STUDIA INFORMATICA 2011
Volume 32 Number 2B (97)

Krzysztof CZAJKOWSKI, Dominik ZIAJKA
Politechnika Krakowska, Instytut Teleinformatyki,
Wydział Fizyki, Matematyki i Informatyki

ZAAWANSOWANE INDEKSOWANIE I WYSZUKIWANIE DANYCH W SYSTEMIE ORACLE

Streszczenie. Artykuł omawia problematykę zaawansowanego indeksowania i wyszukiwania danych w środowiskach baz danych. Zagadnienia zostały przedstawione na przykładzie rozwiązań firmy Oracle: Oracle Text, Oracle Ultra Search i Oracle Secure Enterprise Search. W artykule zaprezentowano przykładowe implementacje ukazujące możliwości tych technologii w różnorodnych zastosowaniach. W pracy zawarto również wyniki eksperymentów wykonanych na przygotowanych aplikacjach.

Słowa kluczowe: zaawansowane indeksowanie, zaawansowane wyszukiwanie, Oracle Text, Oracle Ultra Search, Oracle Secure Enterprise Search

ADVANCED DATA INDEXING AND RETRIEVING IN ORACLE SYSTEM

Summary. The article discusses the issues of advanced indexing and retrieving data in database environments. The issues are presented on the example of Oracle solutions: Oracle Text, Oracle Ultra Search and Oracle Secure Enterprise Search. The article presents examples which showing implementations of these technologies in various applications. The paper also contains results of experiments carried out on the prepared applications.

Keywords: advanced indexing, advanced retrieving, Oracle Text, Oracle Ultra Search, Oracle Secure Enterprise Search

1. Wstęp

Indeksowanie danych oraz ich wyszukiwanie to zagadnienia z natury skomplikowane. Tematyka ta jest obszarem prowadzenia intensywnych badań naukowych oraz realizacji ko-

lejnych implementacji praktycznych. Rozwiązania istniejące wciąż nie są satysfakcjonujące, a ciągły postęp informatyzacji sprawia, że ilość informacji nieustannie wzrasta. Powoduje to, że problemy związane z uzyskiwaniem dostępu do pożądanych informacji nieustannie się komplikują. Ilość gromadzonych informacji, ich złożoność oraz konieczność jak najszybszego uzyskiwania odpowiednich danych sprawiają, że waga problemu wzrasta. Nie wydaje się możliwe opracowanie jednego, uniwersalnego sposobu jego rozwiązania, konieczne są jednak prace mające na celu dostarczenie użytkownikom jak najszerszej palety możliwości w tym zakresie.

Jedną z czołowych firm zajmujących się problematyką gromadzenia i przetwarzania danych jest firma Oracle. Pośród różnych rozwiązań udostępnianych przez tego producenta w zakresie indeksowania oraz wyszukiwania danych, na uwagę zasługują rozwiązania *Oracle Text, Oracle Ultra Search* oraz *Oracle Secure Enterprise Search*.

W artykule omówiono powyższe rozwiązania oraz zaprezentowano aplikacje wykorzystujące je w praktyczny sposób. Przedstawiono również wyniki eksperymentów przeprowadzonych na zbiorach danych o różnych wielkościach.

2. Przeszukiwanie pełnotekstowe

Przeszukiwanie pełnotekstowe (ang. *Full Text Search*) jest techniką przeszukiwania dokumentów, wykorzystującą indeks pełnotekstowy (ang. *Full Text Index*), czyli indeks słów kluczowych zawartych w dokumencie. Tworzenie indeksu pełnotekstowego polega na przetworzeniu dokumentu, utworzeniu wektora słów w nim zawartych, oddzielenia od stop-słów (słowa nieniosące za sobą informacji), lematyzacji i utworzeniu indeksu zawierającego częstość występowania danego słowa kluczowego w konkretnym dokumencie. Przykładowym indeksem jest TF-IDF (ang. *TF – Term Frequency, IDF – Inverse Document Frequency*).

Częstość wystąpień danego termu w tekście (*tf*) wyraża się wzorem:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}, \tag{1}$$

gdzie: n_{ij} – liczba wystąpień termu t_i w dokumencie d_j, $\sum_k n_k$ – suma liczby wystąpień termów w dokumencie d_j.

Odwrotna częstość w dokumentach:

$$idf_i = \log \frac{|D|}{|\{d : t_i \in d\}|}, \tag{2}$$

gdzie: $|D|$ – liczba dokumentów w korpusie, $|\{d:t_j \in d\}|$ – liczba dokumentów zawierających przynajmniej jedno wystąpienie danego termu.

W ostateczności współczynnik *TF-IDF* wyraża się wzorem:

$$(tf - idf)_{i,j} = tf_{i,j} \times idf_i. \quad (3)$$

W efekcie dla każdego termu i dokumentu istnieje współczynnik występowania danego termu w danym dokumencie, służący do szybkiego wyszukiwania fraz.

Technika *FTS* wykorzystywana jest w rozwiązaniach służących do przeszukiwania tekstów, takich jak: wyszukiwarki internetowe, zbiory danych tekstowych czy też relacyjnych baz danych.

FTS został zaimplementowana w bazach danych, takich jak: MySQL (od wersji 3.23.23 w silniku MyISAM [5], od wersji 4.1 wprowadzono również złożone przeszukiwanie [11]), PostgreSQL (od wersji 8.3 natywna obsługa *FTS* [3]), Oracle oraz MSSQL. Występują jednak również systemy nieoparte na RDBMS, takie jak Apache Lucene [2]. W przypadku systemu Oracle, w wersji Oracle 8 wprowadzono moduł *ConText*, realizujący funkcje FTS. W wersji 8i moduł ten wbudowano w system baz danych oraz nazwano *interMedia Text* (był on całkowicie przebudowany w stosunku do pierwowzoru). Wraz z wersją Oracle 9i, rozszerzono jego możliwości, a nazwa uległa zmianie na *Oracle Text* (przy pełnej zgodności z wersją poprzednią) [9].

3. Indeksowanie i wyszukiwanie danych za pomocą Oracle Text

3.1. Zastosowanie

Oracle Text (*OT*) jest narzędziem umożliwiającym indeksowanie oraz wyszukiwanie dokumentów tekstowych. Bazuje na technice *FTS,* tworząc indeks dokumentów zwany Oracle *Text Index*. Udostępnia interfejs SQL do przeszukiwania dokumentów.

OT wspiera trzy typy aplikacji: kolekcje dokumentów (ang. *Document Collection Applications*), informacje katalogowe (ang. *Catalog Information Applications*) oraz klasyfikację dokumentów (ang. *Document Classification Applications*). Każdej aplikacji odpowiada inny indeks, odpowiednio *CONTEXT*, *CTXCAT*, *CTXRULE*. Różnią się one pomiędzy sobą możliwym źródłem danych, wydajnością oraz zastosowaniem.

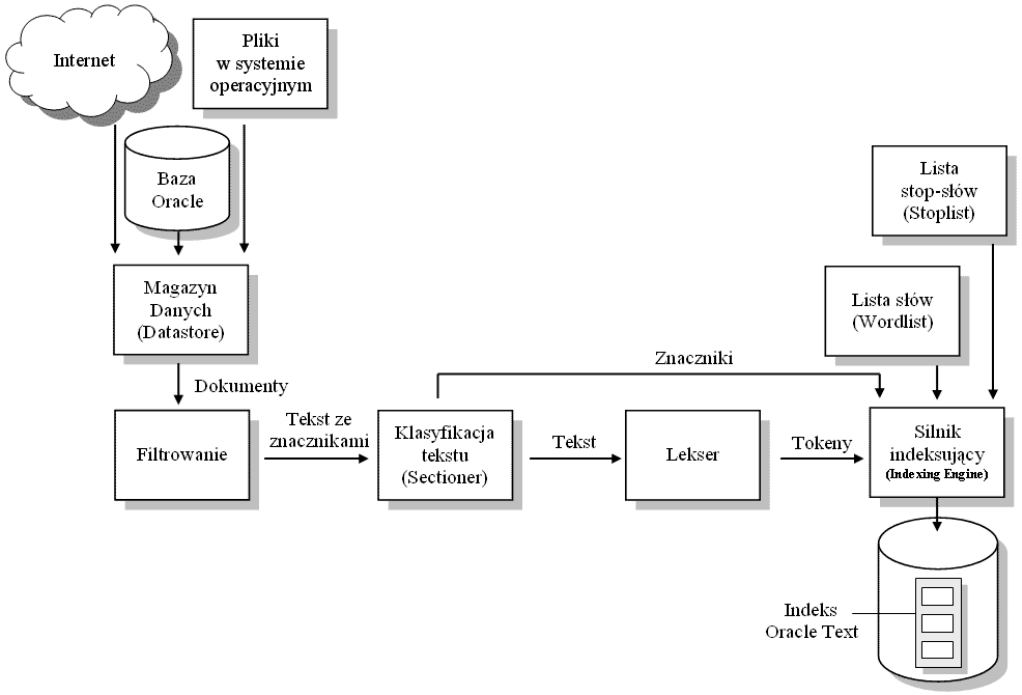
Aplikacje typu kolekcja dokumentu są tworzone w celu magazynowania dużej liczby danych. Wymagają dłuższego czasu indeksowania, oferując szybkie przeszukiwanie. Wykorzystuje się w nich indeks *CONTEXT*, który umożliwia tworzenie zbiorów danych zarówno z czystych danych tekstowych, jak i z materiałów zewnętrznych. W przeciwieństwie do pozostałych indeksów może wykorzystywać kolumny z zapisanymi referencjami do innych materiałów, takich jak linki WWW lub pliki z lokalnego systemu plików.

Kolejnym typem aplikacji są aplikacje zawierające informacje katalogowe, czyli takie, które mają bezpośrednio z sobą powiązane informacje tekstowe i wartości liczbowe, np. liczba sztuk produktu o określonym opisie słownym. Wymagają one szybkiego czasu odpowiedzi na zapytanie w złożonych warunkach – jednoczesne przeszukiwanie po polu tekstowym oraz liczbowym. Wykorzystuje się indeks *CTXCAT*, łączący w sobie dane tekstowe z liczbowymi. Indeks synchronizowany jest automatycznie.

Ostatnim typem są aplikacje, które służą do klasyfikacji dokumentów, korzystając z indeksu *CTXRULE* w celu dopasowania odpowiedniego zasobu do konkretnej czynności, takich jak: wysłanie e-maila z informacją, przypisanie elementu czy też zachowanie w bazie danych.

3.2. Proces tworzenia

Proces tworzenia indeksu rozpoczyna się od uzyskania danych do przetworzenia z magazynu danych (*Datastore*) – rysunek 1. *OT* dopuszcza trzy możliwości pobierania danych: pobieranie z Internetu, lokalnego systemu plików oraz bezpośrednio z bazy danych Oracle. W zależności od typu tworzonego indeksu, dostępne są odpowiednie możliwości. Informacją o tym, skąd mają być pobierane dane, jest typ kolumny w bazie danych.



Rys. 1. Proces tworzenia indeksu
Fig. 1. The process of creating an index

Po uzyskaniu danych następuje ich filtracja, czyli proces ujednolicenia otrzymywanego formatu danych oraz kodowania. Pliki tekstowe, HTML lub XML nie wymagają filtracji formatu.

Pliki binarne (dokumenty formatowane), takie jak pliki programu Microsoft Word (.doc/.docx) czy PDF, muszą zostać przefiltrowane do tekstu ze znacznikami, zachowując jednocześnie możliwość wyróżnienia ważniejszych fraz w tekście.

Klasyfikowanie tekstu (*Sectioner*) polega na rozdzieleniu znaczników od tekstu. Znaczniki służą do obliczania wag danych fraz lub tekstu i są uwzględniane w procesie indeksowania.

W lekserze następuje proces analizy leksykalnej tekstu, który konwertuje słowa występujące w tekście na tokeny, czyli wyrazy posiadające określone znaczenie. Możliwe jest zdefiniowanie znaków rozdzielających tokeny, takich jak np. białe znaki lub zdefiniowanie formy, w jakiej mają zostać przesłane do silnika indeksującego – konwertowane do małych, dużych liter lub w niezmienionej formie.

Silnik indeksujący (*Indexing Engine*) odpowiada za utworzenie odwrotnego indeksu, który dopasowuje słowa do danego dokumentu. Na tym etapie można zdefiniować słowa stopu, które mają zostać ignorowane w procesie budowania indeksu – nie będą wpływać na wyniki przeszukiwania. Dodatkowo można zdefiniować listę słów wykorzystywaną przy procesie tworzenia indeksów prefiksowych lub indeksów zawierających daną frazę.

Po przetworzeniu powstaje indeks słów kluczowych, wykorzystywany podczas przeszukiwania.

3.3. Tworzenie indeksów

Tworzenie indeksu polega na wykonaniu polecania *CREATE INDEX* z dodatkowymi parametrami. Jeśli nie zdefiniowano parametrów automatycznie, mają zastosowanie parametry domyślne. Podstawowe opcje modyfikacji to:

- magazyn danych – sposób przechowywania dokumentów,
- filtr – metoda filtracji do czystego tekstu,
- lekser – definicja języków,
- lista słów – rozszerzanie stemmingu (sprowadzanie słów do ich form podstawowych) i zapytań rozmytych,
- składowanie – określenie sposobu przechowywania indeksu,
- lista stop-słów – definiowanie słów lub fraz, które nie mają zostać zindeksowane,
- grupy sekcji – wykorzystanie sekcji poszczególnych dokumentów.

3.4. Magazyny danych i filtrowania

OT wspiera możliwość definiowania różnych magazynów danych, w których przechowywane są dokumenty:

- *DIRECT_DATASTORE* – dokumenty są składowane bezpośrednio w kolumnie tekstowej, gdzie każdy wiersz jest indeksowany jako osobny dokument. Możliwe typy kolumn: *VARCHAR2, CLOB, BLOB, CHAR, BFILE, XMLType, URIType*. Indeks *CONTEXT* wspiera dodatkowo typ *XMLType*. Jest to magazyn podstawowy.
- *MULTI_COLUMN_DATASTORE* – informacje tekstowe magazynowane są w wielu kolumnach danego wiersza, podczas indeksacji dane są łączone i wspólnie indeksowane.
- *DETAIL_DATASTORE* – uwzględniane są w tym przypadku relacje pomiędzy dokumentami. Wyszczególnia się główny dokument, który może zostać opisywany przez inne dokumenty (relacje) i przechowywany w innych wierszach. Na indeks danego dokumentu składają się tekst głównego dokumentu oraz informacje z powiązań.
- *FILE_DATASTORE* – służy do przechowywania dokumentów w systemie plików. Po stronie bazy danych przechowywane są ścieżki dostępu, gdzie kolejno są one przetwarzane i uwzględniane w procesie indeksowania.
- *NESTED_DATASTORE* – do magazynowania danych wykorzystuje są w tym przypadku tabele zagnieżdżone (ang. *nested table*).
- *URL_DATASTORE* – w tym przypadku przechowywane są odwołania URL do zasobów Internetu lub intranetu. Przetwarzanie polega na pobraniu odpowiednich dokumentów w celu indeksacji. Ze względu na dodatkowy czas na pobranie zewnętrznych danych, czas indeksacji może znacząco wzrosnąć.
- *USER_DATASTORE* – użytkownik definiuje różne źródła danych w celu ich połączenia i przetworzenia.

3.5. Listy słów i stop-słów

Oracle Text wspiera możliwość modyfikacji listy słów indeksowanych, pozwalając rozszerzyć funkcjonalność lub poprawić wydajność przeszukiwania.

W przypadku zwykłej bazy danych (bez modułu *OT*) istnieje możliwość zdefiniowania indeksu przeszukującego według części zadanego słowa. Dostępne są trzy możliwości: dopasowanie lewostronne (np. „%ux” dopasuje do „linux”, „tux”), dopasowanie prawostronne („li%” dopasuje do „linux”, „linus”) i dopasowanie obustronne „%u%” (dopasowujące do „linux”, „linus” etc.). Dla normalnego indeksu lista słów może się znacząco rozrosnąć i wpływać negatywnie na wydajność. *OT* wpiera możliwość wygenerowania tokenów słów prefiksowych lub części ciągu znaków. W tym celu należy zdefiniować odpowiednie ustawienie opcji *BASIC_WORLDLIST*. Początkowo opcja ta nie jest automatycznie ustawiana. Korzystając z niej, można dodatkowo zdefiniować *stemmer* (język, jaki ma zostać wykorzystany) lub metodę dopasowania w przypadku zapytań rozmytych.

Stop-słowa są to słowa najczęściej występujące we wszystkich dokumentach niemających wpływu na wartość przeszukiwania, takich jak „i", „lub". *Oracle Text* dostarcza listę stop-słów dla danego języka, którą użytkownik może modyfikować lub stworzyć własną podczas procesu tworzenia indeksu za pomocą polecenia *CREATE INDEX*. Istnieje możliwość utworzenia listy stop-słów dla wielu języków równocześnie, gdy następuje korzystanie z *MULTI_LEXER*.

3.6. Grupy sekcji i położenie tekstu

Dokumenty, takie jak HTML i XML, zawierają określone sekcje, takie jak np. tag <h1>, określający nagłówek o określonym priorytecie. W *OT* istnieje możliwość tworzenie sekcji ich przeszukiwania. Dostępne sekcje grup:

- *NULL_SECTION_GROUP* – jest to podstawowa grupa wykorzystywana w przypadku braku sekcji lub gdy użytkownik zdefiniuje sekcje *SECTION* lub *PARAGRAPH*.
- *BASIC_SECTION_GROUP* – wykorzystywany do konfiguracji sekcji w formacie <A>...</A>.
- *HTML_SECTION_GROUP* – stosowany do indeksowania plików HTML i definicji sekcji dokumentów HTML.
- *XML_SECTION_GROUP* – sekcje i indeksowanie dokumentów XML.
- *AUTO_SECTION_GROUP* – automatyczne generowanie sekcji plików XML. Sekcje tworzone są na podstawie tagów oraz nazw argumentów. Typ rozróżnia wielkość liter sekcji.
- *PATH_SECTION_GROUP* – wykorzystywany, podobnie jak *AUTO_SECTION_GROUP*, do dokumentów XML, z tą różnicą, że wygenerowane w ten sposób sekcje są możliwe do przeszukania operatorami *INPATH* oraz *HASPATH*.
- *NEWS_SECTION_GROUP* – służy do tworzenia sekcji list dyskusyjnych, opartych na RFC 1036.

3.7. Język dokumentu

OT wspiera pięć możliwości tworzenia leksera dla zadanych tekstów, w zależności od wykorzystywanego języka lub języków:

- *BASIC_LEXER* – podstawowy lekser z opcjami dla wielu języków.
- *AUTO_LEXER* – lekser służący do indeksacji tabeli z dokumentami o różnych językach. Wykrywa język na podstawie danych statycznych struktury języka. W przeciwieństwie do *MULTI_LEXER* nie jest wymagana kolumna językowa.

- *MULTI_LEXER* – lekser służący do indeksacji tabeli z dokumentami o różnych językach. Wymaga zdefiniowania kolumny językowej z wartościami konkretnego języka dla danego dokumentu. Wykorzystywany bezpośrednio do definicji wykorzystywanego języka.
- *USER_LEXER* – możliwość utworzenia własnego leksera przez użytkownika.
- *WORLD_LEXER* – lekser służący do indeksacji tabeli z dokumentami o różnych językach. Nie wymaga kolumny językowej. W stosunku do *AUTO_LEXER* posiada dużo większą granulacje parametrów.

W przypadku niezdefiniowania języka tabel/wierszy, wykorzystywany jest język bazy danych.

3.8. Wyszukiwanie danych z wykorzystaniem Oracle Text

Przeszukiwanie danych polega na porównywaniu frazy przeszukiwanej, często dodatkowymi operatorami, względem zaindeksowanych dokumentów. Silnik *OT* w wyniku zwraca wszystkie dokumenty spełniające podane kryteria wraz z punktacją (ang. *score*) rozumianą jako liczba określająca współczynnik trafności danej frazy w zadanym dokumencie.

Każde przeszukiwanie odbywa się po stronie bazy danych z wykorzystaniem polecenia *SELECT* języka SQL. W zależności od zastosowanego indeksu, stosuje się inny operator przeszukiwania – dla indeksu *CONTEXT* operator *CONTAINS*, dla indeksu *CTXCAT* operator *CATSEARCH*. W celu klasyfikacji dokumentów indeksu *CTXRULE* wykorzystuje się operator *MATCHES*.

4. Oracle Ultra Search i Oracle Secure Enterprise Search

Oracle Ultra Search (US) jest rozszerzeniem do bazy danych *Oracle Database, Oracle iAS/Portal* i *Collaboration Suite*, opartym na *Oracle Database* oraz *Oracle Text*.

Mechanizm ten przemierza zdefiniowane, publiczne źródła danych oraz indeksuje dokumenty w nich zawarte, następnie udostępnia interfejs przeszukiwania przez użytkownika przeanalizowanych zasobów. Dostępnymi źródłami danych mogą być:

- bazy danych (Oracle Database oraz inne przez ODBC),
- serwery pocztowe IMAP,
- dokumenty HTML generowane przez serwery WWW,
- pliki.

US powstał z myślą o zapewnieniu łatwego wyszukiwania informacji w sieciach korporacyjnych.

4.1. Oracle Ultra Search

Rozwiązanie składa się z trzech głównych elementów: *Crawlera* – robota, *Backend* – warstwy przechowywania zgromadzonych informacji, oraz *Middle Tier* – warstwy udostępniającej zgromadzone informacje.

Robot *OUS* (ang. *Oracle Ultra Search Crawler – OUSC*) jest procesem Java, uruchamianym przez serwer Oracle w określonym terminarzu. Uruchamia konfigurowalną liczbę procesów potomnych, które pobierają zdefiniowane źródła danych oraz indeksują je, wykorzystując *OT*. *Crawler* tworzy mapę linków oraz analizuje powiązania pomiędzy nimi. Bazuje na mechanizmie kolejkowania *DBMS_JOB*. W przypadku napotkania dokumentów, których nie można sklasyfikować jako HTML, przetwarza je automatycznie przez mechanizm *OT* w celu ich filtracji oraz indeksowania.

Warstwa *Backend* zawiera w sobie repozytorium przetworzonych dokumentów przez *Crawler* oraz *OT*, który udostępnia mechanizm indeksujący oraz interfejs niższego poziomu (bazującego na SQL) do ich przeszukiwania.

Ostatnią warstwą jest *Middle Tier*. Pełni ona funkcję pośredniczącą pomiędzy użytkownikiem a zgromadzonymi dokumentami w warstwie *Backend*, przez aplikację internetową dostępną przez przeglądarkę oraz zdefiniowane API. Udostępnia aplikację administracyjną (*OUSA – Oracle Ultra Search Administration Tool*), aplikację do przeszukiwania (*QA – Query Application*) oraz API służące do tworzenia własnych aplikacji przeszukiwania. *OUSA* oraz *QA* są to aplikacje zgodne z J2EE i są one trójwarstwowe. Pierwszą stanowi warstwa przeglądarki internetowej, za pomocą której użytkownik otrzymuje dostęp do *OUS*. Kolejną warstwą jest serwer WWW i silnik serweletów. Ostatnią warstwą jest baza danych Oracle.

Oracle Ultra Search Administration Tool jest to aplikacja służąca do zarządzania instancjami *OUS*. Narzędzie to jest niezależne w stosunku do aplikacji przeszukiwania i może być udostępniane przez inny serwer WWW w celu zapewnienia bezpieczeństwa oraz skalowania.

Oracle Ultra Search API udostępnia interfejsy na poziomie języka Java i służy do zarządzania UOS, które dzielą się na:

- API zapytań (ang. *Query API*) – przeszukiwanie informacji bez nadmiarowej warstwy HTML,
- API agent robota (ang. *Crawler Agent API*) – dostępu do przeszukiwania oraz indeksowania zdefiniowanych obiektów,
- API e-mail – dostęp do archiwalnej poczty e-mail, wykorzystywany w aplikacji przeszukiwania do wyświetlenia poczty,
- API do przepisywania URL – używany przez robota do filtrowania i przepisywania znalezionych URL przed dodaniem do kolejki adresów URL do przetworzenia,
- API usługi dokumentów agenta robotów – umożliwia generowanie atrybutów na podstawie treści dokumentów.

Oracle Ultra Search Query Application pełni rolę interfejsu do przeszukiwania danych przez użytkownika.

4.2. Oracle Secure Enterprise Search

Bezpośrednim następcą *OUS* jest *Oracle Secure Enterprise Search (OSES)*. W przeciwieństwie do poprzednika nie jest dodatkiem do istniejącej bazy danych, a odrębnym produktem. Na rynku dostępny jest od 2006 roku. Udostępnia wsparcie dla przeszukiwania prywatnych i chronionych dokumentów. Główne różnice pomiędzy *Oracle Ultra Search* i *Secure Enterprise Search* zamieszczono w tabeli 1.

Tabela 1

Główne różnice pomiędzy Oracle Ultra Search i Secure Enterprise Search [7]

Właściwość	Ultra Search	Secure Enterprise Search
Przeszukiwanie publicznych źródeł danych	Tak	Tak
Przeszukiwanie prywatnych/zabezpieczonych źródeł danych	Nie	Tak
Samodzielny produkt	Nie	Tak
Sposób instalacji	Instalacja modułu bazy danych	Własny instalator
API zapytań	Query Java API	Web Service
Zdalny robot indeksujący	Tak	Nie
Możliwość innego wykorzystania bazy danych	Tak	Nie
Instancje przeszukiwania	Tak	Nie
Agent Robota	Poprzez API	Wykorzystując wtyczki
Aktualizacje	Wraz z wydaniem wersji bazy danych	Osobne aktualizacje
Zintegrowany serwer WWW	Nie	Tak, WebLogic

5. Aplikacja

Aplikacja wykorzystuje rozwiązanie *Oracle Text* w celu indeksowania, przeszukiwania oraz kategoryzacji dokumentów. Możliwe są operacje na danych w następujących formatach:

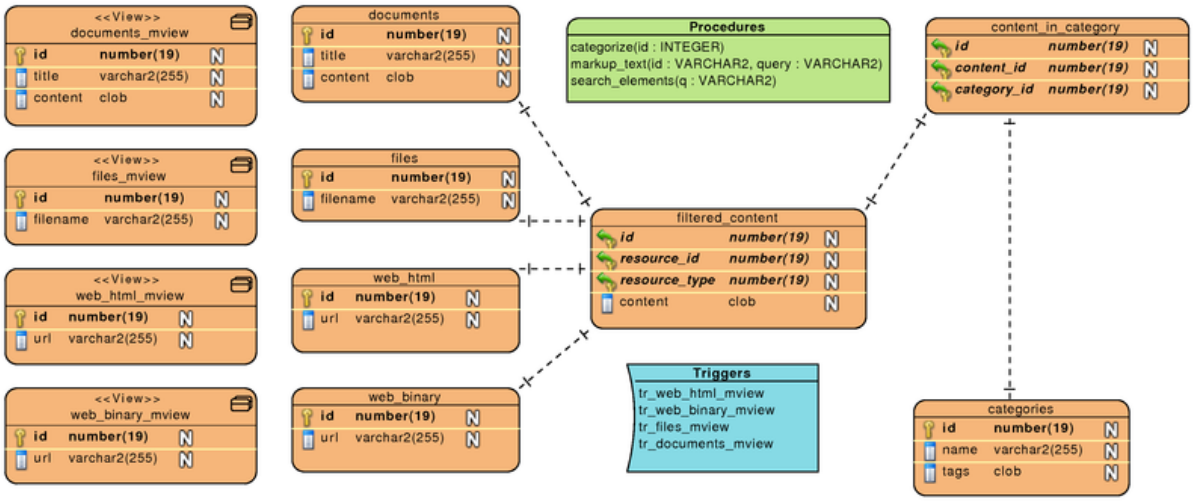
- dokumenty tekstowe wprowadzone bezpośrednio do systemu baz danych,

- dokumenty typu tekstowego: HTML, XML, znajdujące się na serwerach WWW,
- dokumenty binarne: DOC, PDF, znajdujące się w zasobach internetowych,
- dokumenty zmagazynowane w systemie plików.

Oprogramowanie wspomaga kategoryzację dokumentów, podpowiadając użytkownikowi najbardziej trafne kategorie dla danego dokumentu, na podstawie których użytkownik podejmuje decyzję odnośnie przyporządkowania. Działanie aplikacji opiera się na tabelach przechowujących zasoby lub odwołania do nich:

- *DOCUMENTS* – dokumenty tekstowe,
- *FILES* – ścieżki do żądanych plików,
- *WEB_BINARY* – przechowuje adresy URL zasobów binarnych,
- *WEB_HTML* – przechowuje adresy URL zasobów tekstowych.

W celu kategoryzacji dokumentów utworzono dwie tabele: *CATEGORIES*, zawierającą kategorie, oraz *CONTENT_IN_CATEGORIES*, przechowującą relacje pomiędzy kategoriami a dokumentami. Tabela *FILTERED_CONTENT* agreguje wszystkie dokumenty w jednym miejscu, w formie czystego tekstu. Na rysunku 1 przedstawiono schemat bazy danych.



Rys. 2. Schemat bazy danych
Fig. 2. Database schema

Dla każdej z czterech głównych tabel utworzono indeksy *CONTEXT*, odpowiadające typowi danych przechowywanemu w bazie danych. Tabela *DOCUMENTS* przechowuje tytuł oraz treść. Odpowiada jej indeks z wykorzystaniem magazynu danych, zdefiniowanego przez administratora – *USER_DATASTORE*. Tworzenie magazynu danych *USER_DATASTORE* polega na utworzeniu preferencji i zdefiniowaniu funkcji filtrującej. W tym celu utworzono preferencje document_pref oraz procedurę search_filter w pakiecie search_package. Procedura jako argument przyjmuje *ROWID*, pobiera zawartość danego wpisu, łączy kolumny wiersza: tytuł (ang. *title*) oraz treść (ang. *content*), i zwraca je w formacie *CLOB*, który jest indeksowany. Procedura jest skojarzona z magazynem danych *USER_DATASTORE* przy wykorzy-

staniu preferencji document_pref, zawierającej nazwę procedury (parametr *PROCEDURE*) oraz zwracany typ (*OUTPUT_TYPE*):

```
BEGIN
   ctx_ddl.create_preference('document_pref', 'USER_DATASTORE');
   ctx_ddl.set_attribute('document_pref', 'PROCEDURE',
                         search_package.search_filter');
   ctx_ddl.set_attribute('document_pref', 'OUTPUT_TYPE', 'CLOB');
END;
```

Indeks jest tworzony na tabeli *DOCUMENTS*, na kolumnie *title*. Określenie kolumny jest wykorzystywane jedynie przez funkcję przeszukującą, która odwołuje się do kolumny, na której został założony indeks. Procedura filtrująca w tym przypadku wykorzystuje kolumny *title* oraz *content*. Jako parametry indeks przyjmuje magazyn danych. Standardowo indeks jest przebudowywany na żądanie przez administratora, w tym przypadku zostało określone, że ponowna indeksacja ma się odbywać po każdym zatwierdzeniu danych – *SYNC* (*ON COMMIT*). Natychmiastowa synchronizacja indeksu jest celowa, ze względu na dostęp do jak najbardziej aktualnych danych oraz pobrania przefiltrowanych danych z dostępnego dokumentu:

```
CREATE INDEX documents_idx ON documents(title)
   INDEXTYPE IS ctxsys.context
   PARAMETERS ('DATASTORE document_pref SYNC (ON COMMIT)');
```

Tabele *WEB_BINARY* oraz *WEB_HTML* wykorzystują indeks *CONTEXT* z magazynem danych *URL_DATASTORE*. Przechowują w kolumnie *url* adres dokumentu binarnego lub tekstowego, znajdującego się w zasobach internetowych. Analogicznie do przypadku magazynu *USER_DATASTORE* została utworzona preferencja modyfikująca zachowanie magazynu. W tym przypadku został zmieniony czas maksymalnego oczekiwania na odpowiedź (*TIMEOUT*) na maksymalnie 300 sekund:

```
BEGIN
   ctx_ddl.create_preference('pref_url_datastore','URL_DATASTORE');
   ctx_ddl.set_attribute('pref_url_datastore','TIMEOUT','300');
END;
```

Plikom binarnym odpowiada filtr *AUTO_FILTER*, który filtruje dokument na formę tekstową możliwą do indeksacji. W tym przypadku indeks jest tworzony na kolumnie, która zawiera odnośniki do zasobów internetowych:

```
CREATE INDEX web_binary_idx ON web_binary(url)
   INDEXTYPE IS ctxsys.context
   PARAMETERS (
   'DATASTORE pref_url_datastore FILTER ctxsys.AUTO_FILTER SYNC (ON COMMIT)');
```

W przypadku tabeli *FILES*, przechowującej w kolumnie *filename* adres pliku, utworzono indeks *CONTEXT* z magazynem *danych FILE_DATASTORE*. Wykorzystano preferencję *ROOT_DIR* z parametrem *PATH*, określającą ścieżkę katalogu, z którego mają zostać pobrane pliki w przypadku podania ścieżki względnej:

```
BEGIN
   ctx_ddl.create_preference('ROOT_DIR','FILE_DATASTORE');
   ctx_ddl.set_attribute('ROOT_DIR','PATH','&path');
END;
```

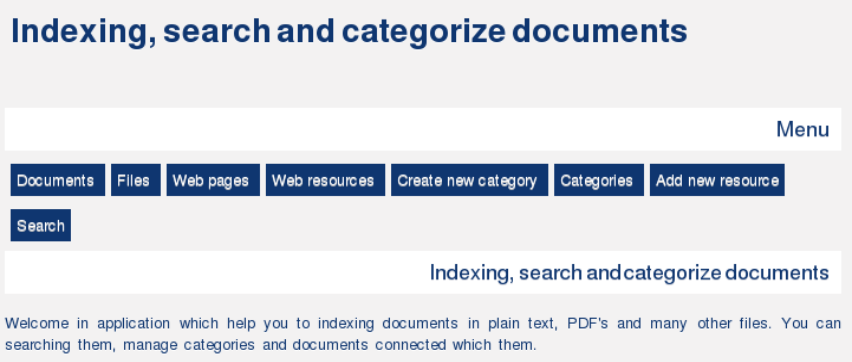
W przypadku oddzielnego przeszukiwania dla każdej tabeli z osobna i złączeniu wyników, wynik funkcji *SCORE* nie odpowiadałby właściwej wartości współczynnika trafności, ze względu na cztery różne indeksy. W tym celu utworzono tabelę *FILTERED_CONTENT*, zawierającą identyfikator zasobu (*resource_id*), typ zasobu *resource_type*) oraz treść (*content*), odpowiadającą przefiltrowanym danym. Indeks *CONTEXT* został utworzony na kolumnie *content*. Agregacja danych odbywa się przez wykorzystanie widoków zmaterializowanych. *OT* pobiera dane do utworzenia indeksów podczas zatwierdzania transakcji. W celu wykorzystania wcześniej utworzonych indeksów, a nie pobierania osobno danych dla każdego zasobu, utworzono widoki zmaterializowane na każdą z tabel *DOCUMENTS*, *FILES, WEB_BINARY* oraz *WEB_HTML*. Widok jest synchronizowany podczas synchronizacji danych (*REFRESH ON COMMIT*). Na tak utworzony widok założono wyzwalacz synchronizujący dane pomiędzy tabelami zawierającymi zasoby a tabelą *FILTERED_CONTENT*, dodatkowo umieszczając przefiltrowane dane w kolumnie *content*.

Przeszukiwanie odbywa się z wykorzystaniem operatora *MATCHES* indeksu CONTEXT. Została utworzona funkcja *SEARCH_ELEMENTS*, która jako argument przyjmuje frazę do przeszukania i zwraca wcześniej utworzony typ *SEARCH_TABLE*, agregujący kolumny z *FILTERED_CONTENT* oraz z tabel zawierających dokumenty. W wyniku wykonania tej funkcji następuje złączenie tabeli *FILTERED_CONTENT* z tabelami dokumentów i zwrócenie wyników przeszukiwania w kolejności od najbardziej trafnego.

W celu oznaczenia przeszukiwanych słów kluczowych wykorzystano procedurę *ctx_doc.markup*. Procedura, jako argumenty, przyjmuje nazwę indeksu (*index_name*), klucz tekstowy elementu (*textkey*), zapytanie tekstowe (*text_query*), zwracaną zmienną (*restab*), określenie, czy zwracana treść ma być w formie bez znaczników (*plaintext*), systemowo zdefiniowaną możliwość oznaczenia słowa kluczowego (*tagset*), początkowy znacznik (*starttag*), końcowy znacznik (*endtag*), znaczniki określające, służące do nawigacji (*prevtag* i *nexttag*).

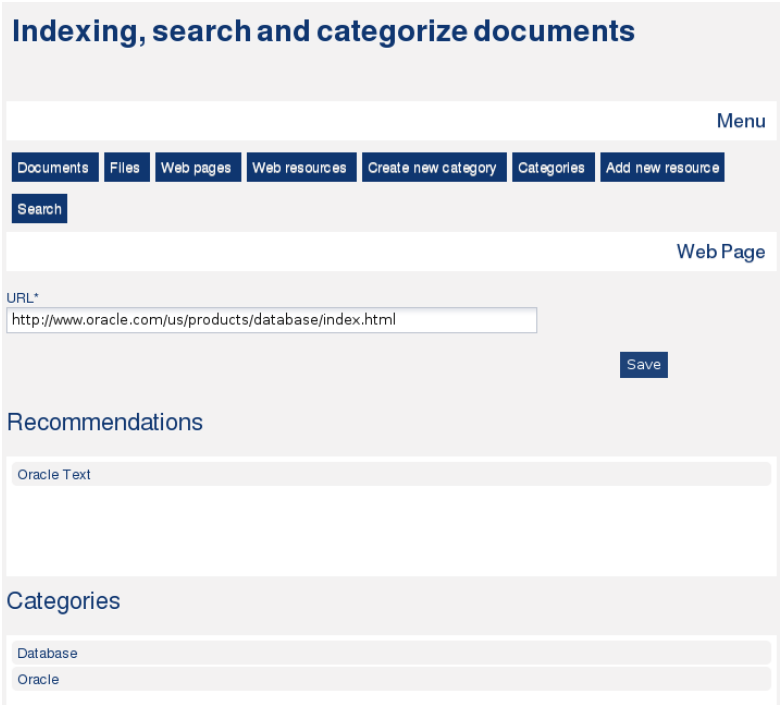
Aplikacja ma charakter internetowy, jest dostępna z poziomu przeglądarki i zawiera odnośniki do (rysunek 3):

- *Documents* – listy z możliwością edycji dokumentów tekstowych,
- *Files* – listy z możliwością edycji plików dodanych do systemu,
- *Web pages* – listy z możliwością edycji dodanych stron internetowych,
- *Web resources* – listy z możliwością edycji zasobów internetowych,
- *Categories* – listy z możliwością edycji kategorii,
- *Add new resource* – dodania nowego zasobu,
- *Search* – przeszukiwania zbiorów dokumentów.



Rys. 3. Główny ekran aplikacji
Fig. 3. The main screen of application

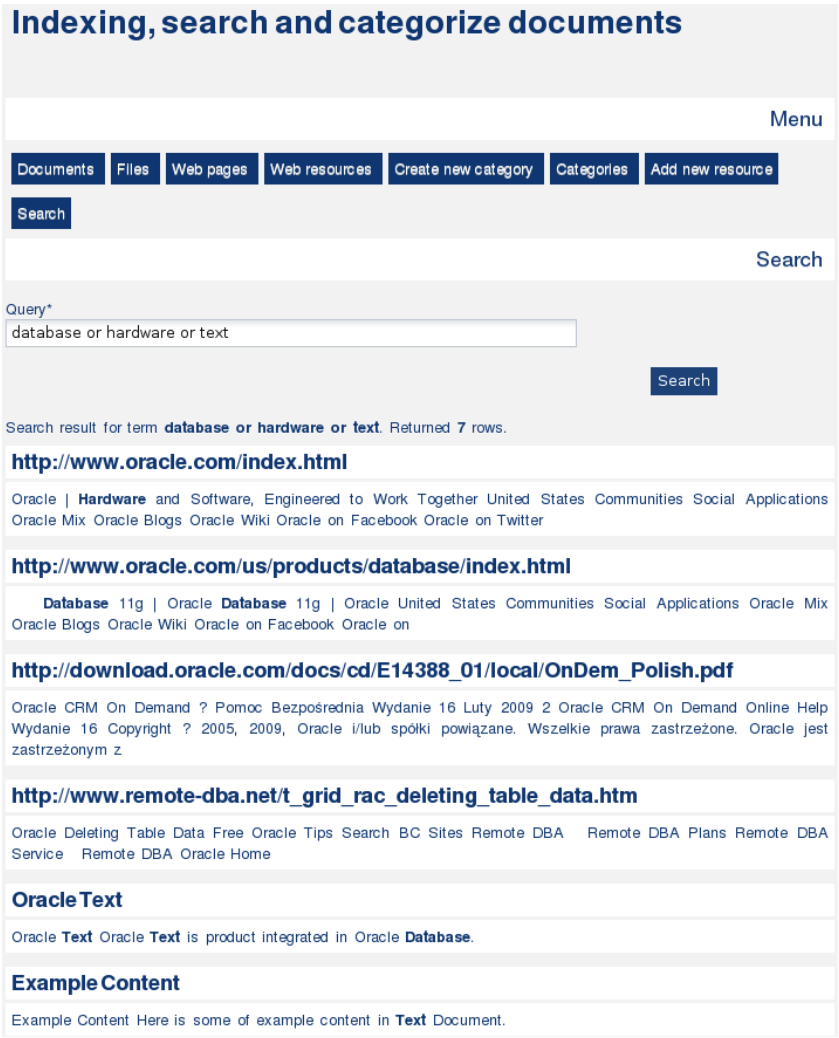
Podczas kategoryzacji użytkownikowi wyświetlane są elementy *Recommendations* oraz *Categories*. *Recommendations* to lista maksymalnie pięciu kategorii, które są najbardziej trafne wobec kategoryzowanego dokumentu. *Categories* to lista kategorii przypisanych dokumentowi. Lista ta nie ma ograniczenia.



Rys. 4. Edycja strony internetowej
Fig. 4. Web page edition

Przypisywanie odbywa się przez przeciągnięcie rekomendowanej kategorii na listę kategorii (mechanizm drag and drop). Usuwanie polega na przesunięciu kategorii w odwrotnym kierunku. Dane są zapisywane w trybie rzeczywistym. Przykład kategoryzacji zaznaczono na rysunku 4.

Użytkownik dokonuje przeszukiwania, podając poszukiwaną frazę i otrzymując listę dokumentów wraz z krótkim opisem, z zaznaczonymi słowami kluczowymi. Na rysunku 5 przedstawiono przykładowy wynik wyszukiwania.



Rys. 5. Edycja strony Internetowej
Fig. 5. Web page edition

6. Podsumowanie

W artykule zaprezentowano zagadnienia dotyczące zaawansowanego indeksowania oraz wyszukiwania informacji. Problematyka została omówiona na przykładzie rozwiązań firmy Oracle. W artykule przedstawiono aplikacje wykorzystujące w praktyczny sposób dostępne rozwiązania, rozszerzając funkcjonalność konkretnego systemu o zaawansowane opcje indeksowania i przeszukiwania zbiorów danych, cechujące się wysoką wydajnością oraz dużymi elastycznością i konfigurowalnością.

Prace nad różnorodnymi technikami wspierającymi efektywne indeksowanie, a co za tym idzie wyszukiwanie, są nieustannie prowadzone przez wszystkich wiodących producentów oprogramowania baz danych. Efekty tych prac pozwalają na tworzenie rozwiązań coraz lepiej spełniających oczekiwania użytkowników, przy uwzględnieniu stale rosnących ilości prze-

twarzanych informacji oraz dynamicznego rozwoju formatów danych. Z uwagi na tempo postępu informatyzacji wszelkich aspektów życia człowieka oraz rozwoju technologii składowania danych, dalsze prace w omawianym zakresie są nieuniknione. Istotną kwestią jest umiejętność prawidłowego i efektywnego wykorzystania dostępnych technologii i narzędzi, co, biorąc pod uwagę ich różnorodność, stanowi coraz istotniejszy problem.

Celem niniejszego artykułu było przybliżenie aktualnego stanu prac w tym zakresie oraz zaprezentowanie przykładowych aplikacji wykorzystujących nowoczesne rozwiązania w praktyczny sposób.

BIBLIOGRAFIA

1. Lane P., Shea C.: Oracle Text. Application Developer's Guide. Oracle 11g, 2010.
2. Apache Fundation. Apache lucene – overview, http://lucene.apache.org/java/docs/index.html.
3. PostgreSQL Global Development Group. Postgresql 8.4: Full text search, http://www.postgresql.org/docs/8.4/static/textsearch.html.
4. Oracle Secure Enterprise Search Group. Oracle Ultra Search vs Oracle Secure Enterprise Search Frequently Asked Questions. OSESG, 2010.
5. Mysql 4.1 reference manual, http://dev.mysql.com/doc/refman/4.1/en/fulltext-search.html.
6. Oracle Secure Enterprise Search 11g – Data Sheet. Oracle, 2010.
7. Oracle Secure Enterprise Search 11g – Version 11.1.2 – White Paper. Oracle, 2010.
8. Secure Searching with Oracle Secure Enterprise Search – White Paper. Oracle, 2010.
9. idevelopment – how oracle text (oracle 9i) relates to intermedia text (oracle 8i), http://www.idevelopment.info/data/Oracle/DBA_tips/Oracle_Text/TEXT_1.shtml.
10. Tf-idf weighting. 2009, http://nlp.stanford.edu/IR-book/html/htmledition/tf-idf-weighting-1.html.
11. Zend developer zone – using mysql full-text searching, http://devzone.zend.com/article/1304.

Recenzenci: Dr inż. Bożena Małysiak-Mrozek
Dr inż. Dariusz Mrozek

Wpłynęło do Redakcji 31 stycznia 2011 r.

Abstract

Indexing and searching of data are inherently complex issues. This subject is an area of intensive scientific research and the realization of successive practical implementation. Existing solutions are not still satisfactory, and the continuous progress of computerization makes the amount of information is constantly increasing. This causes that problems with accessing the desired information are constantly complicated. Number of collected information, the complexity and the need to obtain adequate data as soon as possible, all these factors make that the significance of the problem increases.

The article presents the issues of advanced data indexing and searching. The issue was discussed on the example of Oracle, one of the leading companies dealing with issues of data collection and processing. Among the various solutions provided by the company, as for indexing and searching the data attention should be focused on Oracle Text, Oracle Ultra Search and Oracle Secure Enterprise Search.

The article presents applications using the available solutions in a practical way, extending the functionality of a particular system of advanced indexing and searching data sets, characterized by high efficiency and high flexibility and configurability. Research on various techniques that support efficient indexing, and hence the search, are constantly carried out by all the leading manufacturers of database software. The effects of these works allow you to create solutions that better meet the expectations of users, taking into account the ever-increasing volumes of information and the rapid development of data formats. Given the pace of computerization of all aspects of human life and storage technology development, further works in this area are inevitable. An important issue is the ability to correct and effective use of available technologies and tools and taking into account their diversity, it is an increasingly important issue. The purpose of this article was to present the current state of work in this area and show examples of applications using modern solutions in a practical manner.

Adresy

Krzysztof CZAJKOWSKI: Politechnika Krakowska, Wydział Fizyki, Matematyki i Informatyki, Instytut Teleinformatyki, ul. Warszawska 24, 31-155 Kraków, Polska, kc@pk.edu.pl.
Dominik ZIAJKA: Politechnika Krakowska, Wydział Fizyki, Matematyki i Informatyki, Instytut Teleinformatyki, ul. Warszawska 24, 31-155 Kraków, Polska, dominik.ziajka@gmail.com.