ORACLE

Oracle Text

A Product Overview

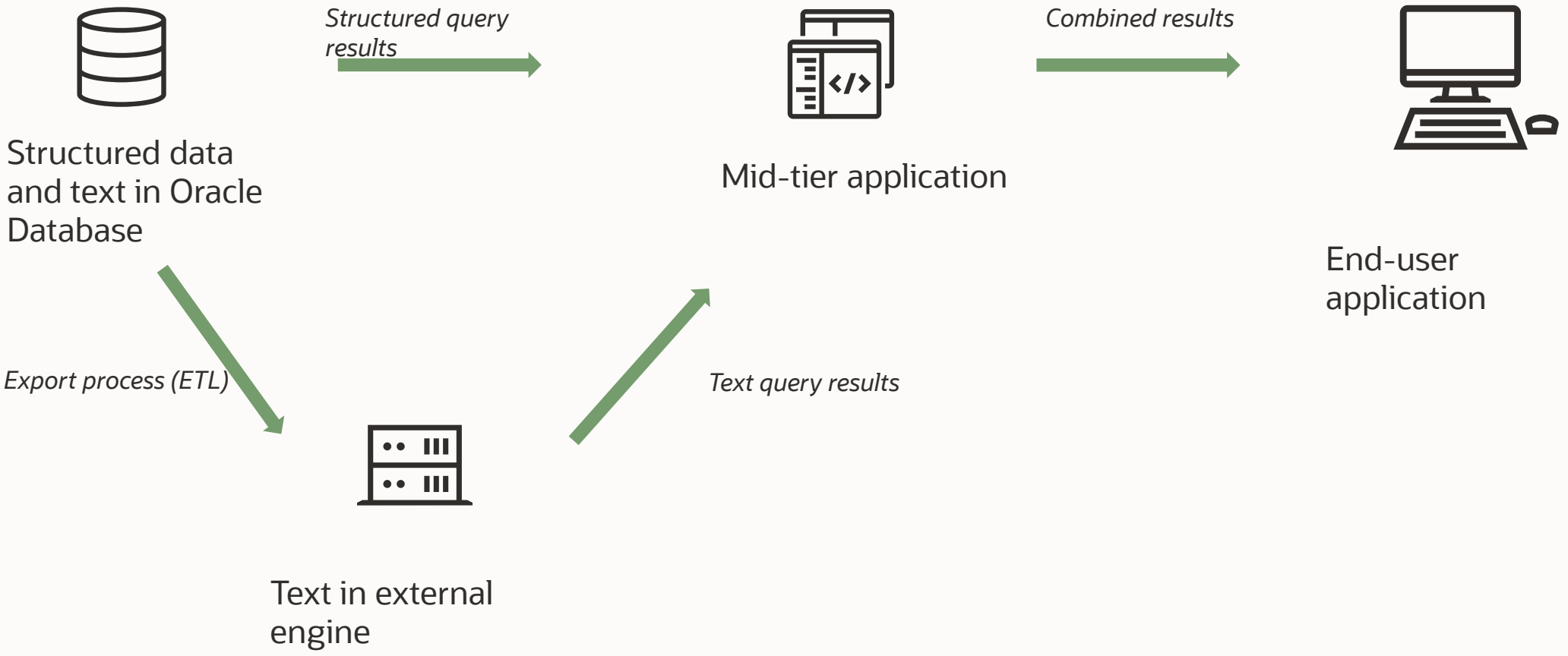
Stéphane Duprat
Converged Database specialist

Do you have text in your database?
Of course you do.

And you have Oracle Text
- so use it.

Oracle Text is a standard part of all versions and editions of Oracle Database.

Why not an external text engine?



Complexity 😠

Different vendors and APIs 😠 😠

Synchronization issues 😠 😠 😠

Oracle Text is …

A SQL-level toolkit

- Any application or language that supports SQL can work with Oracle Text

Available on-premise

- Since Oracle 8i, all versions of Oracle have had Oracle Text

Multilingual

- Works in virtually and language. Specific language support for 37 languages

Available on Cloud

- Autonomous Database, Exadata at Customer, all support Oracle Text

Handling Text Anywhere

Any type of text, in any place

Text Anywhere			
In Database		Outside	
VARCHAR(2)	LOB	File System	URL

File Formats					
Plain Text	HTML	JSON/XML	M/Soft Office	PDF	150 others

Languages and Character Sets		
English/European	Cyrillic, Arabic	Pictogram (Chinese, Japanese, Korean)

So why not use the built-in text engine?

One vendor, one support organization, one API

Full-featured text search in a SQL environment

No need to move data – one source

All indexes and metadata stored in Oracle tables

- Security
- Reliability
- High availability

Full ACID capabilities : Atomicity, Consistency, Isolation, Durability

- "… a set of properties of database transactions intended to guarantee validity even in the event of errors, power failures, etc." - Wikipedia

Oracle Text in Oracle Autonomous Database

Oracle Text is fully enabled in Autonomous Database

Provides full-text search capabilities over Text / JSON / XML content

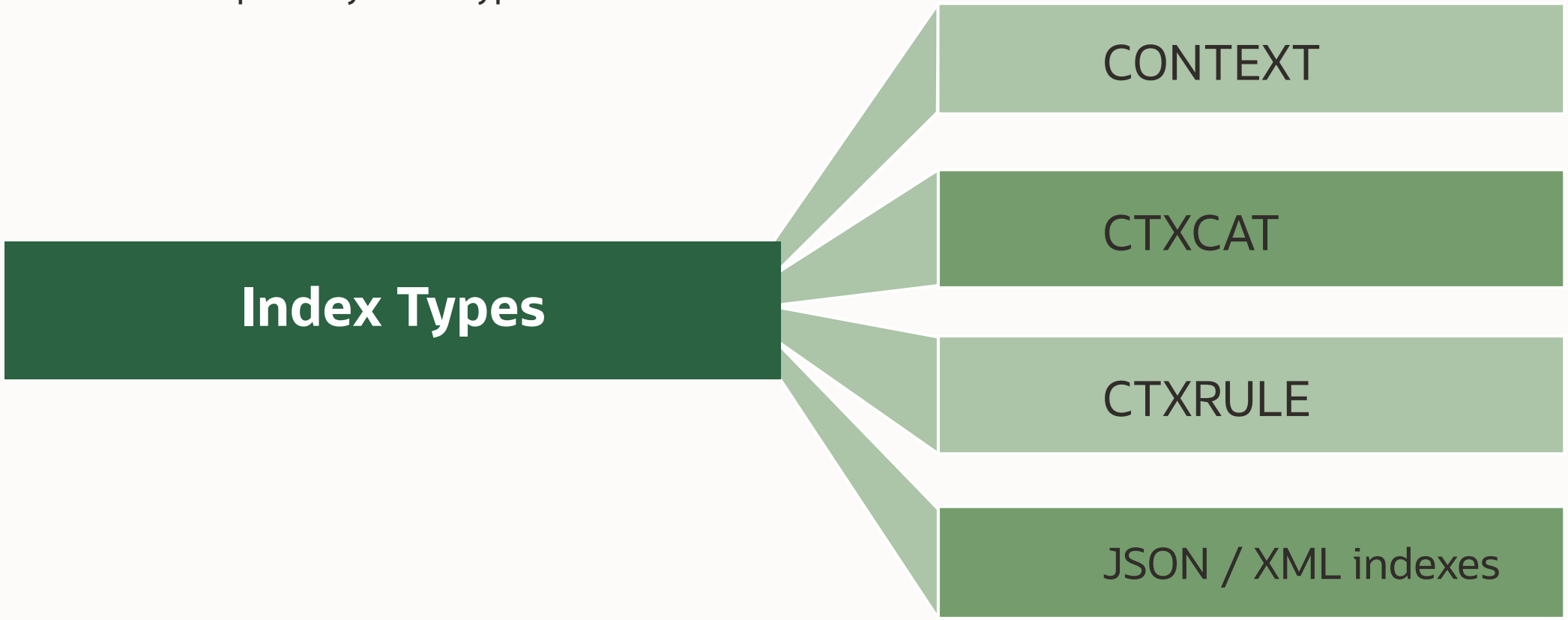
- Extend current apps to make better use of textual fields
- Build new apps specifically targeted at document searching

All the power of Oracle Database and a familiar development environment

Rock-solid autonomous database infrastructure for your text apps

Index Types in Oracle Text

CONTEXT is primary indextype but other are available



Product Architecture

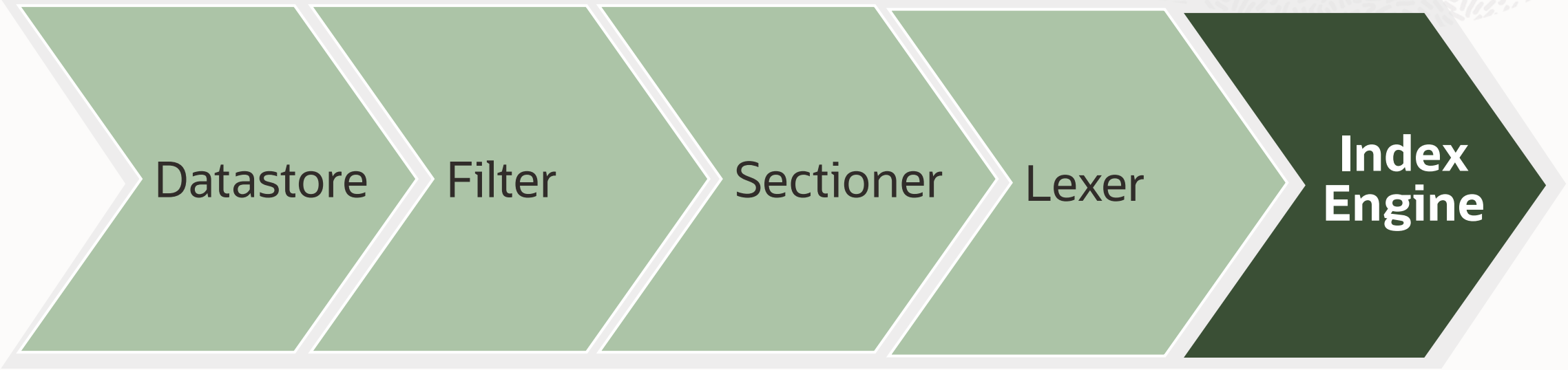
All indexes use the EXTENSIBILITY FRAMEWORK which allows for “Domain Indexes”

Unlike user domain indexes, code resides in Oracle kernel

Oracle Text index data and metadata is stored in Oracle Database tables – no file storage required

Features such as RAC, partitioning, parallel query are all “text aware”

The Indexing Pipeline



- All pipeline stages are configurable by a system of "preferences" and "attributes"
- Most can be replaced by user-written plugin modules in PL/SQL, C or Java

Indexing Pipeline – Stages and Inputs

Pipeline stages

- Datastore – where the data is fetched from
  - e.g. DIRECT_DATASTORE => simple single column index
  - MULTI_COLUMN_DATASTORE => concatenate several columns
- FILTER – defines how binary documents should be converted to text
- SECTIONER – divides documents into sections such as title, body
- LEXER – divides text into "tokens" (words, pictogram groups, etc)

Other inputs

- STORAGE – options for laying out the index on disk
- WORDLIST – defines special attributes, e.g. case sensitivity
- STOPLIST – list of common ("noise") words that should not be indexed

Oracle Text Queries

Can use

- standard SQL “SELECT” query syntax
  - Best for integration into standard applications
- XML-based Result Set Interface
  - Best for facet navigation and summary information

Return a “score” to indicate the relevance of each hit

Use “sections” to mark structured data in documents or in other columns of the table

Can mix structured (numeric, date) with unstructured (full text) searches in a single query expression

Creating and using a simple Oracle Text Index

```
CREATE INDEX prod_name_idx ON
   product_information(product_name)
   INDEXTYPE IS ctxsys.context ;
```

```
SELECT score(99), product_id, product_name
    FROM product_information
     WHERE contains (product_name,
        'monitor NEAR full hd', 99)>0
      ORDER BY score(99) DESC ;
```

```
  SCORE(99) PRODUCT_ID   PRODUCT_NAME
  --------- ----------   ------------------------------
  72        3331         Full HD Monitor 22 inch
  56        3060         Monitor and TV combo, full HD
```

21c ‘Search Index’

21c has new “CONTEXT_V2” indextype and simplified create syntax:

```
CREATE SEARCH INDEX prod_name_idx ON
product_information [for JSON];
```

Used with CONTAINS in the same way as CONTEXT indextype

If ‘for JSON’ is specified, can use either CONTAINS or JSON_TEXTCONTAINS

Main difference is sharding and partitioning support

Some Useful Operators

`STEM ($)` - matches words with the same linguistic base form

`FUZZY (...)` - finds mis-spellings

`NEAR (...)` - proximity search for words close to each other

`WITHIN` *`section`* – simple section search

`SDATA (...)` - performs structured search within text index

`NDATA (...)` - match names (or other similar inexact data)

`MVDATA(...)` – multi-valued section data

`NT, BT, SYN` – thesaurus operators

Thesaurus Support

Full thesaurus support to ISO-2788 and ANSI Z39.19

`CTXLOAD` utility is used to load thesauri from text file
- or use `CTX_THES` PL/SQL package

Thesaurus operators must be explicitly included in queries

- `CONTAINS(col, 'dog OR syn(feline)') > 0`

Thesaurus expansions at query time. Above query equivalent to:

- `CONTAINS (col, 'dog OR (feline=cat=moggy=puss)') > 0`

Index Synchronization

By default, text indexes are *ASYNCHRONOUS*

Index must be sync'd before they can be queried

Sync options:

- SYNC (ON COMMIT) : Sync automatically after each commit
- SYNC (EVERY "freq=secondly;interval=5") : Sync every 5 seconds
- SYNC (manual) : Must call *ctx_ddl.sync_index* to synchronize the index

There is a trade-off between frequency of sync and fragmentation of index

19c has automatic defrag, previous versions must defrag manually

Typically customers choose ON COMMIT or every 5s to 1 minute

23c: Maintenance Auto (MA)

Automatic Index Maintenance

- SYNC-ON-COMMIT incurs a DML overhead to maintain the index
- SYNC EVERY requires the user to know the right sync interval for a given index
- Maintenance Auto (MA) automatically synchronizes an index when it sees new DML
- MA makes optimal use of background workers across all MA indexes

```
CREATE SEARCH INDEX … FOR JSON
PARAMETERS ('MAINTENANCE AUTO');
```

Support:

- Default for newly created 23c indexes even when MAINTENANCE AUTO is not specified explicitly
- Upgraded indexes can be converted to MA mode using ALTER INDEX (no index rebuild needed)
- In future releases, MA will also automatically optimize the index and perform any other maintenance

Denormalization

Several techniques to combine multiple columns or tables into one index

- Materialized Views
- MULTI_COLUMN_DATASTORE for single tables and/or functions
  - Creates a "virtual document" for indexing from a SELECT list
  - COLUMNS = 'name, USER, description, taglist, myfunction(keyvalue)'
  - Each column (or psuedo-column/function output) delimited by <columntag> by default
- USER_DATASTORE procedure in PL/SQL
  - Virtual document constructed by PL/SQL procedure
  - Ultimate flexibility
  - Can combine data from multiple tables or even external sources
  - Called once for each row that is indexed

23c: API: DBMS_SEARCH

- DBMS_SEARCH allows creation of Text Indexes against multiple tables and views
- Creates centralized index with JSON metadata to allow identification of source table
- Requires some knowledge of SQL/JSON to use effectively
- Tables AND Views added must have unique keys

```
create table PRODUCTS ( id number primary key, description  varchar2(2000));
create table CUSTOMERS ( id number primary key, address json );

insert into PRODUCTS values
    (1, 'product1 description');
insert into CUSTOMERS values
    (99, '{ "city": "Richmond", "country": "United Kingdom"}');

-- Create the DBMS_SEARCH index (PROD):
exec DBMS_SEARCH.CREATE_INDEX('PROD')
```

(code continues on next slide)

23c: DBMS_Search (continued)

```
-- Populate the index with tables:
exec DBMS_SEARCH.ADD_SOURCE('PROD', 'PRODUCTS')
exec DBMS_SEARCH.ADD_SOURCE('PROD', 'CUSTOMERS')

-- The data is stored in a text table called PROD, which matches your index name.
-- View the virtual indexed document:
select DBMS_SEARCH.GET_DOCUMENT('PROD',METADATA) from PROD;

-- Run a query which fetches metadata from the index
select metadata from PROD where contains(data,'product1')>0;

-- Now run a query to get stuff where there is match in the customers address
select metadata from PROD where json_textcontains(data, '$.ROGER.CUSTOMERS.ADDRESS', 'Richmond');

-- Now join that with the base table so we get the actual original data back
select c.id, c.address
from PROD P, CUSTOMERS c
where json_textcontains(data, '$.ROGER.CUSTOMERS.ADDRESS', 'Richmond')
and p.metadata."KEY"."ID" = c.id;
```

More information about Oracle Text

LiveLabs

Full-text indexing in Oracle Database

https://bit.ly/3yCeLAb

Documentation

Oracle Text Application Developers Guide, 21c

https://bit.ly/3EzRToX

Documentation

Oracle Text Reference, Oracle Database 21c

https://bit.ly/3CoT2Nq

Text on O.com

Oracle Text

https://bit.ly/3rOtNiI

ORACLE

Our mission is to help people see data in new ways, discover insights, unlock endless possibilities.