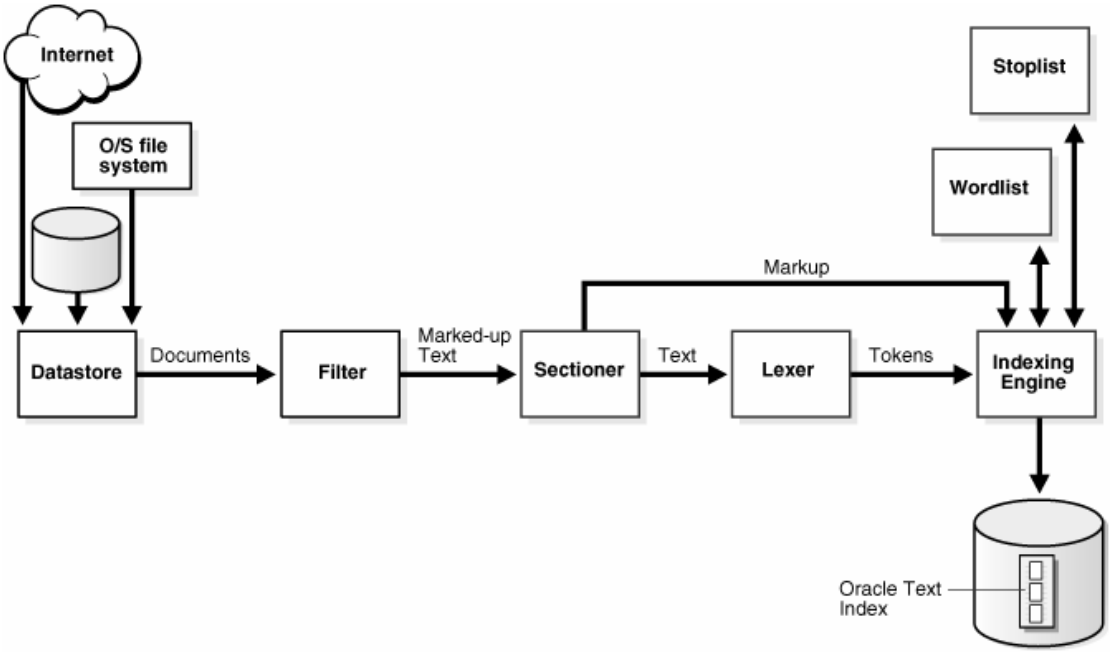
全文检索(oracle text)

Oracle Text 使 Oracle9i 具备了强大的文本检索能力和智能化的文本管理能力，Oracle Text 是 Oracle9i 采用的新名称，在 oracle8/8i 中被称为 oracle intermedia text，oracle8 以前是 oracle context cartridge。Oracle Text 的索引和查找功能并不局限于存储在数据库中的数据。它可以对存储于文件系统中的文档进行检索和查找，并可检索超过 150 种文档类型，包括 Microsoft Word、PDF 和 XML。Oracle Text 查找功能包括模糊查找、词干查找（搜索 mice 和查找 mouse）、通配符、相近性等查找方式，以及结果分级和关键词突出显示等。你甚至可以增加一个词典，以查找搭配词，并找出包含该搭配词的文档。

Oracle text 需要为可检索的数据项建立索引，用户才能够通过搜索查找内容，索引进程是根据管道建模的，在这个管道中，数据经过一系列的转换后，将其关键字会添加到索引中。该索引进程分为多个阶段，如下图



1.数据检索（Datastore）：只是将数据从数据存储（例如 web 页面、数据库大型对象或本地文件系统）中取出，然后作为数据流传送到下一个阶段。

2. 过滤(Filter)：过滤器负责将各种文件格式的数据转换为纯文本格式，索引管道中的其他组件只能处理纯文本数据，不能识别 Ms word 或 excel 等文件格式。

3. 分段(Sectioner)：分段器添加关于原始数据项结构的元数据。

4. 词法分析(Lexer)：根据数据项的语言将字符流分为几个字词。

5. 索引(Index)：最后一个阶段将关键字添加到实际索引中。

测试环境：

Linux AS release 4 (Nahant Update 3)， oracle10g（10.2.0.2.0）

内容简介：

本文档主要以实验为主，文档中包含了大量的实验例子，部分测试用例来自 document，部分来自网友的测试，所有的例子都在 oracle10g 中测试通过。

(1).

配置 oracle text

9i 之前，Oracle Text 不是默认安装的，必须手工安装。检查数据库中是否有 ctxsys 用户和 ctxapp 脚色，如果没有这个用户和角色，这意味着你在创建数据库时没有安装 oracle text 功能，需要先配置上该功能。

9i 安装 text

```
--创建表空间
SQL>create tablespace drsys datafile '/opt/oracle10g/oradata/10gtest/drsys01.dbf' size 100m;
SQL> connect / as sysdba
--创建 ctxsys 用户
SQL> @?/ctx/admin/dr0csys password default_tablespace_name temporary_tablespace_name
SQL> connect ctxsys/password
--创建数据字典
SQL> @?/ctx/admin/dr0inst @?/ctx/lib/libctxx9.so
SQL> @?/ctx/admin/defaults/drdefus.sql ;
```

Oracl0g 安装 text:

```
SQL> connect / as sysdba
SQL> @?/ctx/admin/catctx.sql ctxsys TBS_DRSYS temp unlock
Conn ctxsys/ctxsys
SQL> @?/ctx/admin/defaults/drdefel.sql
```

(2).

创建测试用户 oratext

```
Alter user ctxsys identified by ctxsys account unlock（10g 默认安装，帐号被锁定）;
Create tablespace oratext  datafile '/opt/oracle10g/oradata/10gtest/oratext01.dbf' size 2000m ;
Create user oratext  identified by oratect default tablespace oratext temporary tablespace temp;
Alter database tempfile '/opt/oracle10g/oradata/10gtest/temp01.dbf' resize 2000m;
SQL>Create user oratext  identified by oratext default tablespace oratext temporary tablespace temp;
Grant resource, connect, ctxapp to oratext;
Grant execute on ctxsys.ctx_cls to oratext;
Grant execute on ctxsys.ctx_ddl to oratext;
Grant execute on ctxsys.ctx_doc to oratext;
Grant execute on ctxsys.ctx_output to oratext;
Grant execute on ctxsys.ctx_query to oratext;
Grant execute on ctxsys.ctx_report to oratext;
Grant execute on ctxsys.ctx_thes to oratext;
Grant execute on ctxsys.ctx_ulexer to oratext;
```

查看系统默认的 oracle text 参数

```
Select pre_name, pre_object from ctx_preferences
```

(3).

Oracle Text 索引原理

Oracle text 索引将文本中所有的字符转化成记号（token），如 www.taobao.com 会转化成 www,taobao,com 这样的记号。

Oracle10g 里面支持四种类型的索引，context,ctxcat,ctxrule,ctxxpath

索引类型	描述	查询操作符
CONTEXT	用于对含有大量连续文本数据进行检索。支持 word、html、xml、text 等很多数据格式。支持范围（range）分区，支持并行创建索引（Parallel indexing）的索引类型。支持类型：VARCHAR2,CLOB,BLOB,CHAR,BFILE, XMLType, and URIType. DML 操作后，需要 CTX_DDL.SYNC_INDEX 手工同步索引 如果有查询包含多个词语，直接用空格隔开(如 oracle itpub)	CONTAINS
CTXCAT	适用于混合查询语句（如查询条件包括产品 id，价格，描述等）。适合于查询较小的具有一定结构的文本段。 具有事务性。DML 操作后，索引会自动进行同步。 操作符：and,or,>,<, =,between,in	CATSEARCH
CTXRULE	Use CTXRULE index to build a document classification or routing application. The CTXRULE index is an index created on a table of queries, where the queries define the classification or routing criteria.	MATCHES
CTXXPATH	Create this index when you need to speed up existsNode() queries on an XMLType column.	

1.

Context 索引

Oracle text 索引把全部的 word 转化成记号，context 索引的架构是反向索引（inverted index）,每个记号都映射着包含它自己的文本位置，如单词 dog 可能会有如下的条目

Dog	Doc1 Doc3 Doc5

这表示 dog 在文档 doc1，doc3，doc5 中都出现过。索引建好之后，系统中会自动产生如下 DR$MYINDEX$I,DR$MYINDEX$K,DR$MYINDEX$R,DR$MYINDEX$X,MYTABLE5 个表(假设表为 mytable,索引为 myindx)。Dml 操作后，context 索引不会自动同步，需要利用 ctx_ddl.sync_index 手工同步索引。

例子：

```
Create table docs (id number primary key, text varchar2(200));
Insert into docs values(1, '<html>california is a state in the us.</html>');
Insert into docs values(2, '<html>paris is a city in france.</html>');
Insert into docs values(3, '<html>france is in europe.</html>');
Commit;
/
--建立 context 索引
Create index idx_docs on docs(text)
    indextype is ctxsys.context parameters
    ('filter ctxsys.null_filter section group ctxsys.html_section_group');

--查询
Column text format a40;
Select id, text from docs where contains(text, 'france') > 0;

id   text
---------- --------------------------------
3    <html>france is in europe.</html>
2   <html>paris is a city in france.</html>
--继续插入数据
Insert into docs values(4, '<html>los angeles is a city in california.</html>');
Insert into docs values(5, '<html>mexico city is big.</html>');
commit;

Select  id, text from docs where contains(text, 'city') > 0;--新插入的数据没有查询到

id    text
-------------------------------------------
2   <html>paris is a city in france.</html>
--索引同步
begin
ctx_ddl.sync_index('idx_docs', '2m');
end;
--查询
Column text format a50;
Select id, text from docs where contains(text, 'city') > 0; --查到数据

id    text
-----------------------------------------------
5    <html>mexico city is big.</html>
4    <html>los angeles is a city in california.</html>
2    <html>paris is a city in france.</html>
-- or 操作符
```

```
Select id, text from docs where contains(text, 'city or state ') > 0;
--and 操作符
Select id, text from docs where contains(text, 'city and state ') > 0;
或是
Select id, text from docs where contains(text, 'city  state ') > 0;
--score 表示得分，分值越高，表示查到的数据越精确
SELECT SCORE(1), id, text FROM docs WHERE CONTAINS(text, 'oracle', 1) > 0;
```

Context 类型的索引不会自动同步，这需要在进行 Dml 后，需要手工同步索引。与 context 索引相对于的查询操作符为 contains

语法：

```
Contains(
        [schema.]column,
        text_query      varchar2
        [,label         number])
Return number;
```

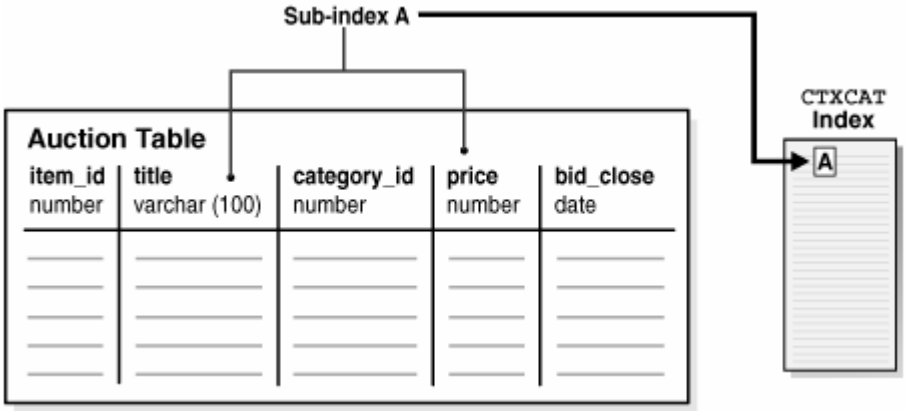
indextype is ctxsys.context：表示所定义的索引类型为 context

2.

Ctxcat 索引

用在多列混合查询中

Ctxcat 可以利用 index set 建立一个索引集，把一些经常与 ctxcat 查询组合使用的查询列添加到索引集中。比如你在查询一个商品名时，还需要查询生产日期，价格，描述等，你可可以将这些列添加到索引集中。oracle 将这些查询封装到 catsearch 操作中，从而提高全文索引的效率。在一些实时性要求较高的交易上，context 的索引不能自动同步显然是个问题，ctxcat 则会自动同步索引（原文：This example creates a catalog index for an auction site that sells electronic equipment such as cameras and CD players. New inventory is added everyday and item descriptions, bid dates, and prices must be stored together.The application requires good response time for mixed queries. The key is to determine what columns users frequently search so that we can create a suitable CTXCAT index. Queries on this type of index are issued with the CATSEARCH operator.）



例子：

```
Create table auction(Item_id number,Title varchar2(100),Category_id number,Price
number,Bid_close date);
Insert into auction values(1, 'nikon camera', 1, 400, '24-oct-2002');
Insert into auction values(2, 'olympus camera', 1, 300, '25-oct-2002');
Insert into auction values(3, 'pentax camera', 1, 200, '26-oct-2002');
Insert into auction values(4, 'canon camera', 1, 250, '27-oct-2002');
Commit;
/
--确定你的查询条件(很重要)
Determine that all queries search the title column for item descriptions
--建立索引集
begin
ctx_ddl.create_index_set('auction_iset');
ctx_ddl.add_index('auction_iset','price'); /* sub-index a*/
end;
--建立索引
Create index auction_titlex on auction(title) indextype is ctxsys.ctxcat
parameters ('index set auction_iset');

Column title format a40;
Select title, price from auction where catsearch(title, 'camera', 'order by price')> 0;

Title             price
--------------- ----------
Pentax camera         200
Canon camera          250
Olympus camera        300
Nikon camera          400

Insert into auction values(5, 'aigo camera', 1, 10, '27-oct-2002');
Insert into auction values(6, 'len camera', 1, 23, '27-oct-2002');
commit;
/
--测试索引是否自动同步
Select title, price from auction where catsearch(title, 'camera',
    'price <= 100')>0;

Title             price
--------------- ----------
aigo camera           10
len camera            23
```

添加多个子查询到索引集：

```
begin
```

```
ctx_ddl.drop_index_set('auction_iset');
ctx_ddl.create_index_set('auction_iset');
ctx_ddl.add_index('auction_iset','price'); /* sub-index A */
ctx_ddl.add_index('auction_iset','price, bid_close'); /* sub-index B */
end;
drop index auction_titlex;
Create index auction_titlex on auction(title) indextype is ctxsys.ctxcat
parameters ('index set auction_iset');

SELECT * FROM auction WHERE CATSEARCH(title, 'camera','price = 200 order by bid_close')>
0;
SELECT * FROM auction WHERE CATSEARCH(title, 'camera','order by price, bid_close')> 0;
```

任何的 Dml 操作后，Ctxcat 的索引会自动进行同步，不需要手工去执行，与 ctxcat 索引相对应的查询操作符是 catsearch.

语法：

```
Catsearch(
[schema.]column,
Text_query       varchar2,
Structured_query varchar2,
Return number;
```

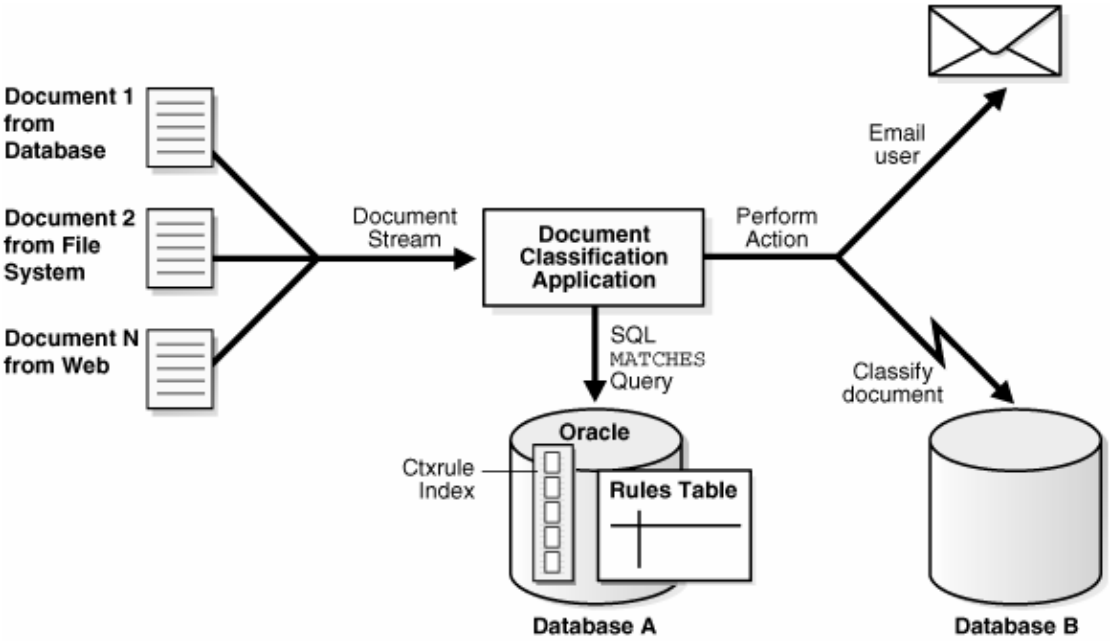
例子：

```
catsearch(text, 'dog', 'foo > 15')
catsearch(text, 'dog', 'bar = ''SMITH''')
catsearch(text, 'dog', 'foo between 1 and 15')
catsearch(text, 'dog', 'foo = 1 and abc = 123')
```

3.

Ctxrule 索引

The function of a classification application is to perform some action based on document content. These actions can include assigning a category id to a document or sending the document to a user. The result is classification of a document.



例子：

```
Create table queries (query_id number,query_string  varchar2(80));
insert into queries values (1, 'oracle');
insert into queries values (2, 'larry or ellison');
insert into queries values (3, 'oracle and text');
insert into queries values (4, 'market share');
commit;
--建立索引
Create index queryx on queries(query_string) indextype is ctxsys.ctxrule;

Column query_string format a35;
Select query_id,query_string from queries
    where matches(query_string,
                'oracle announced that its market share in databases
                 increased over the last year.')>0;

  query_id query_string
---------- -----------------------------------
         1 oracle
         4 market share
```

4.

Ctxxpath 索引

Create this index when you need to speed up existsNode() queries on an XMLType column

以上是 oracle text 索引的几种类型，通常用的比较多的是 context 和 ctxcat 索引。

语法参考:

http://download.oracle.com/docs/cd/B19306_01/text.102/b14218/csql.htm#i997226

接下来，我们来重新看一下索引的内部处理流程,先从 Datasotre 开始

类别	描述
Datastore	从哪儿得到数据
Filter	数据如何转化成文本
Lexer	正在索引什么语言
Wordlist	如何展开词根和模糊查询
Storage	索引怎么存储
Stop List	哪些词和主题不被索引
Section Group	允许分区查询吗，如何定义文档区段

(4).

Datastore 属性

数据检索负责将数据从数据存储（例如 web 页面、数据库大型对象或本地文件系统）中取出，然后作为数据流传送到下一个阶段。Datastore 包含的类型有 Direct datastore, Multi_column_datastore, Detail_datastore, File_datastore, Url_datastore, User_datastore, Nested_datastore。

Datastore Type	Use When
DIRECT_DATASTORE	检索表（数据存储在单列中）
MULTI_COLUMN_DATASTORE	检索表（数据存储在多列中）
DETAIL_DATASTORE	Data is stored internally in the text column. Document consists of one or more rows stored in a text column in a detail table, with header information stored in a master table.
FILE_DATASTORE	检索本地操作系统上的文档
NESTED_DATASTORE	检索嵌套表
URL_DATASTORE	检索 Internet 地址
USER_DATASTORE	Documents are synthesized at index time by a user-defined stored procedure.

1.

Direct datastore

支持存储数据库中的数据,单列查询.没有 attributes 属性

支持类型：char, varchar, varchar2, blob, clob, bfile,or xmltype.

例子：

```
Create table mytable(id number primary key, docs clob);
Insert into mytable values(111555,'this text will be indexed');
Insert into mytable values(111556,'this is a direct_datastore example');
Commit;
--建立 direct datastore
Create index myindex on mytable(docs)
  indextype is ctxsys.context
  parameters ('datastore ctxsys.default_datastore');
Select * from mytable where contains(docs, 'text') > 0;
```

2.

Multi_column_datastore

适用于索引数据分布在多个列中

the column list is limited to 500 bytes

支持 number 和 date 类型，在索引之前会先转化成 textt

raw and blob columns are directly concatenated as binary data.

不支持 long, long raw, nchar, and nclob, nested table

例子：

```
Create table mytable1(id number primary key, doc1 varchar2(400),doc2 varchar2(400),doc3
varchar2(400));
Insert into mytable1 values(1,'this text will be indexed','following example creates a
multi-column ','denotes that the bar column ');
Insert into mytable1 values(2,'this is a direct_datastore example','use this datastore when
your text is stored in more than one column','the system concatenates the text columns');
Commit;
/
--建立 multi datastore
Begin
Ctx_ddl.create_preference('my_multi', 'multi_column_datastore');
Ctx_ddl.set_attribute('my_multi', 'columns', 'doc1, doc2, doc3');
End;
--建立索引
Create index idx_mytable on mytable1(doc1)indextype is ctxsys.context
parameters('datastore my_multi')
Select * from mytable1 where contains(doc1,'direct datastore')>0;
Select * from mytable1 where contains(doc1,'example creates')>0;
--只更新从表，看是否能查到更新的信息
Update mytable1 set doc2='adladlhadad this datastore when your text is stored test' where
id=2;
Begin
Ctx_ddl.sync_index('idx_mytable');
End;
Select * from mytable1 where contains(doc1,'adladlhadad')>0; --没有记录
Update mytable1 set doc1='this is a direct_datastore example' where id=2; --更新主表
```

```
Begin
Ctx_ddl.sync_index('idx_mytable');--同步索引
End;
Select * from mytable1 where contains(doc1,'adladlhadad')>0; -查到从表的更新
```

对于多列的全文索引可以建立在任意一列上，但是，在查询时指定的列必须与索引时指定的列保持一致，只有索引指定的列发生修改，oracle 才会认为被索引数据发生了变化，仅修改其他列而没有修改索引列，即使同步索引也不会将修改同步到索引中.

3.

Detail_datastore

适用于主从表查询（原文：use the detail_datastore type for text stored directly in the database in detail tables, with the indexed text column located in the master table）

因为真正被索引的是从表上的列，选择主表的那个列作为索引并不重要，但是选定之后，查询条件中就必须指明这个列

主表中的被索引列的内容并没有包含在索引中

DETAIL_DATASTORE 属性定义

Attribute	Attribute Value
binary	Specify TRUE for Oracle Text to add no newline character after each detail row. Specify FALSE for Oracle Text to add a newline character (\n) after each detail row automatically.
detail_table	从表
detail_key	从表外键
detail_lineno	Specify the name of the detail table sequence column.
detail_text	从表索引列

例子：

```
create table my_master -建立主表
(article_id number primary key,author varchar2(30),title varchar2(50),body varchar2(1));
create table my_detail -建立从表
(article_id number, seq number, text varchar2(4000),
constraint fr_id foreign key (ARTICLE_ID)  references my_master (ARTICLE_ID));
--模拟数据
insert into my_master values(1,'Tom','expert on and on',1);
insert into my_master values(2,'Tom','Expert Oracle Database Architecture',2);
commit;
insert into my_detail values(1,1,'Oracle will find the undo information for this transaction
either in the cached
undo segment blocks (most likely) or on disk ');
insert into my_detail values(1,2,'if they have been flushed (more likely for very large
transactions).');
insert into my_detail values(1,3,'LGWR is writing to a different device, then there is no
```

```
contention for
redo logs');
insert into my_detail values(2,1,'Many other databases treat the log files as');
insert into my_detail values(2,2,'For those systems, the act of rolling back can be
disastrous');
commit;
--建立 detail datastore
begin
ctx_ddl.create_preference('my_detail_pref', 'DETAIL_DATASTORE');
ctx_ddl.set_attribute('my_detail_pref', 'binary', 'true');
ctx_ddl.set_attribute('my_detail_pref', 'detail_table', 'my_detail');
ctx_ddl.set_attribute('my_detail_pref', 'detail_key', 'article_id');
ctx_ddl.set_attribute('my_detail_pref', 'detail_lineno', 'seq');
ctx_ddl.set_attribute('my_detail_pref', 'detail_text', 'text');
end;
--创建索引
CREATE INDEX myindex123 on my_master(body) indextype is ctxsys.context
parameters('datastore my_detail_pref');
select * from my_master where contains(body,'databases')>0

--只更新从表信息，看是否还能查到
update my_detail set text='undo is generated as a result of the DELETE, blocks are modified,
and redo is sent over to
the redo log buffer' where article_id=2 and seq=1
begin
ctx_ddl.sync_index('myindex123','2m'); --同步索引
end;
select * from my_master where contains(body,'result of the DELETE')>0 -没有查到刚才的更新
--跟新从表后，更新主表信息
update my_master set body=3 where body=2
begin
ctx_ddl.sync_index('myindex123','2m');
end;
select * from my_master where contains(body,'result of the DELETE')>0 -查到数据
```

如果更新了子表中的索引列，必须要去更新主表索引列来使 oracle 认识到被索引数据发生变化（这个可以通过触发器来实现）。

4.

File_datastore

适用于检索本地服务器上的文件（原文：The FILE_DATASTORE type is used for text stored in files accessed through the local file system.）

多个路径标识：Unix 下冒号分隔开如 path1:path2:pathn　Windows 下用分号;分隔开

```
create table mytable1(id number primary key, docs varchar2(2000));
insert into mytable1 values(111555,'1.txt');
```

```
insert into mytable1 values(111556,'2.log');
commit;
--建立 file datastore
begin
 ctx_ddl.create_preference('COMMON_DIR','FILE_DATASTORE');
 ctx_ddl.set_attribute('COMMON_DIR','PATH','/opt/tmp');
end;
--建立索引
create index myindex on mytable1(docs) indextype is ctxsys.context parameters ('datastore
COMMON_DIR');
select * from mytable1 where contains(docs,'oracle')>0; --查询
```

Oracle 只支持在 file_datastore 中设置的路径中进行搜索，不支持指定路径的子目录（如/opt/tmp/log），如果文件发生了变化，只能是通过修改索引列的方式来通知 oracle，被索引数据已经发生了变化。

5.

Url_datastore

适用于检索 internet 上的信息，数据库中只需要存储相应的 url 就可以

Attribute	Attribute Value
timeout	Specify the timeout in seconds. The valid range is 15 to 3600 seconds. The default is 30.
maxthreads	Specify the maximum number of threads that can be running simultaneously. Use a number between 1and 1024. The default is 8.
urlsize	Specify the maximum length of URL string in bytes. Use a number between 32 and 65535. The default is 256.
maxurls	Specify maximum size of URL buffer. Use a number between 32 and 65535. The defaults is 256.
maxdocsize	Specify the maximum document size. Use a number between 256 and 2,147,483,647 bytes (2 gigabytes). The defaults is 2,000,000.
http_proxy	Specify the host name of http proxy server. Optionally specify port number with a colon in the form hostname:port.
ftp_proxy	Specify the host name of ftp proxy server. Optionally specify port number with a colon in the form hostname:port.
no_proxy	Specify the domain for no proxy server. Use a comma separated string of up to 16 domain names.

例子：

```
create table urls(id number primary key, docs varchar2(2000));
insert into urls values(111555,'http://context.us.oracle.com');
insert into urls values(111556,'http://www.sun.com');
insert into urls values(111557,'http://www.itpub.net');
insert into urls values(111558,'http://www.ixdba.com');
```

```
commit;
/
--建立 url datastore
begin
 ctx_ddl.create_preference('URL_PREF','URL_DATASTORE');
 ctx_ddl.set_attribute('URL_PREF','Timeout','300');
end;
--建立索引
create index datastores_text on urls (docs) indextype is ctxsys.context parameters
( 'Datastore URL_PREF' );

select * from urls where contains(docs,'Aix')>0
```

若相关的 url 不存在，oracle 并不会报错，只是查询的时候找不到数据而已。
oracle 中仅仅保存被索引文档的 url 地址，如果文档本身发生了变化，必须要通过修改索引列（url 地址列）的方式来告知 oracle，被索引数据已经发生了变化。

6.

User_datastore

Use the USER_DATASTORE type to define stored procedures that synthesize documents during indexing. For example, a user procedure might synthesize author, date, and text columns into one document to have the author and date information be part of the indexed text.

7.

Nested_datastore

全文索引支持将数据存储在嵌套表中

8.

参考脚本

--建立 direct_store

```
Create index myindex on mytable(docs)
  indextype is ctxsys.context
  parameters ('datastore ctxsys.default_datastore');
```

--建立 mutil_column_datastore

```
Begin
Ctx_ddl.create_preference('my_multi', 'multi_column_datastore');
Ctx_ddl.set_attribute('my_multi', 'columns', 'doc1, doc2, doc3');
End;
Create index idx_mytable on mytable1(doc1)indextype is ctxsys.context
parameters('datastore my_multi')
```

--建立 file_datafilestore

```
begin
 ctx_ddl.create_preference('COMMON_DIR','FILE_DATASTORE');
 ctx_ddl.set_attribute('COMMON_DIR','PATH','/opt/tmp');
```

```
end;
create index myindex on mytable1(docs) indextype is ctxsys.context parameters ('datastore
COMMON_DIR');

--建立 url_datastore
begin
 ctx_ddl.create_preference('URL_PREF','URL_DATASTORE');
 ctx_ddl.set_attribute('URL_PREF','Timeout','300');
end;
create index datastores_text on urls (docs) indextype is ctxsys.context parameters
( 'Datastore URL_PREF' );
```

(5).

Filter 属性

过滤器负责将各种文件格式的数据转换为纯文本格式,索引管道中的其他组件只能处理纯文本数据，不能识别 microsoft word 或 excel 等文件格式，filter 有 charset_filter、inso_filter、null_filter、user_filter、procedure_filter 几种类型。

Filter	When Used
CHARSET_FILTER	Character set converting filter
AUTO_FILTER	Auto filter for filtering formatted documents
NULL_FILTER	No filtering required. Use for indexing plain text, HTML, or XML documents
MAIL_FILTER	Use the MAIL_FILTER to transform RFC-822, RFC-2045 messages in to indexable text.
USER_FILTER	User-defined external filter to be used for custom filtering
PROCEDURE_FILTER	User-defined stored procedure filter to be used for custom filtering.

1.

CHARSET_FILTER

把文档从非数据库字符转化成数据库字符（原文：Use the CHARSET_FILTER to convert documents from a non-database character set to the character set used by the database）

例子：

```
create table hdocs (     id number primary key,     fmt varchar2(10),     cset varchar2(20),
    text varchar2(80)
);

begin
```

```
cxt_ddl.create.preference('cs_filter', 'CHARSET_FILTER');
ctx_ddl.set_attribute('cs_filter', 'charset', 'UTF8');
end

insert into hdocs values(1, 'text', 'WE8ISO8859P1', '/docs/iso.txt');
insert into hdocs values (2, 'text', 'UTF8', '/docs/utf8.txt');
commit;
create index hdocsx on hdocs(text) indextype is ctxsys.context
  parameters ('datastore ctxsys.file_datastore
  filter cs_filter
  format column fmt
  charset column cset');
```

2.

NULL_FILTER

默认属性，不进行任何过滤

oracle 不建议对 html、xml 和 plain text 使用 auto_filter 参数，oracle 建议你使用 null_filter 和 section group type

--建立 null filter

```
create index myindex on docs(htmlfile) indextype is ctxsys.context
  parameters('filter ctxsys.null_filter section group ctxsys.html_section_group');
```

Filter 的默认值会受到索引字段类型和 datastore 的类型的影响，对于存储在数据库中的 varchar2、char 和 clob 字段中的数据，oracle 自动选择了 null_filtel,若 datastore 的属性设置为 file_datastore，oracle 会选择 auto_filter 作为默认值。

3.

AUTO_FILTER

通用的过滤器，适用于大部分文档，包括 PDF 和 Ms word，过滤器还会自动识别出 plain-text, HTML, XHTML, SGML 和 XML 文档

```
Create table my_filter (id number, docs varchar2(1000));
Insert into my_filter values (1, 'Expert Oracle Database Architecture.pdf');
Insert into my_filter values (2, '1.txt');
Insert into my_filter values (3, '2.doc');
commit;
/
--建立 file datastore
Begin
 ctx_ddl.create_preference('test_filter', 'file_datastore');
 ctx_ddl.set_attribute('test_filter', 'path', '/opt/tmp');
End;
--错误信息表
select * from CTX_USER_INDEX_ERRORS
--建立 auto filter
```

```
Create index idx_m_filter on my_filter (docs) indextype is ctxsys.context
parameters ('datastore test_filter filter ctxsys.auto_filter');
select * from my_filter where contains(docs,'oracle')>0
```

AUTO_FILTER 能自动识别出大部分格式的文档，我们也可以显示的通过 column 来指定文档类型，有 text,binary,ignore,设置为 binary 的文档使用 auto_filter,设置为 text 的文档使用 null_filter,设置为 ignore 的文档不进行索引。

```
create table hdocs (id number primary key,fmt varchar2(10),text varchar2(80));
insert into hdocs values(1, 'binary', '/docs/myword.doc');
insert in hdocs values (2, 'text', '/docs/index.html');
insert in hdocs values (2, 'ignore', '/docs/1.txt');
commit;
create index hdocsx on hdocs(text) indextype is ctxsys.context
parameters ('datastore ctxsys.file_datastore filter ctxsys.auto_filter  format column
fmt');
```

4.

MAIL_FILTER

通过 mail_filter 把 RFC-822,RFC-2045 信息转化成索引文本

限制:

文档必须是 us-ascii

长度不能超过 1024bytes

document must be syntactically valid with regard to RFC-822

5.

USER_FILTER

Use the USER_FILTER type to specify an external filter for filtering documents in a column

6.

PROCEDURE_FILTER

Use the PROCEDURE_FILTER type to filter your documents with a stored procedure. The stored procedure is called each time a document needs to be filtered.

7.

参考脚本

--建立 null filter

```
create index myindex on docs(htmlfile) indextype is ctxsys.context
  parameters('filter ctxsys.null_filter section group ctxsys.html_section_group');
```

--建立 auto filter

```
Create index idx_m_filter on my_filter (docs) indextype is ctxsys.context
parameters ('datastore test_filter filter ctxsys.auto_filter');
```

Filter 错误记录表：CTX_USER_INDEX_ERRORS

(6).

Lexer 属性

Oracle 全文检索的 lexer 属性用于处理各种不同的语言，最基本的英文使用 basic_lexer，中文则可以使用 chinese_vgram_lexer 或 chinese_lexer。

Type	Description
BASIC_LEXER	英文或西方语言（使用空格作为分词界定）
MULTI_LEXER	多语言版本
CHINESE_VGRAM_LEXER	中文版本
CHINESE_LEXER	更智能的中文版本
JAPANESE_VGRAM_LEXER	Lexer for extracting tokens from Japanese text.
JAPANESE_LEXER	Lexer for extracting tokens from Japanese text.
KOREAN_MORPH_LEXER	Lexer for extracting tokens from Korean text.
USER_LEXER	Lexer you create to index a particular language.
WORLD_LEXER	Lexer for indexing tables containing documents of different languages; autodetects languages in a document

1.

Basic_lexer

basic_lexer 属性支持如英语、德语、荷兰语、挪威语、瑞典语等以空格作为界限的语言（原文：Use the BASIC_LEXER type to identify tokens for creating Text indexes for English and all other supported whitespace-delimited languages.）

```
Create table my_lex (id number, docs varchar2(1000));
Insert into my_lex values (1, 'this is a example for the basic_lexer');
Insert into my_lex values (2, 'he following example sets Printjoin characters ');
Insert into my_lex values (3, 'To create the INDEX with no_theme indexing and with printjoins characters');
Insert into my_lex values (4, '中华人民共和国');
Insert into my_lex values (5, '中国淘宝软件');
Insert into my_lex values (6, '测试 basic_lexer 是否支持中文');
Commit;
/
--建立 basic_lexer
begin
ctx_ddl.create_preference('mylex', 'BASIC_LEXER');
```

```
ctx_ddl.set_attribute ('mylex', 'printjoins', '_-'); --保留_ -符号
ctx_ddl.set_attribute ( 'mylex', 'index_themes', 'NO');
ctx_ddl.set_attribute ( 'mylex', 'index_text', 'YES');
ctx_ddl.set_attribute ('mylex','mixed_case','yes'); --区分大小写
end;
create index indx_m_lex on my_lex(docs) indextype is ctxsys.context parameters('lexer
mylex');
Select id from my_lex where contains(docs, 'no_theme') > 0;
select docs from my_lex where contains(docs,'中国')>0
```

2.

Mutil_lexer

支持多种语言的文档，比如你可以利用这个 lexer 来定义包含 Endlish,German 和 Japanese 的文档（原文：Use MULTI_LEXER to index text columns that contain documents of different languages. For example, you can use this lexer to index a text column that stores English, German, and Japanese documents.）建立一个 multi_lexer 属性的索引，并通过 language 列设置需要索引的语言，Oracle 会根据 language 列的内容去匹配 add_sub_lexer 过程中指定的语言标识符，如果匹配的上，就使用该 sub_lexer 作为索引的 lexer，如果没有找到匹配的，就使用 default 语言作为索引的 lexer 列，注意客户端 nls_language，可能会影响 lexer 的选择

```
Select * from v$nls_parameters where parameter = 'NLS_LANGUAGE';
```

alter session set nls_language='simplified chinese';

alter session set nls_language='american';

例子：

```
create table globaldoc ( doc_id number primary key,lang varchar2(3),text clob);
--建立 multi_lexer
begin
ctx_ddl.create_preference('english_lexer','basic_lexer');
ctx_ddl.set_attribute('english_lexer','index_themes','yes');
ctx_ddl.set_attribute('english_lexer','theme_language','english');
ctx_ddl.create_preference('german_lexer','basic_lexer');
ctx_ddl.set_attribute('german_lexer','composite','german');
ctx_ddl.set_attribute('german_lexer','mixed_case','yes');
ctx_ddl.set_attribute('german_lexer','alternate_spelling','german');
ctx_ddl.create_preference('japanese_lexer','japanese_vgram_lexer');
ctx_ddl.create_preference('global_lexer', 'multi_lexer');
ctx_ddl.add_sub_lexer('global_lexer','default','english_lexer');
ctx_ddl.add_sub_lexer('global_lexer','german','german_lexer','ger');
ctx_ddl.add_sub_lexer('global_lexer','japanese','japanese_lexer','jpn');
end;

create index globalx on globaldoc(text) indextype is ctxsys.context
parameters ('lexer global_lexer language column lang');
```

3.

chinese_vgram_lexer 和 chinese_lexer

basic_lexer 只能识别出被空格、标点和回车符分隔出来的部分，如果要对中文内容进行索引的话，就必须使用 chinese_vgram_lexer 或是 chinese_lexer

Chinese_lexer 相比 chinese_vgram_lexer 有如下的优点：

产生的索引更小

更好的查询响应时间

产生更接近真实的索引切词，使得查询精度更高

支持停用词

因为 chinese_lexer 采用不同的算法来标记 tokens, 建立索引的时间要比 chinese_vgram_lexer 长.

字符集：支持 al32utf8，zhs16cgb231280，zhs16gbk，zhs32gb18030，zht32euc，zht16big5 zht32tris， zht16mswin950，zht16hkscs，utf8

```
--建立 chinese lexer
Begin
 ctx_ddl.create_preference('my_chinese_vgram_lexer', 'chinese_vgram_lexer');
 ctx_ddl.create_preference('my_chinese_lexer', 'chinese_lexer');
End;
-- chinese_vgram_lexer
Create index ind_m_lex1 on my_lex(docs) indextype is ctxsys.context
Parameters ('lexer ctxsys.my_chinese_vgram_lexer');
Select * from t where contains(docs, '中国') > 0;

-- chinese_lexer
Create index ind_m_lex2 on my_lex(docs) indextype is ctxsys.context
Parameters ('lexer ctxsys.my_chinese_lexer');
Select * from t where contains(docs, '中国') > 0;
```

4.

User_lexer

Use USER_LEXER to plug in your own language-specific lexing solution. This enables you to define lexers for languages that are not supported by Oracle Text. It also enables you to define a new lexer for a language that is supported but whose lexer is inappropriate for your application.

5

Default_lexer

如果数据库在建立的时候指定的是中文则 default_lexer 为 chinese_vgram_lexer，如果是英文，则 default_lexer 为 basic_lexer

6.

Query_procedure

This callback stored procedure is called by Oracle Text as needed to tokenize words in the query. A space-delimited group of characters (excluding the query operators) in the query will be identified by Oracle Text as a word.

7.

参考脚本

--建立 basic_lexer

```
begin
ctx_ddl.create_preference('mylex', 'BASIC_LEXER');
ctx_ddl.set_attribute ('mylex', 'printjoins', '_-');  --保留_ -符号
ctx_ddl.set_attribute ('mylex','mixed_case','yes'); --区分大小写
end;
create index indx_m_lex on my_lex(docs) indextype is ctxsys.context parameters('lexer
mylex');
```

--建立 chinese_vgram_lexer 或是 chinese_lexer

```
Begin
 ctx_ddl.create_preference('my_chinese_vgram_lexer', 'chinese_vgram_lexer');
 ctx_ddl.create_preference('my_chinese_lexer', 'chinese_lexer');
End;
-- chinese_vgram_lexer
Create index ind_m_lex1 on my_lex(docs) indextype is ctxsys.context
Parameters ('lexer ctxsys.my_chinese_vgram_lexer');
```

(7).

Section Group 属性

Section group 支持查询包含内部结构的文档（如 html、xml 文档等），可以指定对文档的某一部分进行查询,你可以将查询范围限定在标题 head 中。在 html、xml 等类似结构的文档中，除了用来显示的内容外，还包括了大量用于控制结构的标识，而这些标识可能是不希望被索引的，这就是 section group 的一个主要功能(原文：In order to issue WITHIN queries on document sections, you must create a section group before you define your sections)

Type	Description
NULL_SECTION_GROUP	系统默认参数，不进行任何过滤，有 SENTENCE ，PARAGRAPH 属性
BASIC_SECTION_GROUP	支持<tag>开头以</tag>结尾的结构的文档
HTML_SECTION_GROUP	支持 html 文档
XML_SECTION_GROUP	支持 xml 文档
AUTO_SECTION_GROUP	Use this group type to automatically create a zone section for each start-tag/end-tag pair in an XML document. The section names derived from XML tags are case sensitive as in XML. Attribute sections are created automatically for XML tags that have attributes. Attribute sections are named in the form tag@attribute. Stop sections, empty tags, processing instructions, and

Type	Description
	comments are not indexed. The following limitations apply to automatic section groups: • You cannot add zone, field, or special sections to an automatic section group. • You can define a stop section that applies only to one particular type; that is, if you have two different XML DTDs, both of which use a tag called FOO, you can define (TYPE1)FOO to be stopped, but(TYPE2)FOO to not be stopped. • The length of the indexed tags, including prefix and namespace, cannot exceed 64 bytes. Tags longer than this are not indexed.
PATH_SECTION_GROUP	Use this group type to index XML documents. Behaves like the AUTO_SECTION_GROUP. The difference is that with this section group you can do path searching with the INPATH and HASPATH operators. Queries are also case-sensitive for tag and attribute names. Stop sections are not allowed.
NEWS_SECTION_GROUP	Use this group for defining sections in newsgroup formatted documents according to RFC 1036.

1.

Null_section_group

系统默认，不进行任何节的过滤

例子：

```
Create table my_sec (id number, docs varchar2(100));
Insert into my_sec values (1, 'a simple section group, test null_section_group attribute.');
Insert into my_sec values (2, 'this record one, can be query in nornal');
Insert into my_sec values (4, 'this record
 are tested for
 the query in paragraph');
Commit;
/
--定义null_section_group
Create index ind_m_sec on my_sec(docs) indextype is ctxsys.context
parameters ('section group ctxsys.null_section_group');

Select * from my_sec where contains(docs, 'record and query') > 0;
```

```
--要预先定义 sentence 或 paragraph'，否则查询会出错
Select * from my_sec where contains(docs, '(record and query) within sentence') > 0;
Begin
 ctx_ddl.create_section_group('test_null', 'null_section_group');
 ctx_ddl.add_special_section('test_null', 'sentence');
 ctx_ddl.add_special_section('test_null', 'paragraph');
End;

drop index ind_m_sec;
Create index ind_m_sec on my_sec(docs) indextype is ctxsys.context
parameters ('section group test_null');

Select * from my_sec where contains(docs, '(record and query) within sentence') > 0;
Select * from my_sec where contains(docs, '(record and query) within paragraph') > 0;
```

2.

Basic_section_group

basic_section_group 才是支持节搜索的最基础的一种属性，但是它只支持以<tag>开头以</tag>结尾的结构的文档

```
Create table my_sec1 (id number, docs varchar2(1000));
Insert into my_sec1 values (1, '<heading>title</heading>
<context>this is the contents of the example.
Use this example to test the basic_section_group.</context>');
Insert into my_sec1 values (2, '<heading>example</heading>
<context>this line incluing the word title too.</context>');
Commit;
/
Create index ind_my_sec1 on my_sec1(docs) indextype is ctxsys.context;
Select * from my_sec1 where contains (docs, 'heading') > 0;
--定义 basic_section_group
Begin
Ctx_ddl.create_section_group('test_basic', 'basic_section_group');
End;
drop index ind_my_sec1;
Create index ind_my_sec1 on my_sec1(docs) indextype is ctxsys.context
parameters ('section group test_basic');

Select * from my_sec1 where contains (docs, 'heading') > 0;
Select * from my_sec1 where contains (docs, 'context') > 0;
Select * from my_sec1 where contains (docs, 'use') > 0;
```

节搜索的另一个主要功能就是可以限制查询的范围，上面的文档包含了两部分，标题和正文，其中标题使用标签<heading>，正文使用标签<context>,我们可以对 basic_section_group 添加区域属性，运行查询在文档的某个范围内进行

```
Drop index ind_my_sec1;
```

```
Begin
 ctx_ddl.add_zone_section('test_basic', 'head', 'heading');
End;
Create index ind_my_sec1 on my_sec1(docs) indextype is ctxsys.context
parameters ('section group test_basic');
Select * from my_sec1 where contains (docs, 'title') > 0;
--在 head 里面查询
Select * from my_sec1 where contains (docs, 'title within head') > 0;
```

3.

Html_section_group

Html 文档具有很多不规范的表示方法，oracle 建议使用 html_section_group 以便能够得到更好的识别

```
--定义 html_section_group
begin
ctx_ddl.create_section_group('htmgroup', 'HTML_SECTION_GROUP');
end;

create index myindex on docs(htmlfile) indextype is ctxsys.context
parameters('filter ctxsys.null_filter section group htmgroup');
```

无论是 field_section 还是 zone_section，表示文档的 tag 标签都是大小写敏感的，其大小写需要和原文中匹配

4.

Xml_section_group

Xml 文档的格式要求比 html 文档严谨、规范，这也使得 xml_section_group 比 html_section_group 具有了更多的功能

例子：

```
Create table my_sec2 (id number, docs varchar2(1000));
Insert into my_sec2 values (1, 'context.xml');
commit;
/
--定义 xml_section_group
Begin
 ctx_ddl.create_preference('test_file', 'file_datastore');
 ctx_ddl.set_attribute('test_file', 'path', '/opt/tmp');
 ctx_ddl.create_section_group('test_html', 'html_section_group');
 ctx_ddl.create_section_group('test_xml', 'xml_section_group');
End;

Create index ind_t_docs on my_sec2 (docs) indextype is ctxsys.context
parameters('datastore ctxsys.test_file filter ctxsys.null_filter section group
ctxsys.test_xml')
```

```
Select * from my_sec2 where contains (docs, 'complete') > 0;
Begin
 ctx_ddl.add_attr_section('test_xml', 'name', 'const@name');
End;
Select * from my_sec2 where contains (docs, 'complete within name') > 0;
```

5.

Auto_section_group

Xml_section_group 的增强型，对于 xml_section_group 用户需要自己添加需要定义的节组，而使用 auto_section_group，则 oracle 会自动添加节组以及属性信息

6.

Path_section_group

和 auto_section_group 十分类似，path_section_group 比 auto_section_group 增加了 haspath 和 inpath 操作，但是 path_section_group 不支持 add_stop_section 属性

7.

参考脚本

--建立 null_section_group

```
Create index ind_m_sec on my_sec(docs) indextype is ctxsys.context
parameters ('section group ctxsys.null_section_group');
```

--建立 basic_section_group

```
Begin
Ctx_ddl.create_section_group('test_basic', 'basic_section_group');
End;
Begin
 ctx_ddl.add_zone_section('test_basic', 'head', 'heading'); --设定节查询
End;
Create index ind_my_sec1 on my_sec1(docs) indextype is ctxsys.context
parameters ('section group test_basic');
```

--建立 Html_section_group

```
begin
ctx_ddl.create_section_group('htmgroup', 'HTML_SECTION_GROUP');
end;
create index myindex on docs(htmlfile) indextype is ctxsys.context
parameters('filter ctxsys.null_filter section group htmgroup');
```

--建立 Xml_section_group

```
Begin
 ctx_ddl.create_section_group('test_xml', 'xml_section_group');
End;
```

```
Create index ind_t_docs on my_sec2 (docs) indextype is ctxsys.context
parameters('filter ctxsys.null_filter section group ctxsys.test_xml')
```

(8).

Storage 属性

Oracle 全文检索通常会生成一系列的辅助表，生成规则是 dr$+索引名+$+表用途标识，由于这些表是 oracle 自动生成的，通常没有办法为这些表指定存储空间。为构造 text 索引所生成的辅助表指定表空间、存储参数（use the storage preference to specify tablespace and creation parameters for tables associated with a text index），oracle 提供了单一的存储类型 basic_storage。

在 mytable1 表中建立了全文索检索 myindex，系统中会自动产生如下 5 个表：DR$MYINDEX$I,DR$MYINDEX$K,DR$MYINDEX$R,DR$MYINDEX$X,MYTABLE1

Attribute	Attribute Value
i_table_clause	指定 dr$indexname$I 的存储参数
k_table_clause	指定 dr$indexname$K 的存储参数，K 表存储键值映射表（keymap）
r_table_clause	指定 dr$indexname$R 的存储参数，R 表存储 rowid 表 The default clause is:’LOB(DATA) STORE AS (CACHE)’
n_table_clause	指定 dr$indexname$N 的存储参数，N 表存储负键值链表（negative list ）
i_index_clause	指定 dr$indexname$X 的存储参数，I 表存储索引数据表(index data table） The default clause is:’COMPRESS 2’which instructs Oracle Text to compress this index table. 另：oracle 建议使用 COMPRESS 2 以节省空间存储
p_table_clause	只有在设置 BASIC_WORDLIST 的 SUBSTRING_INDEX 属性才有用，P 表示是索引组织表

1.

参考脚本

--建立 basic storage

```
Begin
  Ctx_ddl.create_preference('mystore', 'basic_storage'); --建立 storage
  Ctx_ddl.set_attribute('mystore',     --设置参数
                    'i_table_clause',
                    'tablespace foo storage (initial 1k)');
  Ctx_ddl.set_attribute('mystore',
                    'k_table_clause',
                    'tablespace foo storage (initial 1k)');
  Ctx_ddl.set_attribute('mystore',
                    'r_table_clause',
                    'tablespace users storage (initial 1k) lob
```

```
                    (data) store as (disable storage in row cache)');
  Ctx_ddl.set_attribute('mystore',
                    'n_table_clause',
                    'tablespace foo storage (initial 1k)');
  Ctx_ddl.set_attribute('mystore',
                    'i_index_clause',
                    'tablespace foo storage (initial 1k) compress 2');
  Ctx_ddl.set_attribute('mystore',
                    'p_table_clause',
                    'tablespace foo storage (initial 1k)');
End;
--建立索引
Create index indx_m_word on my_word(docs) indextype is ctxsys.context
parameters('storage mystore');
```

(9).

Wordlist 属性

Oracle 全文检索的 wordlist 属性用来设置模糊查询和同词根查询，wordlist 属性还支持子查询和前缀查询，oracle 的 wordlist 属性只有 basic_wordlist 一种（原文：Use the wordlist preference to enable the query options such as stemming, fuzzy matching for your language. You can also use the wordlist preference to enable substring and prefix indexing, which improves performance for wildcard queries with CONTAINS and CATSEARCH.）

1.

例子：

```
Create table my_word (id number, docs varchar2(1000));
Insert into my_word values (1, 'Specify the stemmer used for word stemming in Text queries');
Insert into my_word values (2, 'Specify which fuzzy matching routines are used for the
column');
Insert into my_word values (3, 'Fuzzy matching is currently supported for English');
Insert into my_word values (4, 'Specify a default lower limit of fuzzy score. Specify a
number between 0 and 80');
Insert into my_word values (5, 'Specify TRUE for Oracle Text to create a substring index
matched.');
commit;
/
--建立 wordlist
Begin
   ctx_ddl.drop_preference('mywordlist');
   ctx_ddl.create_preference('mywordlist', 'basic_wordlist');
   ctx_ddl.set_attribute('mywordlist','fuzzy_match','english'); --模糊匹配，英语
```

```
   ctx_ddl.set_attribute('mywordlist','fuzzy_score','0');       --匹配得分
   ctx_ddl.set_attribute('mywordlist','fuzzy_numresults','5000');
   ctx_ddl.set_attribute('mywordlist','substring_index','true'); --左查询，适用%to,%to%
   ctx_ddl.set_attribute('mywordlist','stemmer','english');  --词根
  ctx_ddl.set_attribute('mywordlist', 'prefix_index', 'true');  --右查询，适用 t0%
End;

Create index indx_m_word on my_word(docs) indextype is ctxsys.context
parameters('wordlist mywordlist');
--例子
Select docs from my_word where contains(docs,'$match')>0 ; --词根查询
Select docs from my_word where contains(docs,'MA%')>0;    --匹配查询
```

2.

document 上的例子

```
create table quick( quick_id number primary key,  text  varchar(80) );
--- insert a row with 10 expansions for 'tire%'
insert into quick ( quick_id, text )
  values ( 1, 'tire tirea tireb tirec tired tiree tiref tireg tireh tirei tirej');
commit;
/
begin
    Ctx_Ddl.Create_Preference('wildcard_pref', 'BASIC_WORDLIST');
    ctx_ddl.set_attribute('wildcard_pref', 'wildcard_maxterms', 100) ;
end;
/
create index wildcard_idx on quick(text)  indextype is ctxsys.context
    parameters ('Wordlist wildcard_pref') ;
select quick_id from quick  where contains ( text, 'tire%' ) > 0;
drop index wildcard_idx ;
begin
    Ctx_Ddl.Drop_Preference('wildcard_pref');
    Ctx_Ddl.Create_Preference('wildcard_pref', 'BASIC_WORDLIST');
    ctx_ddl.set_attribute('wildcard_pref', 'wildcard_maxterms', 5) ;--限制最大的匹配数，如果超过这个数量，查询出现报错
end;
/
create index wildcard_idx on quick(text) indextype is ctxsys.context
parameters ('Wordlist wildcard_pref') ;
select quick_id from quick  where contains ( text, 'tire%' ) > 0;
```

3.

参考脚本

--建立 wordlist

```
begin
ctx_ddl.create_preference('mywordlist', 'BASIC_WORDLIST');
ctx_ddl.set_attribute('mywordlist','PREFIX_INDEX','TRUE'); --定义 wordlist 的参数
end;
--删除 wordlist
begin
ctx_ddl.drop_preference('mywordlist');
end;
```

(9).

Stoplist 属性

Stoplist 允许屏蔽某些常用的词，比如 is，a，this，对这些词进行索引用处不大，系统默认会使用和数据库语言相对应的停用词库（原文：Stoplists identify the words in your language that are not to be indexed. In English, you can also identify stopthemes that are not to be indexed. By default, the system indexes text using the system-supplied stoplist that corresponds to your database language.），Oracle text 提供最常用的停用词库语言包括 English, French, German, Spanish, Chinese, Dutch, and Danish
分别有 basic_stoplist，empty_stoplist，default_stoplist，multi_stoplist 几种类型

1.
Basic_stoplist
建立用户自定义的停用词库，文档中关于 stoplist 的介绍相当少，只有寥寥的数行
例子：

```
Create table my_stop (id number, docs varchar2(1000));
Insert into my_stop values (1, 'Stoplists identify the words in your language that are not
to be indexed.');
Insert into my_stop values (2, 'ou can also identify stopthemes that are not to be indexed');
Commit;
/

--建立 basic stoplist
Begin
Ctx_ddl.create_stoplist('test_stoplist', 'basic_stoplist');
End;

Create index ind_m_stop on my_stop(docs) indextype is ctxsys.context
parameters ('stoplist test_stoplist');

Select * from my_stop where contains(docs, 'words') > 0;
```

```
Begin
Ctx_ddl.add_stopword('test_stoplist', 'language'); --添加停用词
ctx_ddl.sync_index('ind_m_stop', '2m');    --同步索引
End;

Select * from my_stop where contains(docs, 'language') > 0; --添加停用词，同步索引后发现还是能查到，需要重新建立索引才能生效
Drop index ind_m_stop;
Create index ind_m_stop on my_stop(docs) indextype is ctxsys.context
parameters ('stoplist test_stoplist');
Select * from my_stop where contains(docs, 'language') > 0; --停用词生效
```

添加停用词，同步索引后发现还是能查到，需要**重新建立**索引才能生效。

2.

Empty_stoplist

停用词库没有任何停用词，适用于不需要过滤的查询中，如不需要过滤 is this,a 等。

2.

Default_stoplist

建立 basic_stoplist 后，里面不包含任何的停用词，而 default_stoplist 在 basic_stoplist 的基础上增加了预定义的默认停用词，对于不同的语言，默认的停用词库数据也不一样

例子：

```
Create table my_stop (id number, docs varchar2(1000));
Insert into my_stop values (1, 'Stoplists identify the words in your language that are not to be indexed.');
Insert into my_stop values (2, 'ou can also identify stopthemes that are not to be indexed');
Commit;
/
--建立 lexer，不同 lexer 属性会默认不同的停用词库
Begin
 ctx_ddl.create_preference('test_b_lexer', 'basic_lexer');
End;

drop index ind_m_word;
--建立默认停用词 default_stoplist
Create index ind_m_word on my_stop(docs) indextype is ctxsys.context
Parameters ('lexer test_b_lexer stoplist ctxsys.default_stoplist');
--检查默认词库中是否存在
Select * from my_stop where contains(docs, 'the') > 0;
Select * from my_stop where contains(docs, 'stopthemes') > 0;
--往默认词库中添加停用词
conn ctxsys/ctxsys;
Begin
 ctx_ddl.add_stopword('default_stoplist', 'stopthemes'); --增加停用词
```

```
 ctx_ddl.add_stopword('default_stoplist', 'words');
 ctx_ddl.remove_stopword('default_stoplist', 'words');--删除停用词
End;
--添加后需重新建立索引才能生效
conn oratext/oratext;
drop index ind_m_word;
Create index ind_m_word on my_stop(docs) indextype is ctxsys.context
Parameters ('lexer test_b_lexer stoplist ctxsys.default_stoplist');
Select * from my_stop where contains(docs, 'words') > 0;
Select * from my_stop where contains(docs, 'stopthemes') > 0;

--相关数据字典
Select * from ctx_preferences where pre_name = 'DEFAULT_LEXER';
Select * from ctx_stopwords where spw_stoplist = 'DEFAULT_STOPLIST';
```

4.

multi_stoplist

多语言停用词，适用在文档中包含不同的语言（A multi-language stoplist is useful when you use the MULTI_LEXER to index a table that contains documents in different languages, such as English, German, and Japanese）

增加停用词时，可以为停用词指定某种语言，只对指定的语言生效，默认情况下停用词对任何语言都是生效的。

--建立 multi_stoplist

```
begin
  ctx_ddl.create_stoplist('multistop1', 'MULTI_STOPLIST');
  ctx_ddl.add_stopword('multistop1', 'Die', 'german');
  ctx_ddl.add_stopword('multistop1', 'Or', 'english');
end;
```

添加停用词，同步索引后发现还是能查到，需要**重新建立**索引才能生效。

5.

参考脚本

--建立 stoplist：

```
Begin
Ctx_ddl.create_stoplist('test_stoplist', 'basic_stoplist');
End;
```

--删除 stoplist：

```
begin
ctx_ddl.drop_stoplist(' test_stoplist ');
end;
```

--增加停用词

```
ctx_ddl.add_stopword('default_stoplist', 'stopthemes'); --增加停用词
```

--删除停用词

```
ctx_ddl.remove_stopword('default_stoplist', 'words');--删除停用词
```

(10).

Theme 主题查询

主题查询的概念是根据文档的含义，而不仅仅是根据某个词的匹配程度来返回查询结果的。比如查询 about('US politics')可能会返回‘US presidential elections’ 和 ‘US foreign policy’之类的结果（原文：An ABOUT query is a query on a document theme. A document theme is a concept that is sufficiently developed in the text. For example, an ABOUT query on US politics might return documents containing information about US presidential elections and US foreign policy. Documents need not contain the exact phrase US politics to be returned.）

10g 只支持两种主题查询语言：English，French

例子：

```
--在 context 中启用主题查询
BEGIN
 CTX_DDL.CREATE_PREFERENCE('TEST_ABOUT', 'BASIC_LEXER');
 CTX_DDL.SET_ATTRIBUTE('TEST_ABOUT', 'INDEX_THEMES', 'YES');
 CTX_DDL.SET_ATTRIBUTE('TEST_ABOUT', 'INDEX_TEXT', 'YES');
END;

CREATE INDEX IND_m_about ON my_about(DOCS) INDEXTYPE IS CTXSYS.CONTEXT
 PARAMETERS ('LEXER CTXSYS.TEST_ABOUT');
--查询
SELECT *  FROM my_about WHERE CONTAINS(DOCS, 'ABOUT(US politics)') > 0;
```

(11).

Highlighting 高亮显示

并不是说将内容高亮显示，而是返回所有命中词在文档中的位置和命中词本身的长度。这样用户在得到文档的同时，还得到了需要高亮显示的内容的长度和偏移量，真正的显示工作需要由用户来完成（原文：In Oracle Text query applications, you can present selected documents with query terms highlighted for text queries or with themes highlighted for ABOUT queries）

例子：

```
Create table my_high (id number primary key, docs varchar2(1000));
insert into my_high values (1, 'this is a oracle text example. And oracle is the key word.');
```

```
insert into my_high values (2, '<title>oracle text</title>
2 <body>this is a oracle ctx_doc hightlight example.</body>');
commit;
/
--建立索引
create index ind_m_high on my_high(docs) indextype is ctxsys.context;
--返回结果的偏移量
set serverout on
declare
v_restab ctx_doc.highlight_tab;
begin
ctx_doc.highlight('ind_m_high', 1, 'oracle', v_restab, true);
for i in 1..v_restab.count loop
dbms_output.put_line('begin with: ' || v_restab(i).offset || ' length: ' ||
v_restab(i).length);
end loop;
end;

begin with: 11 length: 6
begin with: 36 length: 6
```

参考：

http://yangtingkun.itpub.net/post/468/212718

http://download.oracle.com/docs/cd/B19306_01/text.102/b14217/view.htm

(12).

常用的脚本：

1.删除 preference：

```
begin
ctx_ddl.drop_preference('my_lexer');
end;
```

2.索引重建：

```
ALTER INDEX newsindex REBUILD PARAMETERS('replace lexer my_lexer');
```

3.同步索引

```
begin
ctx_ddl.sync_index('myindex', '2M');
end;
```

或通过后台设置同步脚本：

$ORACLE_HOME/ctx/sample/script/drjobdml.sql --后台同步脚本（9i 中有，10g 不知道放哪儿了，文档有问题）

SQL> @drjobdml myindex 360 --表示以周期 360 分钟同步索引 myindex

或通过创建索引加参数实现

```
--表示每小时同步一次，内存 16m
CREATE INDEX IND_m_high ON my_high(DOCS) INDEXTYPE IS CTXSYS.CONTEXT
parameters ('sync (EVERY "TRUNC(SYSDATE)+ 1/24")  memory 16m ') parallel 2 online;
```

(确认这个时间间隔内索引同步能完成，否则，sync job 将处于挂起状态)

```
--还可以是
sync (manual)  --手工同步，默认
sync (on commit)  --dml 后立即同步
```

--通过 job 定义同步

```
declare
  job number;
begin
  dbms_job.submit(job,
                  'ctx_ddl.sync_index(''ind_m_high'');', --索引名
                  interval => 'SYSDATE+1/1440'); --1 分钟一次
  commit;
  dbms_output.put_line('job ' || job || ' has been submitted.');
end;
```

4.索引碎片：

刚开始建立索引后，DOG 可能是如下的条目

DOG DOC1 DOC3 DOC5

新的文档增加到表后，新行会同步到索引中去， 假设新文档中 Doc7 中的 DOG 被同步到索引中去，DOG 可能变成如下条目

DOG DOC1 DOC3 DOC5

DOG DOC7

随后的 DML 操作可能变成：

DOG DOC1 DOC3 DOC5

DOG DOC7

DOG DOC9

DOG DOC11

这就产生了碎片，需要进行索引的优化

查看索引碎片

```
create table output (result CLOB);
declare
  x clob := null;
begin
  ctx_report.index_stats('idx_auction', x);
  insert into output values (x);
  commit;
  dbms_lob.freetemporary(x);
```

```
end;

select * from output
```

5. 索引优化：

快速 fast 优化和全部 full 优化的区别是，快速优化不会删除没用的、过期的数据，而 full 会删除老的数据(deleted rows)

```
--快速优化
begin
  ctx_ddl.optimize_index('myidx','FAST');
end;
--全部优化
begin
  ctx_ddl.optimize_index('myidx','FULL');
end;
--对单个记号进行优化，默认是 full 模式
begin
  ctx_ddl.optimize_index('myidx','token', TOKEN=>'Oracle');
end;
```

6.Online 参数限制：

at the very beginning or very end of this process, dml might fail.

online is supported for context indexes only.

online cannot be used with parallel.

```
--online 索引的时候后面必须要加 parameters，否则会失败
alter index IND_m_high rebuild online parameters ('sync memory 16m' )
```

7.更改索引的属性，但不进行索引重建

Replaces the existing preference class settings, including SYNC parameters, of the index with the settings from new_preference. Only index preferences and attributes are replaced. The index is not rebuilt.

```
ALTER INDEX myidx REBUILD PARAMETERS('replace metadata transactional');
alter index idx_auction_db1 rebuild  PARAMETERS('REPLACE METADATA  SYNC(ON COMMIT)') ;
```

参考文档：

http://download.oracle.com/docs/cd/B19306_01/text.102/b14218/csql.htm#CIHBFDCE

8.Score:

```
--表示得分，分值越高，表示查到的数据越精确
SELECT SCORE(1), id, text FROM docs WHERE CONTAINS(text, 'oracle', 1) > 0;
--根据得分来排序
SELECT SCORE(1), title  from news  WHERE CONTAINS(text, 'oracle', 1) > 0  AND issue_date >=
('01-OCT-97')
ORDER BY SCORE(1) DESC;
```

9.分析表：

```
ANALYZE TABLE <table_name> COMPUTE STATISTICS;
ANALYZE TABLE <table_name> ESTIMATE STATISTICS 1000 ROWS;
ANALYZE TABLE <table_name> ESTIMATE STATISTICS 50 PERCENT;
ANALYZE TABLE <table_name> DELETE STATISTICS;
```

10.数据字典表：

查看系统默认的 oracle text 参数

```
Select pre_name, pre_object from ctx_preferences
```

查询 dml 操作后索引为同步

```
SELECT pnd_index_name, pnd_rowid, to_char(pnd_timestamp, 'dd-mon-yyyy hh24:mi:ss')
timestamp FROM ctx_user_pending;
```

查看错误记录表

```
select * from CTX_USER_INDEX_ERRORS
```

11.Php，Jsp 应用 oracle text：

http://download.oracle.com/docs/cd/B19306_01/text.102/b14217/acase.htm

12.逻辑操作符：

Operator	符号	表达式
AND	&	'cats AND dogs' 'cats & dogs'
OR	|	'cats | dogs' 'cats OR dogs'
NOT	~	'animals ~ dogs'(查询除了 dog 外的所有动物)
ACCUM	,	'dogs, cats, puppies'(查找文档中包含这个这三个词)
EQUIV	=	'German shepherds=alsatians are big dogs'
Near	Near	'near((dog, cat), 6)' 查找 dog 和 cat 必须满足相邻 6 个词之内
()	()	(A B) | C 查找的结果满足(a and b) or c

```
-- or 操作符
Select id, text from docs where contains(text, 'city or state ') > 0;
--and 操作符
Select id, text from docs where contains(text, 'city and state ') > 0;
```

或是

```
Select id, text from docs where contains(text, 'city  state ') > 0;
```

13.索引优化问题

--先看个例子
SQL> exec ctx_ddl.optimize_index('idx_auction_db1','FULL');

PL/SQL procedure successfully completed.

Elapsed: 00:16:16.77
索引优化相当的慢，200 万的数据建立 context 索引需要不到 5 分钟，而优化索引居然要 16 分钟，这么慢的优化速度对一个具有几亿表的数据是很难接受的的。刚开始我以为这是 oracle 的一个 bug，后来查到了一些文档。Oracle10g 引进了 rebuild 优化参数，速度还是很快的。
SQL> exec ctx_ddl.optimize_index('idx_auction_db1','rebuild') ;

PL/SQL procedure successfully completed.

Elapsed: 00:03:18.92

参考文档：
http://www.oracle.com/technology/products/text/x/10g_tech_overview.html#idxmaint_opt_rebuild

The REBUILD mode optimizes an index by building a fresh shadow copy of the index, then swapping it with the existing index.

FULL optimize updates the $I table in place. This can be slow, since we must use SQL inserts and deletes, which are slower than direct path writing. On top of that, the $X index slows down any DML on $I. The new REBUILD mode of optimize is designed to get around these problems. You invoke it like a fast optimize, but specify REBUILD for the mode:

exec ctx_ddl.optimize_index('myidx','REBUILD');

There is currently no symbol CTX_DDL.OPTLEVEL_REBUILD, so you'll just have to write out the literal string REBUILD. Rebuild optimize optimizes the entire index in one shot, so there is no need to specify maxtime, token, and token_type parameters -- you'll just get a syntax error. Rebuild optimize, like FULL, can be run in parallel, so you may specify a parallel_degree parameter if you wish.

So what does it do? Broadly, it creates an empty $I, copies the contents of current $I to the new $I, optimizing each row as it copies, then swaps the new $I and the old $I. This can be faster than FULL because FULL has to insert rows using SQL, then delete old rows, all the time updating $X. REBUILD can insert using direct path load, does not have to delete old rows, and does not have to update a b-tree index.

Here's what REBUILD optimize does, specifically:

1. Create two shadow $I tables. Shadow 1 is named DR$<idx>MI, while shadow 2 is named DR$<idx>NI. Both will use the storage clause for $I, so if the initial extent on that storage clause is really high, you could run out of space here. The MI table will be

partitioned, but have only one partition. It needs to be partitioned so that later we can do a swap with the existing $I. We use special internal calls so that this works even on standard edition, which does not have partitioning. Obviously this precludes your use of storage preferences which would partition the $I table.

2. Create a log table DR$<idx>$L -- this table is used to monitor changes to the existing $I table during the optimization. It gets no storage clause, so uses the index owner's default tablespace's default storage.
3. Create a copy of the current $N table, named DR$<idx>MN. This serves as a snapshot of deleted docids.
4. Create a monitor trigger, DR$<idx>$T, which logs the rowid of all inserts to the existing $I in the $L table
5. Copy the existing $I table to the shadow I tables. This is done in one complex SQL statement -- roughly,

   insert into <shadow I>
   select * from TABLE(optimize_index(CURSOR(select * from $I))

   Let's break this down. The innermost part is the CURSOR. This selects all the rows from $I and passes them through the optimize_index function. The optimize function reads the snapshot $N table, then takes the $I rows coming from the cursor and optimizes them -- removing garbage rows, defragging small rows, etc. It uses essentially the same code as the FULL optimize. The table function returns optimized $I rows which are then shunted by the insert into statement into the shadow tables. (Mostly to shadow 1. Shadow 2 gets those rows with really long token_info, for technical reasons). The insert into uses the efficient direct path insertion, so the I/O costs of populating the shadow tables is minimal. (You do not need to have NOLOGGING in your I_TABLE_CLAUSE in your storage preference, but it might help performance if you do, or if it is part of the tablespace default). At the end of this statement, the shadow tables hold an optimized version of the existing $I table data.

6. Copy all rows from shadow 2 to shadow 1
7. Create a shadow $X index on shadow 1
8. Copy any new rows in $I to shadow 1. While we were doing the optimize, the $I might have changed. Perhaps there was a concurrent sync, or ctx_ddl.add_mdata. Our shadow 1 does not have these new rows. Luckily, their rowids are logged in $L (remember the monitor trigger created in step 4?). So it's a simple matter to copy the new rows from $I to shadow 1 -- insert into shadow1 as select from $I, $L where $L.logged_rowid = $I.rowid.
9. Swap shadow 1 and $I, using alter table exchange partition
10. We've optimized all the docids in the snapshot $N table, but there could have been deletes or updates while we were doing the optimize, so we can't just truncate the $N table. We need to delete all the docids in our snapshot table from the $N. Logically, that's what we do here. Only practically, it is easier to create a shadow $N table (DR$<idx>NN) using create table as select current $N table MINUS snapshot $N table.
11. Swap the second shadow N table and the $N table
12. Drop all shadow objects, which have old data in them

The implication of steps 8 and 10 is that after a rebuild optimize, your index may not be totally optimal -- DML that happens during the optimize cannot be optimized. But this kind of thing is unavoidable, and no interesting index stays pristine for very long, anyway.

So, when do you use FULL vs REBUILD vs TOKEN? Use FULL and its time-limited operation for continuing maintenance, like a cron job that kicks off every night for 4 hours. This can be supplemented by a cron that runs some TOKEN or TOKEN_TYPE optimizes for continuing targeted maintenance, keeping especially dynamic field sections, mdata sections, or text tokens optimal. Use REBUILD for periodic maintenance -- like running it once a month or something, if FULL if unable to keep up with daily activity.

13.事务查询

索引可能不是实时进行同步的，但是查询又要求实时的。

--更改索引为事务性，查询再找索引时还会 dr$unindexed 表，把满足条件还未索引的记录找出来

```
alter index idx_auction_db1 rebuild parameters('replace metadata transactional')
```

例子：

```
select count(*) from table where contains(text, 'someword') > 0; -- 0 hits
insert into table values ('someword');
select count(*) from table where contains(text, 'someword') > 0; -- 1 hit (the one we just entered)
rollback;
select count(*) from table where contains(text, 'someword') > 0; -- 0 hit
```

仅仅是对某个 session 启用

```
exec ctx_query.disable_transactional_query := TRUE;
```

参考文档：

http://tahiti.oracle.com
http://yangtingkun.itpub.net
http://epub.itpub.net/4/1.htm
http://www.oracle.com/global/cn/oramag/oracle/04-sep/o54text.html
http://www.oracle.com/technology/products/text/x/10g_tech_overview.html
http://forums.oracle.com/forums/thread.jspa?messageID=1958708

后记：

本文档是我花了一个星期时间整理出来的，里面可能有很多不对的地方，请大家多多指正，我们共同来完善 oracle text，共同来推广 oracle text 在国内的普及。文档部分内容来自 document 的翻译，部分内容来自网友的文章，还有一部分来出自我的理解，本人名字徐进挺，puber 名 westlife_xu。

2007-8-23