un operazione di portabilità, seguita e preceduta da altre importanti fasi, si è giunto alla possibilità di testare in modo concreto, su dataset appositamente scelti, il software e, nella fase di analisi prestazionale comparata (capitolo 5.3), confrontare l'efficienza di tale implementazione con quella basata sull'algoritmo del Suffix Array.

Si precisa anche che la macchina su cui sono stati svolti i test è descritta nel paragrafo successivo (5.2.1)

5.2.1 L'ambiente e l'inserimento dei dati

Per poter effettuare un test sul software è necessario innanzitutto riempire la Translation Memory del programma, nel nostro caso con versioni precedenti dei manuali a cui suggerire una traduzione.

L'analisi delle performance d'inserimento dei dati ci permette in primo luogo di valutare la velocità d'inserimento dei dati nei rispettivi DBMS. Da notare che l'inserimento per Oracle JI e Oracle JO è perfettamente equivalente.

- Riportiamo che per tutti i DBMS (ove non specificatamente scritto) valgono le seguenti impostazioni e risultati ottenuti (Tab. 5.2.1-a):

Stemming	File:	Frasi inserite	Q-grammi	Q-grammi (sub)
OFF	dp3-all.tm	1499	32915	31416
ON	dp3-all.tm	1499	17970	16471
OFF	Readme1541.tm	610	12554	11944
ON	Readme1541.tm	610	7333	6723
OFF	Whirl.tm	34551	463967	429416
ON	Whirl.tm	34551	283108	248945

(Tab. 5.2.1-a – Impostazioni e risultati di default)

- Riportiamo che i tempi risultanti sono scritti ordinatamente in base alla tabella sovrastante (5.2.1-a) e sono soggetti ai parametri della tabella sottostante (5.2.1-b).

Parametro	Q	Q-sub	K	KSUB	LMINSUB
Valore	3	2	0.2	0.2	4

(Tab. 5.2.1-b – Parametri di default)

- Riportiamo che il tempo proporzionale è rapportato al DBMS Oracle 9i (in quanto prima versione del software e primo test effettuato).

- Riportiamo che tutti i grafici comparati sono soggetti agli stessi parametri (anche se non specificatamente scritti) salvo casi eccezionali comunque notificati.

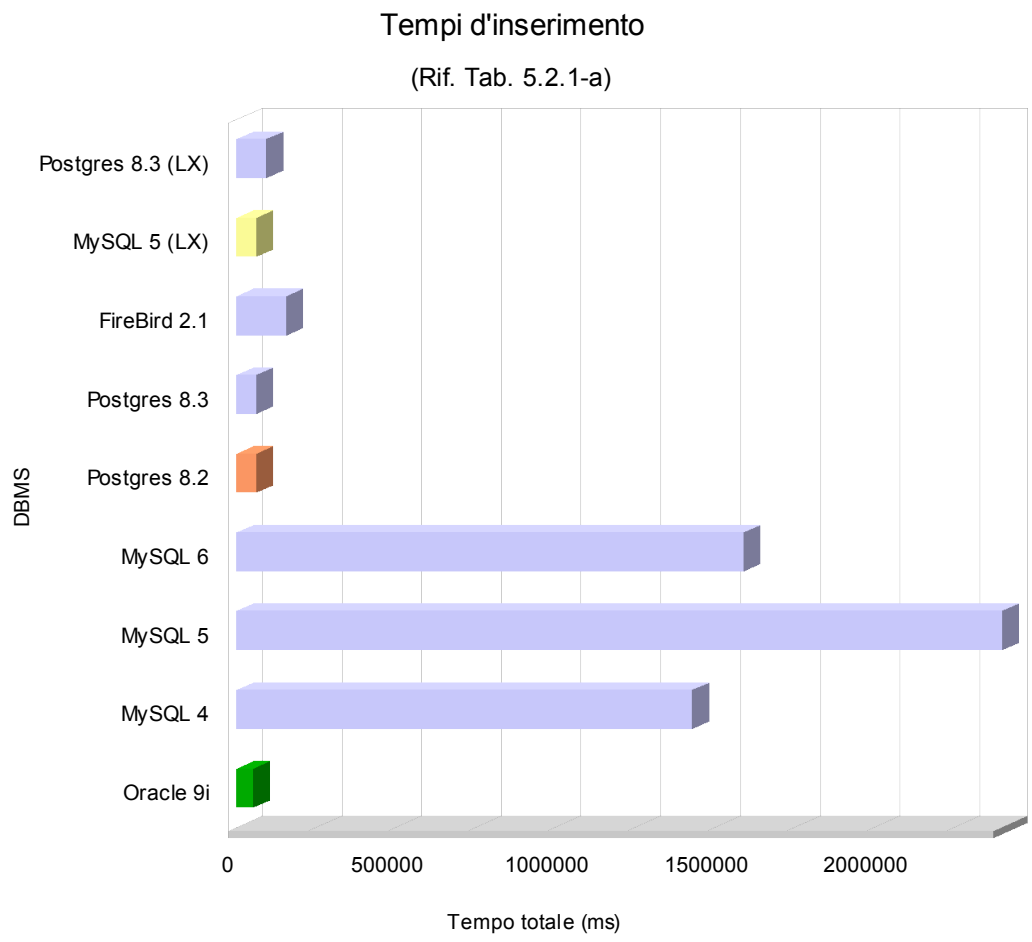
- Riportiamo che tutti i test sono stati svolti sulla seguente macchina:
ASUS-A6Tc mod. AP022H
Note: 1 Gb ram, processore dual-core AMD Turion TL-50 (x64 bit) da 1,60 GHz (x2), 512Kb chache L2, 256Kb chache L1.

– *Il Sistema Operativo (SO o OS) varia a seconda che, dopo il nome nella colonna "Nome del DBMS", sia riportata la sigla "lx" o meno:*
Senza sigla: Windows XP (Home Edition x86, 32Bit), disco NTFS.
Con sigla "lx": Gnu/Linux (Kubuntu 7.04 AMD-X64, Kernel 2.6.20-17) disco Raiserfs

Tempi per dp3-all.tm in modalità stemming OFF:

Nome del DBMS	Tempo Frasi (ms)	Tempo Q-grammi (ms)	Tempo Q-grammi Sub (ms)	Tempo totale (ms)	Pari a .. (min)	.. e .. (sec)	Tempo Propor-zionale	Note
Oracle 9i	2375	25969	25547	53891	0	54	1	
MySQL 4	47141	772766	609844	1429751	23	50	26,53	
MySQL 5	47688	1184984	1194797	2427469	40	27	45,04	
MySQL 5 lx	15301	23598	23061	61960	1	2	1,15	
MySQL 6	47327	848343	694644	1590314	26	30	29,51	
Postgres 8.2	2351	31063	29359	62953	1	3	1,17	
Postgres 8.3	2703	31750	29656	64109	1	4	1,19	
Postgres 8.3 lx	14573	43308	36862	94743	1	35	1,76	
Firebird 2.1	4828	78735	72781	156344	2	36	2,9	

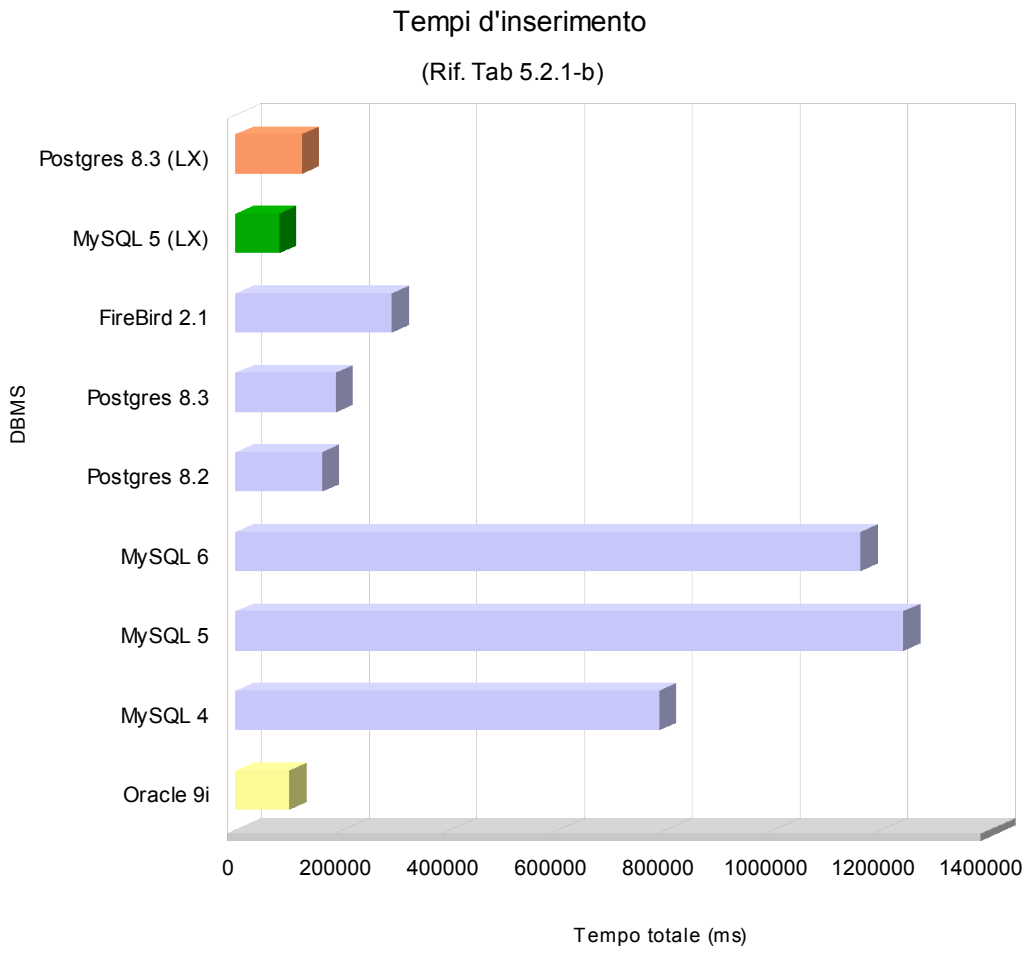
(Tab. 5.2.1-a – DP3 STEM off)



Tempi per dp3-all.tm in modalità stemming ON:

Nome del DBMS	Tempo Frasi (ms)	Tempo Q-grammi (ms)	Tempo Q-grammi Sub (ms)	Tempo totale (ms)	Pari a .. (min)	.. e .. (sec)	Tempo Propor-zionale	Note
Oracle 9i	72125	14703	13891	100719	1	41	1	
MySQL 4	88094	366734	332016	786844	13	7	7,81	
MySQL 5	108907	517343	614766	1241016	20	41	12,32	
MySQL 5 lx	41854	20064	19575	81493	1	21	0,81	
MySQL 6	411219	393500	356718	1161437	19	21	11,53	
Postgres 8.2	128219	16937	16226	161382	2	41	1,6	
Postgres 8.3	152781	17297	16218	186296	3	6	1,85	
Postgres 8.3 lx	66728	29022	27307	123057	2	3	1,22	
Firebird 2.1	210000	42093	38344	290437	4	50	2,88	

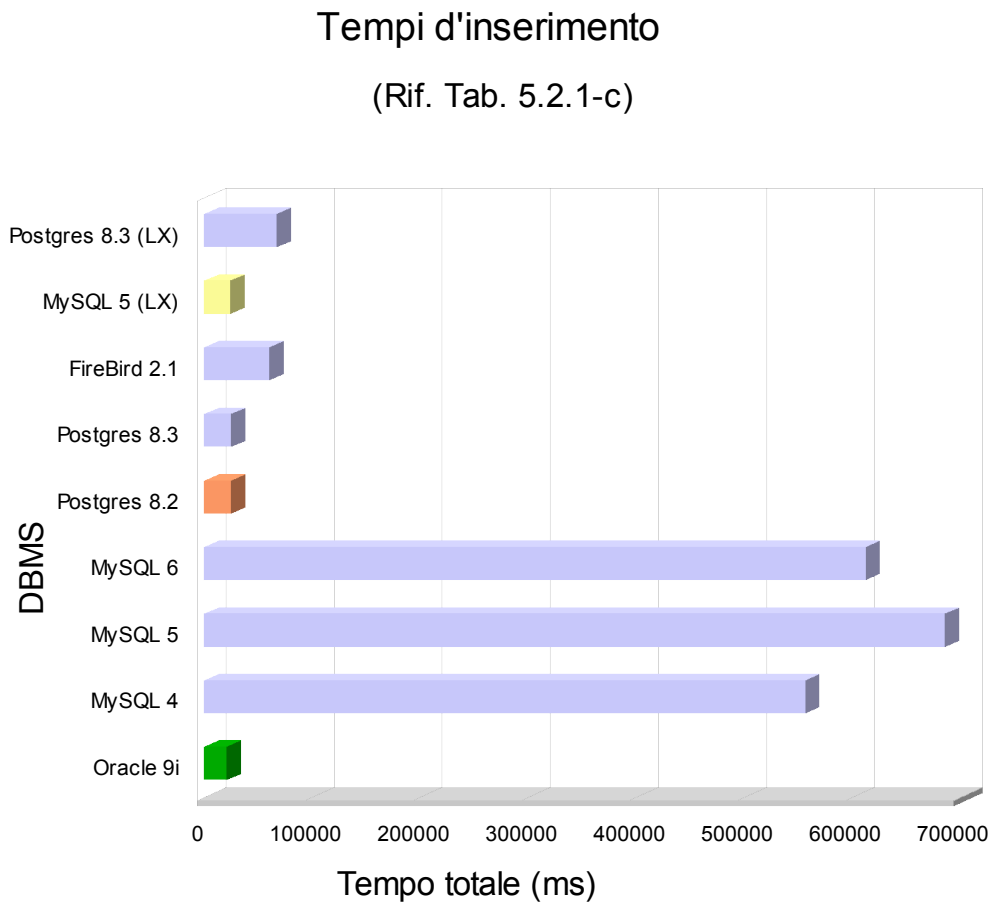
(Tab. 5.2.1-b – DP3 STEM on)



Tempi per README1541.tm in modalità stemming OFF:

Nome del DBMS	Tempo Frasi (ms)	Tempo Q-grammi (ms)	Tempo Q-grammi Sub (ms)	Tempo totale (ms)	Pari a .. (min)	.. e .. (sec)	Tempo Propor-zionale	Note
Oracle 9i	1015	10110	9922	21047	0	21	1	
MySQL 4	19937	300221	236277	556435	9	16	26,44	
MySQL 5	20361	359729	305413	685503	11	26	32,57	
MySQL 5 lx	9698	5667	8994	24359	0	24	1,16	
MySQL 6	20344	321328	270656	612328	10	12	29,09	
Postgres 8.2	1047	12500	11390	24937	0	25	1,18	
Postgres 8.3	1078	12187	11985	25250	0	25	1,2	
Postgres 8.3 lx	16571	28380	22323	67274	1	7	3,2	
Firebird 2.1	2406	30047	28172	60625	1	1	2,88	

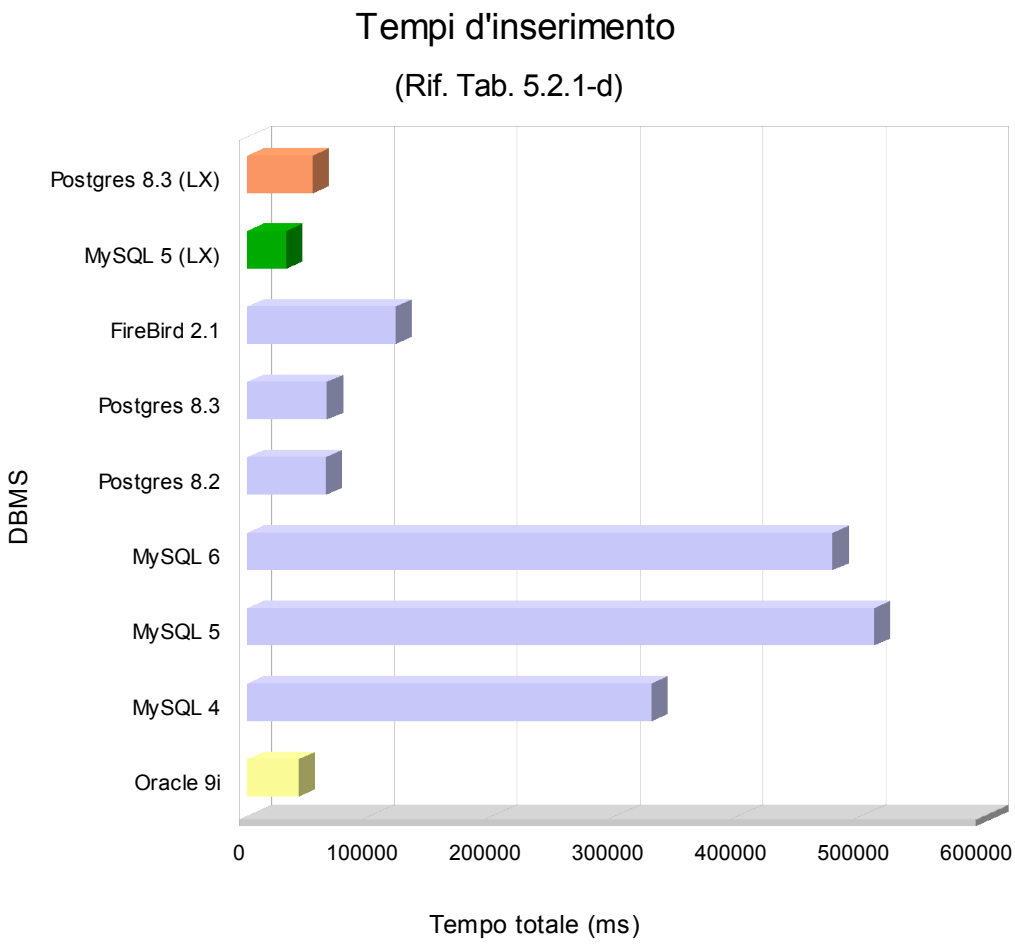
(Tab. 5.2.1-c – NV1541 STEM off)



Tempi per README1541.tm in modalità stemming ON:

Nome del DBMS	Tempo Frasi (ms)	Tempo Q-grammi (ms)	Tempo Q-grammi Sub (ms)	Tempo totale (ms)	Pari a .. (min)	.. e .. (sec)	Tempo Propor-zionale	Note
Oracle 9i	30016	6312	5672	42000	0	42	1	
MySQL 4	19937	300221	236277	556435	9	16	26,44	
MySQL 5	20361	359729	305413	685503	11	26	32,57	
MySQL 5 lx	16191	8204	7942	32337	0	32	0,77	
MySQL 6	160532	167687	148672	476891	7	57	11,35	
Postgres 8.2	50125	7391	6882	64398	1	4	1,53	
Postgres 8.3	50859	7399	7057	65315	1	5	1,56	
Postgres 8.3 lx	44029	5241	4439	53709	0	54	1,28	
Firebird 2.1	86141	18172	16922	121235	2	1	2,89	

(Tab. 5.2.1-d – NV1541 STEM on)



Tempi per Whirl.tm in modalità stemming OFF:

Nome del DBMS	Tempo Frasi (ms)	Tempo Q-grammi (ms)	Tempo Q-grammi Sub (ms)	Tempo totale (ms)	Pari a .. (min)	.. e .. (sec)	Tempo Propor-zionale	Note
Oracle 9i	55282	440828	418562	914672	15	15	1	
MySQL 4	***	***	***	24225087	403	45	26,49	(*)
MySQL 5	***	***	***	35498420	591	38	38,81	(*)
MySQL 5 lx	28402	383271	268344	680017	11	20	0,74	
MySQL 6	***	***	***	26799889	446	40	29,3	(*)
Postgres 8.2	44953	470781	435797	951531	15	52	1,04	
Postgres 8.3	46002	470556	443766	960324	16	0	1,05	
Postgres 8.3 lx	31669	463967	282473	778109	12	58	0,85	
Firebird 2.1	***	***	***	2643402	44	3	2,89	(*)

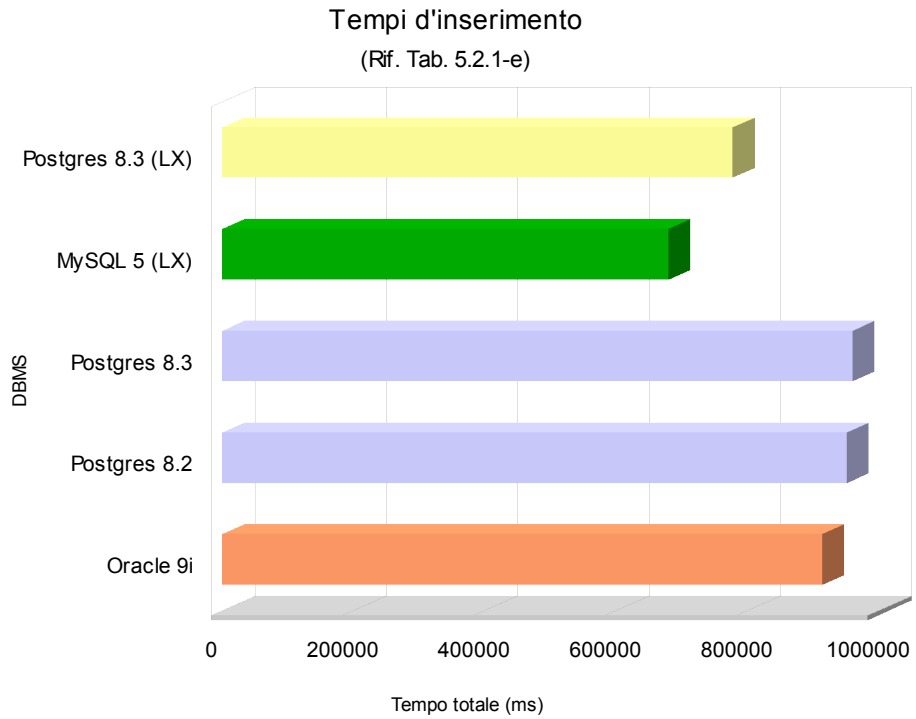
(Tab. 5.2.1-e – Whirl STEM off)

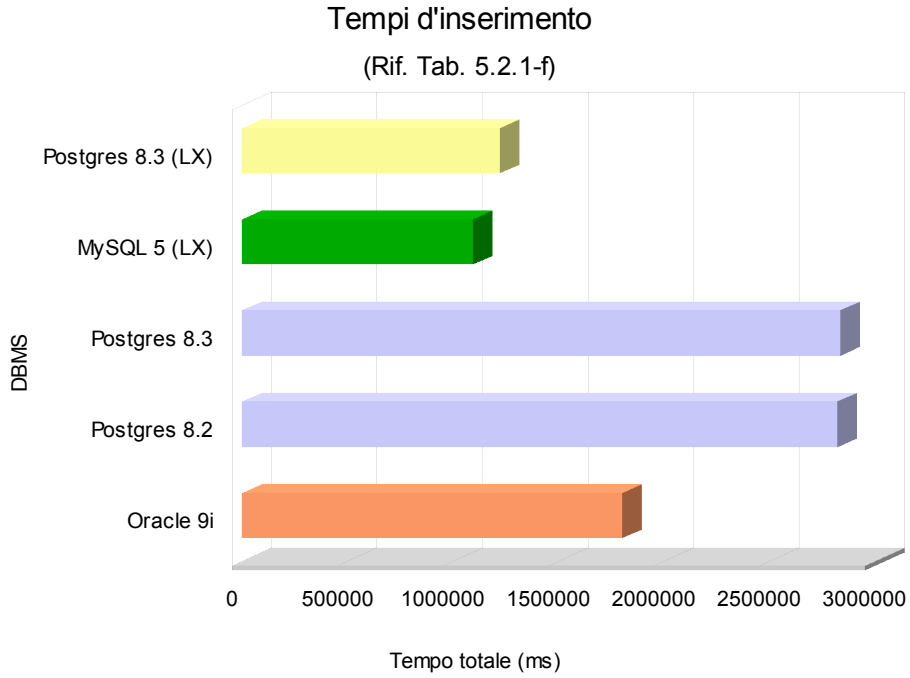
Tempi per Whirl.tm in modalità stemming ON:

Nome del DBMS	Tempo Frasi (ms)	Tempo Q-grammi (ms)	Tempo Q-grammi Sub (ms)	Tempo totale (ms)	Pari a .. (min)	.. e .. (sec)	Tempo Propor-zionale	Note
Oracle 9i	1272500	280406	251094	1804000	30	4	1	
MySQL 4	***	***	***	14125320	235	25	7,83	(*)
MySQL 5	***	***	***	22080960	368	1	12,24	(*)
MySQL 5 lx	670842	277648	147262	1095752	18	16	0,61	
MySQL 6	***	***	***	20637760	343	58	11,44	(*)
Postgres 8.2	2304703	279921	237766	2822390	47	2	1,56	
Postgres 8.3	2314854	280039	240912	2835805	47	16	1,57	
Postgres 8.3 lx	789278	284185	150475	1223938	20	24	0,68	
Firebird 2.1	***	***	***	5213560	86	54	2,89	(*)

(Tab. 5.2.1-f – Whirl STEM on)

Le righe contrassegnate dal seguente simbolo (*) nella colonna “Note” indicano che il test è stato stimato secondo la media dei due inserimenti precedenti, in quanto i tempi di attesa sarebbero stati esorbitanti, alcuni intorno alle 10 ore. Ulteriori test, quali la prova sul dataset ridotto a 1/32 del volume, etc... , sono stati effettuati per verificare l'attendibilità delle stime, che sono risultate abbastanza precise. Nei grafici relativi questi DBMS non sono riportati.


Tempi d'inserimento
(Rif. Tab. 5.2.1-e)
Postgres 8.3 (LX)
MySQL 5 (LX)
Postgres 8.3
Postgres 8.2
Oracle 9i
DBMS
0
200000
400000
600000
800000
1000000
Tempo totale (ms)


Tempi d'inserimento
(Rif. Tab. 5.2.1-f)
Postgres 8.3 (LX)
MySQL 5 (LX)
Postgres 8.3
Postgres 8.2
Oracle 9i
DBMS
0
500000
1000000
1500000
2000000
2500000
3000000
Tempo totale (ms)

Tab. 5.2.1-a:

Dalla tabella ricaviamo che Oracle è il DBMS più veloce per l'inserimento in Translation Memory su questo dataset (DP) in modalità stemming off, mentre MySQL 5.1 per Linux, PostgreSQL 8.2 e 8.3 sono risultati i secondi classificati essendo distaccati di solo 1 secondo tra loro.

Da notare che FireBird 2.1 lavora bene nella prima fase (inserimento frasi), ma perde notevolmente sull'inserimento dei Q-grammi.

Disastrose le versioni 4,5,6 di MySQL per Windows che risultano da 25 a 40 volte più lente della versione 5.1 per Linux.

Tab. 5.2.1-b:

Dalla tabella ricaviamo che MySQL 5.1 per Linux è il DBMS più veloce per l'inserimento in Translation Memory su questo dataset (DP) in modalità stemming on e, insieme a PostgreSQL 8.3 per Linux, è il più rapido nella prima fase (inserimento frasi). Espressi in rapporto, MySQL 5.1 per Linux, con (minuti:secondi) 1:21 è risultato 1,23 volte più rapido di Oracle, secondo classificato con 1:41), che a sua volta è risultato più veloce di PostgreSQL 8.3 per Linux, terzo classificato con 2:03, di 1,22 volte.

Meno disastrose del caso precedente le versioni 4,5,6 di MySQL per Windows che risultano comunque circa 10 volte più lente della versione 5.1 per Linux.

Tab. 5.2.1-c:

Dalla tabella ricaviamo che Oracle è il DBMS più veloce per l'inserimento in Translation Memory su questo dataset (NV) in modalità stemming off, mentre MySQL 5.1 per Linux e PostgreSQL 8.2 sono risultati i secondi classificati essendo distaccati di solo 1 secondo tra loro.

Da notare che MySQL 5.1 per Linux lavora con tempi dimezzati, rispetto agli altri, due sui Q-grammi, mentre impiega circa 10 volte tanto per le frasi.

Disastrose le versioni 4,5,6 di MySQL per Windows che risultano circa 30 volte più lente della versione 5.1 per Linux.

Tab. 5.2.1-d:

Dalla tabella ricaviamo che MySQL 5.1 per Linux è il DBMS più veloce per l'inserimento in Translation Memory su questo dataset (NV) in modalità stemming on, mentre Oracle è risultato il secondo classificato e PostgreSQL 8.3 per Linux il terzo. Per quanto riguarda le proporzioni, MySQL 5.1 per Linux impiega un 23% in meno rispetto ad Oracle che impiega, a sua volta, un 28% in meno rispetto a PostgreSQL 8.3 per Linux.

Da notare che MySQL 5.1 per Linux lavora con tempi dimezzati sulle frasi, mentre impiega circa un 10% in più per i Q-grammi, rispetto agli altri due.

Meno disastrose del caso precedente le versioni 4,5,6 di MySQL per Windows che risultano comunque circa 10 volte più lente della versione 5.1 per Linux.

Tab. 5.2.1-e:

Dalla tabella ricaviamo che MySQL 5.1 per Linux è il DBMS più veloce per l'inserimento in Translation Memory su questo dataset (Whirl) in modalità stemming off, e la sua forza risiede nell'estrazione dei Q-grammi, mentre PostgreSQL 8.3 per Linux è risultato il secondo classificato e Oracle il terzo. Poca differenza anche con le versioni di PostgreSQL 8.2 e 8.3 per Windows che impiegano sui 16 minuti, circa 45 secondi in più di Oracle e 5 minuti in più di MySQL 5.1 per Linux.

Le stime su FireBird 2.1 danno un tempo di circa 45 minuti, ovvero 3-4 volte quello di MySQL 5.1 per Linux.

Sempre disastrose le stime per le versioni 4,5 e 6 di MySQL per Windows, che danno un tempo minimo di 6 ore e 30 minuti e un massimo di 10 ore, ovvero da 30 a 60 volte quello di MySQL 5.1 per Linux.

Tab. 5.2.1-f:

Dalla tabella ricaviamo che MySQL 5.1 per Linux è il DBMS più veloce per l'inserimento in Translation Memory su questo dataset (Whirl) in modalità stemming on, e la sua forza risiede nell'estrazione dei Q-grammi, mentre PostgreSQL 8.3 per Linux è risultato il secondo classificato e Oracle il terzo. Molta differenza con le versioni di PostgreSQL 8.2 e 8.3 per Windows che impiegano sui 47 minuti, ovvero circa 17 minuti in più di Oracle e 25 in più di MySQL 5.1 per Linux.

Le stime su FireBird 2.1 danno un tempo di circa 87 minuti, ovvero circa 4 volte quello di MySQL 5.1 per Linux.

Sempre pessime le stime per le versioni 4,5 e 6 di MySQL per Windows, che danno un tempo minimo di 4 e un massimo di 6 ore, ovvero circa 10 volte quello di MySQL 5.1 per Linux.

Infine, da tutti i dati ottenuti, confrontabili nelle varie tabelle di questo capitolo e meglio visibili nei grafici relativi, si è ricavato che, in ordine di velocità decrescente, i migliori DMBS per l'inserimento dei dati nella Translation Memory sono:

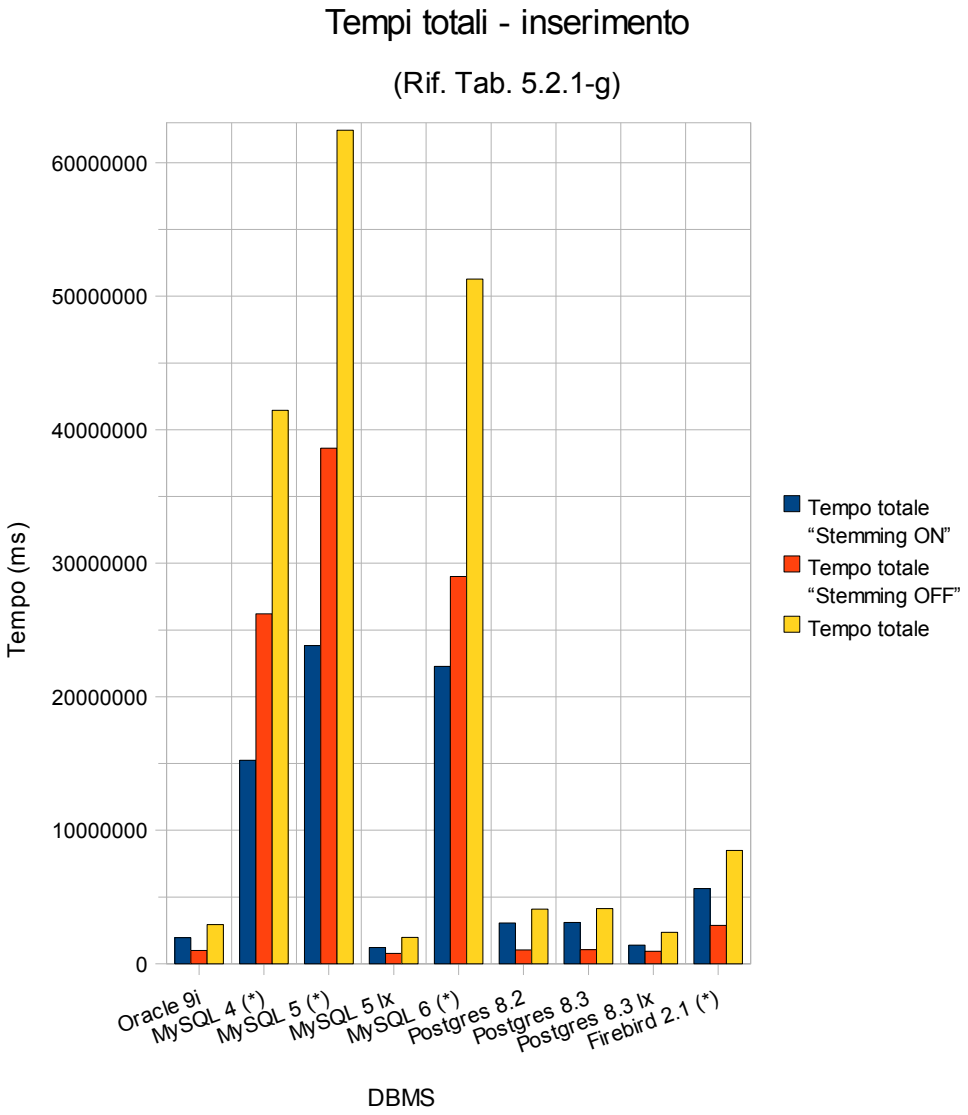
MySQL 5.1 per Linux in modalità stemming on.

Oracle, MySQL 5.1 e PostgreSQL 8.3 per Linux in modalità stemming off.

Concludendo, MySQL 5.1 per Linux risulta la scelta migliore per l'inserimento di dati in Translation Memory, considerando anche il fatto che risulta il più veloce su moli di dati grandi anche in modalità stemming off. Questo dato è visibile dal grafico e dalla tabella sottostanti (5.2.1-g) somma di tutte le precedenti.

Nome del DBMS	Tempo totale "Stemming ON"	Tempo totale "Stemming OFF"	Tempo totale
Oracle 9i	1946719	989610	2936329
MySQL 4 (*)	15241734	26211273	41453007
MySQL 5 (*)	23832602	38611392	62443994
MySQL 5 lx	1209582	766336	1975918
MySQL 6 (*)	22276088	29002531	51278619
Postgres 8.2	3048170	1039421	4087591
Postgres 8.3	3087416	1049683	4137099
Postgres 8.3 lx	1400704	940126	2340830
Firebird 2.1 (*)	5625232	2860371	8485603

(Tab. 5.2.1-g – Tempi totali d'inserimento)



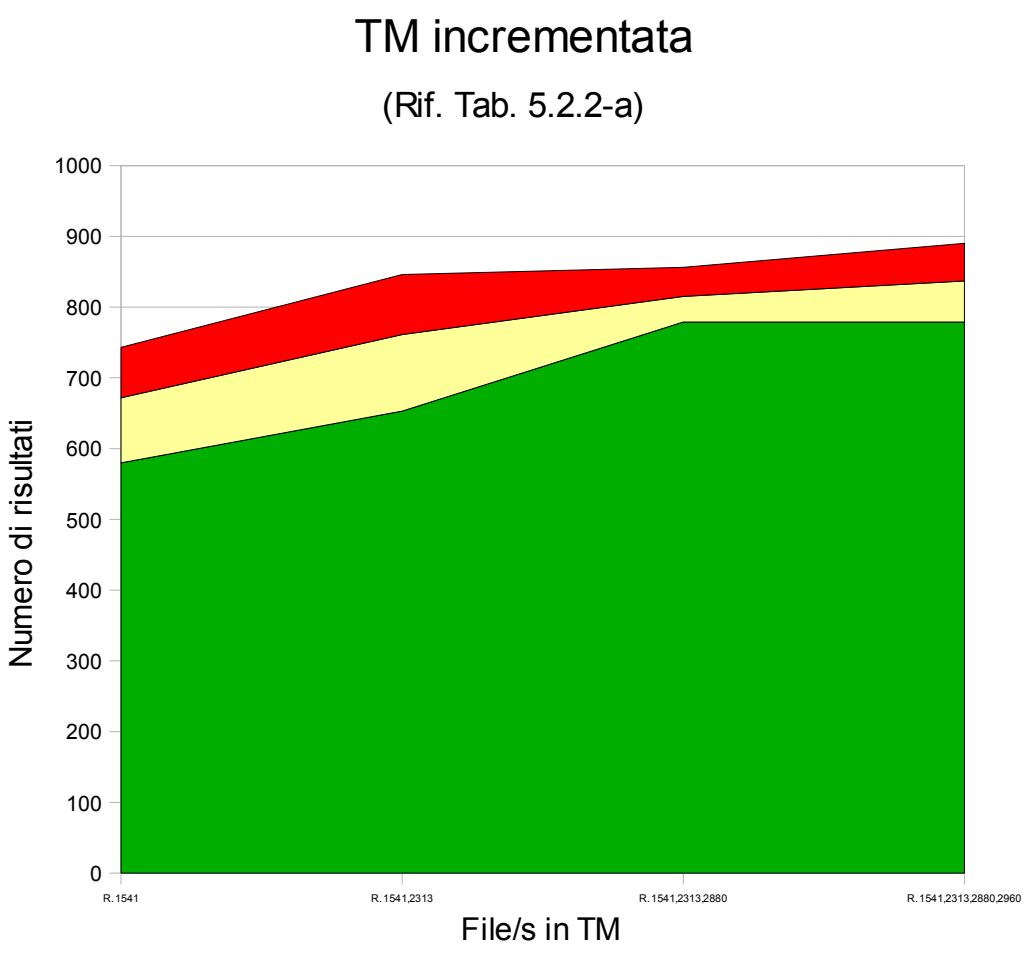
5.2.2 La Translation Memory

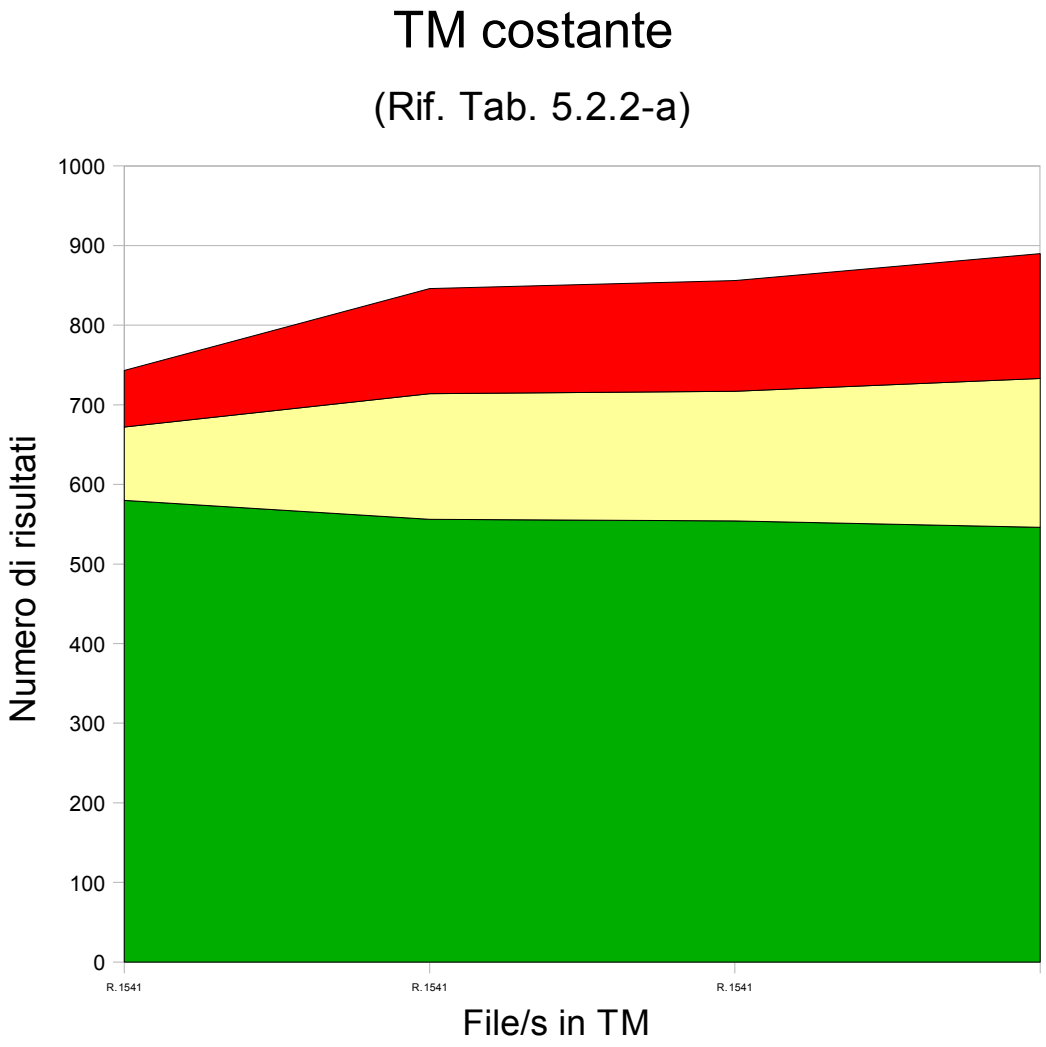
Come descritto nei capitoli precedenti, la Translation Memory (abbreviata con TM) è alla base dei sistemi EBMT come il software utilizzato per questa ricerca.
[3] I sistemi EBMT emulano la traduzione umana, riconoscendo la similarità di una nuova frase nel linguaggio sorgente (da tradurre) con una precedentemente tradotta, ed utilizzando quest'ultima per realizzare una "traduzione per analogia".
L'idea di base dell'EBMT fa riferimento a un database di traduzioni passate, in cui compaiono in parallelo le frasi da tradurre e quelle già tradotte: si tratta proprio della TM. L'accuratezza e la qualità della traduzione dipendono fortemente dalla dimensione e dalla copertura della TM, ma non è necessario che questa assuma le dimensioni dei database richiesti, ad esempio, dai sistemi di traduzione statistica, poiché non è necessario coprire l'intero vocabolario.
I sistemi di tipo TM memorizzano le traduzioni passate e le propongono quando, traducendo nuovo materiale, incontrano frasi identiche o simili. I sistemi TM "imparano" e migliorano con l'utilizzo: più vengono utilizzati più diventano accurati e utili. I sistemi TM, insomma, non traducono da soli. Quando il sistema TM è in procinto di tradurre una nuova frase, svolge una ricerca nella TM e presenta all'utente le traduzioni di frasi identiche o simili che il traduttore ha fatto in passato. Nonostante in molti casi non proponga di suo una traduzione, lo aiuta ad assicurare la consistenza dello stile e della terminologia permettendogli di consultare con facilità le traduzioni passate. In altre parole, i sistemi basati su Translation Memory non traducono ma forniscono consistenza, permettendo al traduttore un lavoro più rapido e di maggiore qualità.

Riportiamo ora un esempio del funzionamento della TM:

Translation Memory	FILE	FULL time (ms)	SUB time (ms)	N° FRASI	FULL	SUB	NO
Tm incrementa							
R. 1541	Readme2313	4469	11860	743	580	92	71
R. 1541,2313	Readme2880	14593	42766	846	653	108	85
R. 1541,2313,2880	Readme2960	27406	3797	856	779	36	41
R. 1541,2313,2880,2960	Readme3123	42078	16250	890	779	58	53
Tm costante							
R. 1541	Readme2313	4469	11860	743	580	92	71
R. 1541	Readme2880	5468	30421	846	556	158	132
R. 1541	Readme2960	5125	31860	856	554	163	139
R. 1541	Readme3123	5687	35609	890	546	187	157

(Tab. 5.2.2-a – TM incrementata e costante)



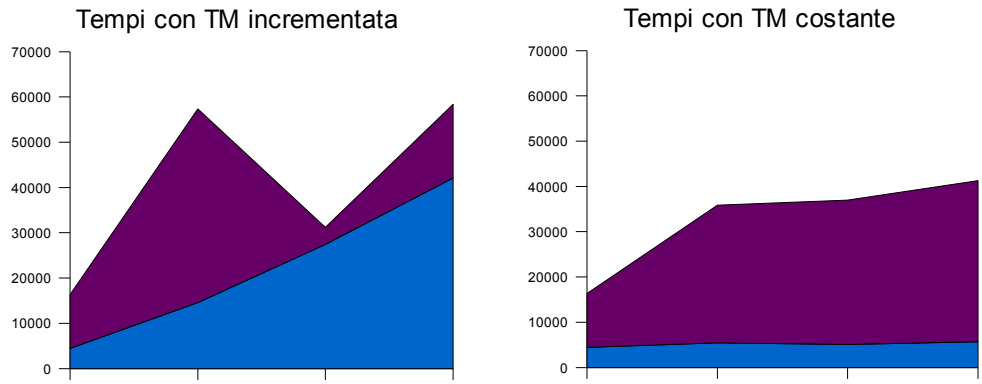


Dai due grafici relativi alla tabella 5.2.2-a, notiamo che il numero di match trovati nei due casi è completamente differente, infatti, nel caso con TM costante otteniamo dai 100 ai 200 risultati in meno per le frasi intere e dai 50 ai 130 risultati in meno per le sotto frasi rispetto al caso con TM incrementata.

Le curve riguardanti le frasi incomplete e quelle non trovate, nel caso con TM incrementa, assumono un andamento che tende a smorzarsi al crescere della TM, mentre nel caso con TM costante l'andamento è crescente in senso stretto.

Ciò ribadisce il concetto sopra elencato, cioè al crescere della TM, considerando esempi utili al fine, cresce anche il numero di match trovati.

Visualizziamo ora un piccolo inconveniente nell'accrescimento della TM:



(Rif. Tab. 5.2.2-a)

Considerando sull'asse delle ascisse i medesimi riferimenti ai grafici sopra associati e sull'asse delle ordinate i tempi espressi in millisecondi (ms), notiamo che:

I tempi di esecuzione della query sulle frasi intere (parte sotto del grafico, in blu) rimangono costanti per il caso con TM costante, nell'altro caso, con TM incrementa, l'andamento è quello di una retta (con una pendenza di circa 20°) cioè crescente in senso stretto.

Nel caso invece della ricerca delle sotto frasi (parte sopra del grafico, in viola), se tralasciamo il primo picco, dovuto principalmente al basso numero di match trovati nell'occasione, vediamo che la curva nel caso di TM incrementa segue l'andamento di quella sottostante (a meno di una costante che non specifichiamo), mentre nel caso della TM costante, il basso numero di match decreta un andamento strettamente crescente del tempo per la ricerca di sotto frasi.

In sostanza, è meglio accrescere la TM che non, ma è ancora meglio mantenerla costantemente aggiornata con esempi “vicini” al testo che si desidera tradurre, in modo da ottimizzare i match ed il tempo impiegato.

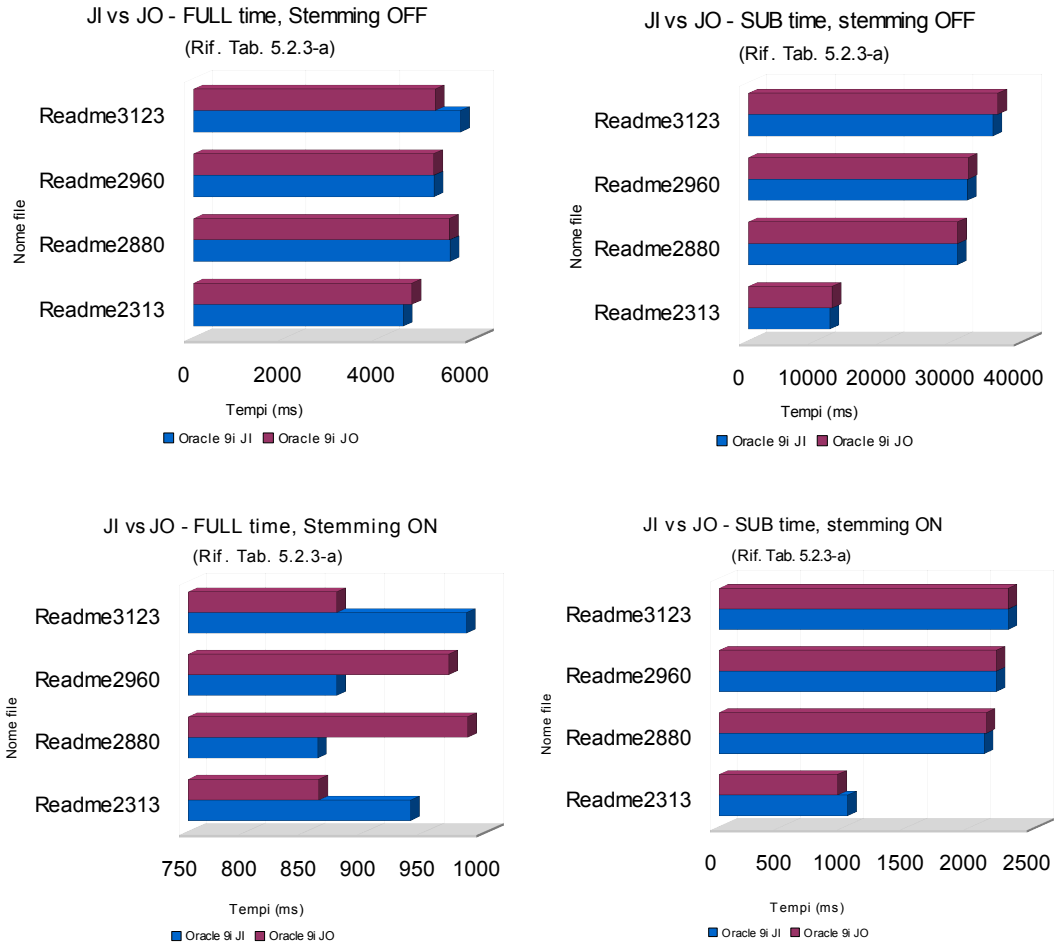
5.2.3 Oracle, Java Inside vs Java Outside

Il primo passo per valutare la strada da scegliere, va fatto nella direzione dell'utilità o meno nell'includere classi Java all'interno dei DBMS.
Uno dei pochi che ci permette questo è Oracle, infatti, la versione originale del software lavorava col metodo "Java Inside" proprio e solo su questo DBMS. Successivamente, modificando il software è stata necessaria una scelta in una direzione precisa, continuare con questa tecnica o "estrarre" le classi dal DBMS e riportarle nel loro contesto originale (Java Outside). I test seguenti risolvono questo piccolo ma fondamentale problema.

FILE	STEMMING		FULL time	FULL time		SUB time	SUB time
			Oracle 9i JI	Oracle 9i JO		Oracle 9i JI	Oracle 9i JO
Readme2313	OFF		4469	4656		11860	12188
Readme2880	OFF		5468	5450		30421	30469
Readme2960	OFF		5125	5119		31860	32004
Readme3123	OFF		5687	5153		35609	36327
Readme2313	ON		937	860		1016	938
Readme2880	ON		859	985		2094	2109
Readme2960	ON		875	969		2188	2188
Readme3123	ON		984	875		2282	2281

(Tab. 5.2.3-a – NV, Oracle JI vs JO)

La tabella sovrastante (5.2.3-a) riguarda il dataset NV, in entrambe le modalità (stemming off e on). È divisa in quattro aree, visibili dalle separazioni delle righe e delle colonne, a cui corrispondono, i quattro grafici successivi.



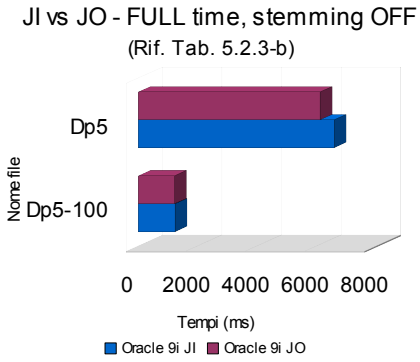
Dai due grafici in modalità “stemming OFF” non notiamo grandi differenze in fatto di performance, così come nel grafico “SUB time” in modalità “stemming ON”, mentre nel grafico “FULL time, stemming ON” notiamo una inversione di tendenze alternate, che comunque bilanciano le due soluzioni.

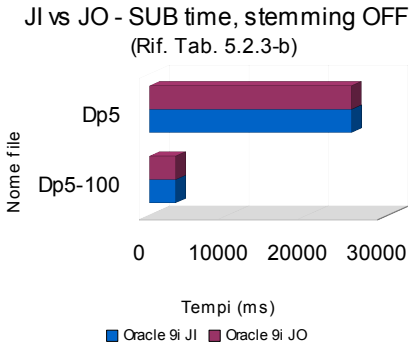
A conferma di ciò, la somma totale dei tempi per la prima soluzione (JI) è di 141734, mentre per la seconda (JO) è di 142571, una differenza di appena 837 ms (0,837 secondi) che influenza il rapporto tra le due versioni di un misero 0,6%.

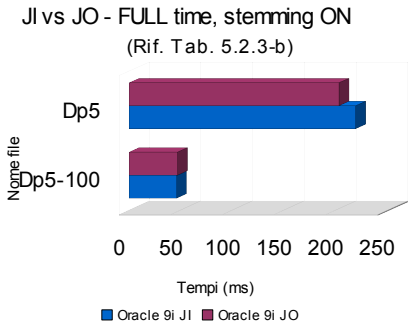
FILE	STEMMING		FULL time	FULL time		SUB time	SUB time
			Oracle 9i JI	Oracle 9i JO		Oracle 9i JI	Oracle 9i JO
Dp5-100	OFF		1234	1219		3344	3359
Dp5	OFF		6578	6109		25499	25469
Dp5-100	ON		46	47		109	109
Dp5	ON		219	203		1047	1047

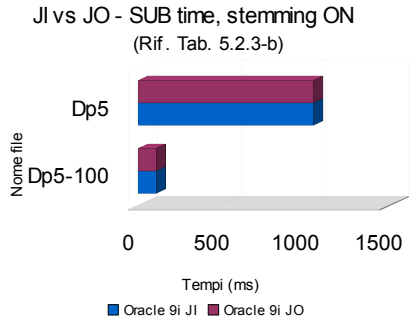
(Tab. 5.2.3-b – DP, Oracle JI vs JO)

La tabella sovrastante (5.2.3-b) riguarda il dataset DP, in entrambe le modalità (stemming off e on). È divisa in quattro aree, visibili dalle separazioni delle righe e delle colonne, a cui corrispondono i quattro grafici successivi.









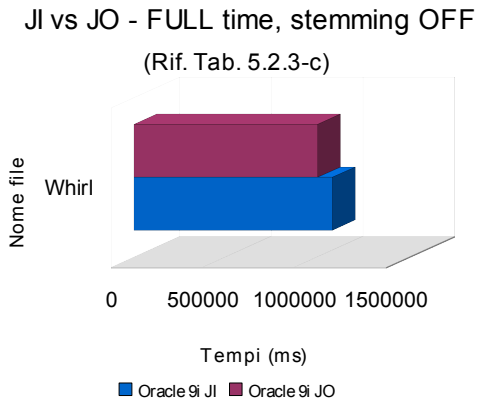
Dai grafici notiamo che la soluzione JO risulta leggermente migliore della JI in quasi tutti i casi, anche se di pochi millisecondi.

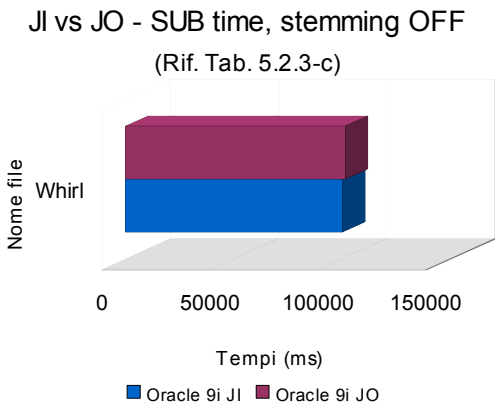
A conferma di ciò, la somma totale dei tempi per la prima soluzione (JI) è di 38076, mentre per la seconda (JO) è di 37562, una differenza di appena 514 ms (0,514 secondi) che influenza il rapporto tra le due versioni di un 1,35%.

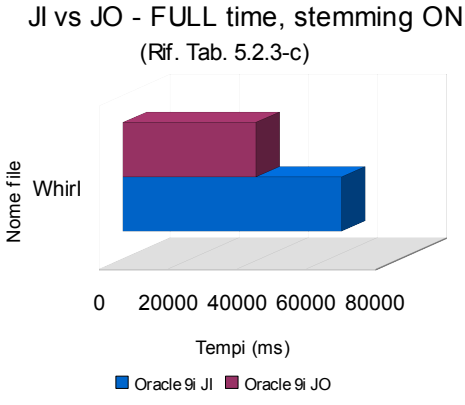
FILE	STEMMING		FULL time	FULL time		SUB time	SUB time
			Oracle 9i JI	Oracle 9i JO		Oracle 9i JI	Oracle 9i JO
Whirl	OFF		1083641	1002594		100719	102014
Whirl	ON		63329	38546		7172	6703

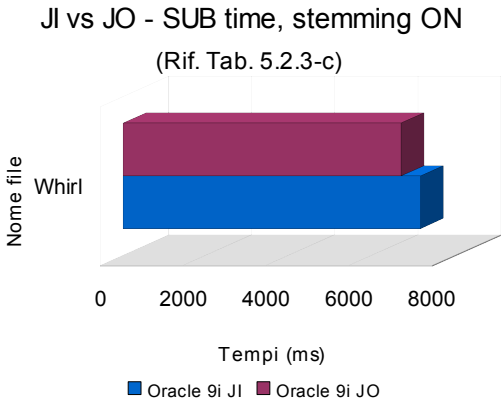
(Tab. 5.2.3-c – Whirl, Oracle JI vs JO)

La tabella sovrastante (5.2.3-c) riguarda il dataset Whirl, in entrambe le modalità (stemming off e on). È divisa in quattro aree, visibili dalle separazioni delle righe e delle colonne, a cui corrispondono i quattro grafici successivi.









Dai grafici notiamo che la soluzione JO risulta nettamente migliore della JI in tutti i casi, ad eccezione di uno dove la differenza è comunque minima.

A conferma di ciò, la somma totale dei tempi per la prima soluzione (JI) è di 1254861, mentre per la seconda (JO) è di 1149867, una notevole differenza di 104994 ms (104,994 secondi) che influenza il rapporto tra le due versioni di un 8,37% facendo preferire di gran lunga la seconda.

A conti fatti dunque, la versione Java Outside (JO) risulta conveniente rispetto alla versione Java Inside (JI), considerando le prime due prove a pari merito e la terza, sul dataset più grande e significativo, vinta dalla versione JO con un margine superiore all'8% sia del caso che del totale.

Con questi dati alla mano, l'implementazione del software, durante il corso di sviluppo, è stata notevolmente influenzata, optando per la versione JO, essendo questa più performante, più semplice e portabile.
Tutti i DBMS che non recano scritte quali “JI” o “JO” sono da considerarsi “JO”.

5.2.4 Il dataset Deluxe Paint

Riportiamo i risultati ottenuti con i vari DBMS sul dataset Deluxe Paint, utilizzando come riferimento sui tempi proporzionali il DBMS Oracle 9i JI, essendo questa la versione originale e necessitando obbligatoriamente di un punto di riferimento. Nei tempi proporzionali si utilizza una tolleranza del 5% per evidenziare il risultato come positivo, neutro o negativo.
(0,01%-95% positivo, 95,01%-104,99% neutro, da 105% in su negativo)

	A	B	C	D	E	F	G	H	I	J
1	DB - file	STEM	FULL time	SUB time	Full time prop.	Sub time prop.	N°frasi	FULL	SUB	NO
2			(ms)	(ms)	(Toll. 5%)	(Toll. 5%)				
3	Oracle 9i JI									
4	Dp5-100	OFF	1234	3344	100%	100%	107	9	53	45
5	Dp5-100	ON	46	109	100%	100%	107	7	18	82
6	Dp5	OFF	6578	25499	100%	100%	400	53	242	105
7	Dp5	ON	219	1047	100%	100%	400	48	119	233
8										
9	Oracle 9i JO									
10	Dp5-100	OFF	1219	3359	98,78%	100,45%	107	9	53	45
11	Dp5-100	ON	47	109	102,17%	100,00%	107	7	18	82
12	Dp5	OFF	6109	25469	92,87%	99,88%	400	53	242	105
13	Dp5	ON	203	1047	92,69%	100,00%	400	48	119	233
14										
15	MySQL 4									
16	Dp5-100	OFF	3375	10140	273,50%	303,23%	107	9	53	45
17	Dp5-100	ON	172	734	373,91%	673,39%	107	7	18	82
18	Dp5	OFF	15078	62953	229,22%	246,88%	400	53	242	105
19	Dp5	ON	884	7796	403,65%	744,60%	400	48	119	233
20										
21	MySQL 5									
22	Dp5-100	OFF	9718	8847	787,52%	264,56%	107	9	53	45
23	Dp5-100	ON	203	594	441,30%	544,95%	107	7	18	82
24	Dp5	OFF	49942	52307	759,23%	205,13%	400	53	242	105
25	Dp5	ON	957	6290	436,99%	600,76%	400	48	119	233

(Tab. 5.2.4-a – parte 1 di 3 – tempi su DP)

In questa parte di tabella (Tab. 5.2.4-a – parte 1 di 3 – tempi su DP) si nota molto bene il confronto tra la versione di Oracle Java Inside e Oracle Java Outside che privilegia la seconda, come già anticipato nel capitolo 5.2.3.
Le versioni di MySQL 4 e 5 risultano pessime come nel caso dell'inserimento (cap. 5.2.1) e si nota che da una, la 4, all'altra, la 5, si guadagna tempo nella fase di SUB match, ma si perde nella fase di FULL match.

	A	B	C	D	E	F	G	H	I	J
1	DB - file	STEM	FULL time	SUB time	Full time prop.	Sub time prop.	N°frasi	FULL	SUB	NO
2			(ms)	(ms)	(Toll. 5%)	(Toll. 5%)				
27	MySQL 6									
28	Dp5-100	OFF	9912	8739	803,24%	261,33%	107	9	53	45
29	Dp5-100	ON	229	601	497,83%	551,38%	107	7	18	82
30	Dp5	OFF	50785	51123	772,04%	200,49%	400	53	242	105
31	Dp5	ON	1078	6216	492,24%	593,70%	400	48	119	233
38										
39	Postgres 8.2									
40	Dp5-100	OFF	1250	3504	101,30%	104,78%	107	9	53	45
41	Dp5-100	ON	109	172	236,96%	157,80%	107	7	18	82
42	Dp5	OFF	4172	26631	63,42%	104,44%	400	53	242	105
43	Dp5	ON	328	1098	149,77%	104,87%	400	48	119	233
44										
45	Postgres 8.3									
46	Dp5-100	OFF	1294	3506	104,86%	104,84%	107	9	53	45
47	Dp5-100	ON	140	219	304,35%	200,92%	107	7	18	82
48	Dp5	OFF	4219	25860	64,14%	101,42%	400	53	242	105
49	Dp5	ON	343	1215	156,62%	116,05%	400	48	119	233
50										

(Tab. 5.2.4-a – parte 2 di 3 – tempi su DP)

Proseguendo il discorso di prima, la versione 6 di MySQL risulta pessima come le altre e si nota lo stesso fatto di prima nel passaggio tra 5 e 6, ed anche tra 4 e 6.

Particolari le versioni di PostgreSQL 8.2 e 8.3 che dimezzano i tempi sulla query in modalità “stemming off”, ma li aumentano di una volta e mezza in modalità “stemming on”, nella parte di FULL match su DP5.

Differenze nel passaggio di versione 8.2 e 8.3, sempre per Postgres. Migliorie nella parte SUB con stemming off, mentre nella parte FULL e nella parte SUB con stemming on riscontriamo un peggioramento molto accentuato.

	A	B	C	D	E	F	G	H	I	J
1	DB - file	STEM	FULL time	SUB time	Full time prop.	Sub time prop.	N°frasi	FULL	SUB	NO
2			(ms)	(ms)	(Toll. 5%)	(Toll. 5%)				
50										
51	FireBird 2.1									
52	Dp5-100	OFF	11563	4110	937,03%	122,91%	107	9	53	45
53	Dp5-100	ON	156	219	339,13%	200,92%	107	7	18	82
54	Dp5	OFF	89281	26685	1357,27%	104,65%	400	53	242	105
55	Dp5	ON	657	1484	300,00%	141,74%	400	48	119	233
56										
57	MySQL 5 (LX)									
58	Dp5-100	OFF	3444	2655	279,09%	79,40%	107	9	53	45
59	Dp5-100	ON	94	55	204,35%	50,46%	107	7	18	82
60	Dp5	OFF	60121	22907	913,97%	89,83%	400	53	242	105
61	Dp5	ON	434	708	198,17%	67,62%	400	48	119	233
62										
63	Postgres 8.3 (LX)									
64	Dp5-100	OFF	1017	2718	82,41%	81,28%	107	9	53	45
65	Dp5-100	ON	65	76	141,30%	69,72%	107	7	18	82
66	Dp5	OFF	3996	20500	60,75%	80,40%	400	53	242	105
67	Dp5	ON	257	836	117,35%	79,85%	400	48	119	233

(Tab. 5.2.4-a – parte 3 di 3 – tempi su DP)

Poco da dire su FireBird 2.1 che risulta migliore delle versioni per Windows di MySQL, ma resta ancora nettamente inferiore agli altri DBMS.

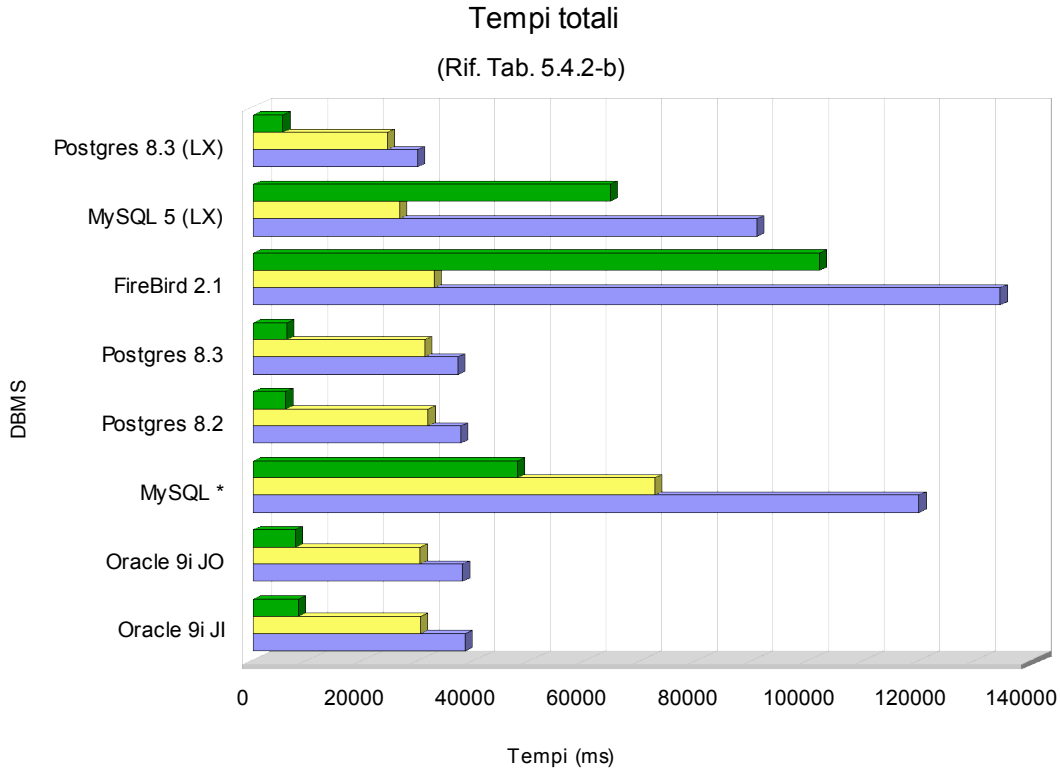
Notevoli le performance di MySQL 5.1 per Linux e di Postgres 8.3 sempre per Linux, che migliorano decisamente le loro versioni per Windows e quasi dimezzano i tempi nella fase SUB rispetto agli altri DBMS.

Da notare i tempi totali registrati per i vari DMBS:

DBMS	Tempo FULL	Tempo SUB	Tempo TOTALE
Oracle 9i JI	8077	29999	38076
Oracle 9i JO	7578	29984	37562
MySQL *(*)*	47444,33	72113,33	119557,67
Postgres 8.2	5859	31405	37264
Postgres 8.3	5996	30800	36796
Firebird 2.1	101657	32498	134155
MySQL 5.1 (LX)	64093	26325	90418
Postgres 8.3 (LX)	5335	24130	29465

(Tab. 5.2.4-b – Tempi totali su DP)

(*) MySQL è da considerarsi una media dei tempi delle versioni 4,5 e 6 per Windows.



(La prima serie del grafico corrisponde al tempo FULL, la seconda al tempo SUB, la terza al tempo TOTALE).

Dal grafico (Rif. Tab. 5.4.2-b) vediamo che:

- Le tre versioni di PostgreSQL sono le più veloci nelle query di Whole match (FULL), mentre Oracle risulta il secondo classificato (meglio JO che JI). Crisi per la versione MySQL 5.1 per Linux che viene superata dalle versioni per Windows; Firebird risulta il peggiore in assoluto.

- Le versioni di Postgres e MySQL per Linux, in ordine, sono le più veloci nella fase di Sub^{2}Match (SUB), come già notato in precedenza. Subito a ruota segue Oracle (JO poi JI) con distacchi di 3-6 secondi, seguito a sua volta dalle versioni Windows di PostgreSQL e da Firebird. Pessime le versioni per Windows di MySQL, ultime classificate con tempi raddoppiati rispetto agli altri DBMS.

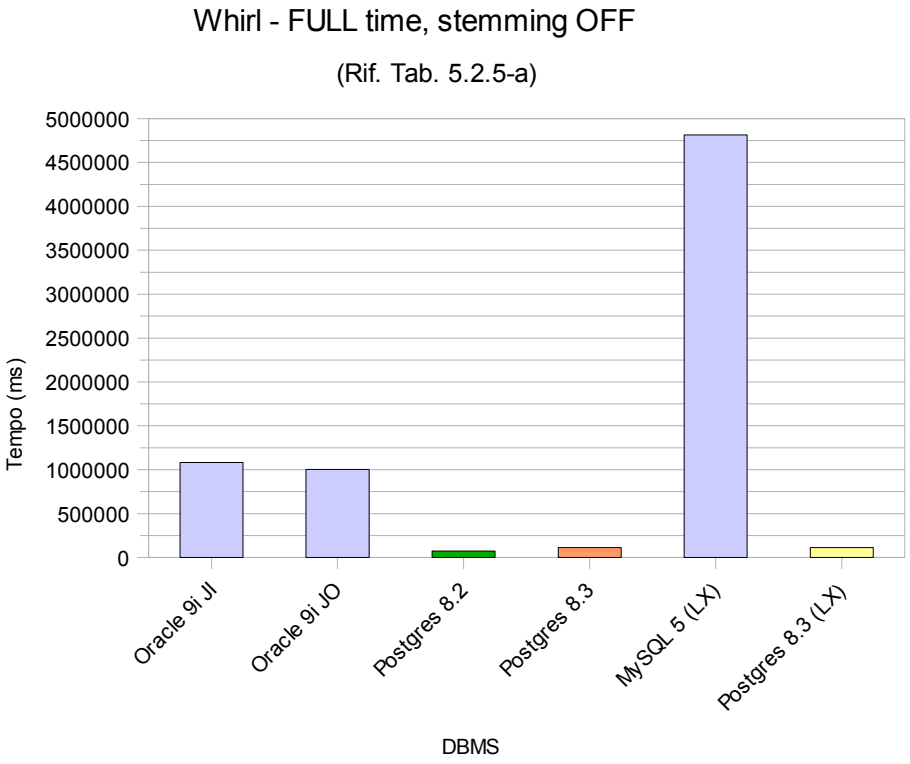
- PostgreSQL 8.3 per Linux è il più veloce nella query di Whole match (FULL), nella query di Sub^{2}Match (SUB) e quindi nel totale.

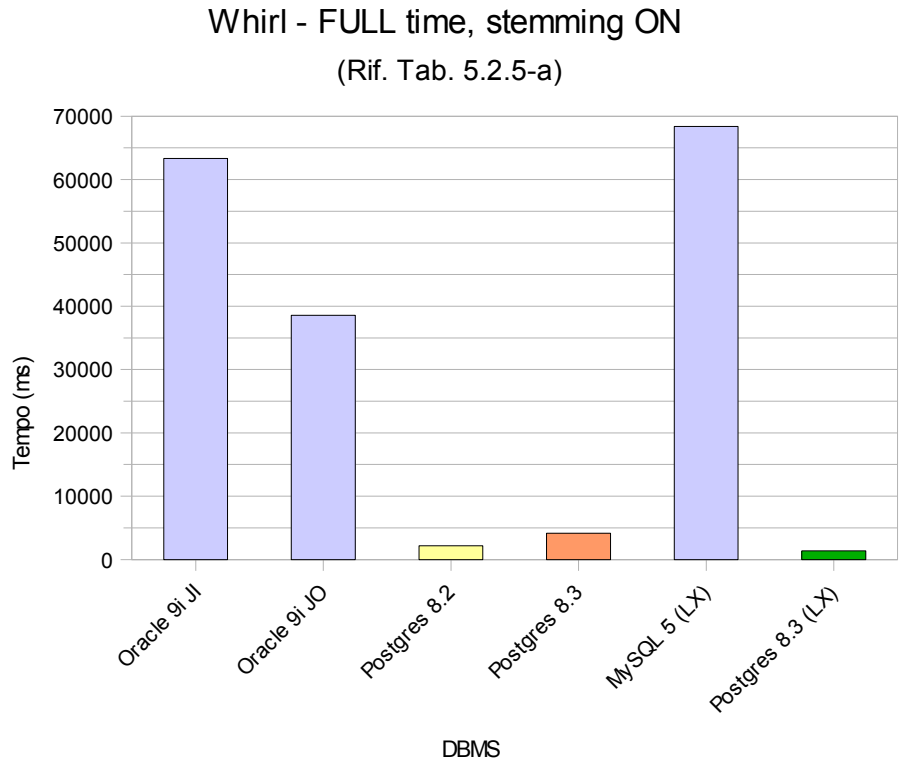
5.2.5 Il dataset Whirlpool

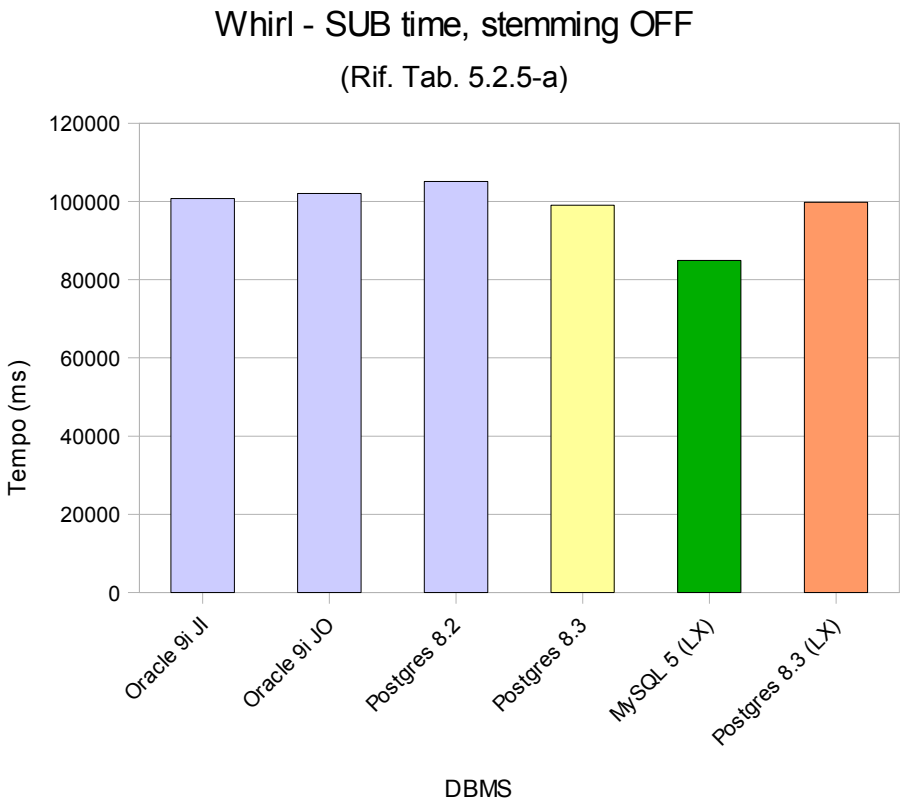
Riportiamo i risultati ottenuti con i vari DBMS sul dataset Whirlpool, utilizzando come riferimento sui tempi proporzionali il DBMS Oracle 9i JI, essendo questa la versione originale e necessitando obbligatoriamente di un punto di riferimento. Nei tempi proporzionali si utilizza una tolleranza del 5% per evidenziare il risultato come positivo, neutro o negativo.
(0,01%-95% positivo, 95,01%-104,99% neutro, da 105% in su negativo).
Evidenziamo anche la mancanza di test e quindi di risultati su alcuni DBMS che risultano nettamente peggiori in confronto a uno qualsiasi di questi selezionati.

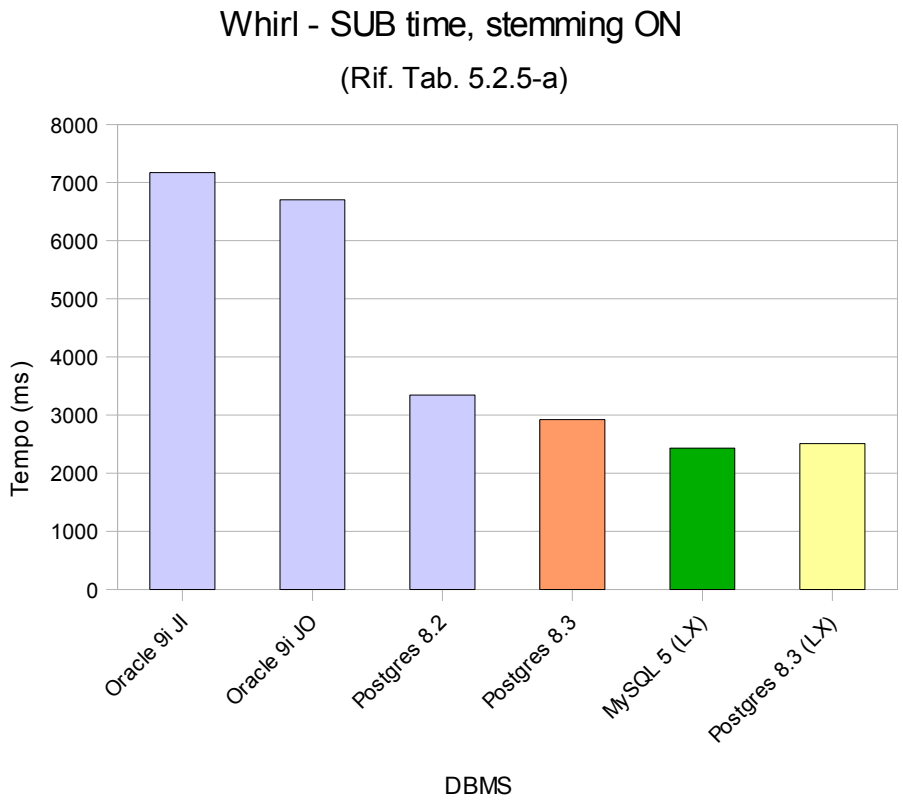
	A	B	C	D	E	F	G	H	I	J
1	**DB – file**	**STEM**	**FULL time**	**SUB time**	**Full time prop.**	**Sub time prop.**	**N° frasi**	**FULL**	**SUB**	**NO**
2					(Toll. 5%)	(Toll. 5%)				
3	Oracle 9i JI									
4	Whirl	OFF	1083641	100719	100%	100%	267	202	62	3
5	Whirl	ON	63329	7172	100%	100%	267	202	55	10
6										
7	Oracle 9i JO									
8	Whirl	OFF	1002594	102014	92,52%	101,29%	267	202	62	3
9	Whirl	ON	38546	6703	60,87%	93,46%	267	202	55	10
10										
11	Postgres 8.2									
12	Whirl	OFF	74734	105157	6,90%	104,41%	267	202	62	3
13	Whirl	ON	2171	3344	3,43%	46,63%	267	202	55	10
14										
15	Postgres 8.3									
16	Whirl	OFF	113861	99047	10,51%	98,34%	267	202	62	3
17	Whirl	ON	4148	2925	6,55%	40,78%	267	202	55	10
18										
19	MySQL 5 (LX)									
20	Whirl	OFF	4812953	84934	444,15%	84,33%	267	202	62	3
21	Whirl	ON	68395	2431	108,00%	33,90%	267	202	55	10
22										
23	Postgres 8.3 (LX)									
24	Whirl	OFF	112953	99806	10,42%	99,09%	267	202	62	3
25	Whirl	ON	1373	2509	2,17%	34,98%	267	202	55	10

(Tab. 5.2.5-a – Tempi su Whirl)


Whirl - FULL time, stemming OFF
(Rif. Tab. 5.2.5-a)
Tempo (ms)
5000000
4500000
4000000
3500000
3000000
2500000
2000000
1500000
1000000
500000
0
Oracle 9i JI
Oracle 9i JO
Postgres 8.2
Postgres 8.3
MySQL 5 (LX)
Postgres 8.3 (LX)
DBMS


Whirl - FULL time, stemming ON
(Rif. Tab. 5.2.5-a)
Tempo (ms)
70000
60000
50000
40000
30000
20000
10000
0
Oracle 9i JI
Oracle 9i JO
Postgres 8.2
Postgres 8.3
MySQL 5 (LX)
Postgres 8.3 (LX)
DBMS


Whirl - SUB time, stemming OFF
(Rif. Tab. 5.2.5-a)
Tempo (ms)
120000
100000
80000
60000
40000
20000
0
Oracle 9i JI
Oracle 9i JO
Postgres 8.2
Postgres 8.3
MySQL 5 (LX)
Postgres 8.3 (LX)
DBMS


Whirl - SUB time, stemming ON
(Rif. Tab. 5.2.5-a)
Tempo (ms)
8000
7000
6000
5000
4000
3000
2000
1000
0
Oracle 9i JI
Oracle 9i JO
Postgres 8.2
Postgres 8.3
MySQL 5 (LX)
Postgres 8.3 (LX)
DBMS

Analizzando le differenze tra i grafici in ordine di disposizione, notiamo che le grandezze sull'asse Y (dei tempi) variano di 2 ordini di grandezza tra il primo e i secondi due, e di un ordine di grandezza tra quest'ultimi ed il quarto. Con questo sottolineiamo che in una stima totale di tempi, sarà privilegiato il DBMS che ha ottenuto il miglior risultato nel primo grafico, senza esagerare negli altri, ovvero PostgreSQL 8.2. Ciò non denota però altre caratteristiche che potranno essere analizzate e dedotte solo da altre informazioni.

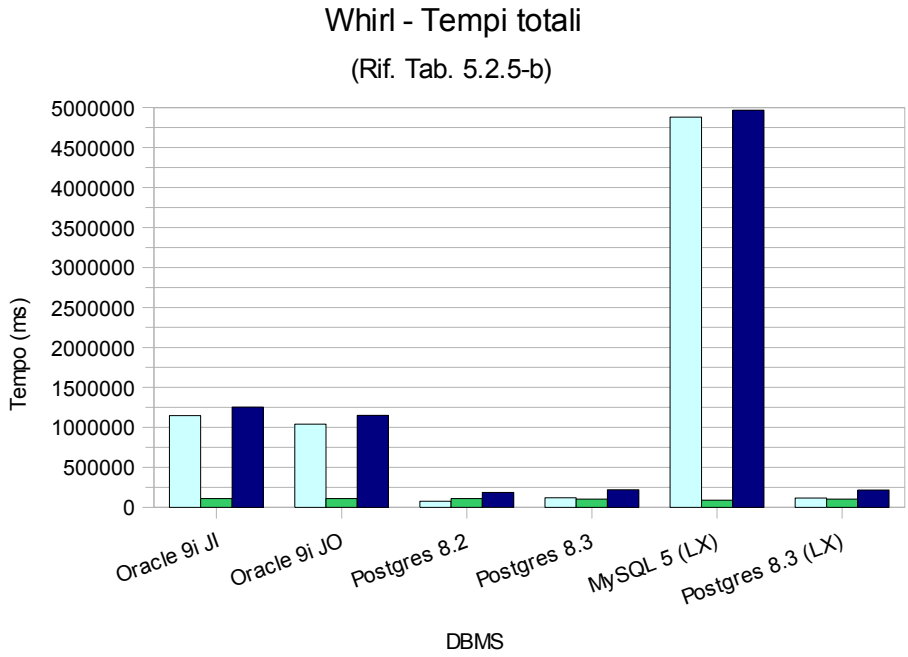
DBMS	Tempo FULL	Tempo SUB	Tempo TOTALE
Oracle 9i JI	1146970	107891	1254861
Oracle 9i JO	1041140	108717	1149857
Postgres 8.2	76905	108501	185406
Postgres 8.3	118009	101972	219981
MySQL 5 (LX)	4881348	87365	4968713
Postgres 8.3 (LX)	114326	102315	216641

(Tab. 5.2.5-b – Tempi totali su Whirl)

L'analisi appena descritta può essere ricavata anche dalla tabella sovrastante (Tab. 5.2.5-b), infatti, i tre DBMS Postgres pur essendo sbilanciati come "medagliere" (Postgres 8.2: 1°,5° - Postgres 8.3: 3°,1° - Postgres 8.3 (LX) - 2°, 2°) a favore della versione 8.3 per Windows, sono ordinati per Tempo FULL. L'influenza dei tempi FULL è ben visibile dal grafico sottostante relativo (Rif. Tab. 5.2.5-b).

Si noti anche che le differenze nei tempi tra le diverse versioni di PostgreSQL risultano appianate, se confrontate ai tempi impiegati dagli altri DBMS.

Da notare anche che la versione di MySQL 5 per Linux è risultata la più performante nella fase di Sub^{2}Match (SUB) come già visto in precedenza.



Dall'analisi globale su questo dataset, di dimensioni pari a 23 volte DP e 57 volte NV da cui emerge un fattore di precisione piuttosto alto, evidenziamo alcuni aspetti dei DBMS utilizzati e delle loro performance:

Oracle 9i JI:	Tempi mediamente buoni nella fase di FULL match. Tempi nella norma nella fase di SUB match. Leggermente peggio della versione JO.
Oracle 9i JO:	Tempi mediamente buoni nella fase di FULL match. Tempi nella norma nella fase di SUB match. Leggermente meglio della versione JI.
Postgres 8.2:	Il migliore nella fase di FULL match. Tempi nella norma nella fase di SUB match.
Postgres 8.3:	Tempi molto buoni nella fase di FULL match. Tempi leggermente inferiori alla norma nella fase di SUB match.
Postgres 8.3 (LX):	Tempi ottimi nella fase di FULL match. Tempi leggermente inferiori alla norma nella fase di SUB match.
MySQL 5 (LX):	Tempi pessimi nella fase di FULL match. Il migliore nella fase di SUB match.

Si conclude che per questo dataset il miglior DBMS da utlizzare è una versione a scelta tra le tre analizzate di PostgreSQL.

5.2.6 Il dataset NVidia

Riportiamo i risultati ottenuti con i vari DBMS sul dataset NVidia, utilizzando come riferimento sui tempi proporzionali il DBMS Oracle 9i JI, essendo questa la versione originale e necessitando obbligatoriamente di un punto di riferimento. Nei tempi proporzionali si utilizza una tolleranza del 5% per evidenziare il risultato come positivo, neutro o negativo.
(0,01%-95% positivo, 95,01%-104,99% neutro, da 105% in su negativo).
Evidenziamo anche la mancanza di test e quindi di risultati su alcuni DBMS che risultano nettamente peggiori in confronto a uno qualsiasi di questi selezionati.

	A	B	C	D	E	F	G	H	I	J
1	DB – file	STEMMING	FULL time	SUB time	Full time prop.	Sub time prop.	N° frasi	FULL	SUB	NO
2			(ms)	(ms)	(Toll. 5%)	(Toll. 5%)				
3	Oracle 9i JI									
4	Readme2313	OFF	4469	11860	100%	100%	743	580	92	71
5	Readme2313	ON	937	1016	100%	100%	743	571	51	121
6	Readme2880	OFF	5468	30421	100%	100%	846	556	158	132
7	Readme2880	ON	859	2094	100%	100%	846	546	85	215
8	Readme2960	OFF	5125	31860	100%	100%	856	554	163	139
9	Readme2960	ON	875	2188	100%	100%	856	545	86	225
10	Readme3123	OFF	5687	35609	100%	100%	890	546	187	157
11	Readme3123	ON	984	2282	100%	100%	890	543	98	249
12										
13	Oracle 9i JO									
14	Readme2313	OFF	5656	12188	126,56%	102,77%	743	580	92	71
15	Readme2313	ON	860	938	91,78%	92,32%	743	571	51	121
16	Readme2880	OFF	5450	30469	99,67%	100,16%	846	556	158	132
17	Readme2880	ON	985	2109	114,67%	100,72%	846	546	85	215
18	Readme2960	OFF	5119	32004	99,88%	100,45%	856	554	163	139
19	Readme2960	ON	969	2188	110,74%	100,00%	856	545	86	225
20	Readme3123	OFF	5153	36327	90,61%	102,02%	890	546	187	157
21	Readme3123	ON	875	2281	88,92%	99,96%	890	543	98	249
22										
23	Postgres 8.2									
24	Readme2313	OFF	3719	12214	83,22%	102,98%	743	580	92	71
25	Readme2313	ON	1187	964	126,68%	94,88%	743	571	51	121
26	Readme2880	OFF	4063	31897	74,31%	104,85%	846	556	158	132
27	Readme2880	ON	1125	2195	130,97%	104,82%	846	546	85	215
28	Readme2960	OFF	4078	32676	79,57%	102,56%	856	554	163	139
29	Readme2960	ON	1093	2240	124,91%	102,38%	856	545	86	225
30	Readme3123	OFF	4266	37186	75,01%	104,43%	890	546	187	157
31	Readme3123	ON	1078	2389	109,55%	104,69%	890	543	98	249
32										
33	Postgres 8.3									
34	Readme2313	OFF	4656	11252	104,18%	94,87%	743	580	92	71
35	Readme2313	ON	1250	965	133,40%	94,98%	743	571	51	121
36	Readme2880	OFF	4312	28728	78,86%	94,43%	846	556	158	132
37	Readme2880	ON	1234	2055	143,66%	98,14%	846	546	85	215
38	Readme2960	OFF	4361	31579	85,09%	99,12%	856	554	163	139
39	Readme2960	ON	1234	2272	141,03%	103,84%	856	545	86	225
40	Readme3123	OFF	4219	35425	74,19%	99,48%	890	546	187	157
41	Readme3123	ON	1187	2367	120,63%	103,72%	890	543	98	249

(Tab. 5.2.6-a parte 1 di 2 - Tempi su NV)

	A	B	C	D	E	F	G	H	I	J
1	DB – file	STEMMING	FULL time	SUB time	Full time prop.	Sub time prop.	N° frasi	FULL	SUB	NO
2			(ms)	(ms)	(Toll. 5%)	(Toll. 5%)				
42										
43	FireBird 2.1									
44	Readme2313	OFF	32891	15275	735,98%	128,79%	743	580	92	71
45	Readme2313	ON	3078	2988	328,50%	294,09%	743	571	51	121
46	Readme2880	OFF	39828	38671	728,38%	127,12%	846	556	158	132
47	Readme2880	ON	3172	6078	369,27%	290,26%	846	546	85	215
48	Readme2960	OFF	40187	39625	784,14%	124,37%	856	554	163	139
49	Readme2960	ON	3140	6172	358,86%	282,08%	856	545	86	225
50	Readme3123	OFF	40656	43937	714,89%	123,39%	890	546	187	157
51	Readme3123	ON	3141	6438	319,21%	282,12%	890	543	98	249
52										
53	MySql 6									
54	Readme2313	OFF	70391	18687	1575,10%	157,56%	743	580	92	71
55	Readme2313	ON	18000	7578	1921,02%	745,87%	743	571	51	121
56	Readme2880	OFF	83953	45696	1535,35%	150,21%	846	556	158	132
57	Readme2880	ON	16110	15519	1875,44%	741,12%	846	546	85	215
58	Readme2960	OFF	78577	48331	1533,21%	151,70%	856	554	163	139
59	Readme2960	ON	16387	16215	1872,80%	741,09%	856	545	86	225
60	Readme3123	OFF	87203	54108	1533,37%	151,95%	890	546	187	157
61	Readme3123	ON	18451	16942	1875,10%	742,42%	890	543	98	249
62										
63	MySql 5 (LX)									
64	Readme2313	OFF	36889	10074	825,44%	84,94%	743	580	92	71
65	Readme2313	ON	920	656	98,19%	64,57%	743	571	51	121
66	Readme2880	OFF	33252	30604	608,12%	100,60%	846	556	158	132
67	Readme2880	ON	1297	1518	150,99%	72,49%	846	546	85	215
68	Readme2960	OFF	35413	25875	690,99%	81,21%	856	554	163	139
69	Readme2960	ON	918	1590	104,91%	72,67%	856	545	86	225
70	Readme3123	OFF	37034	30229	651,20%	84,89%	890	546	187	157
71	Readme3123	ON	847	1660	86,08%	72,74%	890	543	98	249
72										
73	Postgres 8.3 (LX)									
74	Readme2313	OFF	6086	9162	136,18%	77,25%	743	580	92	71
75	Readme2313	ON	720	683	76,84%	67,22%	743	571	51	121
76	Readme2880	OFF	10851	23800	198,45%	78,24%	846	556	158	132
77	Readme2880	ON	853	1597	99,30%	76,27%	846	546	85	215
78	Readme2960	OFF	6182	25035	120,62%	78,58%	856	554	163	139
79	Readme2960	ON	845	1745	96,57%	79,75%	856	545	86	225
80	Readme3123	OFF	5969	28406	104,96%	79,77%	890	546	187	157
81	Readme3123	ON	1371	2356	139,33%	103,24%	890	543	98	249

(Tab. 5.2.6-a parte 2 di 2 - Tempi su NV)

Già analizzando la tabella sovrastante (Tab. 5.2.6-a, parte 1 e 2) notiamo, sopratutto se la vista è a colori, i punti di forza ed i punti deboli dei vari DBMS.
In particolare i DBMS:

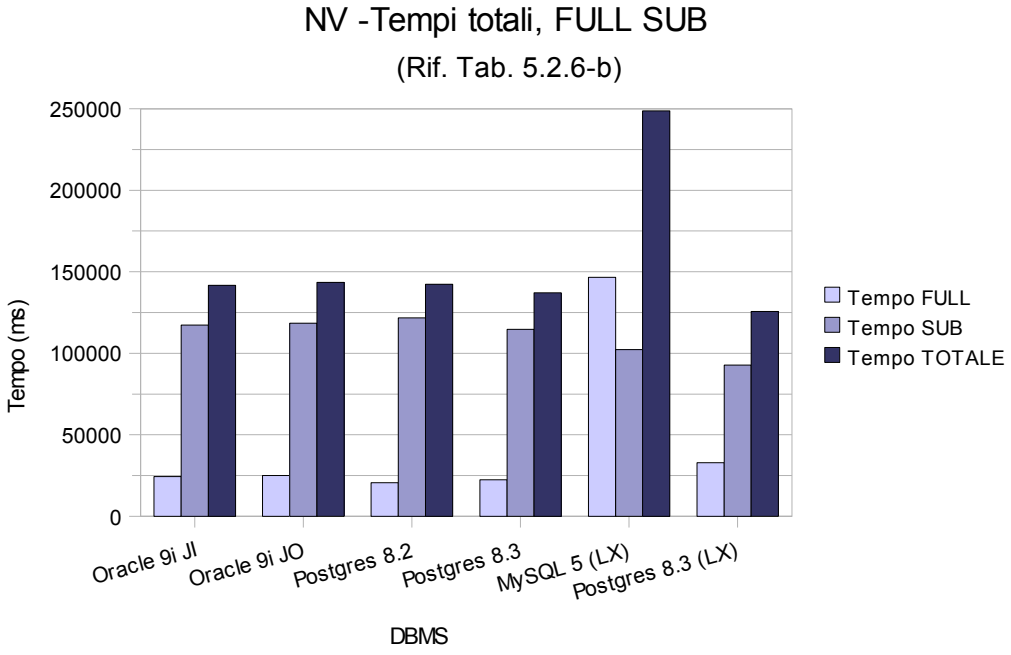
- FireBird 2.1 e MySQL 6 risultano disastrosi, infatti, Firebird impiega circa 7 volte il tempo base nella fase di full match con stemming off, circa 3 volte nella fase di full match con stemming on, circa 1.2 volte nella fase di sub match con stemming off e circa 3 volte nella fase di sub match con stemming on, mentre MySQL 6 ne impiega rispettivamente 15, 19, 1.5 e 7.

- MySQL 5 per Linux e PostgreSQL 8.3 per Linux risultano i migliori nella fase di SUB match, infatti, lavorano con tempi inferiori dal 15 al 36% in meno rispetto al tempo base, e, in pochi casi, migliorano anche i tempi nella fase di FULL match.
- Oracle JO, Postgres 8.2 e 8.3 ottengono molti risultati neutri e qualcuno positivo nella fase di SUB match, mentre nella fase di FULL match ottengono risultati altalenati, ma prevalentemente migliori di quelli base.

Analizziamo ora, in modo leggermente più dettagliato i risultati ottenuti, scartando i DBMS FireBird 2.1 e MySQL 6 risultati evidentemente i peggiori:

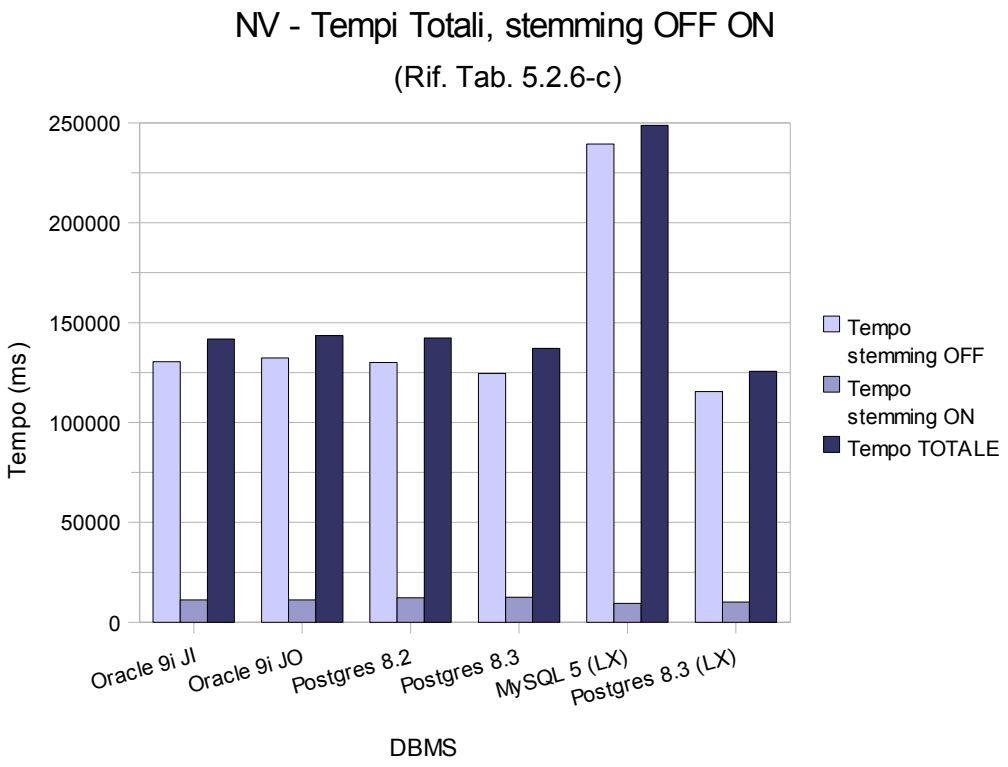
DBMS	Tempo FULL	Tempo SUB	Tempo TOTALE
Oracle 9i JI	24404	117330	141734
Oracle 9i JO	25067	118504	143571
Postgres 8.2	20609	121761	142370
Postgres 8.3	22453	114643	137096
MySQL 5 (LX)	146570	102206	248776
Postgres 8.3 (LX)	32877	92784	125661

(Tab. 5.2.6-b - Tempi totali FULL SUB su NV)



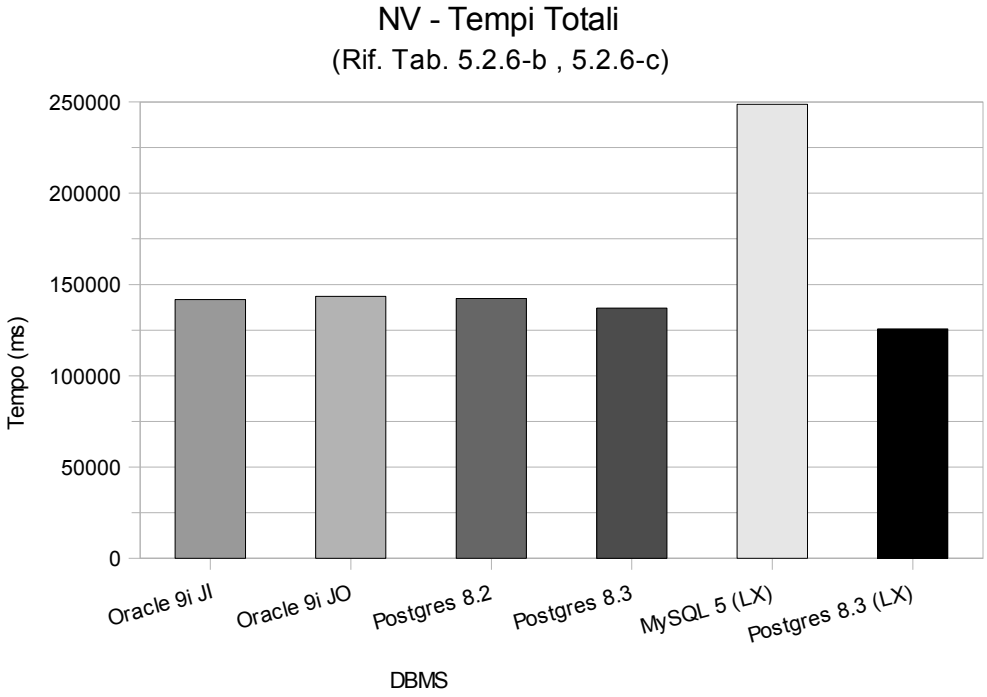
DBMS	Tempo stemming OFF	Tempo stemming ON	Tempo TOTALE
Oracle 9i JI	130499	11235	141734
Oracle 9i JO	132366	11205	143571
Postgres 8.2	130099	12271	142370
Postgres 8.3	124532	12564	137096
MySQL 5 (LX)	239370	9406	248776
Postgres 8.3 (LX)	115491	10170	125661

(Tab. 5.2.6-c - Tempi totali stemming OFF ON su NV)



Utilizzando le tabelle appena viste (5.2.6-b e 5.2.6-c) ed i relativi grafici, derivati dalla tabella generale (5.2.6-a), descriviamo il comportamento dei vari DBMS analizzati, prestando attenzione ai tempi totali che sono eguali sia per la tabella 5.2.6-b che per la tabella 5.2.6-c essendo sommatorie, rispettivamente per righe e per colonne, della stessa tabella (5.2.6-a), quindi degli stessi dati.

- Dalla tabella 5.2.6-b leggiamo che tutti i DBMS per Windows lavorano molto bene sulla query di Whole Match (FULL) a differenza di quelli per Linux che ottengono pessimi risultati (sopratutto MySQL 5 (LX)).
Registriamo, invece, una inversione di tendenza, già analizzata e studiata, per quanto riguarda la query di Sub^{2}Match (SUB), che porta PostgreSQL 8.3 (LX) e MySQL 5 (LX) rispettivamente al primo e secondo posto per velocità.
- Dalla tabella 5.2.6-c leggiamo che tutti i DBMS, ad eccezione di MySQL 5 per Linux, lavorano molto bene in modalità "stemming OFF" ed il migliore risulta essere Postgres 8.3 per Linux.
Per quanto riguarda la modalità "stemming ON", le versioni per Linux risultano le migliori, portando MySQL 5 (LX) al primo posto e PostgreSQL 8.3 (LX) al secondo.
- Dai risultati globali (visibili nel grafico sottostante Rif. Tab. 5.2.6-b , 5.2.6-c), cioè dal tempo totale delle tabelle (5.2.6-b e 5.2.6-c), emerge che i migliori DMBS su questo dataset in ambito generale sono, in ordine crescente di velocità, Oracle JI (che risulta comunque poco distaccato dalla versione Postgres 8.2 e Oracle JO), Postgres 8.3 e Postgres 8.3 per Linux.



5.2.7 Conclusioni generali sull'analisi prestazionale

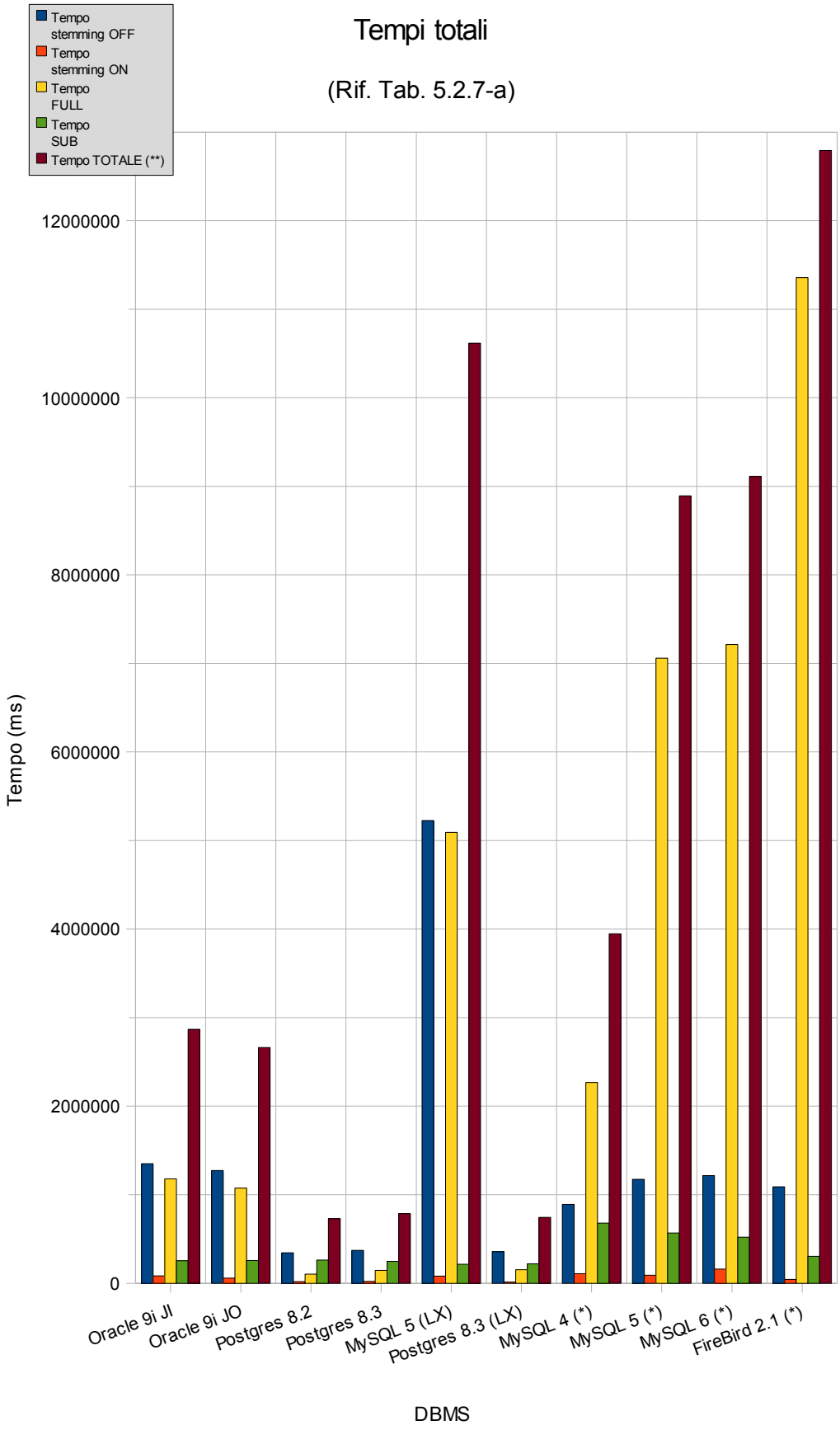
Considerando i capitoli 5.2.4, 5.2.5 e 5.2.6, relativi ai tre dataset dell'analisi prestazionale, ed escludendo la questione dell'inserimento, in quanto può essere svolta indipendentemente o, come spesso avviene per gli aggiornamenti, in fasi in cui l'utente non utilizza il software, analizziamo globalmente i risultati ottenuti.
Come si è visto fino ad ora, le fasi "conclusive" dei vari paragrafi altro non sono che riassunti e deduzioni generali derivate dai singoli dati, valide nello specifico e desumibili grossolanamente dal totale. In questo capitolo si vuole appunto fornire una breve, grossolana, ma utile descrizione globale delle performance dei singoli DBMS, facendo un riassunto di tutti i dati prelevati dai capitoli precedenti e contenuti nella tabella (Tab. A4.1 in appendice A4).

DBMS	Tempo stemming OFF	Tempo stemming ON	Tempo FULL	Tempo SUB	Tempo TOTALE *(**)*
Oracle 9i JI	1351514	83157	1179451	255220	2869342
Oracle 9i JO	1273130	57860	1073785	257205	2661980
Postgres 8.2	345547	19493	103373	261667	730080
Postgres 8.3	372319	21554	146458	247415	787746
MySQL 5 (LX)	5226384	81523	5092011	215896	10615814
Postgres 8.3 (LX)	356481	15286	152538	219229	743534
MySQL 4 *(*)*	889654	109466	2267686	680086	3946892
MySQL 5 *(*)*	1174083	91857	7059082	567495	8892517
MySQL 6 *(*)*	1217440	159573	7213851	520129	9110993
FireBird 2.1 *(*)*	1088700	44851	11356540	303959	12794050

(Tab. 5.2.7-a - Tempi totali, FULL-SUB , stemming OFF-ON)

(*) Evidenziamo subito il fatto che le stime contrassegnate da questo simbolo (e le relative caselle contrassegnate in rosso) sono state calcolate statisticamente da altri risultati in quanto i test non sono realmente stati effettuati sul 100% dei tre dataset in questione. Detto questo, i risultati sono presumibilmente molto attendibili ma non con certezza assoluta.

(**) Il "Tempo TOTALE" è la somma delle prime due colonne o delle seconde due, essendo esse stesse risultato di somme per righe o colonne sui medesimi dati.


Tempi totali
(Rif. Tab. 5.2.7-a)
Tempo stemming OFF
Tempo stemming ON
Tempo FULL
Tempo SUB
Tempo TOTALE (**)
Tempo (ms)
12000000
10000000
8000000
6000000
4000000
2000000
0
Oracle 9i JI
Oracle 9i JO
Postgres 8.2
Postgres 8.3
MySQL 5 (LX)
Postgres 8.3 (LX)
MySQL 4 (*)
MySQL 5 (*)
MySQL 6 (*)
FireBird 2.1 (*)
DBMS

Dall'analisi della tabella (5.2.7-a) e del relativo grafico desumiamo che:

- Nella fase “stemming OFF” i DBMS PostgreSQL risultano i migliori, in particolare la versione 8.2 per Windows risulta la prima, seguita dalla 8.3 per Linux e dalla 8.3 per Windows. MySQL 5.1 per Linux risulta la peggiore, con tempi cinque volte superiori alla media generale. Le altre versioni ottengono tutte risultati simili; nel gruppo leggermente meglio MySQL 4 degli altri.

- Nella fase “stemming ON” i DBMS PostgreSQL risultano i migliori, ma a differenza del punto precedente, la versione 8.3 per Linux risulta la prima, seguita dalle versioni 8.2 e 8.3 per Windows. Al quarto e quinto posto, con tempi raddoppiati rispetto ai primi, Firebird 2.1, molto buono in questo frangente, e Oracle JO. Infine, gli altri DBMS, tra cui Oracle JI, raddoppiano ulteriormente i tempi, quadruplicandoli rispetto ai primi; da segnalare come fanalino di coda MySQL 6 con tempi 1.5 volte superiori alle versioni 4 e 5.

- Nella fase “FULL” i DBMS PostgreSQL risultano i migliori, in particolare la versione 8.2 per Windows risulta la prima, seguita dalla 8.3, sempre per Windows, e dalla 8.3 per Linux. Firebird 2.1 risulta il peggiore, con tempi da due a cento volte superiori rispetto agli altri. Le altre versioni ottengono risultati molto differenti; in ordine di velocità Oracle JO, Oracle JI, MySQL 4 con tempi doppi rispetto ad Oracle, MySQL 5 per Linux con tempi quintupli rispetto ad Oracle ed infine MySQL 5 e 6 con tempi sette volte superiori sempre rispetto Oracle.

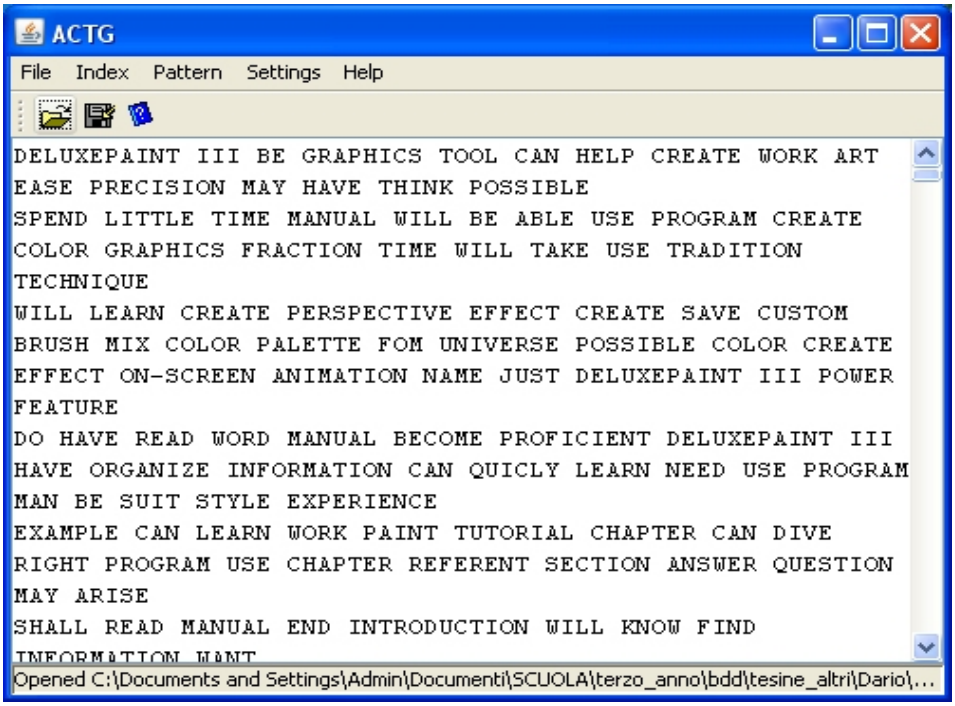
- Nella fase “SUB” i DBMS per Linux risultano i migliori, in particolare la versione MySQL 5 (LX) risulta la prima, seguita dalla versione 8.3 di Postgres sempre per Linux. Da notare le scarse differenze di tempi nei DBMS non contrassegnati da asterisco. Nel gruppo “degli asterischi”, invece, rileviamo un miglioramento di Firebird 2.1 che ottiene risultati simili ai primi, ma comunque inferiori, mentre risultano disastrose le versioni di MySQL per Windows che triplicano i tempi migliori.

- Nel considerare i tempi totali, le versioni di PostgreSQL, in ordine la 8.2, la 8.3 per Linux e la 8.3 per Windows, risultano nettamente le migliori.
  Questo fatto non è in se d'importanza rilevante perché deriva dalla capacità di questo DBMS di ottenere risultati sorprendenti nella fase “FULL” (o di Whole Match) a stemmer spento (“stemming OFF”), “settaggi” poco utilizzati in pratica che influiscono per più del 50% sui tempi totali.
  Ricordiamo comunque che questo DBMS è uno dei migliori, dei più veloci e dei più semplici da installare per il software in questione.

5.3 Analisi prestazionale comparata

L'analisi prestazionale comparata, come intuibile dal nome stesso, è una serie di test prestazionali del software in questione con un altro software (*) che presenta caratteristiche tali da poter essere confrontato. La comparazione, in questo caso, avviene tra il software di questo progetto (Extra [3], nella versione modificata e descritta in questo documento) e il software del progetto di Dario Gelmini [8] chiamato per semplicità ACGT.
Analizzando le tecniche implementative nel capitolo 1, si possono notare le differenze tra i due software, infatti nel caso di ACGT l'implementazione è svolta tramite i Suffix Tree e Suffix Array, mentre in Extra l'implementazione è basata su tutte le restanti tecniche viste. Già quindi dalla struttura del software è possibile comprendere le finalità dello stesso, ovvero, nel caso di ACGT una predisposizione per le sequenze genetiche, mentre nel caso di Extra una predisposizione per i documenti di testo ed in particolare manuali.

Forniamo ora alcune immagini di ACGT:



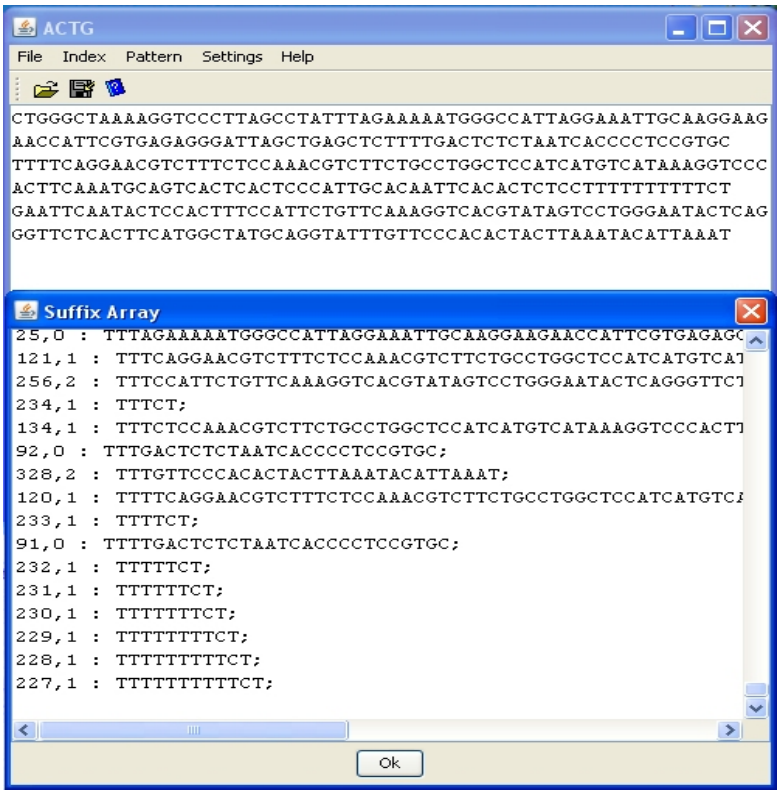
(Fig. 5.3-a – ACGT, main)

(*) .. od anche dello stesso software su diverse piattaforme (come nel caso di diversi DBMS)

Nella figura precedente (Fig. 5.3-a) notiamo la semplicità dell'interfaccia grafica, presa sicuramente da un editor di testi; all'interno dell'area di testo si può notare il file DP3 già passato dalla fase di stemming.

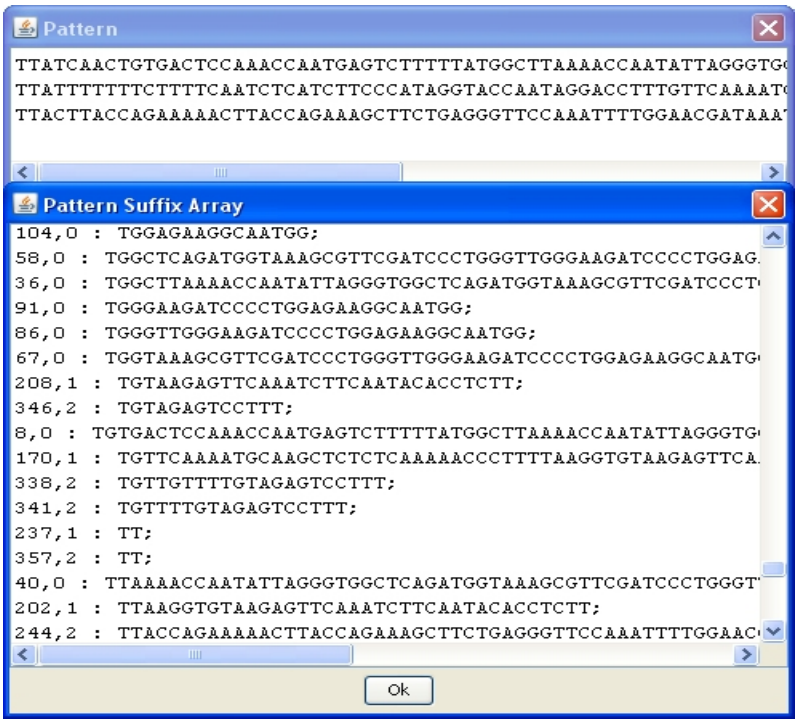
(Fig. 5.3-b – ACGT, SimbolTable)

Nella figura sovrastante (Fig. 5.3-b) vediamo l'insieme di simboli (SimbolTable) che compone il file DP3 già passato dalla fase di stemming.



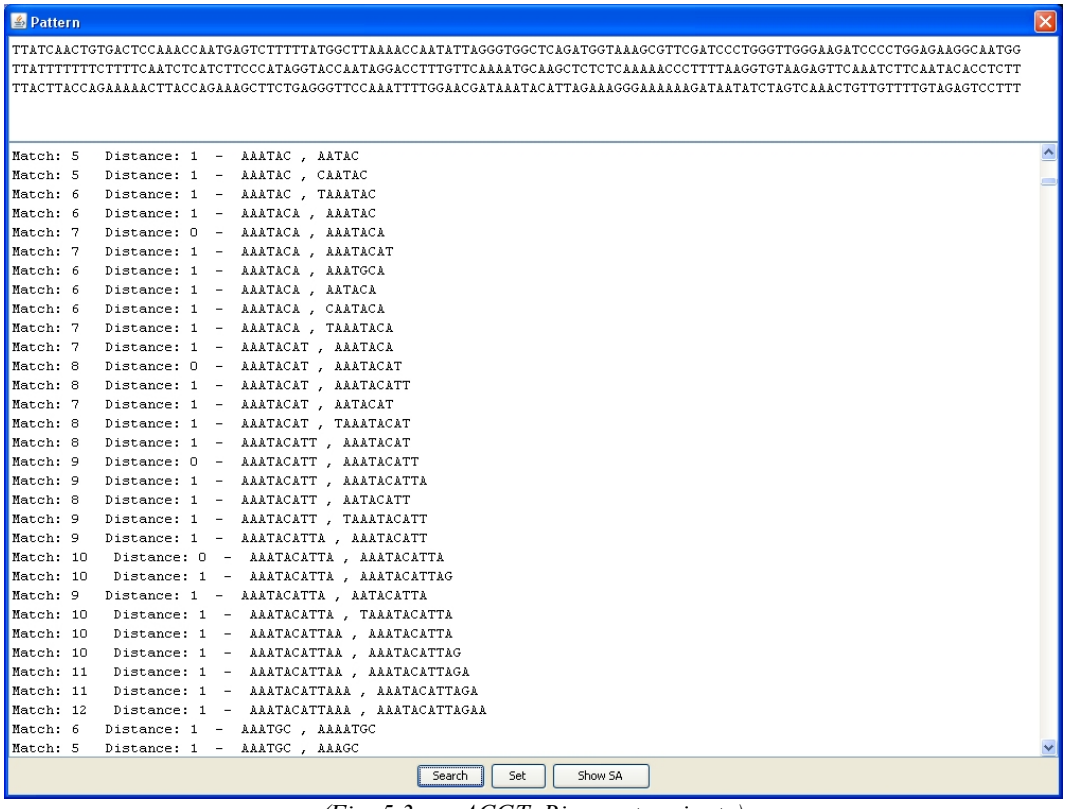
(Fig. 5.3-c – ACGT, Suffix Array del testo)

Nella figura sovrastante (Fig. 5.3-c) vediamo il testo del file DNA3 nell'area di testo in alto e il relativo Suffix Array nella finestra in basso. Nella figura sottostante (Fig. 5.3-d) vediamo le medesime cose, ma riguardanti il pattern.



(Fig. 5.3-d – ACGT, Suffix Array del pattern)

La fase conclusiva, cioè la ricerca del patter nel testo è riportata nell'immagine sottostante (Fig. 5.3-e). Si nota bene la struttura ad albero dei risultati nella parte inferiore, oltre al Match col relativo numero e la distanza espressa da un numero intero (***KINT***). Nella parte sopra il pattern completo.



(Fig. 5.3-e – ACGT, Ricerca terminata)

Facendo riferimento alle immagini riportiamo un esempio del funzionamento del software ACGT dopo averlo lanciato tramite il seguente comando:

java -cp ..\lib\jbcl.jar; -XX:NewSize=128m -XX:MaxNewSize=128m
-Xms256m -Xmx256m com.borland.samples.texteditor.TextEditClass

1. Caricare la Tabella dei Simboli, nuova o pre-generata, dal menù Settings.
2. Caricare un file (testo) nella finestra principale.
3. Creare il Suffix Array del file principale mediante la voce “create SA” del menu Index.

4. Caricare un file (pattern) dal menu “Pattern” ed impostarlo mediante l'apposito pulsante (set). Tale azione creerà automaticamente il Suffix Array del Pattern.
5. Impostare i parametri della ricerca dalla maschera delle opzioni (Options) nel menu Settings.
6. Eseguire la ricerca.

Note:

- Caricamento dei File - [8]

Nei passi 1, 2 e 3 i file possono essere immessi manualmente o caricati. Se il testo e/o il pattern vengono salvati dopo che ne e' stato costruito il Suffix Array, anche quest'ultimo verra' salvato. Saranno cosi' presenti i seguenti tre file:

nomefile = testo (o pattern) cosi' come lo si vede nella finestra.
nomefile.sym = testo (o pattern) codificato mediante la tabella dei simboli.
nomefile.sfx = Suffix Array del testo (o pattern)

da notare che mentre il primo file e' in formato ascii i seguenti due sono in formato binario (sequenze di interi). Al momento del caricamento di un file (testo o pattern) il programma ricerca la presenza degli ultimi due file e, se presenti, li carica immediatamente evitando così di dover ricalcolare la conversione in simboli ed il Suffix Array. Ovviamente ciò può funzionare solo se la tabella dei simboli è caricata correttamente, in caso contrario il programma genererà un errore ed eviterà il caricamento degli ultimi due file.

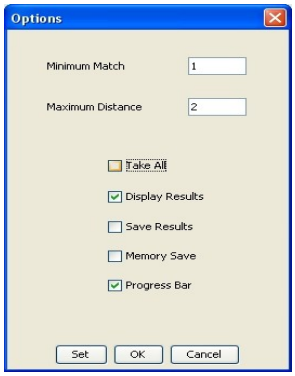
- Uso della Tabella dei Simboli - [8]

La tabella dei simboli nasce per poter processare indistintamente testo e genetica. Essa viene mantenuta in memoria come un insieme di sequenze di uno o più' caratteri ascii separate da un separatore predefinito ([spazio] o [CR]). Su file viene mantenuta la stessa struttura e quindi si presenta come un file ASCII ove i simboli sono separati da un [CR] e sono inoltre ordinati in modo lessicografico. Se volessimo utilizzare il programma per confrontare due testi generici dovremmo

prima concatenarli e quindi importarli come Tabella dei Simboli. Tale operazione consente di non perdere nessuno dei simboli contenuti nei file in esame. Il programma si occuperà quindi di separare i testi in simboli, ordinarli lessicograficamente ed infine eliminare i simboli replicati. È dunque possibile ottenere anche il semplice calcolo del numero di simboli (o parole) contenute all'interno di un testo semplicemente convertendolo in Tabella Simboli ed infine effettuando il conto delle parole.

- Velocità di ricerca -

Per evitare notevoli attese in fase di ricerca è consigliabile deselezionare l'opzione "Take All" nel menu "Settings" alla voce "Options". (Vedi fig. 5.3-f a lato)



(Fig. 5.3-f – ACGT, Options)

Passiamo ora alla scelta del DBMS per il confronto tra Extra e ACGT, infatti, come prima tabella, riportiamo le performance ottenute da Extra con i tre migliori DBMS a disposizione. Notiamo che i tempi proporzionali sono rapportati a Oracle JO (a differenza di tutti i precedenti che sono rapportati a Oracle JI) in quanto miglior DBMS del gruppo. Oltre a questo, notiamo anche l'esplicitazione dei parametri ***LMINSUB*** e ***KINT***.

	A	B	C	D	E	F	G	H	I
1	File	LMINSUB	KINT	SUB time	Sub time prop.	SUB time	Sub time prop.	SUB time	Sub time prop.
2				(ms)	(Toll. 5%)	(ms)	(Toll. 5%)	(ms)	(Toll. 5%)
3				Oracle 9i JO	Oracle 9i JO	Oracle 9i JI	Oracle 9i JI	Postgres 8.2	Postgres 8.2
4	AA1-60	10	1	35935	100,00%	36047	100,31%	36281	100,96%
5		15	1	20171	100,00%	20219	100,24%	20281	100,55%
6		25	2	22547	100,00%	22734	100,83%	23047	102,22%
7	AA1-120	10	1	237391	100,00%	237688	100,13%	237781	100,16%
8		15	1	198187	100,00%	197812	99,81%	198984	100,40%
9		25	2	203609	100,00%	203985	100,18%	205187	100,78%
10	DNA-3	3	1	21000	100,00%	21000	100,00%	21015	100,07%
11		6	1	21015	100,00%	20938	99,63%	20891	99,41%
12		10	1	20547	100,00%	21171	103,04%	20532	99,93%
13	DNA-4	10	1	294273	100,00%	295484	100,41%	295884	100,55%
14		15	1	287991	100,00%	288531	100,19%	290032	100,71%
15		25	2	268740	100,00%	269469	100,27%	270083	100,50%
16	DP3-5	4	1	1813	100,00%	2063	113,79%	1360	75,01%
17		6	1	311	100,00%	312	100,32%	203	65,27%
18		6	2	732	100,00%	735	100,41%	468	63,93%
19				*1634262*	*100,00%*	*1638188*	100,24%	*1642029*	100,48%

(Tab. 5.3-a – Extra e ACGT, scelta del DBMS)

Le performance degli altri DBMS testati non sono riportate in tabella essendo i tempi troppo elevati per un confronto serio. Riportiamo comunque, di seguito, un esempio della lentezza di alcuni di questi DBMS in confronto a quelli scelti:

Sul dataset DNA-3 Postgres 8.3 per Linux impiega rispettivamente 111230, 114594 e 115901 millisecondi, ovvero il 529%, il 545% ed il 564% in più di Oracle 9i JO, mentre MySQL 5 per Linux, impiega, sempre su DNA-3 rispetto ad Oracle JO, il 528%, il 539% ed il 549% di tempo in più.

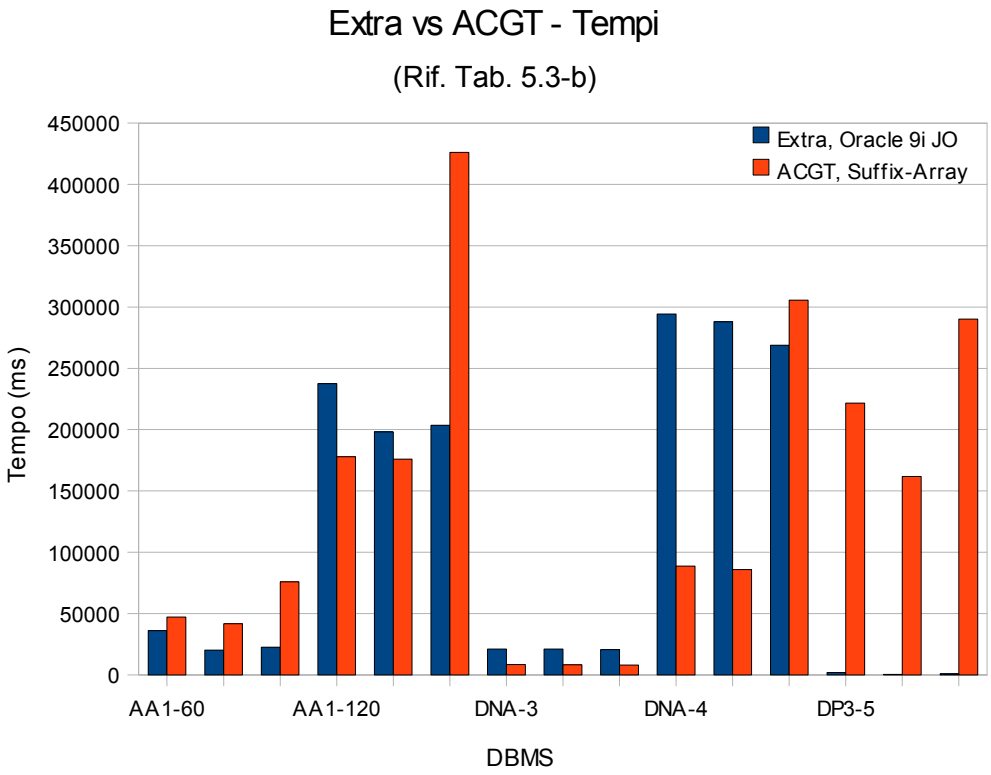
Dopo queste analisi, passiamo a quella prestazionale comparata vera e propria, infatti, riportiamo la tabella successiva (5.3-b) che contiene il confronto tra Extra, supportato dalla miglior versione del DBMS per il dataset in uso, e ACGT, "potenziato" dal punto di vista della memoria RAM da 128Mb a 256Mb per la JVM e da 256Mb a 512Mb per il programma vero e proprio.

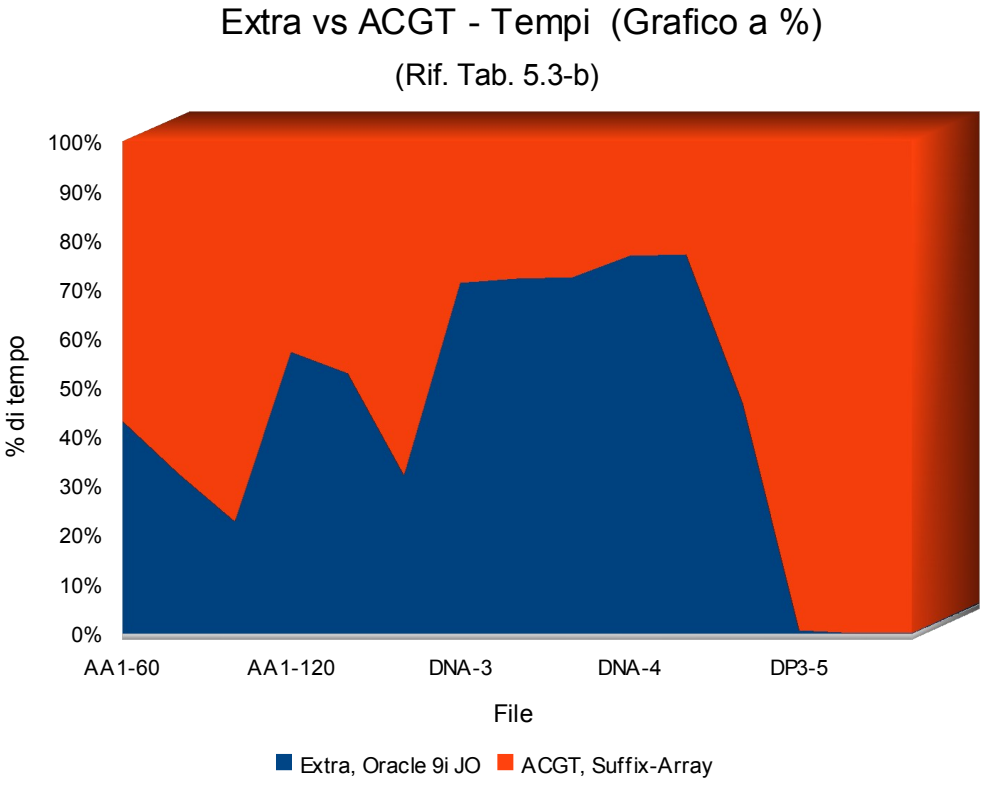
... -XX:NewSize=128m -XX:MaxNewSize=128m -Xms256m -Xmx256m ...
... -XX:NewSize=128m -XX:MaxNewSize=256m -Xms256m -Xmx512m ...
... -XX:NewSize=256m -XX:MaxNewSize=256m -Xms512m -Xmx512m ...

	A	B	C	D	E	F	G
1	File	LMINSUB	KINT	SUB time	Sub time prop.	SUB time	Sub time prop.
2				(ms)	(Toll. 5%)	(ms)	(Toll. 5%)
3				Oracle 9i JO	Oracle 9i JO	Suffix-Array	Suffix-Array
4	AA1-60	10	1	35935	100,00%	47110	131,10%
5		15	1	20171	100,00%	41797	207,21%
6		25	2	22547	100,00%	75969	336,94%
7	AA1-120	10	1	237391	100,00%	177875	74,93%
8		15	1	198187	100,00%	175922	88,77%
9		25	2	203609	100,00%	426172	209,31%
10	DNA-3	3	1	21000	100,00%	8453	40,25%
11		6	1	21015	100,00%	8109	38,59%
12		10	1	20547	100,00%	7843	38,17%
13	DNA-4	10	1	294273	100,00%	88765	30,16%
14		15	1	287991	100,00%	85781	29,79%
15		25	2	268740	100,00%	305547	113,70%
16	DP3-5	4	1	1813	100,00%	221578	12221,62%
17		6	1	311	100,00%	161688	51989,71%
18		6	2	732	100,00%	290094	39630,33%
19				*1634262*	*100,00%*	*2122703*	129,89%

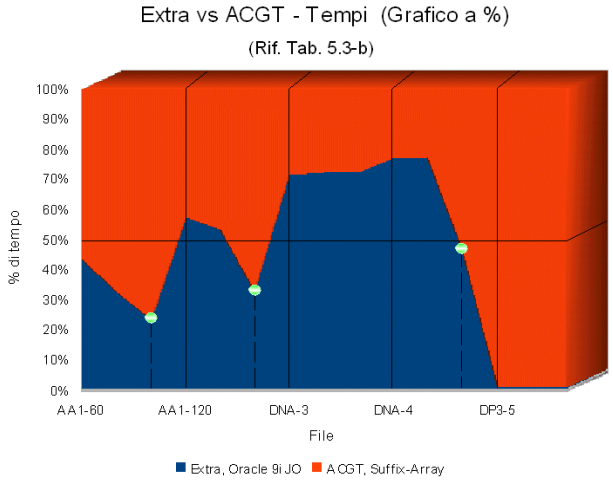
(Tab. 5.3-b – Extra e ACGT, analisi comparata)

Già dalla tabella si notano, disponendo di una vista a colori ancora meglio, i punti di forza di Extra rispetto ad ACGT e viceversa. Sui dataset di tipo "genetico" prevale ACGT, su quelli di tipo "testuale" prevale nettamente Extra. Rimandiamo per ora ulteriori approfondimenti alle pagine successive ed ai relativi grafici.


Extra vs ACGT - Tempi
(Rif. Tab. 5.3-b)
Tempo (ms)
450000
400000
350000
300000
250000
200000
150000
100000
50000
0
Extra, Oracle 9i JO
ACGT, Suffix-Array
AA1-60
AA1-120
DNA-3
DNA-4
DP3-5
DBMS


Extra vs ACGT - Tempi (Grafico a %)
(Rif. Tab. 5.3-b)
% di tempo
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
AA1-60
AA1-120
DNA-3
DNA-4
DP3-5
File
Extra, Oracle 9i JO
ACGT, Suffix-Array

Dai due grafici analizzati (Rif. Tab. 5.3-b), che rappresentano gli stessi dati, ma in modo differente, vediamo che ACGT si comporta meglio di Extra in prevalenza su sequenze genetiche, ma solo se il numero di simboli che le compone è abbastanza basso. Evidenziamo alcuni punti del grafico soprastante (*Extra vs ACGT – Tempi (Grafico a %) (Rif. Tab. 5.3-b)*).



Indichiamo con linee continue i punti d'interruzione tra un dataset e l'altro, con linee tratteggiate i punti problematici per ACGT e con pallini i punti precisi dell'inversione di tendenza.

Guardando il complesso dei grafici e dei dati, riscontriamo che solo nel caso del dataset DNA-3 riscontriamo una totale prevalenza di ACGT su Extra, mentre, sempre nell'ambito di dataset "genetici", all'aumentare della dimensione del problema, ovvero aumentando ***LMINSUB***, ma soprattutto aumentando ***KINT*** da 1 a 2, la performance di ACGT degrada esponenzialmente.

Se analizziamo poi la parte riguardante il dataset di tipo "testuale" (DP) vediamo che l'approccio fornito da ACGT risulta clamorosamente disastroso anche rispetto al peggiore dei DBMS utilizzati con Extra, infatti, i tempi di ricerca, rispetto ad Oracle JO, vengono moltiplicati rispettivamente di una fattore 122, 520 e 396, che espresso in percentuale precisa (Visibile nella tabella 5.3-b) risulta essere 12221,62%, 51989,71% e 39630,33%.

Volendo considerare una stima sul totale, osserviamo la tabella sottostante (Tab. 5.3-c) ed analizziamone i dati:

	Extra	**ACGT**	**Somma**
Genetico	1631406	1449343	3080749
Testuale	2856	673360	676216
Somma	1634262	2122703	3756965

(Tab. 5.3-c – Totale, genetico,testuale X Extra,ACGT)

Le quatto caselle, risultato dell'incrocio delle righe “Genetico” e “Testuale” con le colonne “Extra” e “ACGT”, rappresentano le somme di tutti i dati relativi, a loro volta ulteriormente sommati nelle caselle terminali “Somma” per categoria, e nell'incrocio finale (in basso a destra) per totale.

Da questi incroci vediamo molto bene che:

- ACGT è più veloce di Extra nella parte genetica dell'11% circa
  (1,125 volte tanto)
- Extra è più veloce di ACGT nella parte testuale del 99,58% circa
  (236 volte tanto)
- Nel totale Extra è più veloce di ACGT di circa il 23% (1,299 volte tanto)
  (Il tempo sul totale di Extra è il 43,5% quello di ACGT è il 56,5%)
- I dataset “genetici” influiscono per l'82% sui risultati
  (DP influisce solo per il 18%)

Considerando una parità d'influenza dei dataset (50% - 50%) otteniamo che Extra utilizza solo il 21,96% del tempo totale, mentre ACGT il 78,04%, ovvero Extra risulterebbe quasi 4 volte più veloce su un qualsiasi insieme misto, ma bilanciato.

In generale, se dovessimo ricercare sequenze genetiche con parametri che porterebbero ad una piccola mole di dati converrebbe utilizzare ACGT, mentre in tutti gli altri casi converrebbe utilizzare Extra. Per evidenziare meglio questo fatto riportiamo la tabella sottostante (5.3-d) che aiuta nella scelta migliore del software in base ai parametri. Solo nel caso il Dataset sia di dimensioni pari od inferiori a AA1-120 ed il valore ***KINT*** sia minore o uguale ad 1 è consigliabile utilizzare ACGT. Ricordiamo che i dataset AA1-120 e AA1-60 differiscono solo per la lunghezza ed il numero delle frasi, 27 da 120 contro 54 da 60 appunto, invece il numero di simboli è equivalente, ovvero 3190, ma nel caso di AA1-60 la mole del dataset elaborato dai software risulta maggiore di quella di AA1-120.

Tipo Dataset	Mole Dataset	Valore *KINT*	Software migliore
Genetico	<= AA1-120	<= 1	ACGT
Genetico	0 - ∞	>= 2	Extra
Genetico	>= AA1-60	0 - ∞	Extra
Testuale	0 - ∞	0 - ∞	Extra

(Tab. 5.3-d – Scelta parametrica tra Extra e ACGT)

Conclusioni e sviluppi futuri

Partendo dal caso di studio, seguendo il percorso descritto in questo documento e arrivando infine all'analisi sperimentale comparata, siamo giunti alla conclusione di questo progetto, in cui sono stati raggiunti i seguenti risultati:

- ✔ Sono stati studiati e analizzati algoritmi innovativi per la ricerca di similarità tra frasi, ovvero per individuare, data una frase di query, le frasi di riferimento più simili ad essa, quelle cioè la cui distanza dalla frase cercata non supera una data soglia; Questi algoritmi sono il Whole Match, il Sub^{2}Match e l'approccio basato su Suffix Tree e Suffix Array;

- ✔ Sono stati studiati e analizzati algoritmi per il miglioramento della ricerca di similarità tra frasi, in particolare lo Stemming;

- ✔ Sono state studiate, implementate e analizzate tecniche per la connessione ai vari DBMS, tra cui JDBC e SQLJ, aumentando così anche la conoscenza del linguaggio Java;

- ✔ Sono stati studiati, installati e analizzati numerosi DBMS, sia su sistemi Windows che su sistemi GNU/Linux;

- ✔ È stata sviluppata ed implementata la portabilità su tutti i DBMS studiati ed installati e sui sistemi operativi visti, ampliando così il progetto software originale;

- ✔ È stata sviluppata ed implementata un'interfaccia grafica che permette all'utente di utilizzare i DBMS supportati in tutta facilità, senza più dover creare manualmente le tabelle ed i dati necessari al funzionamento del software, oltre alla possibilità di settare e utilizzare parametri di default quali user, password, tipo di DBMS, etc.., ed avere informazioni su quest'ultimi;

- ✔ Sono state effettuate e commentate numerose analisi prestazionali svolte con le differenti tecniche, i differenti DBMS e i differenti sitemi operativi visti ed implementati, evidenziando nel dettaglio pregi e difetti di ogni singola prova;

- ✔ È stato studiato e analizzato un nuovo software che utilizza tecniche alternative per risolvere il problema del Sequence Matching, il che ha portato ad un'analisi prestazionale comparata, dettagliatamente descritta, tra i diversi algoritmi.

Essendo questo software stato progettato originariamente sia per ricerca scientifica che per il campo professionale (si pensi ad esempio ai team di traduzione impiegati da Logos [3]), l'introduzione delle caratteristiche sviluppate in questo progetto ne porta una migliore integrazione sia dal punto di vista dell'estensibilità che dal punto di vista dell'usabilità.

Per quanto riguarda gli sviluppi futuri, sarebbe interessante:

- ✔ Sviluppare il software per l'utilizzo con altre lingue, infatti, se a livello di traduzione la copertura è praticamente totale in quanto non dipende dagli algoritmi utilizzati, per quanto riguarda la Translation Memory, ma sopratutto la fase di Stemming e Word Sense Disambiguation, l'unica lingua supportata è l'inglese;

- ✔ Ottimizzare ulteriormente le prestazioni della ricerca, lavorando anche su procedure da inserire nei vari DBMS (ad esempio in linguaggio Pg/Sql), oppure utilizzando X-query ovvero query XML, che, in alcuni DBMS, aumentano le prestazioni rispetto a quelle classiche utilizzate;

- ✔ Implementare un metodo basato sulle tecniche viste nel software in questione, specifico per l'ambito di ricerca genetica, essendo risultato particolarmente performante rispetto ad altri software appositamente studiati per questo scopo.

Per concludere, il lavoro svolto in questa tesi ha portato ad una serie d'incontri con esperienze già maturate nell'ambito universitario: Se le tecniche di Sequence Matching hanno portato ad una maggiore conoscenza e dimestichezza con quelle in parte già viste durante il corso di Algoritmica II, il discorso di portabilità, usabilità ed estensione si può riscontrare nel corso di Metodologie per il progetto del Software e, sempre nell'ambito di questo discorso, la possibilità nell'utilizzo di diverse piattaforme è dovuta sia al corso di Sistemi Operativi sia alla portabilità intrinseca di Java, linguaggio appreso in Linguaggi per la programmazione ad oggetti. Per proseguire, lo sviluppo di questa tesi (e soprattutto il Capitolo 3) ha seguito diversi aspetti appresi in Ingegneria del Software, ribaditi anche in Informatica industriale e viceversa. Infine, lo studio, l'utilizzo, la modifica e l'installazione di DBMS, di query SQL, etc.., è stato consentito soprattutto dal corso di Basi di Dati del prof. Riccardo Martoglia, relatore di questa tesi.

Parte III

Appendici (A)

A – Appendici

Nelle pagine successive di questa appendice, presentiamo il codice sorgente del software, tabelle di analisi prestazionali, note e codice sorgente scritto, ma non utilizzato.

Cercando di presentare nel miglior modo possibile tutto questo, si invita un'eventuale persona interessata alle parti tecniche del software ad accedere al materiale in formato elettronico, essendo questo più indicato alla presentazione ed alla copia di codici sorgenti.

Le versioni riportate, pur essendo praticamente identiche a quelle per Linux, sono quelle utilizzate su Windows, pertanto potrebbero essere presenti caratteri non UTF-8.

Nel compact disc originariamente allegato a questo documento (e viceversa) si trova il software aggiornato e completo pronto per la compilazione su entrambi i sistemi operativi visti. Non volendo riportare ulteriori dettagli, si consiglia la lettura degli eventuali file “readme” all'interno di ogni cartella del compact disc.

A.1 Edit Distance (Java – C)

Versione JAVA:

```
package simSearch;

import java.util.*;

public abstract class Distance
{
 // Calcola la Edit Distance (distanza in parole) tra due stringhe
  public static int wordEditDistance (String s, String t)
   {
    int n = DBUtility.wordLen (s);     // numero parole di s
    int m = DBUtility.wordLen (t);     // numero parole di t

                        // casi particolari
    if (n == 0) { return m; }
    if (m == 0) { return n; }

    int d[][] = new int[n+1][m+1];  // matrice di costi
    int i;     // itera in s
    int j;     // itera in t
    String s_i;  // iesima parola di s
    String t_j;  // jesima parola di t
    int cost;  // costo

    ArrayList sv = DBUtility.vectorFromString (s);
    ArrayList tv = DBUtility.vectorFromString (t);

    for (i = 0; i <= n; i++)       // inizializzazione prima colonna
         {d[i][0] = i;}
    for (j = 0; j <= m; j++)       // inizializzazione prima riga
         {d[0][j] = j; }

    for (i = 1; i <= n; i++)         // calcolo matrice dei costi
         {s_i = (String) sv.get (i - 1);
          for (j = 1; j <= m; j++)
               {t_j = (String) tv.get (j-1);
                if (s_i.compareTo(t_j)==0) { cost = 0; }
                else { cost = 1; }
                d[i][j] = DBUtility.minimum
                      (d[i-1][j]+1, d[i][j-1]+1, d[i-1][j-1] + cost);
               }
         }

    return d[n][m];
   }

// Calcola la Edit Distance (distanza in parole) tra due stringhe (versione ottimizzata per calcolo su diagonali)

  public static int wordEditDistanceDiag (String s, String t, double maxDist)
   {
    int n = DBUtility.wordLen (s);     // numero parole di s
    int m = DBUtility.wordLen (t);     // numero parole di t

    final int VUOTO = -10;          // valore di inizializzazione array

                        // casi particolari
    if (n == 0 && m == 0) { return -1; }
    if (n == 0) { return m; }
    if (m == 0) { return n; }

    int[][] d = new int[n+1][m+1];  // matrice di costi

                            // array dei punti di partenza degli stroke
    int startx[] = new int [m+1];
    int starty[] = new int [n+1];

    int stroke=0;                // valore dello stroke di partenza
```

```
int i;                    // variabili per cicli
int j;
int k;

int posx = 0;             // variabili per estremi di calcolo
int posy = 1;
int stopx = m;
int stopy = n;

ArrayList sv = DBUtility.vectorFromString (s);
ArrayList tv = DBUtility.vectorFromString (t);

                          // inizializzazione array
for (i = 0; i <= n; i++)
   {for (j = 0; j <= m; j++)
       {d[i][j] = VUOTO;}
   }

for (i = 0; i <= n; i++)
     {starty[i] = 0;}
for (j = 0; j <= m; j++)
     {startx[j] = 0;}

                          // calcolo della matrice per stroke diagonali
for (stroke=0; (stroke<=Math.round(maxDist*m)) ; stroke++)
   {
                    // calcolo stroke diagonali della matrice triangolare superiore
    k = posx;
    while (k<=stroke && k<=stopx)
      {
      i = startx[k];
      j = k + startx[k];

      do
         {if (i==n && j==m) return stroke;
         d[i][j] = stroke;

         i++;
         j++;
         if (i>n)
            {posx=k+1;
            stopy=-1;
            break;
            }
         if (j>m)
            {stopx=k-1;
            break;
            }
         }
      while (((String) sv.get(i-1)).compareTo((String) tv.get(j-1)) == 0
           || (d[i-1][j] == stroke-1) || (d[i][j-1] == stroke-1));

      startx[k] = i;
      k++;
     }
                    // calcolo stroke diagonali della matrice triangolare inferiore
    k = posy;
    while (k<=stroke && k<=stopy)
      {
      i = k + starty[k];
      j = starty[k];

      do
         {if (i==n && j==m) return stroke;
         d[i][j] = stroke;

         i++;
         j++;
         if (i>n)
            {stopy=k-1;
            break;
            }
         if (j>m)
            {posy=k+1;
```

```
                stopx=-1;
                break;
               }
             }
          while (((String) sv.get(i-1)).compareTo((String) tv.get(j-1)) == 0
               || (d[i-1][j] == stroke-1) || (d[i][j-1] == stroke-1));

          starty[k] = j;
          k++;
         }

      }

   return (-1);
  }
}
```

Versione C:

```
#include <stdio.h>
#include <string.h>
#include <malloc.h>

int nofspace(char *s, int *maxwl)  // numero di parole (o spazi) di una frase
  {
            int slen = strlen(s);
            int wind = 0;
            *maxwl = 0;
            int wlen = 0;
            for(int i=0;i<slen;i++)
            {
                    if(s[i]==' ')
                    {
                        wind++;
                        if(wlen>*maxwl)  *maxwl=wlen;
                        wlen = 0;
                    }
                    else
                            wlen++;
            }
            if(wlen>*maxwl)  *maxwl=wlen;

          return wind;
  }

void vectorFromString(char *s, char **stok) // trasforma una frase in un vettore di parole
  {
          int k=0;
          stok[k] = "";
          int stlen =  strlen(s);
          for(int i=0;i<stlen;i++)
          {
                    if(s[i]!=' ')
                            stok[k] += s[i];
                    else
                    {       k++;
                            stok[k] = "";
                    }
          }
  }

bool comp(char *s,char *t) // Ritorna false se le stringhe sono diverse true altrimenti
{
          int sb = strlen(s);
          if(sb!=strlen(t)) return false;
          for(int i=0;i<sb;i++)
                    if(s[i]!=t[i]) return false;
          return true;
}

int minimum (int a, int b, int c) // ritorna il minimo tra 3 valori
   {
    int min;
```

```
   min = a;
   if (b < min) { min = b; }
   if (c < min) { min = c; }

   return min;
  }

int wordEditDistance (char *s, char *t) // word edit distance
 {
  int sn_maxwl = 0;
              int tm_maxwl = 0;
              int n = nofspace(s,&sn_maxwl);
              int m = nofspace(t,&tm_maxwl);
                      // casi particolari
  if (n == 0) { return m; }
  if (m == 0) { return n; }

  int **d = (int **)malloc((n+1)*sizeof(int));
  for (int i=0;i<n+1;i++)
                d[i] = (int *)malloc((m+1)*sizeof(int));

  int i;    // itera in s
  int j;    // itera in t
  char *s_i;  // iesima parola di s
  char *t_j;  // jesima parola di t
  int cost;  // costo

  //in c
  char **sv = (char **)malloc(n*sizeof(char));
  for(int i=0;i<n;i++)
                sv[i] = (char *)malloc(sn_maxwl*sizeof(char));
  vectorFromString (s,sv);
  //ArrayList sv = DBUtility.vectorFromString (s);

  // in c
  char **tv = (char **)malloc(m*sizeof(char));
  for(int i=0;i<m;i++)
                tv[i] = (char *)malloc(tm_maxwl*sizeof(char));

  vectorFromString (t,tv);
  //ArrayList tv = DBUtility.vectorFromString (t);

  for (i = 0; i <= n; i++)      // inizializzazione prima colonna
       {d[i][0] = i;}
  for (j = 0; j <= m; j++)      // inizializzazione prima riga
       {d[0][j] = j; }

  for (i = 1; i <= n; i++)        // calcolo matrice dei costi
       {s_i = sv[i - 1];
        for (j = 1; j <= m; j++)
             {t_j = tv[j-1];
              if (comp(s_i,t_j)){ cost = 0; }
              else { cost = 1; }
              d[i][j] = minimum(d[i-1][j]+1, d[i][j-1]+1, d[i-1][j-1] + cost);
             }
       }

  return d[n][m];
 }



int wordEditDistanceDiag (char *s, char *t, double maxDist) // word edit distance diagonale
 {
             //java
             //int n = DBUtility.wordLen (s);    // numero parole di s
             //int m = DBUtility.wordLen (t);    // numero parole di t
               int sn_maxwl = 0;
               int tm_maxwl = 0;
               int n = nofspace(s,&sn_maxwl);
               int m = nofspace(t,&tm_maxwl);

             //
  int VUOTO = -10;         // valore di inizializzazione array
```

```
                    // casi particolari
if (n == 0 && m == 0) { return -1; }
if (n == 0) { return m; }
if (m == 0) { return n; }

// int[][] d = new int[n+1][m+1];  // matrice di costi // java
int **d = (int **)malloc((n+1)*sizeof(int));
for (int i=0;i<n+1;i++)
            d[i] = (int *)malloc((m+1)*sizeof(int));

// array dei punti di partenza degli stroke
//int startx[] = new int [m+1];
//int starty[] = new int [n+1];
int *startx = (int *)malloc((m+1)*sizeof(int));
int *starty = (int *)malloc((n+1)*sizeof(int));

int stroke=0;               // valore dello stroke di partenza

int posx = 0;               // variabili per estremi di calcolo
int posy = 1;
int stopx = m;
int stopy = n;

char **sv = (char **)malloc(n*sizeof(char));
for(int i=0;i<n;i++)
            sv[i] = (char *)malloc(sn_maxwl*sizeof(char));
vectorFromString (s,sv);
//ArrayList sv = DBUtility.vectorFromString (s);

// in c
char **tv = (char **)malloc(m*sizeof(char));
for(int i=0;i<m;i++)
            tv[i] = (char *)malloc(tm_maxwl*sizeof(char));

vectorFromString (t,tv);
//ArrayList tv = DBUtility.vectorFromString (t);

// inizializzazione array
int i;
int j;
int k;
for (i = 0; i <= n; i++)
   {for (j = 0; j <= m; j++)
       {d[i][j] = VUOTO;}
   }

for (i = 0; i <= n; i++)
     {starty[i] = 0;}
for (j = 0; j <= m; j++)
     {startx[j] = 0;}

                    // calcolo della matrice per stroke diagonali
int mm = (int)(maxDist*m);
for (stroke=0; (stroke<=mm); stroke++)
   {
                // calcolo stroke diagonali della matrice triangolare superiore
    k = posx;
    while (k<=stroke && k<=stopx)
      {
       i = startx[k];
       j = k + startx[k];

      do
         {if (i==n && j==m) return stroke;
          d[i][j] = stroke;

          i++;
          j++;
          if (i>n)
             {posx=k+1;
              stopy=-1;
              break;
             }
          if (j>m)
```

```
                {stopx=k-1;
                 break;
                }
              }
          while (comp(sv[i-1],tv[j-1]) || (d[i-1][j] == stroke-1) || (d[i][j-1] == stroke-1));

          startx[k] = i;
          k++;
         }
                    // calcolo stroke diagonali della matrice triangolare inferiore
        k = posy;
        while (k<=stroke && k<=stopy)
          {
          i = k + starty[k];
          j = starty[k];

          do
             {
             if (i==n && j==m) return stroke;
             d[i][j] = stroke;

             i++;
             j++;
             if (i>n)
                {stopy=k-1;
                 break;
                }
             if (j>m)
                {posy=k+1;
                 stopx=-1;
                 break;
                }
              }
          while(comp(sv[i-1],tv[j-1]) || (d[i-1][j] == stroke-1) || (d[i][j-1] == stroke-1));

          starty[k] = j;
          k++;
         }

       }

    return (-1);
  }

int main(int argc,char *argv[]) // classe main per testare il funzionamento
{
   float par = 0;
   char s1[1000],s2[1000];
   printf("\nInserici la stringa 1: ");
   gets(s1);
   printf("\nInserici la stringa 2: ");
   gets(s2);
   printf("\nInserici la distanza : ");
   scanf("%f",&par);

   int wed = wordEditDistance(s1,s2);
   int wedd= wordEditDistanceDiag(s1,s2,par);
   printf("\n\nWED su %s , %s : %i",s1,s2,wed);
   printf("\n\nWED su %s , %s , %1.2f :  %i\n\n",s1,s2,par,wedd);
   system("PAUSE");
             return(0);
}
```

A.2 Filtro sub2Count (primo passo sub2match) - Java

```
public Result sub2Count (double k, int q, int lMinSub, boolean qSub) throws SQLException
{
  java.util.Date data1 = new java.util.Date();      // per calcolo tempo
  long inizio = data1.getTime();

  String tab1 = main.ExtraFrame.TABLE1;    // le tabelle delle frasi da utilizzare
  String tab2 = main.ExtraFrame.TABLE2;

  String qtab1 = new String();           // le tabelle dei q-grammi da utilizzare
  String qtab2 = new String();
  if (qSub)
     {qtab1 = "QS"+tab1;
      qtab2 = "QS"+tab2;
     }
  else
     {qtab1 = "Q"+tab1;
      qtab2 = "Q"+tab2;
     }

  int minCount = lMinSub - 1 - ((int) Math.round(k*lMinSub) - 1) * q - (q - 1) * 2;
  if (minCount < 1) minCount = 1;

  // Crea una connessione JDBC
  Connection conn = conn();

  String query =
   "INSERT INTO SUBCOUNT "
     +"SELECT r2q.codice, r1q.codice"
    +" FROM "+qtab1+" r1q, "+qtab2+" r2q"
    +" WHERE r1q.qgram = r2q.qgram"
    +" AND r1q.ext = 'N'"
    +" AND r2q.ext = 'N'"
    +" AND r2q.codice NOT IN (SELECT cod2 FROM FULLMATCH)"
    +" GROUP BY r2q.codice, r1q.codice"
    +" HAVING COUNT(*) >= "+minCount
    +" ORDER BY r2q.codice, r1q.codice";

  Statement stmt = conn.createStatement ();
  stmt.execute (query);

  java.util.Date data2 = new java.util.Date();    // calcolo tempo impiegato
  long fine = data2.getTime();
  long durata = (fine - inizio);

  int conto = 0;
  ResultSet rset = stmt.executeQuery ("SELECT COUNT(*) FROM SUBCOUNT");
  if (rset.next())
     conto = rset.getInt(1);

  rset.close();
  stmt.close();
  conn.close();

  return new Result(conto,durata);
}
```

A.3 Filtro sub2Pos (secondo passo sub2match) - Java

```
public Result sub2MatchPart2 (double k, int lMinSub) throws SQLException
{
  java.util.Date data1 = new java.util.Date();      // per calcolo tempo
  long inizio = data1.getTime();

  int kInt = (int) Math.round(k*lMinSub);

  String tab1 = main.ExtraFrame.TABLE1;    // le tabelle delle frasi da utilizzare
  String tab2 = main.ExtraFrame.TABLE2;

  // Crea una connessione JDBC
  Connection conn = conn();

  Statement stmt = conn.createStatement ();
  ResultSet rset = null;

  ArrayList matches = new ArrayList();   // array delle soluzioni
  ArrayList pMatches[] = null;           // array soluzioni parziali (divise per inizio)
  int curLen=-1;                         // lunghezza frase pretradurre - lMinSub
  int maxFine[] = null;                  // max pos di fine di un match per un dato inizio
  int lastPos=-1;                        // ultima posizione di inizio per multieditdistance
  int cod2Old=-1;                        // codice frase da pretradurre passata

  int conto = 0;

  String query =
      "SELECT r2.codice, r2.frase, r2.wordLen, r1.codice, r1.frase, r1.wordLen"
    +" FROM SUBCOUNT tp, "+tab1+" r1, "+tab2+" r2"
    +" WHERE r1.codice = tp.cod1"
    +" AND r2.codice = tp.cod2";

  rset = stmt.executeQuery (query);

  while (rset.next())
    {
     int cod2 = rset.getInt(1);
     String frase2 = rset.getString(2);
     int len2 = rset.getInt(3);
     int cod1 = rset.getInt(4);
     String frase1 = rset.getString(5);
     int len1 = rset.getInt(6);

     if (frase1 == null || frase2 == null)
       continue;

     if (SimSearch.sub2Pos(frase1,frase2,kInt,lMinSub))      // se coppia passa filtro sub2Pos
       {
        conto++;

        if (cod2Old != cod2)         // nuova frase da pretradurre
          {
           cod2Old = cod2;
           curLen = len2 - lMinSub + 1;
           if (curLen<1)
              continue;
           maxFine = new int[curLen];
           Arrays.fill(maxFine,-1);
           lastPos = curLen -1;

           if (pMatches != null)     // copia soluzioni parziali in vettore globale
             {
              for (int pos=0; pos<pMatches.length; pos++)
                {
                 ArrayList vPMatch = pMatches[pos];
                 if (vPMatch.size()>0)
                     matches.addAll(vPMatch);
                }
             }
```

```
            pMatches = new ArrayList[curLen];
            for (int pos = 0; pos<pMatches.length; pos++)
              pMatches[pos] = new ArrayList();
           }

         lastPos = SimSearch.multiEditDistance (frase2,frase1,cod2,cod1,k,lMinSub,
                                    maxFine,lastPos,pMatches);
         if (lastPos == -1)
           continue;
       }
     }

   if (pMatches != null)
    for (int pos=0; pos<pMatches.length; pos++)     // copia ultime soluzioni parziali
     {
      ArrayList vPMatch = pMatches[pos];
      if (vPMatch.size()>0)
         matches.addAll(vPMatch);
     }

   rset.close();
   stmt.close();
                                    // inserimento submatch in db
   PreparedStatement pstmt = conn.prepareStatement
              ("INSERT INTO SUBMATCH VALUES (?, ?, ?, ?, ?, ?, ?)");

   for (int i=0; i<matches.size(); i++)
     {
      SubMatch match = (SubMatch) matches.get(i);
      pstmt.setInt (1, match.getPCod());
      pstmt.setInt (2, match.getPStart());
      pstmt.setInt (3, match.getPEnd());
      pstmt.setInt (4, match.getTCod());
      pstmt.setInt (5, match.getTStart());
      pstmt.setInt (6, match.getTEnd());
      pstmt.setInt (7, match.getDist());
      pstmt.execute ();                    // inserisce il match
     }

   pstmt.close();
   conn.close();

   java.util.Date data2 = new java.util.Date();    // calcolo tempo impiegato
   long fine = data2.getTime();
   long durata = (fine - inizio);

   return new Result(conto,durata);
}
```

A.4 Tabella A4.1 – Tempi e somme totali

DBMS	DP – FULL	WH – FULL	NV – FULL	TOT FULL	DP – SUB	WH – SUB	NV – SUB	TOT SUB
Oracle 9i JI	8077	1146970	24404	1179451	29999	107891	117330	255220
Oracle 9i JO	7578	1041140	25067	1073785	29984	108717	118504	257205
Postgres 8.2	5859	76905	20609	103373	31405	108501	121761	261667
Postgres 8.3	5996	118009	22453	146458	30800	101972	114643	247415
MySQL 5 (LX)	64093	4881348	146570	5092011	26325	87365	102206	215896
Postgres 8.3 (LX)	5335	114326	32877	152538	24130	102315	92784	219229
MySQL 4 (*)	19509	2126930	121247	2267686	81623	282001	316462	680086
MySQL 5 (*)	60820	6634541	363721	7059082	68038	235666	263791	567495
MySQL 6 (*)	62004	6762775	389072	7213851	66679	230374	223076	520129
FireBird 2.1 (*)	101657	11088790	166093	11356540	32498	112277	159184	303959
DBMS	DP – STEM OFF	WH – STEM OFF	NV – STEM OFF	TOT STEM OFF	DP – STEM ON	WH – STEM ON	NV – STEM ON	TOT STEM ON
Oracle 9i JI	36655	1184360	130499	1351514	1421	70501	11235	83157
Oracle 9i JO	36156	1104608	132366	1273130	1406	45249	11205	57860
Postgres 8.2	35557	179891	130099	345547	1707	5515	12271	19493
Postgres 8.3	34879	212908	124532	372319	1917	7073	12564	21554
MySQL 5 (LX)	89127	4897887	239370	5226384	1291	70826	9406	81523
Postgres 8.3 (LX)	28231	212759	115491	356481	1234	3882	10170	15286
MySQL 4 (*)	91546	463152	334956	889654	9586	30970	68910	109466
MySQL 5 (*)	120814	611225	442044	1174083	8044	25988	57825	91857
MySQL 6 (*)	120559	609935	486946	1217440	8124	26247	125202	159573
FireBird 2.1 (*)	131639	665991	291070	1088700	2516	8128	34207	44851

A.5 Testo dei 3 files di prova (Cap. 1.5)

testo_ita.txt:

Questo è il file di prova in lingua italiana.
Questa è una prova. Seconda frase nel paragrafo.
Altre frasi:
Si scelgono a caso due numeri primi, p e q, l'uno indipendentemente dall'altro, abbastanza grandi da garantire la sicurezza dell'algoritmo.
Si calcola il loro prodotto n=pq, chiamato modulo (dato che tutta l'aritmetica seguente è modulo n).
Si sceglie poi un numero d (chiamato esponente privato), coprimo e più piccolo di (p-1)(q-1).
*Si calcola il numero e (chiamato esponente pubblico) tale che e*d=1 (mod(p-1)(q-1)).*
È nel 1978 che questo sistema trova la sua applicazione reale. Infatti sono 3 ricercatori del MIT (Ronald Rivest, Adi Shamir e Leonard Adleman) che hanno saputo implementare tale logica utilizzando particolari proprietà formali dei numeri primi con alcune centinaia di cifre.
L'algoritmo da loro inventato, denominato RSA per via delle iniziali dei loro cognomi, non è sicuro da un punto di vista matematico teorico, in quanto esiste la possibilità che tramite la conoscenza della chiave pubblica si possa decriptare un messaggio, ma l'enorme mole di calcoli e l'infinito dispendio in termini di tempo necessari per trovare la soluzione, fa di questo algoritmo un sistema di affidabilità pressoché assoluta.

testo_eng.txt:

This is the file test in English.
This is a test. Second sentence in the paragraph.
Other phrases:
We randomly choose two prime numbers, p and q, regardless of a second, large enough to ensure the safety of the algorithm.
It is estimated their product n = pq, called module (given that all the arithmetic is following form n).
It then chooses a number (called private exponent), coprimo and smallest (w-1)(q-1).
*It is estimated the number and (called exponent public) that e*d=1 (mod(p-1)(q-1)).*
It was in 1978 that this system has its actual application. In fact, researchers are 3 MIT (Ronald Rivest, Adi Shamir and Leonard Adleman) who were able to implement this logic using formal properties of prime numbers with a few hundred digits.
The algorithm they invented, called RSA by the initials of their surnames, is not safe from a mathematical point of view theoretical, since there is the possibility that through knowledge of the public key can decrypt a message, but the enormous amount of calculations and the infinite deal of time needed to find the solution, makes this an algorithm system reliability almost absolute.

pattern.txt:

This phrase not compare in the test file.
This is the pattern file in English.
This is a test. Second sentence in this paragraph.
Other phrases follow:
We randomly choose two prime number, p and q, regardless of a second, large enough to ensure the safety of the algorithm.
It is estimated their product n = pq, called module (given that all the arithmetic is following form n).
It then chooses a number (called private exponent), coprimo and smallest (w-1)(q-1).

A.6 Da ConnectSQLJ a JDBC

Mettiamo a confronto in modo esplicito le due versioni tralasciando le parti di codice ininfluenti:

In blu a sinistra il codice nuovo JDBC, in nero a destra il codice originale SQLJ, in rosso le parti di codice diverse, in verde i commenti aggiunti, in viola i commenti originali ed i tre punti (...) che indicano la rimozione di codice ininfluente a questo scopo.

```
package stemming.eng;
import java.sql.*;
// SQLJ non serve più
import java.util.*;
import java.io.*;

class ConnectSqlj
{
// I metodi di questa classe eseguono operazioni di ricerca
sulle tabelle dei vari DBMS

 public static Statement stmt = null;
 public static Connection conn = null;
(...)

 public ConnectSqlj()                  //costruttore
 {
  dbConnect();
 }

  public static void dbConnect()
  {
  (...)
  Class.forName(main.ExtraFrame.DBdriver);
  (...)


conn = DriverManager.getConnection(
main.ExtraFrame.DBconnection,
main.ExtraFrame.DBuser,
main.ExtraFrame.DBpw);

 (...)
  }

  public String cercaVerbo(String parola) throws
SQLException
  {
      PreparedStatement stmt = null;
    (...)
      String ret = null;

  (...)
      ResultSet rset = null;



stmt =
conn.prepareStatement(
"SELECT verb FROM "+main.ExtraFrame.DBtdot+
"VERBI_EN WHERE idc=0
AND id = (SELECT MIN(id)
FROM VERBI_EN WHERE verb=?)");

stmt.setString(1,parola);
rset = stmt.executeQuery();
}

  if(rset.next()) ret = rset.getString(1);

  (...)
  stmt.close();
```

```
package stemming.eng;
import java.sql.*;
import oracle.sqlj.runtime.Oracle;
import java.util.*;
import java.io.*;

class ConnectSqlj
{
// I metodi di questa classe eseguono operazioni di ricerca
sulle tabelle Oracle

// A sinistra, variabili globali di connessione



public ConnectSqlj()                  //costruttore
  {
   dbConnect();
  }

  private static void dbConnect()
  {
   (...)
Oracle.connect(stemming.eng.ConnectSqlj.class,
"../../connect.properties");
   (...)

// A sinistra, i parametri di connessione sono determinati
dinamicamente dalla classe main.ExtraFrame che a sua
volta li carica da SelAndFillDB.SelectDB


}

public String cercaVerbo(String parola) throws
SQLException
{
     String verbo=null;

   class IteratoreVerbo extends
sqlj.runtime.ref.ResultSetIterImpl implements
sqlj.runtime.NamedIterator
{
  public IteratoreVerbo(sqlj.runtime.profile.RTResultSet
resultSet)
    throws java.sql.SQLException
  {
    super(resultSet);
    verbNdx = findColumn("verb");
  }
  public String verb()
    throws java.sql.SQLException
  {
    return resultSet.getString(verbNdx);
  }
  private int verbNdx;
}


{
  sqlj.runtime.ConnectionContext __sJT_connCtx =
sqlj.runtime.ref.DefaultContext.getDefaultContext();
```

```
}
      catch(SQLException v){(...)}
```

```
// La sintassi è completamente differente e nel caso a destra
(SQLJ) è quasi incomprensibile perché generata
automaticamente a partire da testo SQL
```

```
    return ret;

}//end_cercaVerbo
```

```
public String cercaInDizionario(String parola) throws
SQLException
  {
      PreparedStatement stmt = null;
      (...)
      String ret = null;

      try
      {
      ResultSet rset = null;

      (...)
stmt = conn.prepareStatement(
"SELECT COUNT(*) FROM "+main.ExtraFrame.DBtdot+
"DIZ WHERE word=?");

stmt.setString(1,parola);
rset = stmt.executeQuery();
```

```
  if (__sJT_connCtx == null)
sqlj.runtime.error.RuntimeRefErrors.raise_NULL_CONN_
CTX();
  sqlj.runtime.ExecutionContext __sJT_execCtx =
__sJT_connCtx.getExecutionContext();
  if (__sJT_execCtx == null)
sqlj.runtime.error.RuntimeRefErrors.raise_NULL_EXEC_
CTX();
  String __sJT_1 = parola;
  synchronized (__sJT_execCtx) {
    sqlj.runtime.profile.RTStatement __sJT_stmt =
__sJT_execCtx.registerStatement(__sJT_connCtx,
ConnectSqlj_SJProfileKeys.getKey(0), 0);
    try
    {
      __sJT_stmt.setString(1, __sJT_1);
     sqlj.runtime.profile.RTResultSet __sJT_result =
__sJT_execCtx.executeQuery();
     iteraVerbo = new IteratoreVerbo(__sJT_result);
    }
    finally
    {
      __sJT_execCtx.releaseStatement();
    }
  }
}
```

```
     if (iteraVerbo.next()) verbo=iteraVerbo.verb();
      iteraVerbo.close();

      return verbo;

  }//end_cercaVerbo
```

```
public String cercaInDizionario(String parola) throws
SQLException
  {
     int conto=0;
(...)

{
  sqlj.runtime.profile.RTResultSet __sJT_rtRs;
  sqlj.runtime.ConnectionContext __sJT_connCtx =
sqlj.runtime.ref.DefaultContext.getDefaultContext();
  if (__sJT_connCtx == null)
sqlj.runtime.error.RuntimeRefErrors.raise_NULL_CONN_
CTX();
  sqlj.runtime.ExecutionContext __sJT_execCtx =
__sJT_connCtx.getExecutionContext();
  if (__sJT_execCtx == null)
sqlj.runtime.error.RuntimeRefErrors.raise_NULL_EXEC_
CTX();
  String __sJT_1 = parola;
  synchronized (__sJT_execCtx) {
    sqlj.runtime.profile.RTStatement __sJT_stmt =
__sJT_execCtx.registerStatement(__sJT_connCtx,
ConnectSqlj_SJProfileKeys.getKey(0), 1);
    try
    {
      __sJT_stmt.setString(1, __sJT_1);
     sqlj.runtime.profile.RTResultSet __sJT_result =
__sJT_execCtx.executeQuery();
      __sJT_rtRs = __sJT_result;
    }
    finally
    {
      __sJT_execCtx.releaseStatement();
    }
  }
  try
  {
```

```
  if(rset.next())
    if(rset.getInt(1)>0)
          ret = parola;

    (...)
 stmt.close();
    }
    catch(SQLException v){(...)}

    return ret;


 }//end_cercaInDizionario

 public String cercaInStoplist(String parola) throws
SQLException
 {
    PreparedStatement stmt = null;
    (...)
    String ret = null;

    try
    {
    ResultSet rset = null;

    (...)
  stmt = conn.prepareStatement(
"SELECT COUNT(*) FROM "+main.ExtraFrame.DBtdot+
"STOPLIST WHERE word=?");

 stmt.setString(1,parola);
 rset = stmt.executeQuery();
```

```
   sqlj.runtime.ref.ResultSetIterImpl.checkColumns(__sJT_
rtRs, 1);
   if (!__sJT_rtRs.next())
   {
    sqlj.runtime.error.RuntimeRefErrors.raise_NO_ROW_S
ELECT_INTO();
   }
   conto = __sJT_rtRs.getIntNoNull(1);
   if (__sJT_rtRs.next())
   {
    sqlj.runtime.error.RuntimeRefErrors.raise_MULTI_RO
W_SELECT_INTO();
   }
  }
  finally
  {
   __sJT_rtRs.close();
  }
 }

    if(conto>0) return parola;
    else return null;

// La sintassi sopra è equivalente, ma a sinistra è necessario
chiudere lo statement quindi il “return” va fatto alla fine
```

```
}//end_cercaInDizionario

 public String cercaInStoplist(String parola) throws
SQLException
 {
    int conto=0;

    (...)

 {
  sqlj.runtime.profile.RTResultSet __sJT_rtRs;
  sqlj.runtime.ConnectionContext __sJT_connCtx =
sqlj.runtime.ref.DefaultContext.getDefaultContext();
  if (__sJT_connCtx == null)
sqlj.runtime.error.RuntimeRefErrors.raise_NULL_CONN_
CTX();
  sqlj.runtime.ExecutionContext __sJT_execCtx =
__sJT_connCtx.getExecutionContext();
  if (__sJT_execCtx == null)
sqlj.runtime.error.RuntimeRefErrors.raise_NULL_EXEC_
CTX();
  String __sJT_1 = parola;
  synchronized (__sJT_execCtx) {
   sqlj.runtime.profile.RTStatement __sJT_stmt =
__sJT_execCtx.registerStatement(__sJT_connCtx,
ConnectSqlj_SJProfileKeys.getKey(0), 2);
   try
   {
    __sJT_stmt.setString(1, __sJT_1);
    sqlj.runtime.profile.RTResultSet __sJT_result =
__sJT_execCtx.executeQuery();
    __sJT_rtRs = __sJT_result;
   }
   finally
   {
    __sJT_execCtx.releaseStatement();
   }
  }
  try
  {
   sqlj.runtime.ref.ResultSetIterImpl.checkColumns(__sJT_
rtRs, 1);
   if (!__sJT_rtRs.next())
```

```
if(rset.next())
            if(rset.getInt(1)>0)
                        ret = parola;

    (...)
    stmt.close();
    }
    catch(SQLException v){(...)}

    return ret;

}//end_cercaInStoplist


public String cercaInEccezioni(String parola) throws
SQLException
  {
     String risultato=null;
     return risultato;
  }//end_cercaInEccezioni

// Il codice è uguale dove non vi sono connessioni ai DBMS

  public String cercaInAbbreviazioni(String parola) throws
SQLException
  {
     PreparedStatement stmt = null;
     (...)
     String ret = null;

                        try
                        {
                                    ResultSet rset = null;

                        (...)
stmt = conn.prepareStatement(
"SELECT verb FROM "+main.ExtraFrame.DBtdot+
"ABBERV_EN WHERE espressione=?");

stmt.setString(1,parola);
rset = stmt.executeQuery();



if(rset.next())          ret = parola;
```

```
    {
     sqlj.runtime.error.RuntimeRefErrors.raise_NO_ROW_S
ELECT_INTO();
    }
    conto = __sJT_rtRs.getIntNoNull(1);
    if (__sJT_rtRs.next())
    {
     sqlj.runtime.error.RuntimeRefErrors.raise_MULTI_RO
W_SELECT_INTO();
    }
  }
  finally
  {
    __sJT_rtRs.close();
  }
}

(...)
      if(conto>0) return parola;
      else return null;



// La sintassi sopra è equivalente, ma a sinistra è necessario
chiudere lo statement quindi il “return” va fatto alla fine



  }//end_cercaInStoplist


public String cercaInEccezioni(String parola) throws
SQLException
  {
     String risultato=null;
     return risultato;
  }//end_cercaInEccezioni

// Il codice è uguale dove non vi sono connessioni ai DBMS

  public String cercaInAbbreviazioni(String parola) throws
SQLException
  {
     String abbrev=null;

    (...)

class IteratoreAbbrev
extends sqlj.runtime.ref.ResultSetIterImpl
implements sqlj.runtime.NamedIterator
{
  public IteratoreAbbrev(sqlj.runtime.profile.RTResultSet
resultSet)
    throws java.sql.SQLException
  {
    super(resultSet);
    verbNdx = findColumn("verb");
  }
  public String verb()
    throws java.sql.SQLException
  {
    return resultSet.getString(verbNdx);
  }
  private int verbNdx;
}

{
  sqlj.runtime.ConnectionContext __sJT_connCtx =
sqlj.runtime.ref.DefaultContext.getDefaultContext();
  if (__sJT_connCtx == null)
sqlj.runtime.error.RuntimeRefErrors.raise_NULL_CONN_
CTX();
  sqlj.runtime.ExecutionContext __sJT_execCtx =
__sJT_connCtx.getExecutionContext();
```

```
                    (...)
                                stmt.close();
            }
            catch(SQLException v){(...)}
```

```
  if (__sJT_execCtx == null)
sqlj.runtime.error.RuntimeRefErrors.raise_NULL_EXEC_
CTX();
  String __sJT_1 = parola;
  synchronized (__sJT_execCtx) {
    sqlj.runtime.profile.RTStatement __sJT_stmt =
__sJT_execCtx.registerStatement(__sJT_connCtx,
ConnectSqlj_SJProfileKeys.getKey(0), 3);
    try
    {
      __sJT_stmt.setString(1, __sJT_1);
     sqlj.runtime.profile.RTResultSet __sJT_result =
__sJT_execCtx.executeQuery();
      iteraAbbrev = new IteratoreAbbrev(__sJT_result);
    }
    finally
    {
      __sJT_execCtx.releaseStatement();
    }
  }
}
```

```
    return ret;

}//end cercaInAbbreviazioni
```

```
      if (iteraAbbrev.next()) abbrev=iteraAbbrev.verb();
      iteraAbbrev.close();

      return abbrev;

  }//end cercaInAbbreviazioni
```

```
public static void chiudiDB()
  {
    try { conn.close(); } catch (SQLException e) { }
  }//end_chiudiDB
//La chiusura è praticamente identica

public  void inserisci (String frase, int codice) throws
SQLException
  {


PreparedStatement pstmt = null;

      try {
pstmt = conn.prepareStatement(
"UPDATE TERMS SET STEMMED_TERM_EN=?
WHERE CODICE=?");

          pstmt.setString(1,frase);
          pstmt.setInt(2,codice);
          pstmt.executeUpdate();
      }
```

```
public static void chiudiOracle()
  {
    try { Oracle.close(); } catch (SQLException e) { }
  }//end_chiudiOracle
//La chiusura è praticamente identica

public  void inserisci (String frase, int codice) throws
SQLException
  {
    (...)

{
  sqlj.runtime.ConnectionContext __sJT_connCtx =
sqlj.runtime.ref.DefaultContext.getDefaultContext();
  if (__sJT_connCtx == null)
sqlj.runtime.error.RuntimeRefErrors.raise_NULL_CONN_
CTX();
  sqlj.runtime.ExecutionContext __sJT_execCtx =
__sJT_connCtx.getExecutionContext();
  if (__sJT_execCtx == null)
sqlj.runtime.error.RuntimeRefErrors.raise_NULL_EXEC_
CTX();
  String __sJT_1 = frase;
  int __sJT_2 = codice;
  synchronized (__sJT_execCtx) {
    sqlj.runtime.profile.RTStatement __sJT_stmt =
__sJT_execCtx.registerStatement(__sJT_connCtx,
ConnectSqlj_SJProfileKeys.getKey(0), 4);
    try
    {
      __sJT_stmt.setString(1, __sJT_1);
      __sJT_stmt.setInt(2, __sJT_2);
      __sJT_execCtx.executeUpdate();
    }
    finally
    {
      __sJT_execCtx.releaseStatement();
    }
```

```
      finally {
         if (pstmt != null) pstmt.close();
      }




//Qui si vede bene la differenza tra le due lunghezze del
codice.









}//end_inserisci


public float[][] approxMatch(String query) throws
SQLException
  {
    float[][] risultato = new float[10][2];

    for (int j=0;j<10;j++){for (int k=0;k<2;k++) risultato[j]
[k]= 0;}


//La connessione qui esiste già

PreparedStatement pstmt = conn.prepareStatement(query);
    ResultSet rs=pstmt.executeQuery();

    int i =0;
    while ((i<10)&&(rs.next()))
      {risultato[i][0]=rs.getInt(1);

       risultato[i][1]=rs.getInt(2);

       i=i+1;}//end_while

    rs.close();
    pstmt.close();

    return risultato;
  }//end_approxMatch

}//end_connectSqlj



// Le istruzioni per SQLJ a sinistra continuano, mentre per
JDBC sono finite.
```

```
  }
}

}//end_inserisci


public float[][] approxMatch(String query) throws
SQLException
  {
    float[][] risultato = new float[10][2];

    for (int j=0;j<10;j++){for (int k=0;k<2;k++) risultato[j]
[k]= 0;}

    Connection conn=
sqlj.runtime.ref.DefaultContext.getDefaultContext().getCon
nection();
    PreparedStatement pstmt =
conn.prepareStatement(query);
    ResultSet rs=pstmt.executeQuery();

    int i =0;
    while ((i<10)&&(rs.next()))
      {risultato[i][0]=rs.getInt(1);

       risultato[i][1]=rs.getInt(2);

       i=i+1;}//end_while

    rs.close();
    pstmt.close();

    return risultato;
  }//end_approxMatch

}
class ConnectSqlj_SJProfileKeys
{
  private static ConnectSqlj_SJProfileKeys inst = null;
  public static java.lang.Object getKey(int keyNum)
    throws java.sql.SQLException
  {
    if (inst == null)
    {
      inst = new ConnectSqlj_SJProfileKeys();
    }
    return inst.keys[keyNum];
  }
  private final sqlj.runtime.profile.Loader loader =
sqlj.runtime.RuntimeContext.getRuntime().getLoaderForCl
ass(getClass());
  private java.lang.Object[] keys;
  private ConnectSqlj_SJProfileKeys()
    throws java.sql.SQLException
  {
    keys = new java.lang.Object[1];
    keys[0] =
sqlj.runtime.ref.DefaultContext.getProfileKey(loader,
"stemming.eng.ConnectSqlj_SJProfile0");
  }
}
```

A.7 Le modifiche (Rif. Cap. 4.2.1)

Presentiamo ora le cinque macro-modifiche descritte nel paragrafo 4.2.1 includendo e commentando il codice sorgente originale e modificato escludendo le parti ininfluenti a questo scopo:

In blu a sinistra il codice nuovo portabile, in nero a destra il codice originale, in rosso le parti di codice diverse, in verde i commenti aggiunti, in viola i commenti originali ed i tre punti (...) che indicano la rimozione di codice ininfluente a questo scopo.

1. Java Inside (JI) e Java Outside (JO)

Funzione *wholeMatch*:

```
public static Result wholeMatch
(String tab1, String tab2, int q, double k, boolean
countFilter, boolean posFilter, boolean lenFilter, (...))
throws SQLException, IllegalArgumentException
    {
     // controllo su K
     if (k<0) throw new IllegalArgumentException
                        ("K must be >= 0");

     java.util.Date data1 = new java.util.Date();     // per
calcolo tempo
     long inizio = data1.getTime();

     String qtab1 = "Q"+tab1;
     String qtab2 = "Q"+tab2;

     // Crea una connessione JDBC
     Connection conn = conn();

     Statement stmt = conn.createStatement ();

     String approxQuery =

        "SELECT r2.codice AS cod2, r2.frase AS frase2,
r1.codice AS cod1, r1.frase AS frase1, r1.wordLen,
r2.wordLen"

       +" FROM "+tab1+" r1, "+tab2+" r2";

     if (posFilter || countFilter) approxQuery = approxQuery
      +", "+qtab1+" r1q, "+qtab2+" r2q"
      +" WHERE r1.codice = r1q.codice"
      +" AND r2.codice = r2q.codice"
      +" AND r1q.qgram = r2q.qgram";

              String round_r2wLen =
SelAndFillDB.SelectDB.Query(9,null,null,null,0,k); //9
********** QID = 9


 if (posFilter) approxQuery = approxQuery
      + " AND ABS(r1q.pos - r2q.pos) <="
      +round_r2wLen;

     if (lenFilter)
       {
        if ((!posFilter) && (!countFilter)) approxQuery =
approxQuery
      + " WHERE";
        else approxQuery = approxQuery
      + " AND";
        approxQuery = approxQuery
      + " ABS(r1.wordLen - r2.wordLen)"
```

```
public static Result wholeMatch
(String tab1, String tab2, int q, double k, boolean
countFilter, boolean posFilter, boolean lenFilter, (...))
throws SQLException, IllegalArgumentException
    {
     // controllo su K
     if (k<0) throw new IllegalArgumentException
                        ("K must be >= 0");

     java.util.Date data1 = new java.util.Date();     // per
calcolo tempo
     long inizio = data1.getTime();

     String qtab1 = "Q"+tab1;
     String qtab2 = "Q"+tab2;

     // Crea una connessione JDBC
     Connection conn = ods.getConnection();

     Statement stmt = conn.createStatement ();

     String approxQuery =
      "INSERT INTO FULLMATCH "
       +"SELECT r2.codice AS cod2, r1.codice AS cod1,"
       +" wordEditDistanceDiag (r1.frase, r2.frase, "+k+")"
      +" FROM "+tab1+" r1, "+tab2+" r2";

     if (posFilter || countFilter) approxQuery = approxQuery
      +", "+qtab1+" r1q, "+qtab2+" r2q"
      +" WHERE r1.codice = r1q.codice"
      +" AND r2.codice = r2q.codice"
      +" AND r1q.qgram = r2q.qgram";


//A sinistra, il "ROUND" differisce tra DBMS e DBMS
quindi non può essere statico, ma deve essere determinato
da una funzione apposita.

if (posFilter) approxQuery = approxQuery
      + " AND ABS (r1q.pos - r2q.pos) <= ROUND("+k+"*
r2.wordLen)";

     if (lenFilter)
       {
        if ((!posFilter) && (!countFilter)) approxQuery =
approxQuery
      + " WHERE";
        else approxQuery = approxQuery
      + " AND";
        approxQuery = approxQuery
      + " ABS (r1.wordLen - r2.wordLen)"
     +" <= ROUND("+k+"* r2.wordLen)";
};
```

```
      +" <= "+round_r2wLen;
};

      approxQuery = approxQuery
       +" GROUP BY r2.codice, r1.codice, r1.frase, r2.frase,
r1.wordLen, r2.wordLen";

      if (countFilter) approxQuery =approxQuery
       +" HAVING COUNT(*) >= "
       +"(r1.wordLen - 1 - (("
       +round_r2wLen+" - 1) * "+q+"))"
       +" AND COUNT(*) >= "
       +"(r2.wordLen - 1 - (("
       +round_r2wLen+" - 1) * "+q+"))";



ResultSet rset = stmt.executeQuery (approxQuery);

(...)

PreparedStatement pstmt = conn.prepareStatement
("INSERT INTO FULLMATCH VALUES (?, ?, ?)");

      while (rset.next())
        {
         int cod2 = rset.getInt("cod2");
         int cod1 = rset.getInt("cod1");
         String frase2 = rset.getString("frase2");
         String frase1 = rset.getString("frase1");
         int dist =
simSearch.Distance.wordEditDistanceDiag(frase1,
frase2,k);

         if ( dist>=0 )
           {
            pstmt.setInt (1, cod2);
            pstmt.setInt (2, cod1);
            pstmt.setInt (3, dist);
            pstmt.execute ();       // Inserisce il match
           }
        }

       rset.close();
                        pstmt.close();

      }

      stmt.close();
      conn.close();

      java.util.Date data2 = new java.util.Date();    // calcolo
tempo impiegato
      long fine = data2.getTime();
      long durata = (fine - inizio);

      return (new Result (0, durata));
    }
```

```
    approxQuery = approxQuery
      +" GROUP BY r2.codice, r1.codice, r1.frase, r2.frase,
r1.wordLen, r2.wordLen";

    if (countFilter) approxQuery = approxQuery
      +" HAVING COUNT(*) >= "
      +"(r1.wordLen - 1 - (
  ROUND("+k+"* r2.wordLen) - 1) * "+q+"))"
      +" AND COUNT(*) >= "
          +"(r2.wordLen - 1 - (
  ROUND("+k+"* r2.wordLen) - 1) * "+q+"))"
      +" AND ";

    (...)

    stmt.execute (approxQuery);



//L'inserimento a sinistra è fatto alla fine mentre qui è fatto
all'inizio della query sql “INSERT INTO FULLMATCH ”

    stmt.close();
    conn.close();

    java.util.Date data2 = new java.util.Date();    // calcolo
tempo impiegato
    long fine = data2.getTime();
    long durata = (fine - inizio);

    return (new Result (0, durata));
  }
```

Funzione *extractResults*:

```
public static Result extractResults
(String tab1, String tab2, FileWriter outputFile,  boolean
subStr, ArrayList vPhrases, ArrayList preTransData, String
dDelim, String sDelim) throws SQLException,
IOException
   {
    (...)
    // Crea una connessione JDBC
    Connection conn = conn();
    if(conn==null) return null;
```

```
public static Result extractResults
(String tab1, String tab2, FileWriter outputFile,
boolean subStr, ArrayList vPhrases, ArrayList
preTransData, String dDelim, String sDelim) throws
SQLException, IOException
   {
    (...)
    // Crea una connessione JDBC
    Connection conn = ods.getConnection();
```

```
            if(!main.ExtraFrame.JinOracle)
// Se NON è stata scelta l'opzione Oracle 9i (Java inside)
            {
    if (subStr)
    {

    Statement stss = conn.createStatement ();

String ssq =
SelAndFillDB.SelectDB.Query(10,tab1,tab2,null,0,0);
//********** QID = 10

        ResultSet rsss = stss.executeQuery(ssq);

if(!main.ExtraFrame.DBname=="FIREBIRD")){

PreparedStatement pss = conn.prepareStatement("INSERT
INTO "+main.ExtraFrame.DBtdot+"WSS_EXTRES
VALUES(?, ?, ?, ?, ?, ?, ?)");

        while(rsss.next()){


//ORACLE e POSTGRES e FIREBIRD NON USANO I
PUNTI NEI NOMI DELLE COLONNE
            if(main.ExtraFrame.DBname=="ORACLE" ||
main.ExtraFrame.DBname=="POSTGRES")
            {
            String T1 = rsss.getString("p2_spos");
            String T2 = rsss.getString("p1_spos");
            String T3 = rsss.getString("p1_apos");

            String P1 = rsss.getString("p2_frase");
            int P2 = rsss.getInt("sm_i2");
            int P3 = rsss.getInt("sm_f2");
            String P4 = rsss.getString("p2_fraseorig");
            int P5 = DBUtility.transPos(T1,P2);
            int P6 = DBUtility.transPos(T1,P3);
            String P7 = rsss.getString("p1_frase");
            int P8 = rsss.getInt("sm_i1");
            int P9 = rsss.getInt("sm_f1");
            String P10 = rsss.getString("p1_fraseorig");
            int P11 = DBUtility.transPos(T2,P8);
            int P12 = DBUtility.transPos(T2,P9);
            String P13 = rsss.getString("p1_frasetrad");
            int P14 = DBUtility.transPos(T3,P11);
            int P15 = DBUtility.transPos(T3,P12);

         pss.setString(1,DBUtility.wordSubString(P1 ,P2
,P3 ));
        pss.setString(2,DBUtility.wordSubString(P4 ,P5
,P6 ));
        pss.setString(3,DBUtility.wordSubString(P7 ,P8
,P9 ));
        pss.setString(4,DBUtility.wordSubString(P10,P11,P
12));
        pss.setString(5,DBUtility.wordSubString(P13,P14,P
15));
            pss.setInt(6,rsss.getInt("sm_cod1"));
            pss.setInt(7,rsss.getInt("sm_cod2"));
}
else
{
            String T1 = rsss.getString("p2.spos");
            String T2 = rsss.getString("p1.spos");
            String T3 = rsss.getString("p1.apos");
            String P1 = rsss.getString("p2.frase");
            int P2 = rsss.getInt("sm.i2");
            int P3 = rsss.getInt("sm.f2");
      String P4 = rsss.getString("p2.fraseorig");
            int P5 = DBUtility.transPos(T1,P2);
            int P6 = DBUtility.transPos(T1,P3);
            String P7 = rsss.getString("p1.frase");
```

```
Statement stmt = conn.createStatement ();

// Per i DBMS diversi da Oracle versione JI, quindi anche
per Oracle JO, la query viene modificata radicalmente.

// Firebird ha problemi diversi e verrà trattato a parte

// Serve una nuova tabella di appoggio

 // Oracle JO e PostgreSQL non vogliono i nomi delle
colonne risultanti con i .
```

```
                int P8 = rsss.getInt("sm.i1");
                int P9 = rsss.getInt("sm.f1");
        String P10 = rsss.getString("p1.fraseorig");
                int P11 = DBUtility.transPos(T2,P8);
                int P12 = DBUtility.transPos(T2,P9);
         String P13 = rsss.getString("p1.frasetrad");
                int P14 = DBUtility.transPos(T3,P11);
                int P15 = DBUtility.transPos(T3,P12);

        pss.setString(1,DBUtility.wordSubString(P1 ,P2
,P3 ));
        pss.setString(2,DBUtility.wordSubString(P4 ,P5
,P6 ));
        pss.setString(3,DBUtility.wordSubString(P7 ,P8
,P9 ));
        pss.setString(4,DBUtility.wordSubString(P10,P11,P12
));
         pss.setString(5,DBUtility.wordSubString(P13,P14,P1
5));
                pss.setInt(6,rsss.getInt("sm.cod1"));
                pss.setInt(7,rsss.getInt("sm.cod2"));
                }

                pss.execute();
           }

                pss.close();

                }
                        else
//Il solito FireBird che chiude i ResulSet prima del tempo
                        {
                                                (...)
                        }

                        rsss.close();
                        stss.close();

                        QsubStr =               // parte per
sottostringhe
           " UNION"
           +" SELECT sm.cod2 AS codice2,"
                +" p2.frase AS frase2,"
                +" wss.c1 AS sfrase2,"
                +" p2.fraseorig AS fraseorig2,"
                +" wss.c2 AS sfraseorig2,"
                +" sm.cod1 AS codice1,"
                +" p1.frase AS frase1,"
                +" wss.c3 AS sfrase1,"
                +" p1.fraseorig AS fraseorig1,"
                +" wss.c4 AS sfraseorig1,"
                +" p1.frasetrad AS frasetrad1,"
                +" wss.c5 AS sfrasetrad1,"
                +" sm.dist AS dist, (sm.dist / (sm.f2-sm.i2+1))
AS distrel,"
                +" sm.i2 AS inizio, 1 AS sub"
           +" FROM SUBMATCH sm, "+tab1+" p1, "+tab2+"
p2, WSS_EXTRES wss"
           +" WHERE sm.cod1 = p1.codice"
           +" AND sm.cod2 = p2.codice"
           +" AND sm.cod1 = wss.cod1"
           +" AND sm.cod2 = wss.cod2";
                 }
     resultQuery =
           "SELECT fm.cod2 AS codice2,"
                +" p2.frase AS frase2, " AS sfrase2,"
                +" p2.fraseorig AS fraseorig2, " AS sfraseorig2,"
                +" fm.cod1 AS codice1,"
                +" p1.frase AS frase1, " AS sfrase1,"
                +" p1.fraseorig AS fraseorig1, " AS sfraseorig1,"
                +" p1.frasetrad AS frasetrad1, " AS sfrasetrad1,"
                +" fm.dist AS dist, (fm.dist / p2.wordLen) AS
distrel,"
                +" 0 AS inizio, 0 AS sub"
```

```
        +" FROM FULLMATCH fm, "+tab1+" p1,
"+tab2+" p2"
        +" WHERE fm.cod1 = p1.codice"
        +" AND fm.cod2 = p2.codice"
                        +QsubStr;
                        if(!
(main.ExtraFrame.DBname=="FIREBIRD"))
//Errore nella order-by (poco influente)
            resultQuery+=" ORDER BY codice2, sub,
inizio, dist, codice1";
            }
            else
// Se è stata scelta l'opzione Oracle 9i (Java inside)
            {
                        resultQuery =
          "SELECT fm.cod2 AS codice2,"
            +" p2.frase AS frase2, " AS sfrase2,"
            +" p2.fraseorig AS fraseorig2, " AS sfraseorig2,"
            +" fm.cod1 AS codice1,"
            +" p1.frase AS frase1, " AS sfrase1,"
            +" p1.fraseorig AS fraseorig1, " AS sfraseorig1,"
            +" p1.frasetrad AS frasetrad1, " AS sfrasetrad1,"
            +" fm.dist AS dist, (fm.dist / p2.wordLen) AS
distrel,"
            +" 0 AS inizio, 0 AS sub"
        +" FROM FULLMATCH fm, "+tab1+" p1,
"+tab2+" p2"
        +" WHERE fm.cod1 = p1.codice"
        +" AND fm.cod2 = p2.codice";

    if (subStr) resultQuery = resultQuery          // parte per
sottostringhe
        +" UNION"
        +" SELECT sm.cod2 AS codice2,"
            +" p2.frase AS frase2,"
            +" wordSubString(p2.frase, sm.i2, sm.f2) AS
sfrase2,"
            +" p2.fraseorig AS fraseorig2,"
            +" wordSubString(p2.fraseorig,
transPos(p2.spos,sm.i2), transPos(p2.spos,sm.f2))"
                    +" AS sfraseorig2,"
            +" sm.cod1 AS codice1,"
            +" p1.frase AS frase1,"
            +" wordSubString(p1.frase, sm.i1, sm.f1) AS
sfrase1,"
            +" p1.fraseorig AS fraseorig1,"
            +" wordSubString(p1.fraseorig,
transPos(p1.spos,sm.i1), transPos(p1.spos,sm.f1))"
                    +" AS sfraseorig1,"
            +" p1.frasetrad AS frasetrad1,"
            +" wordSubString(p1.frasetrad,
transPos(p1.apos,transPos(p1.spos,sm.i1)),"
                    +"transPos(p1.apos,transPos(p1.spos,sm.f
1))) AS sfrasetrad1,"
            +" sm.dist AS dist, (sm.dist / (sm.f2-sm.i2+1))
AS distrel,"
            +" sm.i2 AS inizio, 1 AS sub"
        +" FROM SUBMATCH sm, "+tab1+" p1, "+tab2+"
p2"
        +" WHERE sm.cod1 = p1.codice"
        +" AND sm.cod2 = p2.codice";

            resultQuery = resultQuery
        +" ORDER BY codice2, sub, inizio, dist, codice1";
            }
Statement stmt = conn.createStatement ();

ResultSet rset = stmt.executeQuery (resultQuery);

    (...)

    rset.close();
    stmt.close();
    conn.close();
```

```
// La query resta identica solo per Oracle JI

String resultQuery =
          "SELECT fm.cod2 AS codice2,"
            +" p2.frase AS frase2, " AS sfrase2,"
            +" p2.fraseorig AS fraseorig2, " AS sfraseorig2,"
            +" fm.cod1 AS codice1,"
            +" p1.frase AS frase1, " AS sfrase1,"
            +" p1.fraseorig AS fraseorig1, " AS sfraseorig1,"
            +" p1.frasetrad AS frasetrad1, " AS sfrasetrad1,"
            +" fm.dist AS dist, (fm.dist / p2.wordLen) AS
distrel,"
            +" 0 AS inizio, 0 AS sub"
        +" FROM FULLMATCH fm, "+tab1+" p1,
"+tab2+" p2"
        +" WHERE fm.cod1 = p1.codice"
        +" AND fm.cod2 = p2.codice";

    if (subStr) resultQuery = resultQuery          // parte per
sottostringhe
        +" UNION"
        +" SELECT sm.cod2 AS codice2,"
            +" p2.frase AS frase2,"
            +" wordSubString(p2.frase, sm.i2, sm.f2) AS
sfrase2,"
            +" p2.fraseorig AS fraseorig2,"
            +" wordSubString(p2.fraseorig,
transPos(p2.spos,sm.i2), transPos(p2.spos,sm.f2))"
                    +" AS sfraseorig2,"
            +" sm.cod1 AS codice1,"
            +" p1.frase AS frase1,"
            +" wordSubString(p1.frase, sm.i1, sm.f1) AS
sfrase1,"
            +" p1.fraseorig AS fraseorig1,"
            +" wordSubString(p1.fraseorig,
transPos(p1.spos,sm.i1), transPos(p1.spos,sm.f1))"
                    +" AS sfraseorig1,"
            +" p1.frasetrad AS frasetrad1,"
            +" wordSubString(p1.frasetrad,
transPos(p1.apos,transPos(p1.spos,sm.i1)),"
                    +"transPos(p1.apos,transPos(p1.spos,sm.f
1))) AS sfrasetrad1,"
            +" sm.dist AS dist, (sm.dist / (sm.f2-sm.i2+1))
AS distrel,"
            +" sm.i2 AS inizio, 1 AS sub"
        +" FROM SUBMATCH sm, "+tab1+" p1, "+tab2+"
p2"
        +" WHERE sm.cod1 = p1.codice"
        +" AND sm.cod2 = p2.codice";
```

```
    (...)

    return (new Result (phraseCount, durata, vCount));
  }
```

```
    resultQuery = resultQuery
        +" ORDER BY codice2, sub, inizio, dist, codice1";


ResultSet rset = stmt.executeQuery (resultQuery);

    (...)

    rset.close();
    stmt.close();
    conn.close();

    (...)

    return (new Result (phraseCount, durata, vCount));
  }
```

2. Le connessioni

Le due funzioni della classe SelectDB per la selezione di driver e stringa di connessione

```
public static String DBDriver(String s)
        {
                main.ExtraFrame.DBtdot = "trans1.";
                s.toUpperCase();
                if(s.contains((CharSequence)("ORACLE")))
              return "oracle.jdbc.driver.OracleDriver";
                if(s.contains((CharSequence)("MYSQL")))
              return "com.mysql.jdbc.Driver";
                if(s.contains((CharSequence)("MONETDB")))
              return "nl.cwi.monetdb.jdbc.MonetDriver";
                if(s.contains((CharSequence)("POSTGRES")))
                {  main.ExtraFrame.DBtdot = "";
              return "org.postgresql.Driver";}
                if(s.contains((CharSequence)("FIREBIRD")))
                {  main.ExtraFrame.DBtdot = "";
              return "org.firebirdsql.jdbc.FBDriver";}
                  // if(s.contains((CharSequence)("DB2")))
            //  return "com.ibm.db2.jcc.DB2Driver";
                  return "ERR";
        }

        public static String DBConn(String s)
        {
                main.ExtraFrame.DBtdot = "trans1.";
                s.toUpperCase();
                if(s.contains((CharSequence)("ORACLE")))
              return "jdbc:oracle:oci8:@";
                if(s.contains((CharSequence)("MYSQL")))
              return "jdbc:mysql://localhost/trans1";
                if(s.contains((CharSequence)("MONETDB")))
              return "jdbc:monetdb://localhost/trans1";
                if(s.contains((CharSequence)("POSTGRES")))
                {  main.ExtraFrame.DBtdot = "";
              return "jdbc:postgresql://localhost/trans1";}
                if(s.contains((CharSequence)("FIREBIRD")))
                {  main.ExtraFrame.DBtdot = "";
              return "jdbc:firebirdsql://localhost/trans1";}
                  // if(s.contains((CharSequence)("DB2")))
            //  return "jdbc:db2://localhost/trans1";
                  return "ERR";
        }
```

Caricamento del file di configurazione per driver, stringa di connessione, user e password

```
  String s = "ORACLE";
  String u = "trans1";
  String p = "trans1";
  // Impostazioni di default e in caso di errore
try{
```

```
 java.io.File f1 = new java.io.File("./config.fil");
java.io.DataInputStream ff1 = new java.io.DataInputStream(new java.io.FileInputStream(f1));
 s = ff1.readLine();
 u = ff1.readLine();
 p = ff1.readLine();
              ff1.close();
              }
// Eventuale lettura del file di configurazione
 catch(java.io.IOException e){
 javax.swing.JOptionPane.showMessageDialog(null, "Errore nell'apertura del file di configurazione.\nDatabase ORACLE
9i impostato di default!\n"+e, "Errore", 0);
 }
// Eventuale messaggio di errore se non è accessibile in modo corretto il file di configurazione
 main.ExtraFrame.DBconnection = SelAndFillDB.SelectDB.DBConn(s);
 main.ExtraFrame.DBdriver = SelAndFillDB.SelectDB.DBDriver(s);
 main.ExtraFrame.DBname = s;
 main.ExtraFrame.DBuser = u;
 main.ExtraFrame.DBpw   = p;
 main.ExtraFrame.STEM = SelAndFillDB.SelectDB.DumpTry(0);
// Impostazione delle variabili d'ambiente
```

3. L'SQL

La funzione *Query* della classe *SelectDB*:

(in arancione le differenze SQL tra i diversi DBMS)

```
// Ritorna la Query con QID (Query ID) passato relativa al DataBase in uso!
public static String Query(int qid,String spar1,String spar2,String spar3,int ipar1,double dpar1){

if(main.ExtraFrame.DBname == "ORACLE")
              switch(qid)
              {
               case 1 : return "CREATE TABLE " + spar1
                  +" (codice NUMBER (10) PRIMARY KEY,"
                +" fraseorig VARCHAR2 (2000),"
                +" frase  VARCHAR2 (2000),"
                +" frasetrad VARCHAR2 (2000),"
                +" wordlen NUMBER (4),"
                +" apos VARCHAR2 (2000),"
                +" ascore VARCHAR2 (2000),"
                +" spos VARCHAR2 (2000)"
                +")";

         case 2 : return "CREATE TABLE " + spar1
              +" (codice   NUMBER(10) REFERENCES "+ spar2 +" ON DELETE CASCADE,"
              +"pos      NUMBER (3),"
              +"ext      VARCHAR2 (2),"
              +"qgram    VARCHAR2 (" + ipar1 + "),"
              +"PRIMARY KEY (codice,pos)"
              +")";

         case 3 : return "CREATE TABLE FULLMATCH"
              +"(cod2    NUMBER(10),"
              +"cod1    NUMBER(10),"
              +"dist    NUMBER(3),"
              +"PRIMARY KEY (cod2, cod1),"
        +"FOREIGN KEY (cod1) REFERENCES PHRASE1 (codice) ON DELETE CASCADE,"
        +"FOREIGN KEY (cod2) REFERENCES PHRASE2 (codice) ON DELETE CASCADE"
                        +")";

          case 4 : return "CREATE TABLE MATCHPOS"
              +"(cod1    NUMBER(10),"
              +"cod2    NUMBER(10),"
              +"nr      NUMBER(3),"
              +"nc      NUMBER(3),"
              +"PRIMARY KEY (cod1, cod2, nr, nc),"
        +"FOREIGN KEY (cod1) REFERENCES PHRASE1 (codice) ON DELETE CASCADE,"
        +"FOREIGN KEY (cod2) REFERENCES PHRASE2 (codice) ON DELETE CASCADE"
                        +")";

            case 5 : return "CREATE TABLE SUBMATCH"
```

```
            +"(cod2    NUMBER(10),"
            +"i2      NUMBER(3),"
            +"f2      NUMBER(3),"
            +"cod1    NUMBER(10),"
            +"i1      NUMBER(3),"
            +"f1      NUMBER(3),"
            +"dist    NUMBER(3),"
            +"PRIMARY KEY (cod2, i2, f2, cod1, i1, f1),"
      +"FOREIGN KEY (cod2) REFERENCES PHRASE2 (codice) ON DELETE CASCADE"
                        +")";

          case 6 : return "CREATE TABLE SUBCOUNT"
            +"(cod2    NUMBER(10),"
            +"cod1    NUMBER(10),"
            +"PRIMARY KEY (cod2, cod1)"
                        +")";

          case 7 : return "CREATE TABLE WSS_EXTRES"
                        +"(c1   VARCHAR2(2000),"
            +"c2   VARCHAR2(2000),"
            +"c3   VARCHAR2(2000),"
            +"c4   VARCHAR2(2000),"
            +"c5   VARCHAR2(2000),"
            +"cod1 NUMBER(10),"
            +"cod2 NUMBER(10)"
            +")";

          case 8 : return "DROP INDEX ind" + spar1;

          case 9 : return "ROUND("+dpar1+"* r2.wordLen)";

          case 10: return "SELECT
p2.frase p2_frase, sm.i2 sm_i2, sm.f2 sm_f2, p2.fraseorig p2_fraseorig, p2.spos p2_spos,"
+" p1.frase p1_frase, sm.i1 sm_i1, sm.f1 sm_f1, p1.fraseorig p1_fraseorig, p1.spos p1_spos, p1.frasetrad p1_frasetrad,
p1.apos p1_apos, sm.cod1 sm_cod1, sm.cod2 sm_cod2"
                              +" FROM
"+main.ExtraFrame.DBtdot+"SUBMATCH sm, "+main.ExtraFrame.DBtdot+""+spar1+" p1,
"+main.ExtraFrame.DBtdot+""+spar2+" p2"
                        +" WHERE sm.cod1 = p1.codice"
                        +" AND sm.cod2 = p2.codice";
}
else
if(main.ExtraFrame.DBname == "MYSQL")
           switch(qid)
           {
            case 1 : return "CREATE TABLE " + spar1
            +" (codice INT (10) PRIMARY KEY,"
            +" fraseorig VARCHAR (2000),"
            +" frase  VARCHAR (2000),"
            +" frasetrad VARCHAR (2000),"
            +" wordlen INT (4),"
            +" apos VARCHAR (2000),"
            +" ascore VARCHAR (2000),"
            +" spos VARCHAR (2000)"
            +")";

          case 2 : return "CREATE TABLE " + spar1
                        +" (codice   INT(10) REFERENCES "+ spar2 +" ON DELETE CASCADE,"
            +"pos     INT (3),"
            +"ext     VARCHAR (2),"
            +"qgram    VARCHAR (" + ipar1 + "),"
            +"PRIMARY KEY (codice,pos)"
            +")";

          case 3 : return "CREATE TABLE FULLMATCH"
                        +"(cod2    INT (10),"
            +"cod1    INT (10),"
            +"dist    INT (3),"
            +"PRIMARY KEY (cod2, cod1),"
          +"FOREIGN KEY (cod1) REFERENCES PHRASE1 (codice) ON DELETE CASCADE,"
          +"FOREIGN KEY (cod2) REFERENCES PHRASE2 (codice) ON DELETE CASCADE"
                        +")";

          case 4 : return "CREATE TABLE MATCHPOS"
```

```
                        +"(cod1    INT (10),"
            +"cod2    INT (10),"
            +"nr      INT (3),"
            +"nc      INT (3),"
            +"PRIMARY KEY (cod1, cod2, nr, nc),"
        +"FOREIGN KEY (cod1) REFERENCES PHRASE1 (codice) ON DELETE CASCADE,"
        +"FOREIGN KEY (cod2) REFERENCES PHRASE2 (codice) ON DELETE CASCADE"
                        +")";

            case 5 : return "CREATE TABLE SUBMATCH"
                        +"(cod2    INT (10),"
            +"i2      INT (3),"
            +"f2      INT (3),"
            +"cod1    INT (10),"
            +"i1      INT (3),"
            +"f1      INT (3),"
            +"dist    INT (3),"
            +"PRIMARY KEY (cod2, i2, f2, cod1, i1, f1),"
          "FOREIGN KEY (cod2) REFERENCES PHRASE2 (codice) ON DELETE CASCADE"
                        +")";

            case 6 : return "CREATE TABLE SUBCOUNT"
                        +"(cod2    INT (10),"
            +"cod1    INT (10),"
            +"PRIMARY KEY (cod2, cod1)"
                        +")";

            case 7 : return "CREATE TABLE WSS_EXTRES"
                                                +"(c1   VARCHAR(2000),"
            +"c2   VARCHAR(2000),"
            +"c3   VARCHAR(2000),"
            +"c4   VARCHAR(2000),"
            +"c5   VARCHAR(2000),"
            +"cod1 INT(10),"
            +"cod2 INT(10)"
            +")";

            case 8 : return "DROP INDEX ind" + spar1 + " ON " + spar1;

            case 9 : return "ROUND("+dpar1+"* r2.wordLen)";

            case 10: return "SELECT
p2.frase, sm.i2, sm.f2, p2.fraseorig, p2.spos,"
+" p1.frase, sm.i1, sm.f1, p1.fraseorig, p1.spos, p1.frasetrad, p1.apos, sm.cod1, sm.cod2"
                                    +" FROM "
+main.ExtraFrame.DBtdot+"SUBMATCH sm, "+main.ExtraFrame.DBtdot+""+spar1+" p1,
"+main.ExtraFrame.DBtdot+""+spar2+" p2"
                            +" WHERE sm.cod1 = p1.codice"
                            +" AND sm.cod2 = p2.codice";
}
else
if(main.ExtraFrame.DBname == "MONETDB") // Non accetta dimensione per INT
            switch(qid)
            {
            case 1 : return "CREATE TABLE " + spar1
             +" (codice INT PRIMARY KEY,"
              +" fraseorig VARCHAR (2000),"
              +" frase  VARCHAR (2000),"
              +" frasetrad VARCHAR (2000),"
              +" wordlen INT,"
              +" apos VARCHAR (2000),"
              +" ascore VARCHAR (2000),"
              +" spos VARCHAR (2000)"
              +")";

            case 2 : return "CREATE TABLE " + spar1
                        +" (codice   INT REFERENCES "+ spar2 +" ON DELETE CASCADE,"
            +"pos      INT,"
            +"ext      VARCHAR (2),"
            +"qgram    VARCHAR (" + ipar1 + "),"
            +"PRIMARY KEY (codice,pos)"
            +")";

            case 3 : return "CREATE TABLE FULLMATCH"
```

```
            +"(cod2    INT,"
            +"cod1    INT,"
            +"dist    INT,"
            +"PRIMARY KEY (cod2, cod1),"
        +"FOREIGN KEY (cod1) REFERENCES PHRASE1 (codice) ON DELETE CASCADE,"
        +"FOREIGN KEY (cod2) REFERENCES PHRASE2 (codice) ON DELETE CASCADE"
                    +")";

          case 4 : return "CREATE TABLE MATCHPOS"
            +"(cod1    INT,"
            +"cod2    INT,"
            +"nr      INT,"
            +"nc      INT,"
            +"PRIMARY KEY (cod1, cod2, nr, nc),"
        +"FOREIGN KEY (cod1) REFERENCES PHRASE1 (codice) ON DELETE CASCADE,"
        +"FOREIGN KEY (cod2) REFERENCES PHRASE2 (codice) ON DELETE CASCADE"
                    +")";

          case 5 : return "CREATE TABLE SUBMATCH"
            +"(cod2    INT,"
            +"i2      INT,"
            +"f2      INT,"
            +"cod1    INT,"
            +"i1      INT,"
            +"f1      INT,"
            +"dist    INT,"
            +"PRIMARY KEY (cod2, i2, f2, cod1, i1, f1),"
         +"FOREIGN KEY (cod2) REFERENCES PHRASE2 (codice) ON DELETE CASCADE"
                    +")";

          case 6 : return "CREATE TABLE SUBCOUNT"
            +"(cod2    INT,"
            +"cod1    INT,"
            +"PRIMARY KEY (cod2, cod1)"
            +")";

          case 7 : return "CREATE TABLE WSS_EXTRES"
                    +"(c1   VARCHAR(2000),"
            +"c2   VARCHAR(2000),"
            +"c3   VARCHAR(2000),"
            +"c4   VARCHAR(2000),"
            +"c5   VARCHAR(2000),"
            +"cod1 INT,"
            +"cod2 INT"
            +")";

          case 8 : return "DROP INDEX ind" + spar1;

          case 9 : return "ROUND("+dpar1+"* r2.wordLen,0)";

          case 10: return "SELECT
p2.frase, sm.i2, sm.f2, p2.fraseorig, p2.spos,"
+" p1.frase, sm.i1, sm.f1, p1.fraseorig, p1.spos, p1.frasetrad, p1.apos, sm.cod1, sm.cod2"
                                    +" FROM "
+main.ExtraFrame.DBtdot+"SUBMATCH sm, "+main.ExtraFrame.DBtdot+""+spar1+" p1,
"+main.ExtraFrame.DBtdot+""+spar2+" p2"
                            +" WHERE sm.cod1 = p1.codice"
                            +" AND sm.cod2 = p2.codice";
}
else
if(main.ExtraFrame.DBname == "POSTGRES")
          switch(qid)
          {
          case 1 : return "CREATE TABLE " + spar1
           +" (codice INT PRIMARY KEY,"
            +" fraseorig VARCHAR (2000),"
            +" frase  VARCHAR (2000),"
            +" frasetrad VARCHAR (2000),"
            +" wordlen INT,"
            +" apos VARCHAR (2000),"
            +" ascore VARCHAR (2000),"
            +" spos VARCHAR (2000)"
            +")";
```

```
            case 2 : return "CREATE TABLE " + spar1
                        +" (codice   INT REFERENCES "+ spar2 +" ON DELETE CASCADE,"
             +"pos      INT,"
             +"ext      VARCHAR (2),"
             +"qgram    VARCHAR (" + ipar1 + "),"
             +"PRIMARY KEY (codice,pos)"
             +")";

            case 3 : return "CREATE TABLE FULLMATCH"
                        +"(cod2    INT,"
             +"cod1    INT,"
             +"dist    INT,"
             +"PRIMARY KEY (cod2, cod1),"
          +"FOREIGN KEY (cod1) REFERENCES PHRASE1 (codice) ON DELETE CASCADE,"
          +"FOREIGN KEY (cod2) REFERENCES PHRASE2 (codice) ON DELETE CASCADE"
                        +")";

            case 4 : return "CREATE TABLE MATCHPOS"
             +"(cod1    INT,"
             +"cod2    INT,"
             +"nr      INT,"
             +"nc      INT,"
             +"PRIMARY KEY (cod1, cod2, nr, nc),"
          +"FOREIGN KEY (cod1) REFERENCES PHRASE1 (codice) ON DELETE CASCADE,"
          +"FOREIGN KEY (cod2) REFERENCES PHRASE2 (codice) ON DELETE CASCADE"
                        +")";

           case 5 : return "CREATE TABLE SUBMATCH"
             +"(cod2    INT,"
             +"i2      INT,"
             +"f2      INT,"
             +"cod1    INT,"
             +"i1      INT,"
             +"f1      INT,"
             +"dist    INT,"
             +"PRIMARY KEY (cod2, i2, f2, cod1, i1, f1),"
           +"FOREIGN KEY (cod2) REFERENCES PHRASE2 (codice) ON DELETE CASCADE"
                        +")";

            case 6 : return "CREATE TABLE SUBCOUNT"
             +"(cod2    INT,"
             +"cod1    INT,"
             +"PRIMARY KEY (cod2, cod1)"
             +")";

            case 7 : return "CREATE TABLE WSS_EXTRES"
                                              +"(c1   VARCHAR(2000),"
             +"c2   VARCHAR(2000),"
             +"c3   VARCHAR(2000),"
             +"c4   VARCHAR(2000),"
             +"c5   VARCHAR(2000),"
             +"cod1 INT,"
             +"cod2 INT"
             +")";

            case 8 : return "DROP INDEX ind" + spar1;

            case 9 : return "ROUND("+dpar1+"* r2.wordLen)";

            case 10: return "SELECT
p2.frase as p2_frase, sm.i2 as sm_i2, sm.f2 as sm_f2, p2.fraseorig as p2_fraseorig, p2.spos as p2_spos,"
+" p1.frase as p1_frase, sm.i1 as sm_i1, sm.f1 as sm_f1, p1.fraseorig as p1_fraseorig, p1.spos as p1_spos, p1.frasetrad as
p1_frasetrad, p1.apos as p1_apos, sm.cod1 as sm_cod1, sm.cod2 as sm_cod2"
                                    +" FROM "
+main.ExtraFrame.DBtdot+"SUBMATCH sm, "+main.ExtraFrame.DBtdot+""+spar1+" p1,
"+main.ExtraFrame.DBtdot+""+spar2+" p2"
                                +" WHERE sm.cod1 = p1.codice"
                                +" AND sm.cod2 = p2.codice";
}
else
if(main.ExtraFrame.DBname == "FIREBIRD")
            switch(qid)
            {
            case 1 : return "CREATE TABLE " + spar1
```

```
            +" (codice INT PRIMARY KEY,"
             +" fraseorig VARCHAR (2000),"
             +" frase  VARCHAR (2000),"
             +" frasetrad VARCHAR (2000),"
             +" wordlen INT,"
             +" apos VARCHAR (2000),"
             +" ascore VARCHAR (2000),"
             +" spos VARCHAR (2000)"
             +")";

        case 2 : return "CREATE TABLE " + spar1
                    +" (codice   INT REFERENCES "+ spar2 +" ON DELETE CASCADE,"
         +"pos      INT,"
         +"ext      VARCHAR (2),"
         +"qgram    VARCHAR (" + ipar1 + "),"
         +"PRIMARY KEY (codice,pos)"
         +")";

        case 3 : return "CREATE TABLE FULLMATCH"
         +"(cod2    INT,"
         +"cod1    INT,"
         +"dist    INT,"
         +"PRIMARY KEY (cod2, cod1),"
      +"FOREIGN KEY (cod1) REFERENCES PHRASE1 (codice) ON DELETE CASCADE,"
      +"FOREIGN KEY (cod2) REFERENCES PHRASE2 (codice) ON DELETE CASCADE"
         +")";

        case 4 : return "CREATE TABLE MATCHPOS"
         +"(cod1    INT,"
         +"cod2    INT,"
         +"nr      INT,"
         +"nc      INT,"
         +"PRIMARY KEY (cod1, cod2, nr, nc),"
      +"FOREIGN KEY (cod1) REFERENCES PHRASE1 (codice) ON DELETE CASCADE,"
      +"FOREIGN KEY (cod2) REFERENCES PHRASE2 (codice) ON DELETE CASCADE"
                    +")";

       case 5 : return "CREATE TABLE SUBMATCH"
         +"(cod2    INT,"
         +"i2      INT,"
         +"f2      INT,"
         +"cod1    INT,"
         +"i1      INT,"
         +"f1      INT,"
         +"dist    INT,"
         +"PRIMARY KEY (cod2, i2, f2, cod1, i1, f1),"
       +"FOREIGN KEY (cod2) REFERENCES PHRASE2 (codice) ON DELETE CASCADE"
                    +")";

        case 6 : return "CREATE TABLE SUBCOUNT"
         +"(cod2    INT,"
         +"cod1    INT,"
         +"PRIMARY KEY (cod2, cod1)"+")";

        case 7 : return "CREATE TABLE WSS_EXTRES"
                                    +"(c1   VARCHAR(2000),"
         +"c2   VARCHAR(2000),"
         +"c3   VARCHAR(2000),"
         +"c4   VARCHAR(2000),"
         +"c5   VARCHAR(2000),"
         +"cod1 INT,"
         +"cod2 INT"
         +")";

        case 8 : return "DROP INDEX ind" + spar1;

        case 9 : return "ROUND("+dpar1+"* r2.wordLen)";

        case 10: return "SELECT
p2.frase as p2_frase, sm.i2 as sm_i2, sm.f2 as sm_f2, p2.fraseorig as p2_fraseorig, p2.spos as p2_spos,"
+" p1.frase as p1_frase, sm.i1 as sm_i1, sm.f1 as sm_f1, p1.fraseorig as p1_fraseorig, p1.spos as p1_spos, p1.frasetrad as
p1_frasetrad, p1.apos as p1_apos, sm.cod1 as sm_cod1, sm.cod2 as sm_cod2"
                                +" FROM SUBMATCH sm, "+spar1+" p1, "+spar2+" p2"
                +" WHERE sm.cod1 = p1.codice"
```

```
                +" AND sm.cod2 = p2.codice";
}

return null;
}
```

4. MonetDB e gli apici

Le istruzioni di elaborazione per MonetDB nella classe *connectSQLJ* per tutte le funzioni:

```
if(main.ExtraFrame.DBname=="MONETDB"){
// Funziona solo se nella stringa tutti gli apici singoli sono raddoppiati
                String pael = "";
                  if(parola.contains((CharSequence)"")) //Sostanzialmente non serve, ma velocizza le operazioni se
la maggioranza delle parole non contengono virgolette singole.
                  for(int i=0,x=0;i<parola.length();i++)
                        {
                              char vir=39;
                              if(parola.charAt(i)==vir)
                                    pael+=vir;
                              pael+=parola.charAt(i);
                        }
                        else
                              pael = parola;
                  stmtm = conn.createStatement();
            rset = stmtm.executeQuery("SELECT (...) FROM (...) WHERE (...) ='"+pael+"')");
}
```

5. FireBird e i ResultSet

Un esempio nella funzione *extractResults* della classe *DBconnect*

```
            int rowCount = 65534;

            String T1[]= new String[rowCount];
            String T2[]= new String[rowCount];
            String T3[]= new String[rowCount];

            String P1[]= new String[rowCount];
            String P4[]= new String[rowCount];
            String P7[]= new String[rowCount];
            String P10[]= new String[rowCount];
            String P13[]= new String[rowCount];

            int P2[]= new int[rowCount];
            int P3[]= new int[rowCount];
            int P5[]= new int[rowCount];
            int P6[]= new int[rowCount];
            int P8[]= new int[rowCount];
            int P9[]= new int[rowCount];
            int P11[]= new int[rowCount];
            int P12[]= new int[rowCount];
            int P14[]= new int[rowCount];
            int P15[]= new int[rowCount];

            int C1[]= new int[rowCount];
            int C2[]= new int[rowCount]; // Si creano tutte le variabili

            int sz = -1;
            while(rsss.next()) // Si itera normalmente il resultset ma...
            {
            sz++;
            T1[sz]= rsss.getString("p2_spos");
            T2[sz]= rsss.getString("p1_spos");
            T3[sz]= rsss.getString("p1_apos");
```

```
            P1[sz]= rsss.getString("p2_frase");
            P2[sz]= rsss.getInt("sm_i2");
            P3[sz]= rsss.getInt("sm_f2");
            P4[sz]= rsss.getString("p2_fraseorig");
            P5[sz]= DBUtility.transPos(T1[sz],P2[sz]);
            P6[sz]= DBUtility.transPos(T1[sz],P3[sz]);
            P7[sz]= rsss.getString("p1_frase");
            P8[sz]= rsss.getInt("sm_i1");
            P9[sz]= rsss.getInt("sm_f1");
            P10[sz]= rsss.getString("p1_fraseorig");
            P11[sz]= DBUtility.transPos(T2[sz],P8[sz]);
            P12[sz]= DBUtility.transPos(T2[sz],P9[sz]);
            P13[sz]= rsss.getString("p1_frasetrad");
            P14[sz]= DBUtility.transPos(T3[sz],P11[sz]);
            P15[sz]= DBUtility.transPos(T3[sz],P12[sz]);
            C1[sz]= rsss.getInt("sm_cod1");
            C2[sz]= rsss.getInt("sm_cod2");

                                    } // I risultati non sono inseriti immediatamente in un'altra tabella



// L'inserimento è fatto a posteriori utilizzando le variabili create all'inizio
PreparedStatement pss = conn.prepareStatement("INSERT INTO "+main.ExtraFrame.DBtdot+"WSS_EXTRES
VALUES(?, ?, ?, ?, ?, ?, ?)");

        for(int j=0;j<=sz;j++)
            {
            pss.setString(1,DBUtility.wordSubString(P1[j],P2[j],P3[j]));
            pss.setString(2,DBUtility.wordSubString(P4[j],P5[j],P6[j]));
            pss.setString(3,DBUtility.wordSubString(P7[j],P8[j],P9[j]));
            pss.setString(4,DBUtility.wordSubString(P10[j],P11[j],P12[j]));
            pss.setString(5,DBUtility.wordSubString(P13[j],P14[j],P15[j]));
            pss.setInt(6,C1[j]);
            pss.setInt(7,C2[j]);
            pss.execute();
            }
pss.close();
```

Questo tipo di intervento è presente nella classe *DBConnect*, funzione, riga:

fillQTable , 513 a 558
extractResults, 736 a 806
wholeMatch, 1109 a 1141

A.8 Classe Expdmp del package SelAndFillDB

```
package SelAndFillDB;

import java.sql.*;
import java.io.*;
import javax.swing.*;

public class Expdmp
{
 public int selTbl(String whtb){ // Seleziona il numero relativo alla tabella/e
        if(whtb.contains((CharSequence)("ABBERV_EN")))   return 1;
        if(whtb.contains((CharSequence)("STOPLIST")))    return 2;
        if(whtb.contains((CharSequence)("VERBI_EN")))    return 3;
        if(whtb.contains((CharSequence)("DIZ")))         return 4;
        return 0;
 }

 public Expdmp(String whtb,String whdb,boolean outputen) // Costruttore
 {
        boolean dbok = false;
        String DBdriver = SelectDB.DBDriver(whdb);
        String DBconn = SelectDB.DBConn(whdb);
        if(!DBdriver.contains((CharSequence)("ERR"))&&!DBconn.contains((CharSequence)("ERR")))
                dbok = true;


        whtb = whtb.toUpperCase();
        boolean all = false;
        int sel = selTbl(whtb);
        if(whtb.contains((CharSequence)("ALL")))
                all = true;

        if((all||(sel!=0))&&dbok) // Se è stata selezionata almeno una tabella e un database
 {
        try
  {
   java.util.Date datag = new java.util.Date();      // per calcolo tempo
   long inizio = datag.getTime();


        Class.forName(DBdriver);
        Connection mdb = DriverManager.getConnection(DBconn,main.ExtraFrame.DBuser,main.ExtraFrame.DBpw);

        if(all||(sel==1)) //************************************************
        {

        Statement mdbS = mdb.createStatement();
        PreparedStatement mdbPS = null;

        String query = "";

        try{
                query = "DROP TABLE "+main.ExtraFrame.DBtdot+"ABBERV_EN";
                mdbS.executeUpdate(query);

        }
        catch(SQLException e){}

        if(whdb.contains((CharSequence)("FIREBIRD")))
                query = "CREATE TABLE ABBERV_EN (ID INT, ESPRESSIONE VARCHAR(50) NOT NULL,
VERB VARCHAR(200))";
        else
                if(whdb.contains((CharSequence)("ORACLE")))
                        query = "CREATE TABLE ABBERV_EN (ID NUMBER(10), ESPRESSIONE
VARCHAR2(50), VERB VARCHAR2(200))";
                else
                        query = "CREATE TABLE ABBERV_EN (ID INT, ESPRESSIONE VARCHAR(50),
VERB VARCHAR(200))";

        mdbS.executeUpdate(query);
```

```
            mdbS.close();

            File f1 = new File("../dump/abberv_en.csv");
            DataInputStream ff1 = new DataInputStream(new FileInputStream(f1));
            if(outputen)
            {
            JFileChooser fileChooserf1 = null;

            fileChooserf1 = new JFileChooser();


            System.out.println("\nINFO: Apri il file CSV abberv_en.csv!");
            int response = fileChooserf1.showOpenDialog(null);

   if(response == 0)
   try
    {
                        f1 = fileChooserf1.getSelectedFile();
               ff1 = new DataInputStream(new FileInputStream(f1));

    }
   catch(Exception e)
    {
              JOptionPane.showMessageDialog(null, "Errore nell'apertura del file.\n"+e, "Errore", 0);
    }
   else
            return;
            }

            boolean vread = true;
            String val = ff1.readLine();
            while(vread)
            {
                        try
                        {
                        val = ff1.readLine();
                        if(val!=null)
                        {

                        int beg=0;
                        while(val.charAt(beg)!=',') beg++;
                        int id = Integer.parseInt(val.substring(0,beg));

                        int end = beg+1;
                        while(val.charAt(end)!=',') end++;
                        String espressione = val.substring(beg+2,(end-1));

                        String verb = val.substring(end+2,(val.length()-1));

                        if(outputen)
                                    System.out.println(id+" "+espressione+" "+verb);

                        try
                        {
                        mdbPS = mdb.prepareStatement("INSERT INTO "+main.ExtraFrame.DBtdot+"ABBERV_EN
VALUES(?,?,?)");
                        mdbPS.setInt(1,id);
                        mdbPS.setString(2,espressione);
                        mdbPS.setString(3,verb);
                        mdbPS.execute();
                        mdbPS.close();
                        }
                        catch(SQLException s){JOptionPane.showMessageDialog(null, "Errore SQL "+s,"Errore",0); vread
= false; return;}

                        }
                        else
                        {
                                    vread = false;
                                    ff1.close();
                                    if(outputen)
                                                System.out.println("INSERIMENTO "+whtb+" ("+sel+") OK!");
                        }
                        }
```

```
                    catch(IOException e){JOptionPane.showMessageDialog(null, "INSERIMENTO "+whtb+" ("+sel+")
FALLITO!", "Errore", 0); vread = false;}

            }

                    try{
                            Statement mdbSi = mdb.createStatement();
                            mdbSi.executeUpdate("CREATE INDEX INDX_ABBREV_EN ON ABBERV_EN
(VERB)");
                            mdbSi.executeUpdate("ALTER TABLE ABBERV_EN ADD PRIMARY KEY
(ESPRESSIONE)");
                            mdbSi.close();
                    }catch(SQLException se){JOptionPane.showMessageDialog(null, "OPERAZIONI ALTER/INDEX
SU "+whtb+" ("+sel+") FALLITE!\n"+se, "Errore", 0);}

            }


            if(all||(sel==2)) //***********************************************
            {
            Statement mdbS = mdb.createStatement();
            PreparedStatement mdbPS = null;

            String query = "";

            try{
                    query = "DROP TABLE "+main.ExtraFrame.DBtdot+"STOPLIST";
                    mdbS.executeUpdate(query);

            }
            catch(SQLException e){}

            if(whdb.contains((CharSequence)("FIREBIRD")))
                    query = "CREATE TABLE STOPLIST (WORD VARCHAR(200) NOT NULL)";
            else
                    if(whdb.contains((CharSequence)("ORACLE")))
                            query = "CREATE TABLE STOPLIST (WORD VARCHAR2(200))";
                    else
                            query = "CREATE TABLE STOPLIST (WORD VARCHAR(200))";

            mdbS.executeUpdate(query);

            mdbS.close();

            File f1 = new File("../dump/stoplist.csv");
            DataInputStream ff1 = new DataInputStream(new FileInputStream(f1));
            if(outputen)
            {
            JFileChooser fileChooserf1 = null;

            fileChooserf1 = new JFileChooser();

            System.out.println("Apri il file CSV stoplist.csv!");
            int response = fileChooserf1.showOpenDialog(null);

  if(response == 0)
  try
    {
                    f1 = fileChooserf1.getSelectedFile();
              ff1 = new DataInputStream(new FileInputStream(f1));

    }
  catch(Exception e)
    {
             JOptionPane.showMessageDialog(null, "Errore nell'apertura del file.\n"+e, "Errore", 0);
    }
  else
            return;
            }

            boolean vread = true;
            String val = ff1.readLine();
            while(vread)
            {
                    try
```

```
                    {
                    val = ff1.readLine();
                    if(val!=null)
                    {

                    String word = val.substring(1,(val.length()-1));
                    if(outputen)
                            System.out.println(word);

                    try
                    {
                    mdbPS = mdb.prepareStatement("INSERT INTO "+main.ExtraFrame.DBtdot+"STOPLIST
VALUES(?)");
                    mdbPS.setString(1,word);
                    mdbPS.execute();
                    mdbPS.close();
                    }
                    catch(SQLException s){JOptionPane.showMessageDialog(null, "Errore SQL "+s,"Errore",0); vread
= false; return;}

                    }
                    else
                    {
                            vread = false;
                            ff1.close();
                            if(outputen)
                                    System.out.println("INSERIMENTO "+whtb+" ("+sel+") OK!");
                    }
                    }
                    catch(IOException e){JOptionPane.showMessageDialog(null,"INSERIMENTO "+whtb+" ("+sel+")
FALLITO!","Errore",0); vread = false;}

            }

                    try{
                            Statement mdbSi = mdb.createStatement();
                            mdbSi.executeUpdate("CREATE INDEX INDX_STOPLIST ON STOPLIST (WORD)");
                            mdbSi.executeUpdate("ALTER TABLE STOPLIST ADD PRIMARY KEY (WORD)");
                            mdbSi.close();
                    }catch(SQLException se){JOptionPane.showMessageDialog(null,"OPERAZIONI ALTER/INDEX
SU "+whtb+" ("+sel+") FALLITE!","Errore",0);}
            }

            if(all||(sel==3)) //************************************************
            {
            Statement mdbS = mdb.createStatement();
            PreparedStatement mdbPS = null;

            String query = "";

            try{
                    query = "DROP TABLE "+main.ExtraFrame.DBtdot+"VERBI_EN";
                    mdbS.executeUpdate(query);

            }
            catch(SQLException e){}

            if(whdb.contains((CharSequence)("FIREBIRD")))
                    query = "CREATE TABLE VERBI_EN (ID INT NOT NULL, VERB VARCHAR(200), IDC INT
NOT NULL)";
            else
                    if(whdb.contains((CharSequence)("ORACLE")))
                            query = "CREATE TABLE VERBI_EN (ID NUMBER(10), VERB VARCHAR2(200),
IDC NUMBER(10))";
                    else
                            query = "CREATE TABLE VERBI_EN (ID INT, VERB VARCHAR(200), IDC INT)";

            mdbS.executeUpdate(query);

            mdbS.close();

            File f1 = new File("../dump/verbi_en.csv");
            DataInputStream ff1 = new DataInputStream(new FileInputStream(f1));
            if(outputen)
            {
```

```
            JFileChooser fileChooserf1 = null;

            fileChooserf1 = new JFileChooser();

            System.out.println("Apri il file CSV verbi_en.csv!");
            int response = fileChooserf1.showOpenDialog(null);

   if(response == 0)
   try
    {
                        f1 = fileChooserf1.getSelectedFile();
              ff1 = new DataInputStream(new FileInputStream(f1));

    }
   catch(Exception e)
    {
               JOptionPane.showMessageDialog(null, "Errore nell'apertura del file.\n"+e, "Errore", 0);
    }
   else
            return;
            }

            boolean vread = true;
            String val = ff1.readLine();
            while(vread)
            {
                        try
                        {
                        val = ff1.readLine();
                        if(val!=null)
                        {

                        int beg=0;
                        while(val.charAt(beg)!=',') beg++;
                        int id = Integer.parseInt(val.substring(0,beg));

                        int end = beg+2;
                        while(val.charAt(end)!="") end++;
                        String verb = val.substring(beg+2,end);

                        int idc = Integer.parseInt(val.substring(end+2,val.length()));

                        if(outputen)
                                    System.out.println(id+" "+verb+" "+idc);

                        try
                        {
                        mdbPS = mdb.prepareStatement("INSERT INTO "+main.ExtraFrame.DBtdot+"VERBI_EN
VALUES(?,?,?)");
                        mdbPS.setInt(1,id);
                        mdbPS.setString(2,verb);
                        mdbPS.setInt(3,idc);
                        mdbPS.execute();
                        mdbPS.close();
                        }
                        catch(SQLException s){JOptionPane.showMessageDialog(null, "Errore SQL "+s,"Errore",0); vread
= false; return;}

                        }
                        else
                        {
                                    vread = false;
                                    ff1.close();
                                    if(outputen)
                                                System.out.println("INSERIMENTO "+whtb+" ("+sel+") OK!");
                        }
                        }
                        catch(IOException e){JOptionPane.showMessageDialog(null,"INSERIMENTO "+whtb+" ("+sel+")
FALLITO!","Errore",0); vread = false;}

            }

                        try{
                                    Statement mdbSi = mdb.createStatement();
                                    mdbSi.executeUpdate("CREATE INDEX INDX_VERBI_ID ON VERBI_EN (ID)");
```

```
                                        mdbSi.executeUpdate("CREATE INDEX IND_VERB_VERB ON VERBI_EN
(VERB)");
                                        mdbSi.executeUpdate("ALTER TABLE VERBI_EN ADD PRIMARY KEY (ID, IDC)");
                                        mdbSi.close();
                                }catch(SQLException se){JOptionPane.showMessageDialog(null,"OPERAZIONI ALTER/INDEX
SU "+whtb+" ("+sel+") FALLITE!","Errore",0);}
                }

                if(all||(sel==4)) //************************************************
                {
                Statement mdbS = mdb.createStatement();
                PreparedStatement mdbPS = null;

                String query = "";

                try{
                                query = "DROP TABLE "+main.ExtraFrame.DBtdot+"DIZ";
                                mdbS.executeUpdate(query);

                }
                catch(SQLException e){}

                if(whdb.contains((CharSequence)("FIREBIRD")))
                                query = "CREATE TABLE DIZ (ID INT, WORD VARCHAR(700) NOT NULL)";
                else
                                if(whdb.contains((CharSequence)("ORACLE")))
                                                query
= "CREATE TABLE DIZ (ID NUMBER(10), WORD VARCHAR2(700))";
                                else
                                                query = "CREATE TABLE DIZ (ID INT, WORD VARCHAR(700))";

                mdbS.executeUpdate(query);

                mdbS.close();

                File f1 = new File("../dump/diz.csv");
                DataInputStream ff1 = new DataInputStream(new FileInputStream(f1));
                if(outputen)
                {
                JFileChooser fileChooserf1 = null;

                fileChooserf1 = new JFileChooser();

                System.out.println("Apri il file CSV diz.csv!");
                int response = fileChooserf1.showOpenDialog(null);

  if(response == 0)
  try
    {
                                f1 = fileChooserf1.getSelectedFile();
                  ff1 = new DataInputStream(new FileInputStream(f1));

    }
  catch(Exception e)
    {
                 JOptionPane.showMessageDialog(null, "Errore nell'apertura del file.\n"+e, "Errore", 0);
    }
  else
                return;
                }

                boolean vread = true;
                String val = ff1.readLine();
                while(vread)
                {
                                try
                                {
                                val = ff1.readLine();
                                if(val!=null)
                                {

                                int beg=0;
                                while(val.charAt(beg)!=',') beg++;
                                int id = Integer.parseInt(val.substring(0,beg));
```

```
                        String word = val.substring(beg+2,(val.length()-1));

                        if(outputen)
                                System.out.println(id+" "+word);

                        try
                        {
                        mdbPS = mdb.prepareStatement("INSERT INTO "+main.ExtraFrame.DBtdot+"DIZ
VALUES(?,?)");
                        mdbPS.setInt(1,id);
                        mdbPS.setString(2,word);
                        mdbPS.execute();
                        mdbPS.close();
                        }
                        catch(SQLException s){JOptionPane.showMessageDialog(null, "Errore SQL "+s,"Errore",0); vread
= false; return;}

                        }
                        else
                        {
                                vread = false;
                                ff1.close();
                                if(outputen)
                                        System.out.println("INSERIMENTO "+whtb+" ("+sel+") OK!");
                        }
                        }
                        catch(IOException e){JOptionPane.showMessageDialog(null,"INSERIMENTO "+whtb+" ("+sel+")
FALLITO!","Errore",0); vread = false;}

                }

                        try{
                                Statement mdbSi = mdb.createStatement();
                                mdbSi.executeUpdate("CREATE INDEX INDX_DIZ ON DIZ (WORD)");
                                mdbSi.executeUpdate("ALTER TABLE DIZ ADD PRIMARY KEY (WORD)");
                                mdbSi.close();
                        }catch(SQLException se){JOptionPane.showMessageDialog(null,"OPERAZIONI ALTER/INDEX
SU "+whtb+" ("+sel+") FALLITE!","Errore",0);}

                }
          mdb.close();
          datag = new java.util.Date();
          if(outputen)
                        System.out.println("Total time: "+(datag.getTime()-inizio));
          }
           catch(Exception e){JOptionPane.showMessageDialog(null, "Errore "+e,"Errore",0);}
          }
          else
          {     if(!dbok)
                        System.out.println("PARAMETRO [TABELLA] NON CORRETTO!");
                else
                        System.out.println("PARAMETRO [DATABASE] NON CORRETTO!");
}
                return;
         }// Fine costruttore

          public static void main(String[] args)
          {
                if(args.length>1){      new Expdmp(args[0],args[1],false);              }
                else
                {System.out.println("\nSELEZIONA IL DATABASE: \n");
                 System.out.println("[Oracle 9] or [MySQL 4,5,6] or [MonetDB 5] or [Postgres 8.x] or [Firebird 2.x]\n");
                 BufferedReader in0=new BufferedReader(new InputStreamReader(System.in));
                 String whichDB = "ERR";
                 try{whichDB=in0.readLine();}catch(IOException e){}
                 System.out.println("\nINSERISCI IL NOME DELLA TABELLA: \n");
                 System.out.println("[DIZ] or [VERBI_EN] or [ABBERV_EN] or [STOPLIST] or [ALL]\n");
                 BufferedReader in1=new BufferedReader(new InputStreamReader(System.in));
                 String passa = "ERR";
                 try{passa=in1.readLine();}catch(IOException e){}
                 new Expdmp(passa,whichDB,true);
                }
          } // Fine Main
} // Fine classe Expdmp
```

A.9 ExtraFrame_ConfigureDialog.java – solo modifiche

```
package main;

import (...)

public class ExtraFrame_ConfigureDialog extends JDialog
{
 JPanel panel1 = new JPanel();
 JPanel contentPane = new JPanel();

 (...)

 //**** SELECTDB  vars ****
 JPanel jPanelConfSelDB = new JPanel();
 JButton   jButtonConfSelDBsetdefault = new JButton();
 JRadioButton jRadioButtonSelDB00 = new JRadioButton();// Oracle 9 Java Inside
 JButton infSelDB00 = new JButton();
 JRadioButton jRadioButtonSelDB0 = new JRadioButton(); // Oracle 9 Java Outside
 JButton infSelDB0 = new JButton();
 JRadioButton jRadioButtonSelDB1 = new JRadioButton(); // MySql 4, 5 o 6
 JButton infSelDB1 = new JButton();
 JRadioButton jRadioButtonSelDB2 = new JRadioButton(); // MonetDB 5
 JButton infSelDB2 = new JButton();
 JRadioButton jRadioButtonSelDB3 = new JRadioButton(); // dFirebird 2.x
 JButton infSelDB3 = new JButton();
 JRadioButton jRadioButtonSelDB4 = new JRadioButton(); // Postgres 8.x
 JButton infSelDB4 = new JButton();
 JRadioButton jRadioButtonSelDB5 = new JRadioButton(); // disponibile *****
 JButton infSelDB5 = new JButton();
 JLabel jLabSelDB = new JLabel();
 boolean dbok = false;
 JLabel jLabImpStem = new JLabel();
 JLabel jLabImpStemStatus0 = new JLabel();
 JButton   jButtonImpStem0 = new JButton();
 JLabel jLabImpStemStatus1 = new JLabel();
 JButton   jButtonImpStem1 = new JButton();
 JLabel jLabImpStemStatus2 = new JLabel();
 JButton   jButtonImpStem2 = new JButton();
 JLabel jLabImpStemStatus3 = new JLabel();
 JButton   jButtonImpStem3 = new JButton();
 JLabel jLabImpStemStatus4 = new JLabel();
 JButton   jButtonImpStem4 = new JButton();

 String e[] = new String[5];

 JPasswordField jTextPW = new JPasswordField();
 JTextField jTextUSER = new JTextField();
 JLabel jLabUSERPW = new JLabel();
 JLabel jLabSetDefUSERPW = new JLabel();
 JLabel jLabUSER = new JLabel();
 JLabel jLabPW = new JLabel();
 JButton jButtonSetUSERPW = new JButton();
 JButton jButtonSetUSERPW2 = new JButton();
 //****  _end ****

 (...)

  int infbx = 40;
  int infby = 20;

  infSelDB00.setPreferredSize(new Dimension(infbx,infby));
  infSelDB00.setText("?");
  infSelDB00.addActionListener(new java.awt.event.ActionListener()
  {
   public void actionPerformed(ActionEvent e)
   {
    infSelDBx_actionPerformed(e,-1);
   }
  });
 infSelDB0.setText("?");
 infSelDB0.setPreferredSize(new Dimension(infbx,infby));
```

```
infSelDB0.addActionListener(new java.awt.event.ActionListener()
{
 public void actionPerformed(ActionEvent e)
 {
  infSelDBx_actionPerformed(e,0);
 }
});
infSelDB1.setText("?");
infSelDB1.setPreferredSize(new Dimension(infbx,infby));
infSelDB1.addActionListener(new java.awt.event.ActionListener()
{
 public void actionPerformed(ActionEvent e)
 {
  infSelDBx_actionPerformed(e,1);
 }
});
infSelDB2.setText("?");
infSelDB2.setPreferredSize(new Dimension(infbx,infby));
infSelDB2.addActionListener(new java.awt.event.ActionListener()
{
 public void actionPerformed(ActionEvent e)
 {
  infSelDBx_actionPerformed(e,2);
 }
});
infSelDB3.setText("?");
infSelDB3.setPreferredSize(new Dimension(infbx,infby));
infSelDB3.addActionListener(new java.awt.event.ActionListener()
{
 public void actionPerformed(ActionEvent e)
 {
  infSelDBx_actionPerformed(e,3);
 }
});
infSelDB4.setText("?");
infSelDB4.setPreferredSize(new Dimension(infbx,infby));
infSelDB4.addActionListener(new java.awt.event.ActionListener()
{
 public void actionPerformed(ActionEvent e)
 {
  infSelDBx_actionPerformed(e,4);
 }
});
infSelDB5.setText("?");
infSelDB5.setPreferredSize(new Dimension(infbx,infby));
infSelDB5.addActionListener(new java.awt.event.ActionListener()
{
 public void actionPerformed(ActionEvent e)
 {
  infSelDBx_actionPerformed(e,5);
 }
});

jRadioButtonSelDB00.setText("Oracle 9i (Java inside)"); // ********************************* I
jRadioButtonSelDB00.addItemListener(new java.awt.event.ItemListener()
{
 public void itemStateChanged(ItemEvent e)
 {
  jRadioButtonSelDB00_itemStateChanged(e);
 }
});
jRadioButtonSelDB0.setText("Oracle 9i (Java outside)");
jRadioButtonSelDB0.addItemListener(new java.awt.event.ItemListener()
{
 public void itemStateChanged(ItemEvent e)
 {
  jRadioButtonSelDB0_itemStateChanged(e);
 }
});
jRadioButtonSelDB1.setText("MySql 4, 5 o 6");
jRadioButtonSelDB1.addItemListener(new java.awt.event.ItemListener()
{
 public void itemStateChanged(ItemEvent e)
 {
  jRadioButtonSelDB1_itemStateChanged(e);
```

```
    }
  });
  jRadioButtonSelDB2.setText("MonetDB 5");
  jRadioButtonSelDB2.addItemListener(new java.awt.event.ItemListener()
  {
    public void itemStateChanged(ItemEvent e)
    {
      jRadioButtonSelDB2_itemStateChanged(e);
    }
  });

  jRadioButtonSelDB3.setText("FireBird 2.x");
  jRadioButtonSelDB3.addItemListener(new java.awt.event.ItemListener()
  {
    public void itemStateChanged(ItemEvent e)
    {
      jRadioButtonSelDB3_itemStateChanged(e);
    }
  });
  jRadioButtonSelDB4.setText("Postgres 8.x");
  jRadioButtonSelDB4.addItemListener(new java.awt.event.ItemListener()
  {
    public void itemStateChanged(ItemEvent e)
    {
      jRadioButtonSelDB4_itemStateChanged(e);
    }
  });
  jRadioButtonSelDB5.setText("VUOTO - DB2?"); //************************** NON USATO
  jRadioButtonSelDB5.addItemListener(new java.awt.event.ItemListener()
  {
    public void itemStateChanged(ItemEvent e)
    {
      jRadioButtonSelDB5_itemStateChanged(e);
    }
  });
  // ******************************** F

  (...)

  //*****************************************************************************DB SEL - IMP
STEM
  java.awt.Font fontdbEstem = new java.awt.Font("Dialog", 0, 12);
  jTabbedPane1.add(jPanelConfSelDB, "Database selection");
  jPanelConfSelDB.setLayout(xYLayout2);
  jLabSelDB.setFont(fontdbEstem);
  jLabSelDB.setText("Select a DataBase:");
  jPanelConfSelDB.add(jLabSelDB, new XYConstraints(5, 10, -1, -1));

  jRadioButtonSelDB00.setSelected(main.ExtraFrame.JinOracle);
  if(main.ExtraFrame.DBname.contains((CharSequence)("ORACLE")))
          jRadioButtonSelDB0.setSelected(true);
  else if(main.ExtraFrame.DBname.contains((CharSequence)("MYSQL")))
          jRadioButtonSelDB1.setSelected(true);
  else if(main.ExtraFrame.DBname.contains((CharSequence)("MONETDB")))
          jRadioButtonSelDB2.setSelected(true);
  else if(main.ExtraFrame.DBname.contains((CharSequence)("FIREBIRD")))
          jRadioButtonSelDB3.setSelected(true);
  else if(main.ExtraFrame.DBname.contains((CharSequence)("POSTGRES")))
          jRadioButtonSelDB4.setSelected(true);
  //else if(main.ExtraFrame.DBname.contains((CharSequence)("VUOTO")))
  //          jRadioButtonSelDB5.setSelected(true);

  int xpos = 5;
  int xinf = 180;
  int ydbpos = 35;
  int ydbspace = 25;

  jPanelConfSelDB.add(jRadioButtonSelDB00, new XYConstraints(xpos, ydbpos, -1, -1));
  jPanelConfSelDB.add(infSelDB00,                    new XYConstraints(xinf, ydbpos, -1, -1));
  ydbpos += ydbspace;
  jPanelConfSelDB.add(jRadioButtonSelDB0,  new XYConstraints(xpos, ydbpos, -1, -1));
  jPanelConfSelDB.add(infSelDB0,                              new XYConstraints(xinf, ydbpos, -1, -1));
  ydbpos += ydbspace;
  jPanelConfSelDB.add(jRadioButtonSelDB1,  new XYConstraints(xpos, ydbpos, -1, -1));
  jPanelConfSelDB.add(infSelDB1,                              new XYConstraints(xinf, ydbpos, -1, -1));
```

```
ydbpos += ydbspace;
jPanelConfSelDB.add(jRadioButtonSelDB2,  new XYConstraints(xpos, ydbpos, -1, -1));
jPanelConfSelDB.add(infSelDB2,                                  new XYConstraints(xinf, ydbpos, -1, -1));
ydbpos += ydbspace;
jPanelConfSelDB.add(jRadioButtonSelDB4,  new XYConstraints(xpos, ydbpos, -1, -1));
jPanelConfSelDB.add(infSelDB4,                                  new XYConstraints(xinf, ydbpos, -1, -1));
ydbpos += ydbspace;
jPanelConfSelDB.add(jRadioButtonSelDB3,  new XYConstraints(xpos, ydbpos, -1, -1));
jPanelConfSelDB.add(infSelDB3,                                  new XYConstraints(xinf, ydbpos, -1, -1));
ydbpos += ydbspace;
//jPanelConfSelDB.add(jRadioButtonSelDB5,  new XYConstraints(xpos, ydbpos, -1, -1));
//jPanelConfSelDB.add(infSelDB5,                    new XYConstraints(xinf, ydbpos, -1, -1));

jTextPW.setText(main.ExtraFrame.DBpw);
jTextPW.setColumns(15);
jTextUSER.setText(main.ExtraFrame.DBuser);
jTextUSER.setColumns(15);
jLabUSERPW.setFont(fontdbEstem);
jLabUSERPW.setText("Select DataBase connection's parameters:");
jLabUSER.setText("User:");
jLabPW.setText("Password:");
jButtonSetUSERPW.setText("Set now");
jButtonSetUSERPW.setPreferredSize(new Dimension(80,25));
jButtonSetUSERPW.setBounds(new Rectangle(495, 9, 77, 22));
jButtonSetUSERPW.addActionListener(new java.awt.event.ActionListener()
{
 public void actionPerformed(ActionEvent e)
 {
  jButtonSetUSERPW_actionPerformed(e);
 }
});

jButtonConfSelDBsetdefault.setText("DataBase");
jButtonConfSelDBsetdefault.setPreferredSize(new Dimension(160,25));
jButtonConfSelDBsetdefault.setBounds(new Rectangle(495, 9, 77, 22));
jButtonConfSelDBsetdefault.addActionListener(new java.awt.event.ActionListener()
{
 public void actionPerformed(ActionEvent e)
 {
  jButtonConfSelDBsetdefault_actionPerformed(e);
 }
});

jButtonSetUSERPW2.setText("User & Password");
jButtonSetUSERPW2.setPreferredSize(new Dimension(160,25));
jButtonSetUSERPW2.setBounds(new Rectangle(495, 9, 77, 22));
jButtonSetUSERPW2.addActionListener(new java.awt.event.ActionListener()
{
 public void actionPerformed(ActionEvent e)
 {
  jButtonSetUSERPW2_actionPerformed(e);
 }
});

int xAlign_Impstem_Clear = 290;
int xAlign_Impstem_Status = 480;

jPanelConfSelDB.add(jLabUSERPW, new XYConstraints(xAlign_Impstem_Clear, 200, -1, -1));
jPanelConfSelDB.add(jLabUSER, new XYConstraints(xAlign_Impstem_Clear, 225, -1, -1));
jPanelConfSelDB.add(jTextUSER, new XYConstraints(xAlign_Impstem_Clear+60, 225, -1, -1));
jPanelConfSelDB.add(jLabPW, new XYConstraints(xAlign_Impstem_Clear, 250, -1, -1));
jPanelConfSelDB.add(jTextPW, new XYConstraints(xAlign_Impstem_Clear+60, 250, -1, -1));
jPanelConfSelDB.add(jButtonSetUSERPW, new XYConstraints(xAlign_Impstem_Clear+200, 235, -1, -1));

jLabSetDefUSERPW.setFont(fontdbEstem);
jLabSetDefUSERPW.setText("Set selected item as default:");
jPanelConfSelDB.add(jLabSetDefUSERPW, new XYConstraints(5, 200, -1, -1));
jPanelConfSelDB.add(jButtonConfSelDBsetdefault, new XYConstraints(5, 225, -1, -1));
jPanelConfSelDB.add(jButtonSetUSERPW2, new XYConstraints(5, 255, -1, -1));

jLabImpStem.setText("Clear & import stemmer's data:     Status:");
jLabImpStem.setFont(fontdbEstem);
jPanelConfSelDB.add(jLabImpStem, new XYConstraints(xAlign_Impstem_Clear, 10, -1, -1));

estatus();
```

```
jButtonImpStem0.setText("ALL tables");
jButtonImpStem0.setPreferredSize(new Dimension(120,25));
jLabImpStemStatus0.setText(e[0]);
jButtonImpStem0.setBounds(new Rectangle(495, 9, 77, 22));
jButtonImpStem0.addActionListener(new java.awt.event.ActionListener()
{
 public void actionPerformed(ActionEvent e)
 {
  jButtonImpStem_actionPerformed(e,"ALL");
 }
});
jPanelConfSelDB.add(jButtonImpStem0, new XYConstraints(xAlign_Impstem_Clear, 40, -1, -1));
jPanelConfSelDB.add(jLabImpStemStatus0, new XYConstraints(xAlign_Impstem_Status, 47, -1, -1));

jButtonImpStem1.setText("Table DIZ");
jButtonImpStem1.setPreferredSize(new Dimension(120,25));
jLabImpStemStatus1.setText(e[1]);
jButtonImpStem1.setBounds(new Rectangle(495, 9, 77, 22));
jButtonImpStem1.addActionListener(new java.awt.event.ActionListener()
{
 public void actionPerformed(ActionEvent e)
 {
  jButtonImpStem_actionPerformed(e,"DIZ");
 }
});
jPanelConfSelDB.add(jButtonImpStem1, new XYConstraints(xAlign_Impstem_Clear, 70, -1, -1));
jPanelConfSelDB.add(jLabImpStemStatus1, new XYConstraints(xAlign_Impstem_Status, 77, -1, -1));

//2
jButtonImpStem2.setText("Table ABBERV_EN");
jButtonImpStem2.setPreferredSize(new Dimension(120,25));
jLabImpStemStatus2.setText(e[2]);
jButtonImpStem2.setBounds(new Rectangle(495, 9, 77, 22));
jButtonImpStem2.addActionListener(new java.awt.event.ActionListener()
{
 public void actionPerformed(ActionEvent e)
 {
  jButtonImpStem_actionPerformed(e,"ABBERV_EN");
 }
});
jPanelConfSelDB.add(jButtonImpStem2, new XYConstraints(xAlign_Impstem_Clear, 100, -1, -1));
jPanelConfSelDB.add(jLabImpStemStatus2, new XYConstraints(xAlign_Impstem_Status, 107, -1, -1));

//3
jButtonImpStem3.setText("Table STOPLIST");
jButtonImpStem3.setPreferredSize(new Dimension(120,25));
jLabImpStemStatus3.setText(e[3]);
jButtonImpStem3.setBounds(new Rectangle(495, 9, 77, 22));
jButtonImpStem3.addActionListener(new java.awt.event.ActionListener()
{
 public void actionPerformed(ActionEvent e)
 {
  jButtonImpStem_actionPerformed(e,"STOPLIST");
 }
});
jPanelConfSelDB.add(jButtonImpStem3, new XYConstraints(xAlign_Impstem_Clear, 130, -1, -1));
jPanelConfSelDB.add(jLabImpStemStatus3, new XYConstraints(xAlign_Impstem_Status, 137, -1, -1));

// 4
jButtonImpStem4.setText("Table VERBI_EN");
jButtonImpStem4.setPreferredSize(new Dimension(120,25));
jLabImpStemStatus4.setText(e[4]);
jButtonImpStem4.setBounds(new Rectangle(495, 9, 77, 22));
jButtonImpStem4.addActionListener(new java.awt.event.ActionListener()
{
 public void actionPerformed(ActionEvent e)
 {
  jButtonImpStem_actionPerformed(e,"VERBI_EN");
 }
});
jPanelConfSelDB.add(jButtonImpStem4, new XYConstraints(xAlign_Impstem_Clear, 160, -1, -1));
jPanelConfSelDB.add(jLabImpStemStatus4, new XYConstraints(xAlign_Impstem_Status, 167, -1, -1));
```

```
   //*****************************************************************************DB SEL - IMP
STEM
  }

 (...)

   if (jSliderElab.getValue() == 0)
      main.ExtraFrame.STEM = false;
   else
     {
            if(SelAndFillDB.SelectDB.DumpTry(0))
            {
            main.ExtraFrame.STEM = true;
            main.ExtraFrame.WSD = false;
            if (jSliderElab.getValue() == 2)
                  main.ExtraFrame.WSD = true;
            estatus();
      }
      else
      {
            jSliderElab.setValue(0);
            javax.swing.JOptionPane.showMessageDialog(this, "Si prega di inserire i dati STEMMER nel DB!", "Errore",
0);
            estatus();
      }
     }

(...)

  void jButtonImpStem_actionPerformed(ActionEvent e, String whtb)
  {
                  try
                  {
                  String wt = "Wait...";
                  this.jLabImpStemStatus0.setText(wt);
            if(whtb=="DIZ"||whtb=="ALL")
            this.jLabImpStemStatus1.setText(wt);
            if(whtb=="ABBERV_EN"||whtb=="ALL")
            this.jLabImpStemStatus2.setText(wt);
            if(whtb=="STOPLIST"||whtb=="ALL")
            this.jLabImpStemStatus3.setText(wt);
            if(whtb=="VERBI_EN"||whtb=="ALL")
            this.jLabImpStemStatus4.setText(wt);
                  }
                  finally
                  {
                  try
                  {
      contentPane.setCursor(new Cursor(Cursor.WAIT_CURSOR));
                  int risp = JOptionPane.showConfirmDialog(null, "Are you sure to re/fill table/s "+whtb+"?", "Fill
Table", JOptionPane.YES_NO_OPTION);
                  if(risp == 0)
                        {
                        String args[] = new String[2];
                        args[0] = whtb;
                        args[1] = main.ExtraFrame.DBname;
                        SelAndFillDB.Expdmp.main(args);
                        }
                  }
                  finally
                  {
                  estatus();
                  contentPane.setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
                  }
                  }
  }

  void jButtonSetUSERPW_actionPerformed(ActionEvent e)
  {
                  main.ExtraFrame.DBpw = this.jTextPW.getText();
                  main.ExtraFrame.DBuser = this.jTextUSER.getText();

                  if((main.ExtraFrame.DBname!="ERR")&(!
SelAndFillDB.SelectDB.DBTry(main.ExtraFrame.DBname)))
                  {
```

```
                                jRadioButtonSelDB00.setSelected(false);
                                jRadioButtonSelDB0.setSelected(false);
            jRadioButtonSelDB1.setSelected(false);
            jRadioButtonSelDB2.setSelected(false);
            jRadioButtonSelDB3.setSelected(false);
            jRadioButtonSelDB4.setSelected(false);
            jRadioButtonSelDB5.setSelected(false);
            main.ExtraFrame.DBconnection = SelAndFillDB.SelectDB.DBConn("ERR");
            main.ExtraFrame.DBdriver = SelAndFillDB.SelectDB.DBDriver("ERR");
            main.ExtraFrame.DBname = "ERR";
            main.ExtraFrame.JinOracle = false;
                                estatus();
                                JOptionPane.showMessageDialog(this, "User and password are NOT
CORRECT!\nConnection reset!", "Errore", 0);
                        }
                        else
                        {
                                String asterisk = "";
                                for(int i=0;i<main.ExtraFrame.DBpw.length();i++)
                                        asterisk += "*";
                                JOptionPane.showMessageDialog(this, "New user: "+main.ExtraFrame.DBuser+"\nNew
password: "+asterisk, "Info", 1);
                        }
 }

 void jButtonSetUSERPW2_actionPerformed(ActionEvent e)
 {
            try
            {
  java.io.File f1 = new java.io.File("./config.fil");

            java.io.DataInputStream ff0 = new java.io.DataInputStream(new java.io.FileInputStream(f1));
  String d = ff0.readLine();
  ff0.close();

  java.io.DataOutputStream ff1 = new java.io.DataOutputStream(new java.io.FileOutputStream(f1));
            ff1.writeBytes(d+"\n");
            ff1.writeBytes(main.ExtraFrame.DBuser+"\n");
            ff1.writeBytes(main.ExtraFrame.DBpw+"\n");
            ff1.close();
            }
            catch(java.io.IOException er){
                        javax.swing.JOptionPane.showMessageDialog(this, "Errore nella scrittura del file di configurazione!\
n"+er, "Errore", 0);
                        return;
            }
            javax.swing.JOptionPane.showMessageDialog(this, "User "+main.ExtraFrame.DBuser+" and current password
are now set as default!", "Info", 1);
 }

 void jButtonConfSelDBsetdefault_actionPerformed(ActionEvent e)
 {
            try
            {
  java.io.File f1 = new java.io.File("./config.fil");

  java.io.DataInputStream ff0 = new java.io.DataInputStream(new java.io.FileInputStream(f1));
  ff0.readLine();
  String u = ff0.readLine();
  String p = ff0.readLine();
  ff0.close();

            java.io.DataOutputStream ff1 = new java.io.DataOutputStream(new java.io.FileOutputStream(f1));
            ff1.writeBytes(main.ExtraFrame.DBname+"\n");
            ff1.writeBytes(u+"\n");
            ff1.writeBytes(p+"\n");
            ff1.close();
            }
            catch(java.io.IOException er){
                        javax.swing.JOptionPane.showMessageDialog(this, "Errore nella scrittura del file di configurazione!\
n"+er, "Errore", 0);
                        return;
            }
            javax.swing.JOptionPane.showMessageDialog(this, "Database "+main.ExtraFrame.DBname+" is now set as
default!", "Info", 1);
```

```
}

public void estatus(){
        this.e[0] = "OK";
 for(int i=1;i<5;i++)
        if(!SelAndFillDB.SelectDB.DumpTry(i))
        {
                this.e[i] = "ERROR";
                this.e[0] = "ERROR";
        }
        else
        {
                this.e[i] = "OK";
        }
        this.jLabImpStemStatus0.setText(e[0]);
        this.jLabImpStemStatus1.setText(e[1]);
        this.jLabImpStemStatus2.setText(e[2]);
        this.jLabImpStemStatus3.setText(e[3]);
        this.jLabImpStemStatus4.setText(e[4]);
}

(...)

void infSelDBx_actionPerformed(ActionEvent e, int whInf)
{
        String info = "NO DB SELECTED";
        String msg = "";
        int star = 0;
        if(whInf==-1)           {info = "Oracle JI";                star = 4; msg="Global vote: 4/5\nSpeed:
4/5\nInstallation: 0/5\nComment: fast; only for expert.";}
        else if(whInf==0){info = "Oracle JO";                       star = 5; msg="Global vote: 5/5\nSpeed:
5/5\nInstallation: 1/5\nComment: very fast; only for semi-expert.";}
        else if(whInf==1){info = "MySql 4,5 o 6";  star = 2; msg="Global vote: 2/5\nSpeed: 1/5\nInstallation:
4/5\nComment: very slow; simple.";}
        else if(whInf==2){info = "MonetDB 5";                       star = 0; msg="Global vote: 0/5\nSpeed:
2/5\nInstallation: 2/5\nComment: sometimes not work; poor.";}
        else if(whInf==3){info = "FireBird 2.x";    star = 3; msg="Global vote: 3/5\nSpeed: 2/5\nInstallation:
3/5\nComment: easy but not very fast.";}
        else if(whInf==4){info = "Postgres 8.x";    star = 5; msg="Global vote: 5/5\nSpeed: 4/5\nInstallation:
5/5\nComment: simple, fast and complete.";}
        else if(whInf==5){info = "NO INFO";                                 star = 0; msg="Global
vote: ?/5\nSpeed: ?/5\nInstallation: ?/5\nComment: .";}

        JOptionPane.showMessageDialog(this,msg+"\n","Info about: "+info,0,new
ImageIcon(ExtraFrame_AboutBox.class.getResource("../star"+star+".gif")));
}

void jRadioButtonSelDB00_itemStateChanged(ItemEvent e)
{
 if (jRadioButtonSelDB00.isSelected())
   {
        String wdb = "ORACLE";
    this.dbok = SelAndFillDB.SelectDB.DBTry(wdb);
    jRadioButtonSelDB00.setSelected(dbok);
    if(!dbok)
        JOptionPane.showMessageDialog(this, "Errore nell'apertura del DataBase "+wdb+"!\n", "Errore", 0);
    else
    {
        jRadioButtonSelDB0.setSelected(false);
        jRadioButtonSelDB1.setSelected(false);
        jRadioButtonSelDB2.setSelected(false);
        jRadioButtonSelDB3.setSelected(false);
        jRadioButtonSelDB4.setSelected(false);
        jRadioButtonSelDB5.setSelected(false);
        main.ExtraFrame.DBconnection = SelAndFillDB.SelectDB.DBConn(wdb);
        main.ExtraFrame.DBdriver = SelAndFillDB.SelectDB.DBDriver(wdb);
        main.ExtraFrame.DBname = wdb;
        main.ExtraFrame.JinOracle = true;
        estatus();
    }
   }
  else
        if(!(jRadioButtonSelDB0.isSelected()||jRadioButtonSelDB1.isSelected()
                ||jRadioButtonSelDB2.isSelected()||jRadioButtonSelDB3.isSelected()
                ||jRadioButtonSelDB4.isSelected()||jRadioButtonSelDB5.isSelected()))
```

```
                {
                        main.ExtraFrame.DBname = "ERR";
                        main.ExtraFrame.DBconnection = "ERR";
                        main.ExtraFrame.DBdriver = "ERR";
                }

}

void jRadioButtonSelDB0_itemStateChanged(ItemEvent e)
{
 if (jRadioButtonSelDB0.isSelected())
   {
        String wdb = "ORACLE";
    this.dbok = SelAndFillDB.SelectDB.DBTry(wdb);
    jRadioButtonSelDB0.setSelected(dbok);
    if(!dbok)
        JOptionPane.showMessageDialog(this, "Errore nell'apertura del DataBase "+wdb+"!\n", "Errore", 0);
    else
    {
        jRadioButtonSelDB00.setSelected(false);
        jRadioButtonSelDB1.setSelected(false);
        jRadioButtonSelDB2.setSelected(false);
        jRadioButtonSelDB3.setSelected(false);
        jRadioButtonSelDB4.setSelected(false);
        jRadioButtonSelDB5.setSelected(false);
        main.ExtraFrame.DBconnection = SelAndFillDB.SelectDB.DBConn(wdb);
        main.ExtraFrame.DBdriver = SelAndFillDB.SelectDB.DBDriver(wdb);
        main.ExtraFrame.DBname = wdb;
        main.ExtraFrame.JinOracle = false;
        estatus();
    }
   }
   else
        if(!(jRadioButtonSelDB00.isSelected()||jRadioButtonSelDB1.isSelected()
                ||jRadioButtonSelDB2.isSelected()||jRadioButtonSelDB3.isSelected()
                ||jRadioButtonSelDB4.isSelected()||jRadioButtonSelDB5.isSelected()))
                {
                        main.ExtraFrame.DBname = "ERR";
                        main.ExtraFrame.DBconnection = "ERR";
                        main.ExtraFrame.DBdriver = "ERR";
                }
}

void jRadioButtonSelDB1_itemStateChanged(ItemEvent e)
{
 if (jRadioButtonSelDB1.isSelected())
   {
        String wdb = "MYSQL";
    this.dbok = SelAndFillDB.SelectDB.DBTry(wdb);
    jRadioButtonSelDB1.setSelected(dbok);
    if(!dbok)
        JOptionPane.showMessageDialog(this, "Errore nell'apertura del DataBase "+wdb+"!\n", "Errore", 0);
    else
    {
        jRadioButtonSelDB00.setSelected(false);
        jRadioButtonSelDB0.setSelected(false);
        jRadioButtonSelDB2.setSelected(false);
        jRadioButtonSelDB3.setSelected(false);
        jRadioButtonSelDB4.setSelected(false);
        jRadioButtonSelDB5.setSelected(false);
        main.ExtraFrame.DBconnection = SelAndFillDB.SelectDB.DBConn(wdb);
        main.ExtraFrame.DBdriver = SelAndFillDB.SelectDB.DBDriver(wdb);
        main.ExtraFrame.DBname = wdb;
        main.ExtraFrame.JinOracle = false;
        estatus();
    }
   }
   else
        if(!(jRadioButtonSelDB0.isSelected()||jRadioButtonSelDB00.isSelected()
                ||jRadioButtonSelDB2.isSelected()||jRadioButtonSelDB3.isSelected()
                ||jRadioButtonSelDB4.isSelected()||jRadioButtonSelDB5.isSelected()))
                {
                        main.ExtraFrame.DBname = "ERR";
                        main.ExtraFrame.DBconnection = "ERR";
                        main.ExtraFrame.DBdriver = "ERR";
```

```
                }
}

void jRadioButtonSelDB2_itemStateChanged(ItemEvent e)
{
 if (jRadioButtonSelDB2.isSelected())
   {
        String wdb = "MONETDB";
    this.dbok = SelAndFillDB.SelectDB.DBTry(wdb);
    jRadioButtonSelDB2.setSelected(dbok);
    if(!dbok)
        JOptionPane.showMessageDialog(this, "Errore nell'apertura del DataBase "+wdb+"!\n", "Errore", 0);
    else
    {
        jRadioButtonSelDB00.setSelected(false);
        jRadioButtonSelDB0.setSelected(false);
        jRadioButtonSelDB1.setSelected(false);
        jRadioButtonSelDB3.setSelected(false);
        jRadioButtonSelDB4.setSelected(false);
        jRadioButtonSelDB5.setSelected(false);
        main.ExtraFrame.DBconnection = SelAndFillDB.SelectDB.DBConn(wdb);
        main.ExtraFrame.DBdriver = SelAndFillDB.SelectDB.DBDriver(wdb);
        main.ExtraFrame.DBname = wdb;
        main.ExtraFrame.JinOracle = false;
        estatus();
    }
   }
   else
        if(!(jRadioButtonSelDB0.isSelected()||jRadioButtonSelDB1.isSelected()
                ||jRadioButtonSelDB00.isSelected()||jRadioButtonSelDB3.isSelected()
                ||jRadioButtonSelDB4.isSelected()||jRadioButtonSelDB5.isSelected()))
                {
                        main.ExtraFrame.DBname = "ERR";
                        main.ExtraFrame.DBconnection = "ERR";
                        main.ExtraFrame.DBdriver = "ERR";
                }
}

 void jRadioButtonSelDB3_itemStateChanged(ItemEvent e)
{
 if (jRadioButtonSelDB3.isSelected())
   {
        String wdb = "FIREBIRD";
    this.dbok = SelAndFillDB.SelectDB.DBTry(wdb);
    jRadioButtonSelDB3.setSelected(dbok);
    if(!dbok)
        JOptionPane.showMessageDialog(this, "Errore nell'apertura del DataBase "+wdb+"!\n", "Errore", 0);
    else
    {
        jRadioButtonSelDB00.setSelected(false);
        jRadioButtonSelDB0.setSelected(false);
        jRadioButtonSelDB1.setSelected(false);
        jRadioButtonSelDB2.setSelected(false);
        jRadioButtonSelDB4.setSelected(false);
        jRadioButtonSelDB5.setSelected(false);
        main.ExtraFrame.DBconnection = SelAndFillDB.SelectDB.DBConn(wdb);
        main.ExtraFrame.DBdriver = SelAndFillDB.SelectDB.DBDriver(wdb);
        main.ExtraFrame.DBname = wdb;
        main.ExtraFrame.JinOracle = false;
        estatus();
    }
   }
   else
        if(!(jRadioButtonSelDB0.isSelected()||jRadioButtonSelDB1.isSelected()
                ||jRadioButtonSelDB2.isSelected()||jRadioButtonSelDB00.isSelected()
                ||jRadioButtonSelDB4.isSelected()||jRadioButtonSelDB5.isSelected()))
                {
                        main.ExtraFrame.DBname = "ERR";
                        main.ExtraFrame.DBconnection = "ERR";
                        main.ExtraFrame.DBdriver = "ERR";
                }
}

 void jRadioButtonSelDB4_itemStateChanged(ItemEvent e)
{
```

```
 if (jRadioButtonSelDB4.isSelected())
   {
        String wdb = "POSTGRES";
    this.dbok = SelAndFillDB.SelectDB.DBTry(wdb);
    jRadioButtonSelDB4.setSelected(dbok);
    if(!dbok)
        JOptionPane.showMessageDialog(this, "Errore nell'apertura del DataBase "+wdb+"!\n", "Errore", 0);
    else
    {
        jRadioButtonSelDB00.setSelected(false);
        jRadioButtonSelDB0.setSelected(false);
        jRadioButtonSelDB1.setSelected(false);
        jRadioButtonSelDB2.setSelected(false);
        jRadioButtonSelDB3.setSelected(false);
        jRadioButtonSelDB5.setSelected(false);
        main.ExtraFrame.DBconnection = SelAndFillDB.SelectDB.DBConn(wdb);
        main.ExtraFrame.DBdriver = SelAndFillDB.SelectDB.DBDriver(wdb);
        main.ExtraFrame.DBname = wdb;
        main.ExtraFrame.JinOracle = false;
        estatus();
    }
   }
   else
        if(!(jRadioButtonSelDB0.isSelected()||jRadioButtonSelDB1.isSelected()
                ||jRadioButtonSelDB2.isSelected()||jRadioButtonSelDB3.isSelected()
                ||jRadioButtonSelDB00.isSelected()||jRadioButtonSelDB5.isSelected()))
                {
                        main.ExtraFrame.DBname = "ERR";
                        main.ExtraFrame.DBconnection = "ERR";
                        main.ExtraFrame.DBdriver = "ERR";
                }
}

 void jRadioButtonSelDB5_itemStateChanged(ItemEvent e)
 {
 if (jRadioButtonSelDB5.isSelected())
   {
        String wdb = "VUOTO";
    this.dbok = SelAndFillDB.SelectDB.DBTry(wdb);
    jRadioButtonSelDB5.setSelected(dbok);
    if(!dbok)
        JOptionPane.showMessageDialog(this, "Errore nell'apertura del DataBase "+wdb+"!\n", "Errore", 0);
    else
    {
        jRadioButtonSelDB00.setSelected(false);
        jRadioButtonSelDB0.setSelected(false);
        jRadioButtonSelDB1.setSelected(false);
        jRadioButtonSelDB2.setSelected(false);
        jRadioButtonSelDB3.setSelected(false);
        jRadioButtonSelDB4.setSelected(false);
        main.ExtraFrame.DBconnection = SelAndFillDB.SelectDB.DBConn(wdb);
        main.ExtraFrame.DBdriver = SelAndFillDB.SelectDB.DBDriver(wdb);
        main.ExtraFrame.DBname = wdb;
        main.ExtraFrame.JinOracle = false;
        estatus();
    }
   }
   else
        if(!(jRadioButtonSelDB0.isSelected()||jRadioButtonSelDB1.isSelected()
                ||jRadioButtonSelDB2.isSelected()||jRadioButtonSelDB3.isSelected()
                ||jRadioButtonSelDB4.isSelected()||jRadioButtonSelDB00.isSelected()))
                {
                        main.ExtraFrame.DBname = "ERR";
                        main.ExtraFrame.DBconnection = "ERR";
                        main.ExtraFrame.DBdriver = "ERR";
                }
 }

 (...)
 }
}
```

A.10 Note varie

Note legali ed altro:

Microsoft e MS-DOS sono marchi registrati e Windows è un marchio registrato di Microsoft Corporation.
Oracle è un marchio registrato di Oracle Corporation.
UNIX è un marchio registrato di UNIX Systems Laboratories.
MySQL è un marchio registrato di MySQL AB negli Stati Uniti e in altri Paesi.
Altri nomi di prodotti possono essere marchi registrati dei rispettivi proprietari.

I loghi per quanto proprietari non sono utilizzati per fini espressamente vietati dalle compagnie stesse.
Le compagnie proprietarie dei marchi registrati non sono responsabili del contenuto di questo documento.
Qualsiasi affermazione può essere verificata, discussa o smentita contattando l'autore del documento.
Le reali caratteristiche dei sistemi proprietari si possono verificare seguendo le note bibliografiche.

In queste pagine possono essere rappresentati marchi registrati.
Si ritiene che essi possano essere riprodotti su questo documento - limitatamente alle voci che riguardano direttamente l'azienda proprietaria e i prodotti della stessa - in osservanza del Codice della proprietà industriale e della rimanente normativa di settore, in quanto utilizzato a meri fini informativi e descrittivi e comunque per finalità non commerciali e non apposto su prodotti, di alcuna sorta, destinati al mercato commerciale.
La natura del segno distintivo marchio registrato si desume dall'immagine stessa, recante il contrassegno ™ o ®
Queste immagini e files sono utilizzati rispettando i requisiti imposti dalla Exemption Doctrine Policy.
Si ritiene che non sia ragionevolmente possibile ottenere files equivalenti dal punto di vista illustrativo e dotati di una licenza libera.
L'immagine deve rispettare le seguenti indicazioni:
*Dev'essere indicata la fonte e specificato chi è il detentore del copyright (qualora non esplicitato dal tag di licenza) * * VEDI NOTE BIBLIOGRAFICHE.*
Il file dev'essere usato solo nel namespace principale, in particolare nelle voci a cui il suo contenuto si riferisce direttamente.
Le immagini non possono essere inserite nelle gallerie di immagini.
Se non è evidente, occorre specificare dettagliatamente il motivo per cui il file non possa essere sostituibile con uno sotto licenza libera. Se non c'è un motivo razionale, ne si richieda la cancellazione immediata.

Bibliografia:

[1] G. Navarro,
A guided Tour to Approximate String Matching.
Technical report, Dept. of Computer Science, University of Chile, 2001.

[2] E. Ukkonen,
Algorithms for Approximate String Matching.
In Information and Control, 1985.

[3] R. Martoglia,
EXTRA: Progetto e Sviluppo di un Ambiente per Traduzioni Multilingua Assistite.
Tesi di Laurea, Università degli Studi di Modena e Reggio Emilia, 2000/2001.

[4] F. Gavioli,
Progetto ed Implementazione di un Algoritmo per Ricerca di Similarità tra Frasi.
Tesi di laurea, Università degli studi di Modena e Reggio Emilia, 1999/2000.

[5] F. Mandreoli, R. Martoglia, P. Tiberio,
A Syntatic Ap-proach for Searching Similarities whithin Sentences.
Proc. of the 11th ACM Conference of Information and Knowledge Management (ACM CIKM), anno 2002.

[6] E. Ukkonen,
Approximate String Matching with q-grams and Maximal Matches.
Theoretical Computer Science, anno 1992.

[7] L. Gravano, P. Ipeirotis, H. Jagadish, N. Koudas, S. Muthukrishnan, D. Srivastava,
Approximate String Joins in a Database (Almost) for Free.
In Proceedings of 27th VLDB Conference, 2001.

[8] D. Gelmini,
All-Against-All Sequence Matching: Implementazione mediante Suffix Array e Analisi Prestazionale Comparata.
Tesi di laurea, Università degli studi di Modena e Reggio Emilia, 2002/2003.

[9] E.Sutien, J.Tarhio,
On Using q-gram Locations in Approximate String Matching.
Proc. of 3rd Annual European Symposium, 1995.

[10] E.Sutien, J.Tarhio,
Filtration with q-samples in Approximate String Mat-ching,
Proc. of the 7th annual Symposium on Combinatorial Pattern Matching, 1996.

[11] F. Mandreoli R. Martoglia P. Tiberio,
Searching Similar (Sub)Sentences for Example-Based Machine Translation.
In Atti del Decimo Convegno Nazionale su Sistemi Evoluti per Basi di Dati (SEBD 2002).

[12] E. Stefanini,
Risoluzione di ambiguità semantiche per la ricerca di similarità tra frasi.
Tesi di Laurea Università degli studi di Modena e Reggio Emilia,2002/2003

[13] N. Ide, J. Veronis,
Word Sense disambiguation: The State of Art.
Computational Linguistics, vol. 24, number 1, p. 1-40 (1998).

[14] Information Systems Group
University of Modena and Reggio Emilia
(http://www.isgroup.unimo.it)

[15] Oracle 8i,
SQL Reference, 2000.
JDBC Reference, 2000.
SQLJ Developer's Guide, 1999.
Java Developer's Guide, 2000.
Java Stored Procedures Developer's Guide, 1999.
InterMedia Text - Reference, 1999.
Oracle Corporation.

[16] Oracle 9i

1) Products references:
http://www.oracle.com/technology/products/oracle9i/index.html
Oracle Corporation

2) M. Bigatti, *L'Oracolo e l'Open Source.*

[17] MySQL

1) Reference Manuals at: *http://dev.mysql.com/doc/#refman*

Reference Manuals v. 4.x at: *http://dev.mysql.com/doc/refman/4.1/en/*
Reference Manuals v. 5.x at:
http://dev.mysql.com/doc/refman/5.0/en/index.html or
http://dev.mysql.com/doc/refman/5.1/en/index.html
Reference Manuals v. 6.x at: *http://dev.mysql.com/doc/refman/6.0/en/*

MySQL AB.

2) R. J.Yarger, G. Reese, T. King, *MySQL & mSQL.*

[18] FireBird (riferimeni alla versione 2.1)
Mauali ufficiali in italiano: http://www.firebirdsql.org/manual/it/
Guida veloce:
http://www.firebirdsql.org/pdfmanual/Firebird-2.0-QuickStart.pdf
Firebird Project

[19] MonetDB/SQL (riferimenti alla verisione 5.4)
Documentation:
http://monetdb.cwi.nl/projects/monetdb/SQL/Documentation/index.html
CWI

[20] PostgreSQL (riferimenti alla versione 8.2 e 8.3)

1) Documentation: http://www.postgresql.org/docs/
PostgreSQL Global Development Group

2) H. Chen, H. Lim, J. Xiao,
Concept Architecture of PostgreSQL.
University of Waterloo