Unlocking Hidden Gems in Oracle Text

Presenter: Bill Coulam (www.dbartisans.com)

THE CHURCH OF
JESUS CHRIST
OF LATTER-DAY SAINTS

Agenda

- What is Oracle Text
- Installation
- CONTEXT Indexes & Features
- CONTAINS Queries
- Multi-column index
- Multi-table, multi-column index

Oracle Text

- Built into Oracle DB (PE, SE, SE One, EE)
- Free to use with existing DB license
- ConText Cartridge (8) interMedia Text (8i)
- Just "Oracle Text" since 9i
- Technology built into Oracle that extends indexing capabilities to text, XML, CLOB, documents stored as BLOB, BFILE and web pages.
- Build document classification and cataloging applications.
- Special XML and HTML features as well.

Oracle Text

- Has searching text columns ever been hindered by the limitations of =, LIKE, SUBSTR, INSTR?
  - User misspellings
  - Case problems
  - International characters
  - Search many columns at once for term(s)
  - Querying large LOB columns inefficient.
- Oracle CONTEXT indexes solves these

Misspelled Search Terms/Data

- Traditional:
  - Maybe SOUNDEX
  - Generally "No Data for You!"
- Oracle Text:
  - Partial term with basic CONTAINS, substring index or wildcarding
  - EQUIV, Fuzzy, Stemming, Soundex, Thesaurus

Case Problems

- Traditional:
  - Store original source as all one case.
    - All searches converted to single case.
    - Data looks nasty when displayed, printed
  - Store single case copy columns in table or MV
    - Extra space required, trigger, moving parts, violates redundancy.
- Oracle Text:
  - All tokens indexed in UPPER case by default.
  - Searches are case-insensitive by default

International Characters/Diacritics

- Traditional:
  - Equivalence (=) by

```
NLSSORT(UPPER(column),'NLS_SORT=BINARY')
```

  - LIKE by removing diacritics from both sides of the equation using frontend libraries and/or Oracle TRANSLATE:

```
TRANSLATE(column,'ÀÁÂÃÄÅÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖØÙÚÛÜÝàáâãäåçşèéêëìíîïðñóôõöøùúûüýÿ',
                 'AAAAAACEEEEIIIIDNOOOOOOUUUUYaaaaaacseeeeiiiionooooouuuuyy')
```

- Oracle Text:
  - base_letter attribute of context index
  - Host of features for language nuances

Proximity, Relevance, Theme, Exclusion Searches

- Traditional:
  - Only exclusion by use of AND NOT LIKE
- Oracle Text:
  - ACCUM, SCORE, ABOUT, boolean operators, MINUS, NOT, NEAR, and THRESHOLD.

Search multiple columns or tables

- Traditional:
  - Joins with OR LIKE
  - View using UNION [ALL] or concatenation
  - Materialized view
  - Concatenated copy column
- Oracle Text:
  - MULTI_COLUMN_DATASTORE
  - USER_DATASTORE

Check Installation

- Installed by default if DB created with DBCA.
- CTXSYS schema
  - 9iR2: 260+ objects
  - 10gR2: 340+ objects
  - Security issues fixed in 10g
  - Account usually in locked state
- Check version

```
SELECT * FROM ctxsys.ctx_version;
```

Installation

- Drop CTXSYS if it exists.
- As SYS, run

```
$ORACLE_HOME/ctx/admin/catctx.sql SYSAUX TEMP NOLOCK
```

- As CTXSYS, run

```
$ORACLE_HOME/ctx/admin/defaults/drdefus.sql
```

- Now check your installation
  - Approximately 260 objects in 9iR2
  - Approximately 340 objects in 10gR2

```
SELECT status, COUNT(*) FROM dba_objects WHERE owner = 'CTXSYS' GROUP
  BY status;
```

First Context Index

- Prerequisite:
  - A source of text to index:
    - Character column (VARCHAR2, CLOB, etc)
    - Binary column that contains text (BLOB, BFILE)
    - XMLType
    - URIType
    - Column that is path to file
    - Column that is path to web page

First Context Index

- Set up/verify the account that contains the data to index:

```
CREATE USER edw IDENTIFIED BY "edw"
  DEFAULT TABLESPACE users
  TEMPORARY TABLESPACE temp
  PROFILE default
  QUOTA UNLIMITED ON users;
GRANT EXECUTE ON ctxsys.ctx_ddl TO edw;
GRANT CTXAPP TO edw; -- only necessary to run CTXSYS pkgs
GRANT CREATE PROCEDURE TO edw;
GRANT CREATE SESSION TO edw;
GRANT CREATE TABLE TO edw;
GRANT SELECT ANY DICTIONARY TO edw;
GRANT CREATE JOB to edw; -- only if using SYNC
GRANT CREATE TRIGGER to edw;
CONNECT edw/edw
CREATE SYNONYM ctx_ddl FOR ctxsys.ddl;
```

First Context Index

- Set up/verify the data to be indexed:

```
CREATE TABLE places
(
  place_id           NUMBER(10) NOT NULL,
  place_nm           VARCHAR2(60 CHAR) NOT NULL,
  place_type_nm      VARCHAR2(30 CHAR),
  parent_place_id    NUMBER(10) REFERENCES places (place_id),
  place_notes        CLOB,
  CONSTRAINT places_pk PRIMARY KEY (place_id),
  CONSTRAINT places_uk UNIQUE (place_nm, place_type_nm,
  parent_place_id)
);
```

First Context Index

```
CREATE OR REPLACE FUNCTION get_wiki(i_wiki_page IN VARCHAR2) RETURN CLOB
AS
   l_http_request  utl_http.req;
   l_http_response utl_http.resp;
   l_clob          CLOB;
   l_text          VARCHAR2(32767);
BEGIN
   dbms_lob.createtemporary(l_clob, FALSE);
   l_http_request  := utl_http.begin_request('http://en.wikipedia.org/wiki/'||i_wiki_page);
   l_http_response := utl_http.get_response(l_http_request);
   BEGIN
      LOOP
         utl_http.read_text(l_http_response, l_text, 32767);
         dbms_lob.writeappend(l_clob, LENGTH(l_text), l_text);
      END LOOP;
   EXCEPTION
      WHEN utl_http.end_of_body THEN
         utl_http.end_response(l_http_response);
   END;
   RETURN l_clob;
EXCEPTION
   WHEN OTHERS THEN
      utl_http.end_response(l_http_response);
      dbms_lob.freetemporary(l_clob);
      RAISE;
END get_wiki;
```

First Context Index

```
INSERT INTO places VALUES (251,'United States','Country',NULL,
   get_wiki('UnitedStates'));
INSERT INTO places VALUES (35,'Ohio','State',251,get_wiki('Ohio'));
INSERT INTO places VALUES (1011,'Toledo','City',35,get_wiki('Toledo,_Ohio'));
INSERT INTO places VALUES (51,'District of Columbia','Political District',251,
   get_wiki('District_of_Columbia'));
INSERT INTO places VALUES (270,'Northern Mariana Islands','Territory',251,
   get_wiki('Northern_Mariana_Islands'));
INSERT INTO places VALUES (231,'Spain','Country',NULL,get_wiki('Spain'));
INSERT INTO places VALUES (3260,'Málaga','Province',231,get_wiki('Malaga'));
INSERT INTO places VALUES (3273,'Toledo','Province',231,get_wiki('Toledo,_Spain'));
INSERT INTO places VALUES (146,'Iceland','Country',NULL,get_wiki('Iceland'));
INSERT INTO places VALUES (3839,'Höfuðborgarsvæði','Political District',146,
   get_wiki('Greater_Reykjavík_Area'));
INSERT INTO places VALUES (236,'Sweden','Country',NULL,get_wiki('Sweden'));
INSERT INTO places VALUES (3393,'Skåne','Province',236,get_wiki('Scania'));
INSERT INTO places VALUES (997,'Spam','Country',NULL,get_wiki('Spam_(electronic)'));
INSERT INTO places VALUES (998,'Spam-Spam_and_Bacon','Province',997,
   get_wiki('Spam_(food)'));
INSERT INTO places VALUES (999,'spam egg spam spam bacon and spam','Township',998,
   get_wiki('Spam_(Monty_Python)'));
COMMIT;
```

First Context Index

- Note, for the previous inserts to work on 11g, you need to grant access to the 'net for the UTL packages using something like this:

```
BEGIN
   DBMS_NETWORK_ACL_ADMIN.CREATE_ACL('users.xml',
          'ACL that lets users use the UTL packages.',
          'EDW', TRUE, 'connect');
END;
/

BEGIN

  DBMS_NETWORK_ACL_ADMIN.ADD_PRIVILEGE('/sys/acls/users.xml','EDW
  ',TRUE,'resolve');
END;
/
```

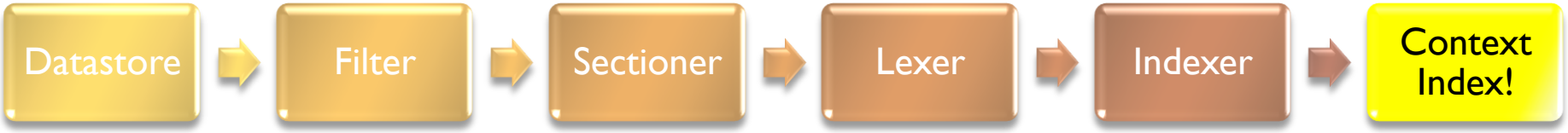
First Context Index(es)

```
CREATE INDEX place_nm_cidx ON places(place_nm)
  INDEXTYPE IS CTXSYS.CONTEXT;


CREATE INDEX place_notes_cidx ON places(place_notes)
  INDEXTYPE IS CTXSYS.CONTEXT;
```

What Happened?

- What is going on when using the defaults?
  1. Detects the column type and filters binary column types.
  2. Decides text language is same as DB lang
  3. Uses the default stoplist
  4. Enables fuzzy and stemming queries
  5. Feeds data from datastore to filter, sectioner, lexer, then indexer.



What Happened?

- Added some metadata to tables in CTXSYS:

  DR$INDEX, DR$INDEX_VALUE and DR$INDEX_OBJECT

- Created some DR$*indexname*$* tables in index-owning account:

  DR$PLACE_NM_CIDX$I (tokens)

  DR$PLACE_NM_CIDX$K (keymap)

  DR$PLACE_NM_CIDX$N (negative list)

  DR$PLACE_NM_CIDX$R (rowid)

  DR$*indexname*$P (substrings)

CONTAINS Queries

- Having a Context index opens up the world of CONTAINS queries for you.
- CONTAINS(*indexed_column, query expr [, label]*)
  - Returns a numeric relevance between 0 and 100. If 0, the row will not be included in the result set.
- SCORE(*label*)
  - Optional. Used in the SELECT list, returns the relevance as a virtual column

```
SELECT col1, col2, SCORE(i) FROM table
  WHERE CONTAINS(col1,expr,i);
```

- CONTAINS sports all sorts of options and flavors.

CONTAINS Queries

```
-- Simple CONTAINS finds records where the term is included at least once.
SELECT place_nm FROM places t WHERE CONTAINS(place_nm,'Egg') > 0;
-- SCORE with optional, matching CONTAINS label, returns the relevance as a column
SELECT place_nm, SCORE(1) FROM places t WHERE CONTAINS(place_nm,'Egg',1) > 0;
-- "Mixed" or Structured CONTAINS (regular SQL predicate with CONTAINS operator)
SELECT place_nm, place_type_nm FROM places t WHERE CONTAINS(place_nm,'Toledo')>0 AND place_type_nm='City';
-- ABOUT looks for themes in an indexed document
SELECT place_nm, SCORE(1) FROM places t WHERE CONTAINS(place_notes,'about(monarchy)',1) > 0;
-- ACCUM scores doc better by number of times ACCUM term(s) appear(s), a term can be weighted
SELECT place_nm, SCORE(1) FROM places t WHERE CONTAINS(place_nm,'spam',1) > 0;
SELECT place_nm, SCORE(1) FROM places t WHERE CONTAINS(place_nm,'bacon*3 ACCUM spam',1) > 0;
  -- alternative ACCUM syntax is term1,term2
SELECT place_nm, SCORE(1) FROM places t WHERE CONTAINS(place_nm,'bacon*3,spam',1) > 0;
-- CONTAINS queries can use Boolean operators, parenthesis and order of operations
SELECT place_nm, SCORE(1) FROM places t WHERE CONTAINS(place_nm,'Egg & Spam | Bacon',1) > 0;
SELECT place_nm, SCORE(1) FROM places t WHERE CONTAINS(place_nm,'(Egg AND Spam) OR Bacon',1) > 0;
-- EQUIV helps with alternate spellings or names the user might be unware of
  -- Scania is a transliteration of Sweden's Skåne
SELECT place_nm, SCORE(1) FROM places t WHERE CONTAINS(place_nm,'Skåne=Scania',1) > 0;
-- FUZZY(?) helps with misspellings and alternate spellings, finding similar word forms
  -- Syntax: fuzzy(term, [score], [numresults], [weight|noweight]) Old Syntax: ?term
SELECT place_nm, SCORE(1) FROM places t WHERE CONTAINS(place_notes,'?united',1) > 0;
SELECT place_nm, SCORE(1) FROM places t WHERE CONTAINS(place_notes,'fuzzy(united,40,5,W)',1) > 0;
SELECT place_nm, SCORE(1) FROM places t WHERE CONTAINS(place_notes,'fuzzy(united)',1) > 0;
```

CONTAINS Queries

```
-- MINUS(-) ranks documents lower that contain the unwanted term

SELECT place_nm, SCORE(1) FROM places t WHERE CONTAINS(place_nm,'spam - egg',1) > 0;

SELECT place_nm, SCORE(1) FROM places t WHERE CONTAINS(place_nm,'spam MINUS egg',1) > 0;

-- NOT(~), similar to MINUS searches for one term, but excludes the other

SELECT place_nm, SCORE(1) FROM places t WHERE CONTAINS(place_nm,'spam ~ egg',1) > 0;

SELECT place_nm, SCORE(1) FROM places t WHERE CONTAINS(place_nm,'spam NOT (egg OR bacon)',1) > 0;

-- NEAR(;) enables word proximity searching

  -- Syntax: NEAR((term1,term2,...,termN)[,max_span[,order]]) Old Syntax: word1 ; word2

SELECT place_nm, SCORE(1) FROM places t WHERE CONTAINS(place_nm,'islands;northern',1) > 0;

SELECT place_nm, SCORE(1) FROM places t WHERE CONTAINS(place_nm,'NEAR((islands,northern),4)',1) > 0;

  -- to ensure matches with words in the order specified, third parameter must be TRUE

SELECT place_nm, SCORE(1) FROM places t WHERE CONTAINS(place_nm,'NEAR((northern,islands),4,TRUE)',1) > 0;

SELECT place_nm, SCORE(1) FROM places t WHERE CONTAINS(place_notes,'NEAR((northern,islands),4,TRUE) AND
    commonwealth',1) > 0 ORDER BY SCORE(1) DESC;

-- SOUNDEX(!) operator expands to words that have similar sounds (works best in English)

  -- search term mispelled by user, but found Spain anyway

SELECT place_nm, SCORE(1) FROM places t WHERE CONTAINS(place_nm,'!Spayne',1) > 0;

-- STEM($) expands to related terms with the same linguistic root

  -- BASIC_LEXER (the default) supports English, French, Spanish, Italian, German, and Dutch

SELECT place_nm, SCORE(1) FROM places t WHERE CONTAINS(place_notes,'$arab',1) > 0;

-- THRESHOLD(>) brings back matches whose expression or word score is greater than threshold

SELECT place_nm, SCORE(1) FROM places t WHERE CONTAINS(place_nm,'Toledo*8 > 30',1) > 0;

-- Multiple CONTAINS in a single query

SELECT place_nm, place_type_nm FROM places t WHERE CONTAINS(place_nm,'Toledo',1) > 0 AND
    CONTAINS(place_notes,'Moorish',2) > 0;
```

Customizing the Context Index

```
CREATE INDEX... name ON table(column(s))
  INDEXTYPE IS CTXSYS.CONTEXT PARAMETERS ('
 [DATASTORE datastore_pref]
 [FILTER filter_pref]
 [CHARSET COLUMN charset_column_name]
 [FORMAT COLUMN format_column_name]
 [LEXER lexer_pref]
 [LANGUAGE COLUMN language_column_name]
 [WORDLIST wordlist_pref]
 [STORAGE storage_pref]
 [STOPLIST stoplist]
 [SECTION GROUP section_group]
 [MEMORY memsize]
 [POPULATE | NOPOPULATE] -- 11g
 [SYNC (MANUAL | EVERY "interval" | ON COMMIT)] -- 10g
 [TRANSACTIONAL]') -- 10g
```

Keeping a Text Index Current

- Until 10g, not kept in-sync with the data automatically
- Write jobs to sync
  - CTX_DDL.sync_index
  - CTX_DDL.optimize_index
- Allow Oracle to write the job
  - SYNC
  - TRANSACTIONAL

Multi-Column Text Index

- Pre-requisites:
  - Data
  - Dummy column
  - Trigger
  - Preferences

Multi-Column Text Index (Data)

```
DROP INDEX cust_strings_cidx;

DROP TABLE customer CASCADE CONSTRAINTS;

CREATE TABLE customer(cust_id NUMBER PRIMARY KEY, first_nm
  VARCHAR2(100), mid_nm VARCHAR2(100), last_nm VARCHAR2(100),
  email_addr VARCHAR2(100));

INSERT INTO customer VALUES(1, 'John','Morgan','Smith',
  'smithjm@harpers.com');

INSERT INTO customer VALUES(2, 'Morgan',NULL,'Smythe',
  'morgan_smythe@reuters.com');

INSERT INTO customer VALUES(3, 'William','Ånjo','Morgan',
  'william-a-morgan@hotmail.com');

INSERT INTO customer VALUES(4, 'Rip','William','Van-winkle',
  'rip.vanwinkle@narcoleptics.org');

COMMIT;
```

Multi-Column Text Index (Prefs)

```
EXEC ctx_ddl.drop_preference('cust_multids');
EXEC ctx_ddl.drop_preference('cust_lexer');
EXEC ctx_ddl.drop_preference('cust_wordlist');
EXEC ctx_ddl.drop_preference('cust_storage');
BEGIN
   ctx_ddl.create_preference('cust_multids', 'MULTI_COLUMN_DATASTORE');
   ctx_ddl.set_attribute('cust_multids', 'columns', 'first_nm||CHR(32)||mid_nm||CHR(32)||
    last_nm||CHR(32)||email_addr as fullstring');
   ctx_ddl.create_preference('cust_lexer','BASIC_LEXER');
   ctx_ddl.set_attribute('cust_lexer','printjoins','-_'); -- keeps hyphenation
   ctx_ddl.set_attribute('cust_lexer','base_letter','YES'); -- removes diacritics
   ctx_ddl.create_preference('cust_wordlist','BASIC_WORDLIST');
   ctx_ddl.set_attribute('cust_wordlist','prefix_index','TRUE');
   ctx_ddl.set_attribute('cust_wordlist','prefix_min_length','3');
   ctx_ddl.set_attribute('cust_wordlist','prefix_max_length','6');
   ctx_ddl.create_preference('cust_storage', 'BASIC_STORAGE');
   ctx_ddl.set_attribute('cust_storage','i_table_clause','TABLESPACE &&cust_data');
   ctx_ddl.set_attribute('cust_storage','k_table_clause','TABLESPACE &&cust_data');
   ctx_ddl.set_attribute('cust_storage','n_table_clause','TABLESPACE &&cust_data');
   ctx_ddl.set_attribute('cust_storage','r_table_clause','TABLESPACE &&cust_data LOB (data) STORE AS (cache)');
   ctx_ddl.set_attribute('cust_storage','i_index_clause','TABLESPACE &&cust_index COMPRESS 2');
END;
/
```

Multi-Column Text Index (Dummy)

- Since Text needs to track updates and queries on a single column, we have to give it one:

```
ALTER TABLE customer
  ADD otx_upd_flg VARCHAR2(1 BYTE) DEFAULT 'N';
```

- And update it whenever the text changes:

```
CREATE OR REPLACE TRIGGER customer_biu
  BEFORE UPDATE OR INSERT ON customer
  FOR EACH ROW
DECLARE
BEGIN
   IF (INSERTING OR UPDATING) THEN
      :new.otx_upd_flg := 'Y';
   END IF;
END customer_biu;
/
```

Multi-Column Text Index

```
CREATE INDEX cust_strings_cidx ON customer(otx_upd_flg)
  INDEXTYPE IS CTXSYS.CONTEXT
  PARAMETERS ('DATASTORE cust_multids
               LEXER cust_lexer
               WORDLIST cust_wordlist
               STORAGE cust_storage');
COMMIT; -- necessary after Text index creation
```

Multi-Column Text Index

- CONTAINS queries with our multi-column index

```
-- user can't remember spelling
SELECT c.* FROM customer c WHERE CONTAINS(otx_upd_flg,'!smith') > 0;
-- user only has "Morgan" to go on to find customer
SELECT c.* FROM customer c WHERE CONTAINS(otx_upd_flg,'morgan') > 0;
-- user wants to find hyphenated names (hyphens and underscores need to be escaped)
SELECT c.* FROM customer c WHERE CONTAINS(otx_upd_flg,'van\-winkle') > 0;
SELECT c.* FROM customer c WHERE CONTAINS(otx_upd_flg,'morgan\_smythe') > 0;
-- user only has part of the customer's email address
SELECT c.* FROM customer c WHERE CONTAINS(otx_upd_flg,'Narc%') > 0;
-- user has name portions, but not sure about order
SELECT c.* FROM customer c WHERE CONTAINS(otx_upd_flg,'NEAR((Smythe,Morgan))') > 0;
-- user has name portions and is sure about their order
SELECT c.* FROM customer c WHERE CONTAINS(otx_upd_flg,'NEAR((Morgan,Smith),1,TRUE)') >
   0;
-- user doesn't know how to enter the nordic diacritical marks
SELECT c.* FROM customer c WHERE CONTAINS(otx_upd_flg,'Anjo') > 0;
```

Multi-Column, Multi-Table Index

- Pre-requisites:
  - More tables, more data
  - Preferences and optional Section Group
  - Procedure to concatenate text
    - IN ROWID
    - OUT [CLOB, BLOB, CLOB_LOC, BLOB_LOC, or VARCHAR2]
  - Dummy column
  - Triggers

Multi-Column, Multi-Table Index (Data)

```
DROP INDEX cust_strings_cidx FORCE;

DROP TABLE customer_contact;
CREATE TABLE customer_contact (
  cust_id NUMBER NOT NULL REFERENCES customer(cust_id),
  contact_dt DATE NOT NULL,
  contact_type VARCHAR2(10 BYTE) CHECK (contact_type IN ('EMAIL','INCALL','OUTCALL','LETTER')),
  note VARCHAR2(4000));
INSERT INTO customer_contact VALUES (1,SYSDATE-5,'INCALL','Called to check warranty length.');
INSERT INTO customer_contact VALUES (2,SYSDATE-4,'EMAIL','Wants callback at home, 435-234-5555');
INSERT INTO customer_contact VALUES (2,SYSDATE-3,'OUTCALL','Called. Spoke to Veronica Smythe.');
INSERT INTO customer_contact VALUES (3,SYSDATE-2,'LETTER','Sent latest privacy notice.');
INSERT INTO customer_contact VALUES (4,SYSDATE-1,'LETTER','Sent 81st late payment notice.');
INSERT INTO customer_contact VALUES (4,SYSDATE,'INCALL','Fell asleep for 20 yrs. Wants interest
   waived.');
COMMIT;
```

Multi-Column, Multi-Table Index (Procedure)

```
CREATE OR REPLACE PACKAGE cust_util AS
PROCEDURE concat_columns(i_rowid IN ROWID, io_text IN OUT
     NOCOPY VARCHAR2);
END cust_util;
/
CREATE OR REPLACE PACKAGE BODY cust_util AS
PROCEDURE concat_columns(i_rowid IN ROWID, io_text IN OUT
     NOCOPY VARCHAR2)
AS
   lr_cust customer%ROWTYPE;

   CURSOR cur_cust (i_cust_id IN customer.cust_id%TYPE) IS
   SELECT first_nm, mid_nm, last_nm, email_addr FROM
     customer
   WHERE cust_id = i_cust_id;

   CURSOR cur_notes (i_cust_id IN
     customer_contact.cust_id%TYPE) IS
   SELECT note FROM customer_contact WHERE cust_id =
     i_cust_id;

   PROCEDURE add_piece(i_add_str IN VARCHAR2) IS
      lx_too_big EXCEPTION;
      PRAGMA EXCEPTION_INIT(lx_too_big, -6502);
   BEGIN
      io_text := io_text||' '||i_add_str;
   EXCEPTION WHEN lx_too_big THEN NULL; -- silently don't
     add the string.
   END add_piece;

BEGIN
   BEGIN
      SELECT * INTO lr_cust FROM customer WHERE ROWID = i_rowid;
   EXCEPTION
      WHEN NO_DATA_FOUND THEN
         RETURN;
   END;

   add_piece('<FULLNAME>'||lr_cust.first_nm||CHR(32)||
                           lr_cust.mid_nm||CHR(32)||
                           lr_cust.last_nm||'</FULLNAME>');
   add_piece('<EMAIL_ADDR>'||lr_cust.email_addr||'</EMAIL_ADDR>');

   -- Now collect text from any calls or letters from the customer
   FOR lr_note IN cur_notes(lr_cust.cust_id) LOOP
      add_piece('<NOTE>'||lr_note.note||'</NOTE>');
   END LOOP;

END concat_columns;
END cust_util;
/
```

Multi-Column, Multi-Table Index (Prefs)

```
EXEC ctx_ddl.drop_section_group('cust_sectioner');
EXEC ctx_ddl.drop_preference('cust_user_ds');
BEGIN
   ctx_ddl.create_section_group('cust_sectioner', 'BASIC_SECTION_GROUP');
   ctx_ddl.add_field_section('cust_sectioner', 'fullname', 'fullname', TRUE);
   ctx_ddl.add_field_section('cust_sectioner', 'email_addr', 'email_addr', TRUE);
   ctx_ddl.add_field_section('cust_sectioner', 'note', 'note', TRUE);

   ctx_ddl.create_preference('cust_user_ds', 'USER_DATASTORE');
   ctx_ddl.set_attribute('cust_user_ds', 'procedure',
   sys_context('userenv','current_schema')||'.'||'cust_util.concat_columns');
   ctx_ddl.set_attribute('cust_user_ds', 'output_type', 'VARCHAR2');

END;
/
```

Multi-Column, Multi-Table Index (Trigger & Index)

```
-- Again add a trigger or two to update otx_upd_flg when
   columns in the index change
-- <snipped to save space, just copy and modify from previous
   example>

CREATE INDEX cust_strings_cidx ON
  customer(otx_upd_flg)
   INDEXTYPE IS CTXSYS.CONTEXT
   PARAMETERS ('DATASTORE cust_user_ds
                SECTION GROUP cust_sectioner
                LEXER cust_lexer
                STORAGE cust_storage
                SYNC (EVERY "SYSDATE+6/24")
                TRANSACTIONAL');
COMMIT;
```

Multi-Column, Multi-Table Index

- We can now efficiently search all the columns at once, as if they were all part of one column:

```
SELECT c.* FROM customer c WHERE CONTAINS(otx_upd_flg,'Morgan AND !Smith') > 0;
SELECT c.* FROM customer c WHERE CONTAINS(otx_upd_flg,'reuters') > 0;
SELECT c.* FROM customer c WHERE CONTAINS(otx_upd_flg,'Veronica') > 0;
SELECT c.* FROM customer c WHERE CONTAINS(otx_upd_flg,'435%5555') > 0;
-- queries can still be targeted at the individual fields in the indexed virtual
   document
SELECT c.* FROM customer c WHERE CONTAINS(otx_upd_flg,'reuters WITHIN email_addr') > 0;
SELECT c.* FROM customer c WHERE CONTAINS(otx_upd_flg,'asleep WITHIN note') > 0;
SELECT c.* FROM customer c WHERE CONTAINS(c.otx_upd_flg,'waived') > 0;
-- Shows that changes to a member of the multi-table index are immediately reflected
-- in CONTAINS queries, using the TRANSACTIONAL parameter, even though the index
-- has not been resync'd yet.
DELETE FROM customer_contact WHERE cust_id = 4;
SELECT c.* FROM customer c WHERE CONTAINS(c.otx_upd_flg,'waived') > 0;
```


Questions?

Thank You for Attending!

- Please fill out your evaluation form.
- Contact: bcoulam@yahoo.com