Как работать с картотекой (набором данных с краткими описаниями) ?

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

... в отделе «Что случилось за день» нонпарелью было напечатано:
Попал под лошадь. Вчера на площади Свердлова попал под лошадь извозчика №8974 гражданин О. Бендер. Пострадавший отделался легким испугом.

И. Ильф, Е. Петров. Двенадцать стульев

Введение

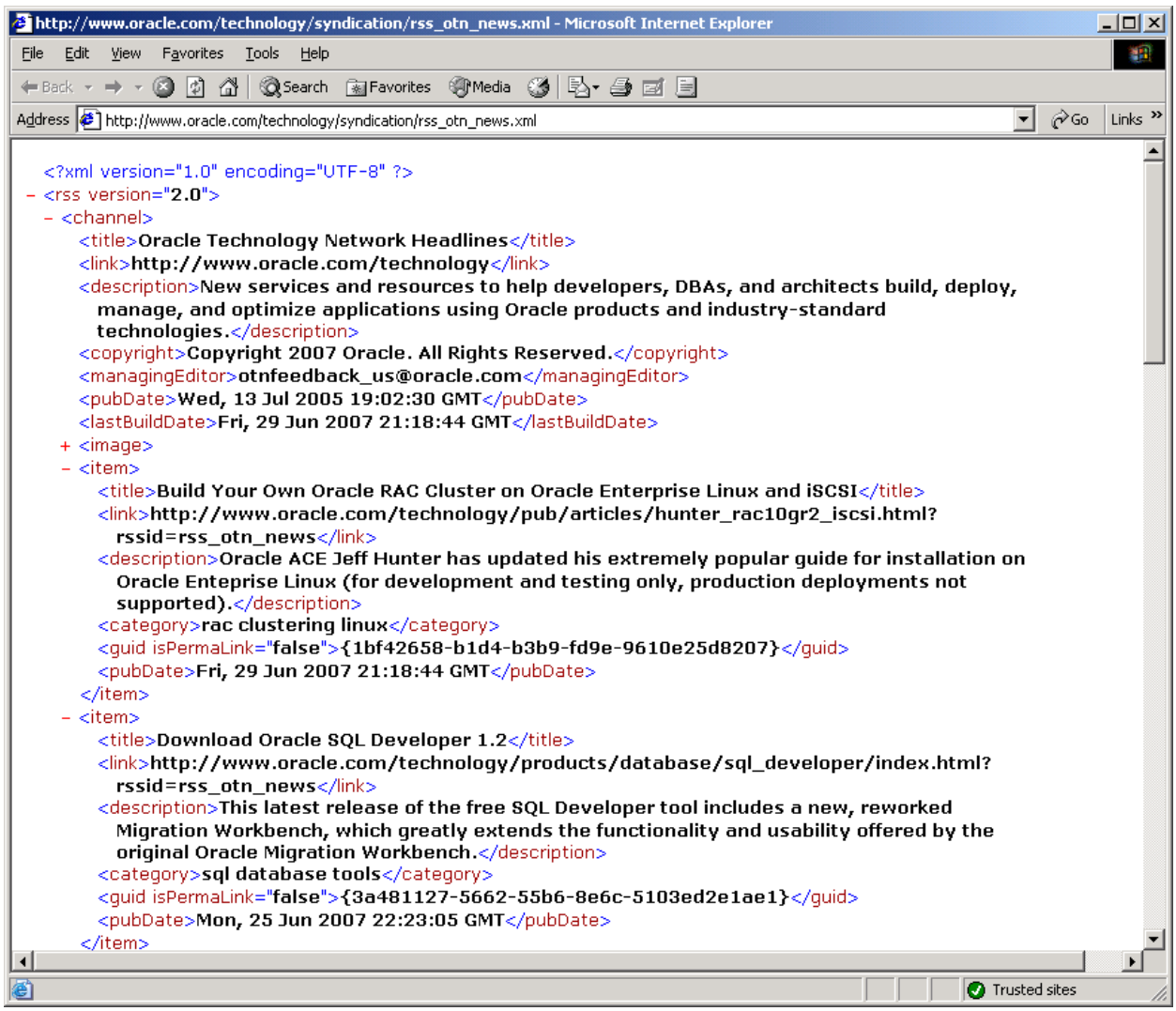
Помимо полнотекстового поиска в документах, объем которых может быть очень велик, встроенная в СУБД Oracle текстовая машина («Oracle Text») отдельно обеспечивает возможность поиска информации для одного специального случая. Речь идет о поиске в наборе записей, содержащих краткие текстовые описания. Подобное устройство данных хорошо известно по традиционным картотекам, постепенно уходящим в прошлое, но еще встречающимся в унаследованых институтах, таких как библиотеки. Там картотеки организованы в виде продолговатых ящичков, где хранятся карточки с кратким описанием единиц хранения и указанием месторасположения. Более современный пример того же устройства данных – каналы новостей RSS, придавшие своим появлением заметный импульс динамике информации в интернете. Сама же фирма Oracle использует для именования такого устройства данных метафору «каталога», образованного записями с описаниями, помимо прочих реквизитов, объектов каталогизации.

Если подобную картотеку/каталог промоделировать таблицею в БД Oracle, сама собой возникает мысль проиндексировать поля с такими описаниями предметным (DOMAIN) индексом типа CTXSYS.CONTEXT и воспользоваться для запросов к данным возможностями оператора CONTAINS. Однако если описания кратки (одна – две строки), это неэкономно: индекс получается чрезмерно большим, а возможности запросов избыточными. Для такого случая и предложен специальный тип предметного индекса: CTXSYS.CTXCAT. Он дает одновременно и большую по отношению к полнотекстовому индексу компактность, и более простой (в основном варианте) язык запросов, обеспеченный специальным оператором CATSEARCH.

Ниже рассматривается пример организации в БД «картотеки», построения предметного индекса типа CTXSYS.CTXCAT и составления запросов к данным.

Исходные данные и план действий

В качестве исходных данных возьмем реальную ленту новостей «Oracle Technology Network Headlines» по адресу *http://www.oracle.com/technology/syndication/rss_otn_news.xml* с содержанием на момент написания статьи:



Документ имеет общее описание канала новостей (разметка `title`, `link`, `description`, `copywrite` и т. д.) и перечисление сведений об отдельных новостях элементами `title`:

```
<title>
   <link>...</link>
   <description>...</description>
   <category>...</category>
  ...
</title>
```

План дальнейших действий таков:

- Перенесем эти данные в БД в таблицу с полями LINK, DESCRIPTION и PUBTIME, данные в которых описывают каждую конкретную новость.
- Построим «каталожный» индекс типа CTXSYS.CTXCAT по полю DESCRIPTION.
- Приведем примеры запросов.

Для лучшего понимания существа сам пример будет использовать ряд технических упрощений и исходных допущений. В частности, обратите внимание, что (а) исходные данные представлены в формате RSS 2.0, и в статье не учитывается возможность существования в прочих каналах новостей других форматов RSS с иной разметкой; (б) в конкретном новостном канале элемент `description` невелик по объему и лучше подходит под «краткое содержание», нежели чем элемент `title`.

Кроме того, некоторых целей, преследуемых далее, можно достичь иными путями. Так, первое, что приходит в голову – программирование, которого дальше, однако, я сознательно избегаю.

Для удобства занесем адрес канала в переменную SQL*Plus:

```
VARIABLE url VARCHAR2 ( 1000 )
EXECUTE :url :=
'http://www.oracle.com/technology/syndication/rss_otn_news.xml'
```

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')
```

Загрузка данных в БД

Загрузка данных о новостях будет осуществляться с помощью функции по форме, а по сути – процедуры, DBMS_XMLSTORE.INSERTXML. Она требует привести данные для загрузки к следующему виду:

```
<ROWSET>
<ROW>
  <link>...</link><description>...</description><pubDate>...</pubDate>
</ROW>
<ROW>
  <link>...</link><description>...</description><pubDate>...</pubDate>
</ROW>
...
</ROWSET>
```

Это будет сделано с помощью преобразования XQUERY, но для того, чтобы не усложнять его формулу, я использую следующий трюк. Создадим таблицу с полями LINK, DESCRIPTION и PUBDATE, а загрузку произведем в однотипную производную таблицу (view), имена столбцов которых сохраняют регистр букв соответствующих меток в исходном документе. Объекты для хранения и для загрузки:

```
CREATE TABLE otnnews (
  link        VARCHAR2 ( 4000 )
, description VARCHAR2 ( 4000 )
, pubdate     VARCHAR2 ( 30 )
);

CREATE VIEW otnnews_view ( "link", "description", "pubDate" )
AS SELECT * FROM otnnews
;
```

В следующем блоке на PL/SQL сначала оператор SELECT берет из интернета документ XML и преобразует его в требуемый вид, а последующая серия вызовов подпрограмм пакета DBMS_XMLSTORE организует занесение данных (элемент `title`) из полученного документа XML в таблицу OTNNEWS_VIEW; фактически – в OTNNEWS:

```
VARIABLE rowsnumber NUMBER

DECLARE
  xmltypedoc    XMLTYPE;
  updatecontext DBMS_XMLSTORE.CTXTYPE;
BEGIN
SELECT
  XMLQUERY (
  '<ROWSET> {
     for $a in /rss/channel/item
     return <ROW>{$a/link, $a/description, $a/pubDate}</ROW>
   }
   </ROWSET>'
  PASSING HTTPURITYPE ( :url ).GETXML ( )
  RETURNING CONTENT
  )
```

```
INTO xmltypedoc
FROM dual
;

updatecontext := DBMS_XMLSTORE.NEWCONTEXT ( 'CTX.OTNNEWS_VIEW' );

DBMS_XMLSTORE.CLEARUPDATECOLUMNLIST ( updatecontext );
DBMS_XMLSTORE.SETUPDATECOLUMN ( updatecontext, 'link' );
DBMS_XMLSTORE.SETUPDATECOLUMN ( updatecontext, 'description' );
DBMS_XMLSTORE.SETUPDATECOLUMN ( updatecontext, 'pubDate' );

:rowsnumber := DBMS_XMLSTORE.INSERTXML ( updatecontext, xmltypedoc );

DBMS_XMLSTORE.CLOSECONTEXT ( updatecontext );
END;
/
```

В таблице OTNNEWS появилось 11 записей. Вот, например, даты новостей:

```
CTX> SELECT pubdate FROM otnnews;

PUBDATE
------------------------------
Fri, 29 Jun 2007 21:18:44 GMT
Mon, 25 Jun 2007 22:23:05 GMT
Tue, 19 Jun 2007 16:18:01 GMT
Tue, 12 Jun 2007 15:26:52 GMT
Fri, 01 Jun 2007 16:24:42 GMT
Fri, 01 Jun 2007 16:23:00 GMT
Fri, 25 May 2007 15:37:53 GMT
Fri, 25 May 2007 15:34:52 GMT
Tue, 15 May 2007 23:11:50 GMT
Tue, 15 May 2007 23:09:37 GMT
Fri, 01 Jun 2007 16:16:47 GMT

11 rows selected.
```

Создание индекса

В простом случае индекс создается просто:

```
CREATE INDEX otnnews_idx
ON otnnews ( description )
INDEXTYPE IS CTXSYS.CTXCAT
;
```

Вот какие структуры БД и сегменты хранения появились в результате:

```
CTX> COLUMN object_name  FORMAT A30
CTX> COLUMN object_type  FORMAT A30
CTX> COLUMN segment_name FORMAT A30
CTX> COLUMN segment_type FORMAT A30
CTX> SELECT object_name, object_type FROM user_objects ORDER BY 2, 1;

OBJECT_NAME                    OBJECT_TYPE
------------------------------ ------------------------------
DR$OTNNEWS_IDX$R               INDEX
DR$OTNNEWS_IDX$X               INDEX
OTNNEWS_IDX                    INDEX
DR$OTNNEWS_IDX$I               TABLE
OTNNEWS                        TABLE
```

```
DR$OTNNEWS_IDXTC                TRIGGER
OTNNEWS_VIEW                    VIEW

CTX> SELECT segment_name, segment_type FROM user_segments ORDER BY 2, 1;

SEGMENT_NAME                    SEGMENT_TYPE
------------------------------ ------------------------------
DR$OTNNEWS_IDX$R                INDEX
DR$OTNNEWS_IDX$X                INDEX
DR$OTNNEWS_IDX$I                TABLE
OTNNEWS                         TABLE
```

Очевидно, что формально индекс типа CTXSYS.CTXCAT компактнее индекса типа CTXSYS.CONTEXT: он реализован не из черырех, а из трех служебных таблиц, вдобавок не имеющих полей LOB. Еще одно отличие: создание «каталожного» индекса сопряжено с автоматическим появлением триггерной процедуры (в нашем случае DR$OTNNEWS_IDXTC) вида AFTER EACH ROW INSERT OR UPDATE, связанной с базовой таблицей (все это легко установить в качестве самостоятельного *упражнения*). Наличие этой триггерной процедуры дает основание предположить, что еще одно отличие «каталожного» индекса от полнотекстового в том, что он автоматически корректируется при изменении данных в таблице. Это предположение легко проверяется.

Запросы

Оператором для предъявления запроса (формально к таблице, а фактически – к индексу), является CATSEARCH. В отличие от CONTAINS, он может употребляться в SQL только в составе условного выражения. В основном применении он допускает собственные обозначения операторов запроса (логические AND, OR и NOT, обозначаемые через «пробел», | и -; * для произвольной подстановки и заключения в кавычки). Тем не менее, синтаксис, применяемый в CONTAINS, тоже допускается, но путем определенных ухищрений, из-за которых эта возможность здесь не рассматривается.

Выдадим:

```
SELECT description FROM otnnews WHERE CATSEARCH ( description, &1, &2 ) > 0
.
SAVE catsearch
COLUMN description FORMAT A75 TRUNCATED
SET VERIFY OFF
```

Примеры запросов:

```
CTX> @catsearch "'java'" NULL

DESCRIPTION
---------------------------------------------------------------------------
Get an introduction to using Oracle Data Integrator, Java-based middleware

CTX> @catsearch "'java | web'" NULL

DESCRIPTION
---------------------------------------------------------------------------
Get an introduction to using Oracle Data Integrator, Java-based middleware
Learn how to secure PHP-based Web applications via database-based authentic

CTX> @catsearch "'(java | web) - php'" null

DESCRIPTION
---------------------------------------------------------------------------
Get an introduction to using Oracle Data Integrator, Java-based middleware
```

В виде *упражнения* можно выдать следующие команды и сравнить результаты запросов:

```
VARIABLE query1 VARCHAR2 ( 1000 )
```

```
VARIABLE query2 VARCHAR2 ( 1000 )
EXECUTE :query1 := 'oracle *developer'
EXECUTE :query2 := '"oracle *developer"'
@catsearch :query1 NULL
@catsearch :query2 NULL
```

Индекс по нескольким полям

Особенностью «каталожного» индекса является то, что он позволяет учитывать дополнительные, помимо основного, столбцы. В качестве такого столбца рассмотрим дату новости; этот же реальный пример заодно покажет и некоторые «овраги», не всегда заметные на «бумаге» фирменной документации Oracle.

Формат представления даты новости очевиден из показанных выше исходного документа XML и ответа на запрос к таблице. К сожалению, он не годится для нашей цели – включить поле PUBDATE в состав индексируемых, – и по двум причинам: (а) текст PUBDATE слишком длинен и (б) он не допускает сравнения значений времени как строк текста. Это можно рассматривать как ограничения языка запросов оператора CATSEARCH и индекса типа CTXSYS.CTXCAT, разрешающих использовать только дополнительные столбцы типа «число» и «строка», да и то, во втором случае не очень длинная. Поэтому чтобы продемонстрировать индексирование по нескольким полям нам придется этот столбец переделать. В качестве упрощения сочтем, что даты новостей *всегда* приводятся относительно Гринвичского часового пояса (как это есть в *имеющихся* данных), а далее и вовсе не будем принимать часовой пояс во внимание.

Удалим старый индекс и добавим новый столбец таблице:

```
DROP INDEX otnnews_idx;

ALTER TABLE otnnews ADD ( pubtime VARCHAR2 ( 30 ) );

UPDATE otnnews
SET pubtime = TO_CHAR (
                TO_TIMESTAMP_TZ ( pubdate, 'Dy, DD Mon YYYY HH24:MI:SS TZD' )
              , 'YYYY:MM:DD:HH24'
              )
;
```

Учет в дополнительных полей оформляется через механизм параметров индекса, а необходимые параметры заводятся последовательностью вызовов системных процедур:

```
BEGIN
   CTX_DDL.CREATE_INDEX_SET ( 'otntab_fields' );
   CTX_DDL.ADD_INDEX ( 'otntab_fields', 'pubtime' );
-- CTX_DDL.ADD_INDEX ( 'otntab_fields', '...' ); -- если надо, и другие столбцы
END;
/
```

Сформированное «предпочтение» OTNTAB_FIELDS укажем параметром новому индексу:

```
CREATE INDEX otnnews_idx
ON otnnews ( description )
INDEXTYPE IS CTXSYS.CTXCAT
PARAMETERS ( 'index set OTNTAB_FIELDS' )
;
```

В качестве *упражнения*, для созданного индекса можно устроить проверку типа выполнявшейся выше для «простого» «каталожного» индекса и убедиться в том, что добавление дополнительных столбцов индексации несколько усложняет техническую организацию индекса.

Проверка запросами

Первый запрос ничем не отличается от одного из предыдущих ни по форме, ни по результату:

```
CTX> @catsearch "'java | web'" NULL

DESCRIPTION
--------------------------------------------------------------------------------
Get an introduction to using Oracle Data Integrator, Java-based middleware
Learn how to secure PHP-based Web applications via database-based authentic
```

Следующий запрос показывает возможность сформировать критерий упорядочения, включающий обращение к *дополнительному* индексированному столбцу, в третьем аргументе оператора CATSEARH:

```
CTX> @catsearch "'java | web'" "'order by pubtime desc'"

DESCRIPTION
--------------------------------------------------------------------------------
Learn how to secure PHP-based Web applications via database-based authentic
Get an introduction to using Oracle Data Integrator, Java-based middleware
```

Следующий запрос показывает возможность сформировать условное выражение, включающее обращение к дополнительному индексированному столбцу, в третьем аргументе оператора CATSEARH:

```
CTX> VARIABLE texpr VARCHAR2 ( 256 )
CTX> BEGIN
  2  :texpr := TO_CHAR ( SYSDATE - 35, 'YYYY:MM:DD:HH24' );
  3  :texpr := 'pubtime < ''' || :texpr || '''';
  4  DBMS_OUTPUT.PUT_LINE ( :texpr );
  5* END;
CTX> /
pubtime < '2007:05:31:15'

PL/SQL procedure successfully completed.

CTX> @catsearch "'java | web'" :texpr

DESCRIPTION
--------------------------------------------------------------------------------
Get an introduction to using Oracle Data Integrator, Java-based middleware
```

К сожалению формулировки выражений для третьего аргумента CATSEARCH могут быть только очень простыми; в частности, в отличие от выражений SQL, они не терпят обращений к функциям. Тем не менее, возможно объединение фразы упорядочения и фильтрующего условного выражения.

Упражнение. Проверить это, сформировав уточнение запроса следующим значением третьего аргумента CATSEARCH: «order by pubtime desc AND pubtime < '2007:05:31:15'».

Реакция СУБД на ошибки работы с каталожным индексом (а в примерах выше их не было) может изнурить программиста программиста, к чему надо быть готовым.