Oracle: работать с текстовыми документами очень просто

Владимир Пржиялковский

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

Все началось с маленькой бумажки, которую принес в брезентовой разносной книге ленивый скороход из коммунотдела.

И. Ильф, Е. Петров. Золотой теленок.

Введение

СУБД Oracle известна в первую очередь как система управления «фактографическими» данными, но с первой половины 90-х годов в ней стали появляться возможности хранить и обрабатывать «сложно устроенные» данные. Одной из первых таких возможностей стала работа в версии 7.3 с частично структурированными данными: текстовыми документами.

До наших дней возможность работы с текстовыми документами в Oracle несколько раз поменяла название (SQL*TextRetrieval –> Text Server –> Oracle ConText –> Oracle Text) и существенно развилась. Начиная с версии 9 она встроена в обычную поставку СУБД Oracle, не требует, как ранее, отдельного лицензирования и автоматически включается в состав типовой БД. При отсутствии же в БД эту возможность можно установить самостоятельно либо при помощи DBCA либо прогоном сценария *dr0inst.sql* (версия 9 и предшествующие) или же *catctx.sql* (с версии 10) в *[ORACLE_HOME]/ctx/admin*.

Текстовые возможности Oracle находят внутреннее употребление, например в Oracle Ultra Search, Content Management (ранее iFS) или в XML DB.

Текстовые возможности СУБД Oracle основаны на использовании специального вида индекса, являющегося одним из встроенных в систему вариантов «предметного» индекса (domain index), используемого для организации работы со сложно устроенными данными. Oracle Text имеет в готовом виде три вида текстового индекса:

- CTXSYS.CONTEXT – для выполнения полнотекстового поиска по текстовым документам;
- CTXSYS.CTXCAT – для выполнения упрощенного и ускоренного поиска по «каталогам» (одно-двустрочным текстовым описаниям);
- CTXSYS.CTXRULE – для построения «классификаций» документов при том, что класс описывается набором характерных запросов.

Здесь рассматриваются общие возможности наиболее популярной разновидности индекса CTXSYS.CONTEXT. Этот вид текстового индекса позволяет хранить в БД текстовые документы и выполнять полнотекстовые запросы к документам как внутреннего хранения, так и внешнего (файловая система, интернет).

Простой пример

Подготовка данных

Для удобства создадим специального пользователя:

```
> CONNECT / AS SYSDBA

SYS> CREATE USER ctx IDENTIFIED BY ctx DEFAULT TABLESPACE users;
```

```
SYS> GRANT connect, resource, ctxapp TO ctx;

SYS> CONNECT ctx/ctx

CTX>
```

Роли CONNECT и RESOURCE приписаны пользователю CTX для простоты примера, и использовать их в рабочей БД неправильно; роль же CTXAPP употреблена по существу, так как без нее пользоваттель CTX не сможет обращаться к необходимым объектам схемы CTXSYS. Выполним:

```
CREATE TABLE docs ( doc_id NUMBER ( 10 ), vc2doc VARCHAR2 ( 4000 ) );

INSERT INTO docs VALUES ( 1, 'Mary had a little lamb' );
INSERT INTO docs VALUES ( 2, 'Twinkle, twinkle little star' );
INSERT INTO docs VALUES ( 3, 'This Lamb is my lamb' );

CREATE INDEX docs_vc2doc_idx ON docs ( vc2doc ) INDEXTYPE IS ctxsys.context;
```

Обратите внимание: индекс DOCS_VC2DOC_IDX – не простой, а «прикладной» (domain); точнее – предопределенного типа CTXSYS.CONTEXT, то есть «текстовый». В общем случае создание такого индекса содержит указание ряда специальных параметров (примеры будут далее), но для первого знакомства довольно положиться на умолчательные характеристики.

Примеры запросов

Основной для запросов к документам по индексу типа CTXSYS.CONTEXT является «оператор» CONTAINS. По своему употреблению оператор Oracle SQL практически не отличается от функции. Оператор CONTAINS возвращает меру, иначе степень, соответствия документа текстовому запросу («relevance»).

Несколько поясняющих примеров. Подготовка:

```
SELECT CONTAINS ( vc2doc, '&1' ) AS score, vc2doc FROM docs
.
SAVE simplequestion REPLACE
COLUMN vc2doc FORMAT A60
SET VERIFY OFF
```

Проверка:

```
CTX> @simplequestion 'star'

     SCORE VC2DOC
---------- ------------------------------------------------------------
         0 Mary had a little lamb
         4 Twinkle, twinkle little star
         0 This Lamb is my lamb

CTX> @simplequestion 'little'

     SCORE VC2DOC
---------- ------------------------------------------------------------
         4 Mary had a little lamb
         4 Twinkle, twinkle little star
         0 This Lamb is my lamb

CTX> @simplequestion 'twinkle'

     SCORE VC2DOC
---------- ------------------------------------------------------------
         0 Mary had a little lamb
```

```
         9 Twinkle, twinkle little star
         0 This Lamb is my lamb

CTX> @simplequestion 'lamb'

     SCORE VC2DOC
---------- ----------------------------------------------------------------
         4 Mary had a little lamb
         0 Twinkle, twinkle little star
         7 This Lamb is my lamb

CTX> @simplequestion 'mary AND lamb'

     SCORE VC2DOC
---------- ----------------------------------------------------------------
         4 Mary had a little lamb
         0 Twinkle, twinkle little star
         0 This Lamb is my lamb

CTX> @simplequestion 'mary lamb'

     SCORE VC2DOC
---------- ----------------------------------------------------------------
         0 Mary had a little lamb
         0 Twinkle, twinkle little star
         0 This Lamb is my lamb
```

Обратите внимание, что степень соответствия документа запросу не является простой частотой употребления в документе слова. Она зависит также от общего количества запрашиваемых документов и от количества документов, где есть искомые словоформы. Ее вычисление в Oracle основано на формуле Сэлтона (*http://www.ra.pae.osd.mil/doclib/servlet/HowWorksSalton*). Результат, который дает формула, отображается на диапазон целых чисел между 0 и 100.

Следующие несколько примеров, выполненных самостоятельно, помогут пояснить поведение оператора CONTAINS и получить представление о некоторых дополнительных возможностях контекстного запроса:

```
@simplequestion 'MARY AND LAMB'
@simplequestion 'MaRy AnD lAmB'
@simplequestion '%le'
@simplequestion 'lamb NOT mary'
@simplequestion 'NEAR ((lamb, mary) ,3)'
@simplequestion 'NEAR ((lamb, mary) ,2)'
@simplequestion 'mary ACCUM lamb'
@simplequestion 'mary ACCUM little'
@simplequestion 'mary ACCUM little lamb'
@simplequestion 'lamb OR little'
```

Полный перечень и описания реализованых операторов для составления контекстного запроса по документам (название «оператор» здесь неудачно совпадает с именованием «оператором» самой функции CONTAINS) имеется в документации по Oracle.

Возможности иной формулировки

На практике использование обращения к CONTAINS в выражениях для формирования столбцов в предложении SELECT не всегда удобно и не способствует эффективности. Вынужденная в этом отношении мера – использование функции («оператора») SCORE, возвращающей тот же результат, что и CONTAINS, но которую можно повторять в запросе многократно без боязни замедлить вычисление. Однако поскольку операторов CONTAINS в запросе может встречаться несколько, придумана специальная техника числовых «меток», устанавливающих соответствие операторов SCORE и CONTAINS в рамках запроса SQL. Метки указываются как параметр операторов (еще одна вынужденная и не очень элегантная мера) и выбираются произвольно. Примеры этой техники:

```
CTX> SELECT SCORE ( 1 ), vc2doc
  2  FROM docs
  3  WHERE CONTAINS ( vc2doc, 'lamb', 1 ) > 0
  4  ORDER BY SCORE ( 1 ) DESC
  5  ;

  SCORE(1) VC2DOC
---------- ------------------------------------------------
         7 This Lamb is my lamb
         4 Mary had a little lamb

CTX> SELECT SCORE ( 1 ), SCORE ( 15 ), vc2doc
  2  FROM docs
  3  WHERE
  4     CONTAINS ( vc2doc, 'lamb', 1 ) > 0
  5  OR CONTAINS ( vc2doc, 'lamb AND mary', 15 ) > 0
  6  ORDER BY
  7     SCORE ( 15 ) DESC
  8  ;

  SCORE(1)  SCORE(15) VC2DOC
---------- ---------- ----------------------------------
         4          4 Mary had a little lamb
         7          0 This Lamb is my lamb
```

Текстовый индекс

Практически обработку текстовой информации в Oracle Text обеспечивает текстовый индекс. Содержательно он организует хранение «обращенного списка», который по предъявленному поисковому слову выдает список пар <документ, словоместо>. Для этого он хранит список документов, позиций словоформ в документах и одно или несколько индексируемых слов в каждой позиции.

Технически текстовый индекс устроен сложнее обычных B-древовидного, или же поразрядного индексов хотя бы тем, что реализован сразу группой объектов и группой структур хранения. В этом легко удостовериться:

```
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
------------------------------ ------------------------------
DOCS_VC2DOC_IDX                INDEX
DR$DOCS_VC2DOC_IDX$X           INDEX
SYS_IOT_TOP_51619              INDEX
SYS_IOT_TOP_51624              INDEX
SYS_LOB0000051616C00006$$      LOB
SYS_LOB0000051621C00002$$      LOB
DOCS                           TABLE
DR$DOCS_VC2DOC_IDX$I           TABLE
DR$DOCS_VC2DOC_IDX$K           TABLE
DR$DOCS_VC2DOC_IDX$N           TABLE
DR$DOCS_VC2DOC_IDX$R           TABLE

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

SEGMENT_NAME                   SEGMENT_TYPE
------------------------------ ------------------------------
DR$DOCS_VC2DOC_IDX$X           INDEX
SYS_IOT_TOP_51619              INDEX
```

```
SYS_IOT_TOP_51624               INDEX
SYS_IL0000051616C00006$$        LOBINDEX
SYS_IL0000051621C00002$$        LOBINDEX
SYS_LOB0000051616C00006$$       LOBSEGMENT
SYS_LOB0000051621C00002$$       LOBSEGMENT
DOCS                            TABLE
DR$DOCS_VC2DOC_IDX$I            TABLE
DR$DOCS_VC2DOC_IDX$R            TABLE
```

В обоих запросах все объекты БД и структуры хранения, кроме DOCS, принадлежат текстовому индексу. Точнее, в результате команды `CREATE INDEX docs_vc2doc_idx ...` индекс DOCS_VC2DOC_IDX (типа DOMAIN) появился только как логический объект в БД; технически его реализуют четыре возникшие служебные таблицы:

- Таблица **DR$**DOCS_VC2DOC_IDX**$I**. Хранит перечень всех словоформ, попавших в индекс, внутренний номер документа («DOCID») и список позиций словоформ в документе. Вторичные, связанные с ней объекты:
  - индекс **DR$**DOCS_VC2DOC_IDX**$X** (обычный, типа NORMAL),
  - сегменты типа LOBSEGMENT и LOBINDEX для хранения данных поля TOKEN_INFO типа BLOB.
- Таблица **DR$**DOCS_VC2DOC_IDX**$K**. Хранит соответствие DOCID адресу ROWID строки с текстом или ссылкой на текст. Индексно-организованная таблица, хранится в структуре индекса.
- Таблица **DR$**DOCS_VC2DOC_IDX**$R**. Хранит список для обратного поиска: ROWID по DOCID. Вторичные, связанные с ней объекты:
  - сегменты типа LOBSEGMENT и LOBINDEX для хранения данных поля DATA типа BLOB.
- Таблица **DR$**DOCS_VC2DOC_IDX**$N**. Хранит список удаленных документов (DOCID) пополняющийся при оптимизации текстового индекса. Индексно-организованная таблица, хранится в структуре индекса.

Пример выдачи из таблицы DR$DOCS_VC2DOC_IDX$I:

```
CTX> SELECT token_text, token_count FROM dr$docs_vc2doc_idx$i;

TOKEN_TEXT                                         TOKEN_COUNT
-------------------------------------------------- -----------
LAMB                                                         2
LITTLE                                                       2
MARY                                                         1
STAR                                                         1
TWINKLE                                                      1
```

Еще одно отличие текстового индекса от обычного в том, что он не правится автоматически при правке документа. Например:

```
CTX> UPDATE docs SET vc2doc = 'This Land is my land' WHERE doc_id = 3;

1 row updated.

CTX> COMMIT;

Commit complete.

CTX> SELECT token_text, token_count FROM dr$docs_vc2doc_idx$i;

TOKEN_TEXT                                         TOKEN_COUNT
-------------------------------------------------- -----------
LAMB                                                         2
LITTLE                                                       2
MARY                                                         1
STAR                                                         1
TWINKLE                                                      1
```

В силу громоздкости текстового индекса сведения о необходимых исправлениях собираются в отдельной

таблице, а сама правка выполняется по мере надобности вручную:

```
CTX> SELECT pnd_index_name, pnd_rowid FROM ctx_user_pending;

PND_INDEX_NAME                 PND_ROWID
------------------------------ ------------------
DOCS_VC2DOC_IDX                AAAMm2AAEAAAABAAAC

CTX> EXECUTE CTX_DDL.SYNC_INDEX ( 'docs_vc2doc_idx' )

PL/SQL procedure successfully completed.

CTX> /

no rows selected

CTX> SELECT token_text, token_count FROM dr$docs_vc2doc_idx$i;

TOKEN_TEXT                                         TOKEN_COUNT
-------------------------------------------------- -----------
LAMB                                                         2
LAND                                                         1
LITTLE                                                       2
MARY                                                         1
STAR                                                         1
TWINKLE                                                      1
```

(Синхронизировать индекс можно и командой ALTER INDEX, но сейчас фирма Oracle этого не советует).

Стандартный прием – создать задание для плановой корректировки текстового индекса по расписанию.

Планы выполнения запросов

Неуклюжесть (отчасти вынужденная) правки текстового индекса компенсируется высокой скоростью обращения к нему при запросах к СУБД. Однако наблюдать план выполнения запроса приходится в этом случае своеобразно. Обычная команда EXPLAIN PLAN много не даст, но обращение к текстовому («прикладному») индексу она отметит:

```
CTX> EXPLAIN PLAN FOR
  2  SELECT * FROM docs
  3  WHERE CONTAINS ( vc2doc, 'twinkle AND star' ) > 0;

Explained.

CTX> SELECT * FROM TABLE ( dbms_xplan.display );

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

Plan hash value: 3477406887

--------------------------------------------------------------------------------
| Id | Operation                   | Name           |Rows|Bytes|Cost (%CPU)|Time    |
--------------------------------------------------------------------------------
|   0| SELECT STATEMENT            |                |   1| 2027|    4   (0)|00:00:01|
|   1|  TABLE ACCESS BY INDEX ROWID| DOCS           |   1| 2027|    4   (0)|00:00:01|
|*  2|   DOMAIN INDEX              | DOCS_VC2DOC_IDX |    |     |    4   (0)|00:00:01|
--------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("CTXSYS"."CONTAINS"("VC2DOC",'twinkle AND star')>0)
```

```
Note
-----
   - dynamic sampling used for this statement
```

(Форма выдачи плана соответствует версии 10, по которой готовился материал).

Детали отработки самого текстового (не SQL) запроса наблюдаются через специальную таблицу, а не привычную PLAN_TABLE. Создать ее можно примерно так:

```
CREATE GLOBAL TEMPORARY TABLE ctx_explain (
  explain_id  VARCHAR2 ( 30 )
, id          NUMBER
, parent_id   NUMBER
, operation   VARCHAR2 ( 30 )
, options     VARCHAR2 ( 30 )
, object_name VARCHAR2 ( 64 )
, position    NUMBER
, cardinality NUMBER
)
ON COMMIT PRESERVE ROWS
;
```

Просмотр плана для конкретного обращения к конкретному индексу делается через специальную процедуру из системного пакета CTX_QUERY:

```
BEGIN
ctx_query.explain (
  index_name    => 'docs_vc2doc_idx'
, text_query    => 'twinkle AND star'
, explain_table => 'ctx_explain'
, explain_id    => 'twinkle star'
);
END;
/
```

Пример просмотра сформированного в CTX_EXPLAIN плана обработки текстового запроса:

```
CTX> SELECT
  2    explain_id
  3  , id
  4  , parent_id
  5  , operation
  6  , options
  7  , object_name
  8  , position
  9  FROM
 10    ctx_explain
 11  ORDER BY
 12    id
 13  /

EXPLAIN_ID      ID  PARENT_ID OPERATION OPTIONS OBJECT_NAME    POSITION
-------------- --- ---------- --------- ------- -------------- ----------
twinkle star     1          0 AND                                     1
twinkle star     2          1 WORD              TWINKLE                1
twinkle star     3          1 WORD              STAR                   2
```