Как классифицировать текстовые документы в Oracle

Владимир Пржиялковский

Преподаватель технологий Oracle
prz@yandex.ru
http://www.ccas.ru/prz

Писание же твое приято бысть и разумлено внятельно.

Первое послание Ивана Грозного Курбскому

Введение

В настоящее время встроенная в СУБД Oracle поисковая текстовая машина Oracle Text поддерживает работу с тремя разновидностями предметного (DOMAIN), текстового индекса: типов CTXSYS.CONTEXT, CTXSYS.CTXCAT и CTXSYS.CTXRULE. Первые два обеспечивают поиск, соответственно, полнотекстовый – в полноценных документах, и в «картотеке» с краткими описаниями (так сказать, в «каталоге») – по предъявленному к тексту запросу. Тип же индекса CTXSYS.CTXRULE по отношению к ним не совсем обычен и может рассматриваться как «оборотный» к типу CTXSYS.CONTEXT. Он строится по набору запросов, а не по документам, и его назначение – определить результативность каждого запроса из этого применительно к предъявляемому документу. Запросы в наборе, по которому строится индекс, вольно иначе назвать «правилами» (отсюда слово *rule* в названии индекса[1]), и проверку соответствия определенного документа тому или иному правилу вольно, опять-таки, рассматривать как классифицирование документа.

Некоторые основные возможности использования индекса типа CTXSYS.CTXRULE будут рассмотрены ниже на двух примерах: простом и более реалистичном.

Простой пример

Существенным технологическим отличием «оборотного» индекса CTXSYS.CTXRULE от CTXSYS.CONTEXT является то, что последний можно строить для документов как внутри БД, так и вне ее (файловая система, интернет), а первый – только для документов, «внутри», то есть хранящихся либо в переменной программы, либо в столбце типа VARCHAR2 или же CLOB таблицы БД (только эти два типа и допускает оператор MATCHES). Причина такого ограничения разработчиками не раскрывается. В этом простом примере будем считать, что документы хранятся в программе, в строке типа VARCHAR2.

Подготовка и проведение опыта с запросами

Построим таблицу с набором запросов, которую потом будем использовать в качестве таблицы «правил», или же «таблицы классификации» документов. Выдадим в SQL*Plus:

```
CREATE TABLE rules ( id NUMBER PRIMARY KEY, query VARCHAR2 ( 1000 ) );

INSERT INTO rules VALUES ( 1, 'lamb' );
INSERT INTO rules VALUES ( 2, 'little OR lamb' );
INSERT INTO rules VALUES ( 3, 'lamb NOT mary' );
INSERT INTO rules VALUES ( 4, 'little lamb' );
INSERT INTO rules VALUES ( 5, 'lamb little' );
```

1 Возможно более уместно здесь было бы привести для слова *rule* другой, более специфичный перевод «направляющая».

```
INSERT INTO rules VALUES ( 6, 'lamb NEAR mary' );
```

Строим индекс и готовим сценарный файл для опытов:

```
CREATE INDEX rules_idx ON rules ( query ) INDEXTYPE IS CTXSYS.CTXRULE;

COLUMN category FORMAT 99999
COLUMN query    FORMAT A50
SELECT id category, query FROM rules WHERE MATCHES ( query, '&1' ) > 0
.
SAVE matches REPLACE
SET VERIFY OFF
```

Проверяем соответствие трех «документов» шести заведенным «классификационным признакам»:

```
CTX> @matches 'Mary had a little lamb'

CATEGORY QUERY
-------- --------------------------------------------------
       1 lamb
       6 lamb NEAR mary
       2 little OR lamb
       4 little lamb

CTX> @matches 'Twinkle, twinkle little star'

CATEGORY QUERY
-------- --------------------------------------------------
       2 little OR lamb

CTX> @matches 'This Lamb is my lamb'

CATEGORY QUERY
-------- --------------------------------------------------
       1 lamb
       2 little OR lamb
       3 lamb NOT mary
```

Дальнейшие опыты рекомендуется провести самостоятельно. При необходимости следует воспользоваться известной из предыдущего материала процедурой CTX_DDL.SYNC_INDEX перестройки индекса. Также рекомендуется удостовериться, что в плане обработки наших запросов стоит обращение к индексу RULES_IDX (имеющийся там шаг обращения к таблице RULES вызван нашим желанием выдать значение поля ID строки из этой таблицы; если этого не сделать, обращение к таблице RULES пропадет).

Техническая организация индекса

Обращение к таблицам USER_OBJECTS и USER_SEGMENTS позволяет уточнить технику реализации индекса типа CTXSYS.CTXRULE. Вот примерно каким в нашем случае будет список логических объектов, возникших в результате выдачи команды CREATE INDEX rules_idx ...:

```
OBJECT_NAME                    OBJECT_TYPE
------------------------------ -----------------------
DR$RULES_IDX$I                 TABLE
DR$RULES_IDX$K                 TABLE
DR$RULES_IDX$N                 TABLE
DR$RULES_IDX$R                 TABLE
RULES                          TABLE
DR$RULES_IDX$X                 INDEX
RULES_IDX                      INDEX
SYS_IOT_TOP_53386              INDEX
SYS_IOT_TOP_53391              INDEX
SYS_LOB0000053383C00006$$      LOB
```

```
SYS_LOB0000053388C00002$$       LOB
```

А вот примерно какие появятся структуры хранения:

```
SEGMENT_NAME                    SEGMENT_TYPE
------------------------------- ------------------------------
DR$RULES_IDX$X                  INDEX
SYS_C005906                     INDEX
SYS_IOT_TOP_53386               INDEX
SYS_IOT_TOP_53391               INDEX
SYS_IL0000053383C00006$$        LOBINDEX
SYS_IL0000053388C00002$$        LOBINDEX
SYS_LOB0000053383C00006$$       LOBSEGMENT
SYS_LOB0000053388C00002$$       LOBSEGMENT
DR$RULES_IDX$I                  TABLE
DR$RULES_IDX$R                  TABLE
RULES                           TABLE
```

Очевидно, техническая организация индекса типа CTXSYS.CTXRULE почти та же, что и для типа CTXSYS.CONTEXT, то есть это четыре таблицы и необходимые для них служебные структуры. (Почти – потому что в таблице DR$RULES_IDX$I в нашем случае появилось дополнительное поле TOKEN_EXTRA. Дальнейшее изучение предлагается предпринять самостоятельно).

В наборах из нескольких запросов-«правил», подобно нашему случаю, это приводит к чудовищному перерасходу дисковой памяти; очевидно, сама возможность рассчитана на большие наборы.

Пример с реальными документами

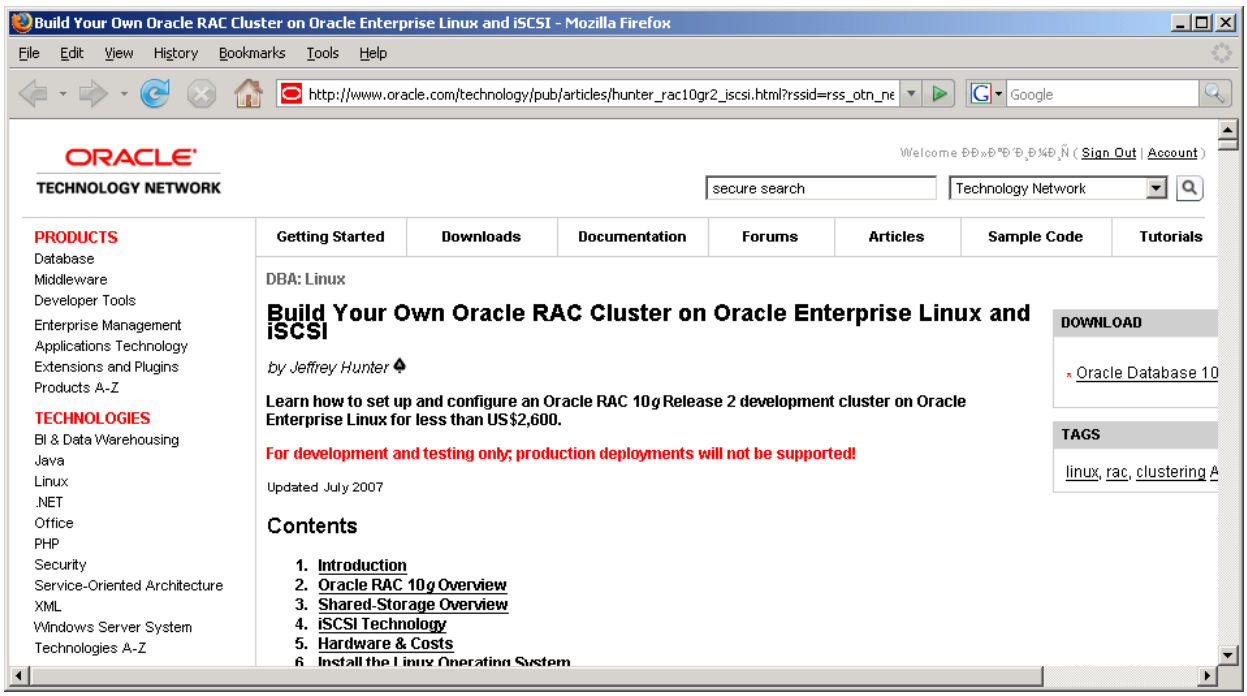
Простой пример выше позволяет понять логику классификации и технические моменты. Теперь попытаемся рассмотреть более жизненный пример. Жизненной будет ситуация, в то время как технологически для лучшего понимания существенного далее все же будет сделан ряд допущений и технических упрощений.

В статье «*Как работать с картотекой (набором данных с краткими описаниями) ?*» рассматривалось заведение в БД и индексирование «картотеки» с краткими описаниями новостей, полученных из канала RSS в интернете для Oracle Technology Network, и со ссылками на источник. Возьмем одну такую ссылку:

```
CTX> COLUMN link FORMAT A75
CTX> SELECT link FROM otnnews WHERE ROWNUM = 1;

LINK
---------------------------------------------------------------------------
http://www.oracle.com/technology/pub/articles/hunter_rac10gr2_iscsi.html?rs
sid=rss_otn_news
```

Вот начало документа HTML, расположенного по этой ссылке:



Проделаем следующее:

- Извлечем по этой ссылки документ в БД.
- Построим более реальный набор классификационных правил.
- Проверим документ на соответствие правилам.

Получаем документ из интернета

Если обращение в интернет будет осуществляться через приближенный (proxy) сервер, требуется предварительно выдать что-то вроде:

```
EXECUTE UTL_HTTP.SET_PROXY ('http://имя:пароль@адрес:порт')
```

Для извлечения документа из интернета мне очень хотелось бы воспользоваться типом HTTPURITYPE, но проверка показывает, что этот тип не способен посылать запросы HTTP методом POST, а для нашей страницы, опять-таки показывает проверка, требуется именно это. Притом, страница динамическая: обратите внимание на завершение адреса текстом `?rssid=rss_otn_news`. (Несложная проверка параметров ответа при обращении по этому адресу показывает, что получателем запроса является сервлет на Oracle Application Server). Поэтому придется прибегнуть к пакету UTL_HTTP и программированию.

Для удобства занесем основную часть адреса и параметры в переменные SQL*Plus:

```
VARIABLE url         VARCHAR2 ( 1000 )
VARIABLE parameters  VARCHAR2 ( 1000 )
VARIABLE length      VARCHAR2 ( 1000 )

EXECUTE :url := -
'http://www.oracle.com/technology/pub/articles/hunter_rac10gr2_iscsi.html'
EXECUTE :parameters := 'rssid=rss_otn_news'
EXECUTE :length := '18'
```

Для простоты количество символов в подстроке параметров (18) я посчитал вручную.

Заведем переменную SQL*Plus для хранения документа:

```
VARIABLE htmlclob CLOB
EXECUTE :htmlclob := EMPTY_CLOB ( )
```

При работе учтите, что значение переменной SQL*Plus типа CLOB потеряется, как только вы завершите сеанс связи с СУБД, например в результате выдачи CONNECT. (В действительности, переменная HTMLCLOB есть переменная-«локатор», указывающий на временный LOB-объект со временем жизни сеанса; этот-то LOB-объект и пропадает без нашей воли по завершению сеанса, после чего локатор начнет указывать «в никуда»).

Следующий блок на PL/SQL прочтет по нужному адресу в интернете документ и разместит его в переменной SQL*Plus:

```
DECLARE
req      UTL_HTTP.REQ;
resp     UTL_HTTP.RESP;
name     VARCHAR2 ( 256 );
value    VARCHAR2 ( 4000 );

BEGIN
req := UTL_HTTP.BEGIN_REQUEST ( :url, method => 'POST' );
UTL_HTTP.SET_HEADER ( r => req, name => 'Content-Type', value =>
'text/html' );
UTL_HTTP.SET_HEADER ( r => req, name => 'Content-Length', value => :length );
UTL_HTTP.WRITE_LINE ( r => req, data => :parameters );
resp := UTL_HTTP.GET_RESPONSE ( req );

DBMS_OUTPUT.PUT_LINE ( 'HTTP response status code: ' || resp.status_code );
DBMS_OUTPUT.PUT_LINE ( 'HTTP response reason: ' || resp.reason_phrase );

LOOP
  UTL_HTTP.READ_LINE ( resp, value, TRUE );
  :htmlclob := :htmlclob || value;
END LOOP;

EXCEPTION
  WHEN UTL_HTTP.END_OF_BODY
  THEN UTL_HTTP.END_RESPONSE ( resp );
END;
/
```

Выдача на экран добавлена для контроля. Проверить, что документ действительно прочитался, можно, например, так:

```
CTX> EXECUTE DBMS_OUTPUT.PUT_LINE ( DBMS_LOB.GETLENGTH ( :htmlclob ) )
166426

PL/SQL procedure successfully completed.
```

При желании можно было пометить документ в таблицу, но здесь довольно и оставить его в переменной SQL*Plus.

Проводим классификацию

Создадим таблицу классификации, заполним ее правилами и построим индекс:

```
CREATE TABLE category ( id NUMBER, query VARCHAR2 ( 2000 ) );

INSERT INTO category VALUES ( 1, 'rac | "real application clusters"' );
INSERT INTO category VALUES ( 2, 'linux | unix' );
INSERT INTO category VALUES ( 3, 'installation | configuration' );
INSERT INTO category VALUES ( 4, 'ms windows OR microsoft NEAR windows' );
INSERT INTO category VALUES ( 5, 'standby AND switchover' );

CREATE INDEX category_idx ON category ( query ) INDEXTYPE IS CTXSYS.CTXRULE;
```

Проверка:

```
CTX> COLUMN query FORMAT A60
CTX> SELECT id, query FROM category WHERE MATCHES ( query, :htmlclob ) > 0;

        ID QUERY
---------- ------------------------------------
         1 rac | "real application clusters"
         2 linux | unix
         3 installation | configuration
         4 ms windows OR microsoft NEAR windows
```

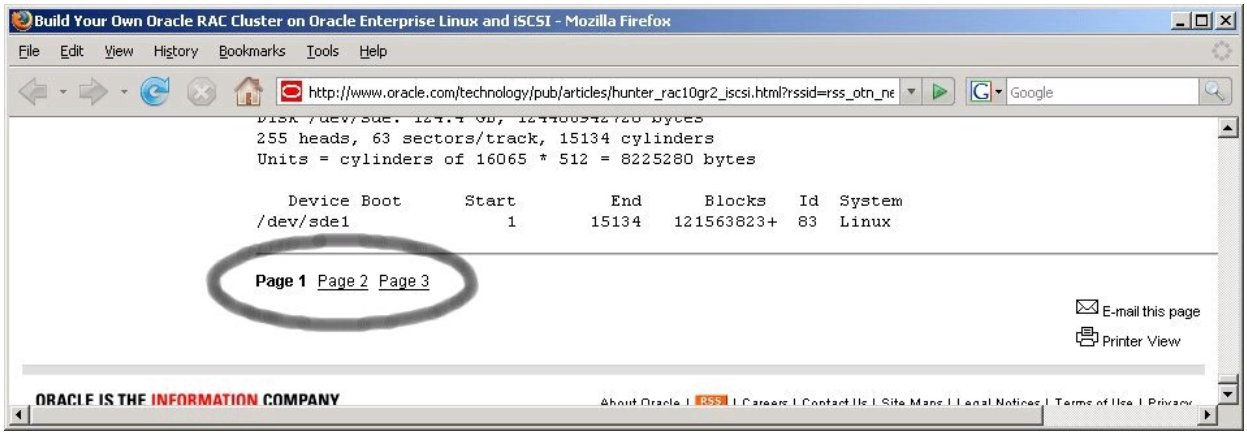
В данном случае документ удовлетворяет четырем категориям имеющейся классификации под номерами 1 – 4 и не удовлетворяет категории под номером 5.

В жизни может быть сложнее

Реальность нередко оказывается сложнее, чем хотелось бы. Вот примеры.

Составной документ

Вернемся воочию к документу, который только что проанализировали. Обратите внимание на его завершение:



Интерес привлекает фрагмент в конце страницы, который я обвел эллипсом. Судя по всему, наш документ не исчерпывается одной страницей HTML и состоит на деле из трех страниц ! Фактический переход по ссылкам Page 2 и Page 3 подтверждает эту догадку. Это – одна из сложностей, которая может нас подстерегать при организации автоматического классифицирования.

Технически проблему документа, логически единого, но разнесенного по нескольким страницам, можно учесть дополнительным программированием, включающим выявление подобного рода продолжений, но при этом мы вынуждены будем учитывать особенности вполне конкретного канала новостей RSS, так как в другом канале ссылки на продолжение документа могут оказаться организованы совсем иначе. Это лишит создаваемую программу автоматического классифицирования общности.

«Словесный шум»

Если документ, как в нашем случае, представлен страницей HTML в интернете, он как правило содержит не относящуюся к делу информацию (то есть, для нас – шум), в том числе в виде текста. Ненужные нам слова могут появиться вследствие желания разработчиков места в интернете показать на странице направления дальнейшей навигации, или же попросту могут относиться к рекламе. Рассмотренный способ фактически анализирует текст страницы HTML, а не документа, и как отфильтровать не относящиеся к делу слова, мне неизвестно. Остается только надеяться, что подобное словесное сопровождение документа, как это нередко

бывает, будет порождаться средствами JavaScript и программой чтения «документа» (страницы) останется незамеченым.

Если документ представлен файлом формата PDF, RTF, простого текста или иным, проблема попадания в поле зрения не относящихся к документу слов не возникает.

Иные форматы

Использованный выше способ классификации можно применять только к документам, представленным простым текстом (plain text), либо в формате HTML. К другим форматам представления документа оператор MATCHES напрямую не применим. Если выясняется, что новостная ссылка указывает на документ в ином формате (например, PDF, – а выяснение этого уже требует дополнительного программирования), остается только привести документ перед анализом в текстовый вид с помощью процедуры CTX_DOC.IFILTER. (Неявно как раз эта процедура выполняет преобразование документа при полнотекстовом индексировании и при использовании в индексе типа CTXSYS.CONTEXT фильтра CTXSYS.AUTO_FILTER). Это технически возможно, но требует дальнейшего дополнительного программирования и увеличит расходование ресурсов СУБД на выполнение анализа.

Заключение

Материала настоящей статьи, а также трех предшествующих, посвященных возможностям Oracle Text, достаточно для понимания того, как запрограммировать автоматический сбор новостей из интернета и выполнять автоматическую классификацию.

Выполнять полнотекстовый поиск, а также классификацию, можно к тому же применительно к документам, находящимся не только в интернете, но и в БД, и в файловой системе.

В то же время затронутые возможности являются базовыми, начальными, и не исчерпывают содержание Oracle Text. Например, за рамками рассмотрения оказались средства «интеллектуального анализа» документов, которые по аналогии с Data Mining получили название Text Mining. Описывать эти средства мимоходом нереально.