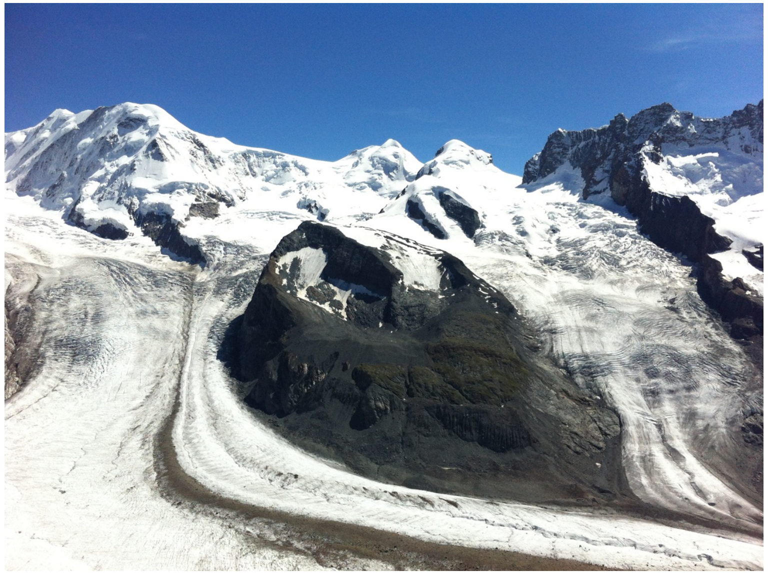
# Search Indexes and Ubiquitous Search in 23c

Niall Mc Phillips - Long Acre sàrl

*niall.mcphillips@longacre.ch*

*@Niall_McP*



# About me: Niall Mc Phillips

Owner - Long Acre sàrl

Irish 🇮🇪 / 🇨🇭 Swiss Living in Geneva, Switzerland.

- Oracle ACE Pro
- Oracle Developer and DBA for >30 years
- Developing web applications with Oracle DB since 1995
- Developing with APEX since 2005 (HTML DB 1.6)
- Organizer of the Swiss APEX Meetup group

*@Niall_McP*

✉ niall.mcphillips@longacre.ch

Oracle ACE Program

**500+ technical experts helping peers globally**

The **Oracle ACE Program** recognizes and rewards community members for their technical and community contributions to the

Oracle ACE Director | Oracle ACE Pro | Oracle ACE Associate

Oracle ACE

[ace.oracle.com/nominate](ace.oracle.com/nominate)

[ace.oracle.com](ace.oracle.com)

Connect: ✉ **aceprogram_ww@oracle.com** Facebook.com/OracleACEs @oracleace

# _SYMPOSIUM 42

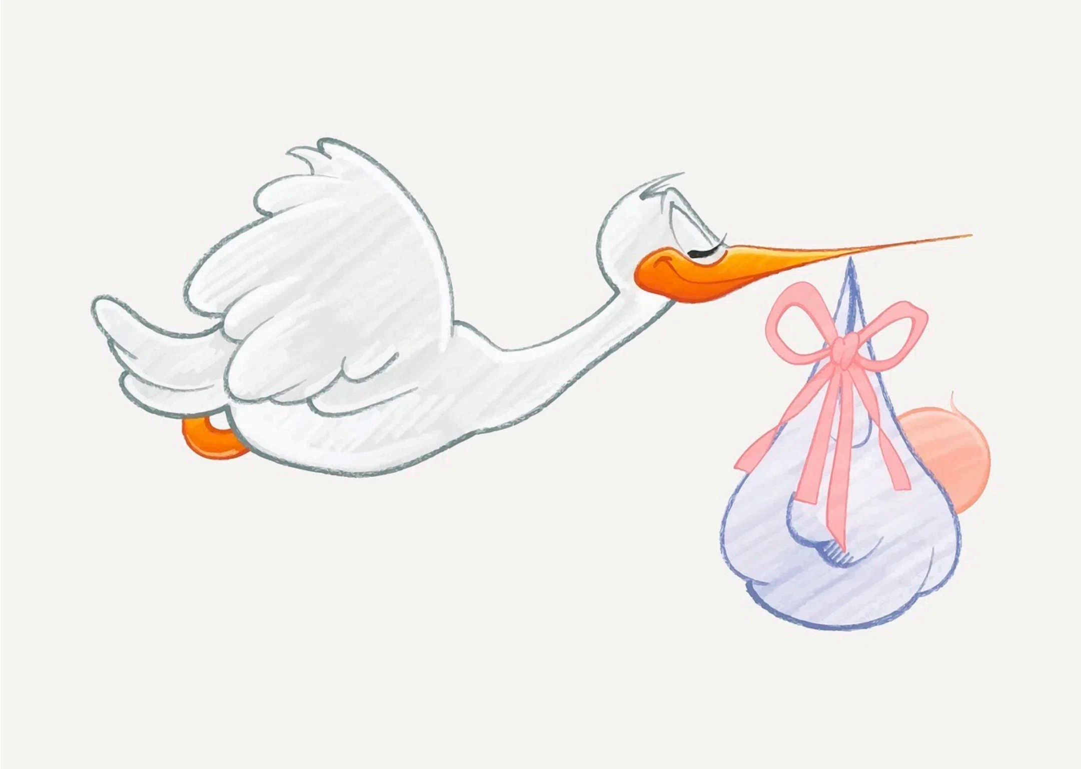
Created by the community, to support the community

Sharing of reliable knowledge
Supporting the various user groups and individuals

@sym_42

https://sym42.org/

# Where do Search Indexes come from?



# Where do Search Indexes come from?

- Oracle8 (1997) - Oracle ConText
- Oracle8i (1999) - Oracle Intermedia Text.
- Oracle9i (2001) - Oracle Text
- Oracle23c (2023) - Search Indexes
- An integral part of all Oracle database editions

# Search Indexes - Built on a rock-solid foundation

long acre
solutions today

# Using Search Indexes

- *really fast* and quite easy to start using
- just create an index and start searching
- index varchar2, XML, JSON, clobs and blobs (like pdfs)
- uses the “contains” clause for querying
- allows AND/OR and more complex logic
- + many more advanced features…

# Searching with “like”

This is the basic “naïve” textual search that can work for very small datasets.

- it will not use an index if there is a wildcard at the start of the search string

  `where mytext like ‘%dog%’`

- it is case-sensitive
  - *`where lower(mytext) like ‘%dog%’`*

# Creating a simple Search Index

```
create search index indexname
       on tablename (columnname);
```

# Creating a simple Search Index - examples

```
create search index si_judgments on judgments(description);

Index SI_JUDGMENTS created.


create search index si_hist_events on hist_events (description);

Index SI_HIST_EVENTS created.
```

## Searching with contains

```
select * from tablename
 where
 contains(searchcolumn,'searchtext') > 0;
```

long acre
solutions today

# Scoring search results

- The **score** of a search result gives an idea of the relevance of the result. High score indicates a higher relevance.
- Scores are always in the 1 to 100 range
- Scores have absolutely no meaning outside of their own query and cannot be compared between different queries, sub-queries or datasets.

long acre
solutions today

# Scoring search results - syntax

```
select score(1), t.* from tablename t
where
contains(searchcolumn,‘searchtext’,1) > 0
order by 1 desc;
```

Note that the (1) in score(1) matches the ,1) in the contains clause

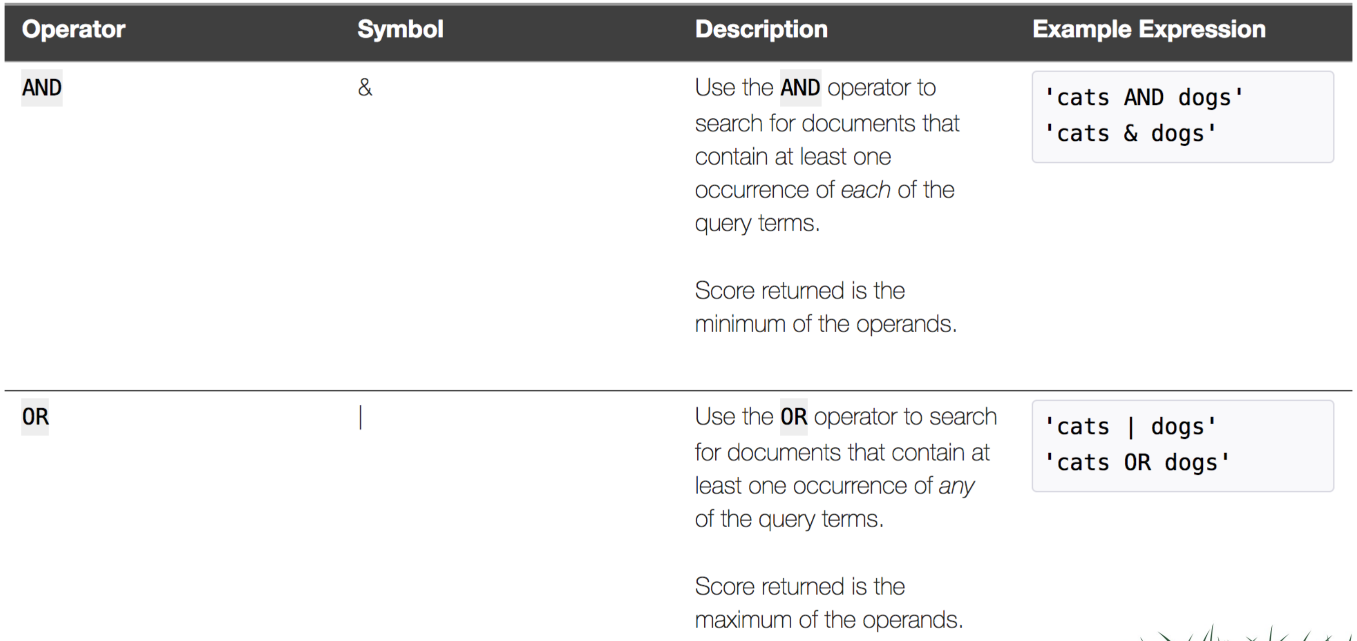
# Cloud DEMO on Autonomous DB – Basic Searches

long acre
solutions today

# Oracle Text operator grammar and syntax

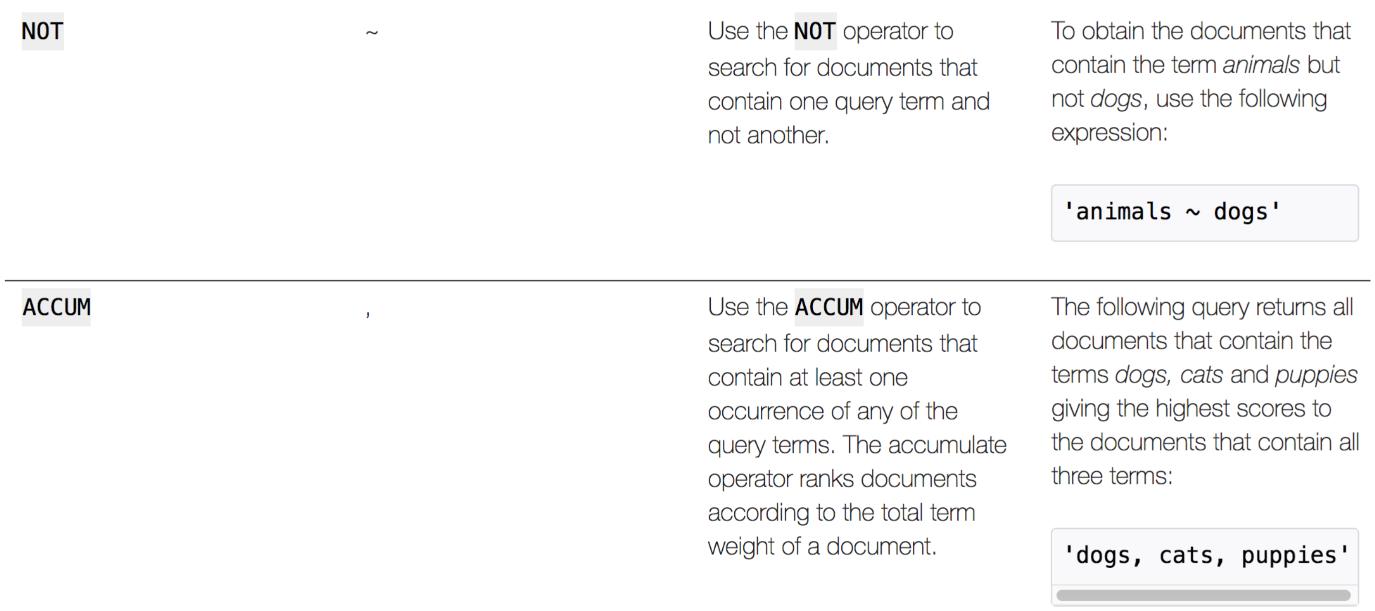
long acre
solutions today

# Searching with AND and OR operators



| Operator | Symbol | Description | Example Expression |
|---|---|---|---|
| `AND` | & | Use the `AND` operator to search for documents that contain at least one occurrence of *each* of the query terms.<br>Score returned is the minimum of the operands. | `'cats AND dogs'`<br>`'cats & dogs'` |
| `OR` | \| | Use the `OR` operator to search for documents that contain at least one occurrence of *any* of the query terms.<br>Score returned is the maximum of the operands. | `'cats \| dogs'`<br>`'cats OR dogs'` |

# Searching with NOT and ACCUM operators



| | | | |
|---|---|---|---|
| `NOT` | ~ | Use the `NOT` operator to search for documents that contain one query term and not another. | To obtain the documents that contain the term *animals* but not *dogs*, use the following expression:<br>`'animals ~ dogs'` |
| `ACCUM` | , | Use the `ACCUM` operator to search for documents that contain at least one occurrence of any of the query terms. The accumulate operator ranks documents according to the total term weight of a document. | The following query returns all documents that contain the terms *dogs*, *cats* and *puppies* giving the highest scores to the documents that contain all three terms:<br>`'dogs, cats, puppies'` |

# Principal operators

| | |
|---|---|
| AND | & |
| OR | \| |
| NOT | ~ |

# Some other operators

EQUIValence (=)

MINUS (-)

Fuzzy

NEAR (;)

stem ($)

soundex (!)

and many more…

*full details in Oracle Text Reference at:*

*https://docs.oracle.com/en/database/oracle/oracle-database/23/ccref/index.html#Oracle%C2%AE-Text*

# Demo

- Examples of searches with CONTAINS

# Escaping terms entered

search for

- Africa and Near East
- “Near” is also an operator so we escape the search words using curly brackets {}

**{Africa}&{Near East}**

long acre
solutions today

# Preparing text for search

- It can quickly become quite complex to parse and prepare the search text that users enter
- Normally some type of pre-processing is required for real-world scenarios

# Pre-processing user-input text for Google-like searches

Baseline principles:

- End-users should not need to know or understand Search Index grammar

- Everyone wants their searches to work “just like Google”

# Pre-processing user-input text for Google-like searches

One approach to pre-processing

```
FOR i IN 1..50 LOOP  -- try to get rid of multiple spaces
         v_text := replace(v_text,'  ',' ');
     END LOOP;
     v_text := replace(v_text,'*','%');  -- wildcard chars
     v_text := replace(v_text,'?','_');  -- wildcard chars
     v_text := replace(v_text,'"',null);
     v_text := replace(v_text,'''',null);
     v_text := replace(v_text,',',null);
     v_text := replace(v_text,';',null);
     v_text := replace(v_text,'.',null);
     v_text := replace(v_text,'+','&');
     v_text := replace(v_text,' &','&');
     v_text := replace(v_text,'& ','&’);
     etc...
```

## Pre-processing user-input text for Google-like searches

- While researching for this presentation I found a great PL/SQL package* written and made freely available by Roger Ford of Oracle.

PARSER package:

*https://blogs.oracle.com/searchtech/oracle-text-query-parser*

**I really wish I had found this a few years ago - I would have saved so much time that I spent writing my own ;)*

# The PARSER package

*We will use the parser.simpleSearch function to transform "Google-like" syntax into Oracle Text syntax.*

*e.g. "Ad Hoc Committee" becomes*

({Ad Hoc Committee})

# PARSER examples

assessment damages becomes

({assessment},{damages})

+assessment +damages becomes

({assessment}&{damages})

+assessment -damages becomes

({assessment}) ~{damages}

long acre

solutions today

# Stoplists

Stoplists are lists containing words “stopwords” that should be ignored when searching.

i.e. frequently occurring words such as ”the”, “also”, “their”, …

# Stoplists - First create a Lexer

Create a lexer called “HrOUG_lexer” of type basic_lexer:

```
ctx_ddl.create_preference('HrOUG_lexer',
   'basic_lexer’);
```

# Stoplists - Creating a stoplist

You can create your own stoplist and add any words that are appropriate for your application.

Create a stoplist called “hrougstoplist”:

```
ctx_ddl.create_stoplist(‘hrougstoplist','BASIC_STOPLIST’);
```

# Stoplists - Adding words to a stoplist

```
ctx_ddl.add_stopword('hrougstoplist’, ‘HrOUG');
ctx_ddl.add_stopword('hrougstoplist', 'APEX');
ctx_ddl.add_stopword('hrougstoplist’, 'Database');
ctx_ddl.add_stopword('hrougstoplist’, 'Oracle’);
```

***Some stopword sources:***

https://github.com/stopwords-iso

http://www.stopwords.org/

# Stoplists - Create an index using the stoplist

```
create search index ind_decisions$1 on decisions(decision)
        parameters ('lexer hroug_lexer  stoplist hroug');
```

long acre
solutions today

## Stoplists - cloud demo

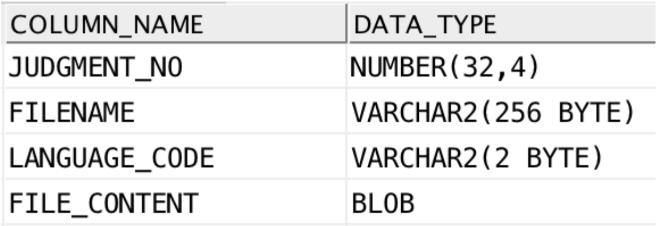
***Demonstrate management of stopwords for a stoplist.***

long acre
solutions today

# Indexing BLOB columns

Sample table JUDGMENT_DOCUMENTS

- Blob column FILE_CONTENT contains PDF files for each judgment



| COLUMN_NAME | DATA_TYPE |
|---|---|
| JUDGMENT_NO | NUMBER(32,4) |
| FILENAME | VARCHAR2(256 BYTE) |
| LANGUAGE_CODE | VARCHAR2(2 BYTE) |
| FILE_CONTENT | BLOB |

# Indexing PDF files stored in BLOB columns

```
create search index txt_judgment_documents$1
       on judgment_documents(file_content);
```

# Searching the BLOB documents

- BLOB Searches are the same as with any other column

```
select score(1) as the_score,
       j.*  from judgments j
 inner join judgment_documents jd on (jd.judgment_no = j.judgment_no)
 where contains(jd.file_content,:P8_SEARCHTEXT_PROCESSED, 1) > 0
 order by 1 desc;
```

# Retrieving Snippets from PDF documents

## Using the CTX_DOC package

```
CTX_DOC.SNIPPET(
   index_name IN VARCHAR2,
   textkey IN VARCHAR2,
   text_query IN VARCHAR2,
   starttag IN VARCHAR2 DEFAULT '<b>’,
   endtag IN VARCHAR2 DEFAULT '</b>’,
   entity_translation IN BOOLEAN DEFAULT TRUE,
   separator IN VARCHAR2 DEFAULT '<b>...</b>’ )
return varchar2;
```

# Retrieving Snippets from PDF documents

```
CTX_DOC.SNIPPET(
   index_name   => ‘TXT_JUDGMENT_DOCUMENTS$3’,
   textkey      => jd.rowid,
   text_query   => :P8_SEARCHTEXT_PROCESSED);
```

# Add the snippet to the query

```
select score(1) as the_score,
       ‘<h4>Judgment no.: '||to_char(j.judgment_no)||' - '
            ||to_char(j.publication_date,'YYYY-MM-DD’)
            ||'</h4>'
            ||'... '
ctx_doc.snippet(index_name   => 'TXT_JUDGMENT_DOCUMENTS$3',
                textkey      => jd.rowid,
                text_query   => :P8_SEARCHTEXT_PROCESSED)
            ||' ...' as snippet
  from judgments j
 etc…
```

# Advanced snippets in PL/SQL

***For multiple snippets within a single result use CTX_DOC to retrieve an array of snippets.  Usually you will built a custom-fuction to return these in the desired way.***

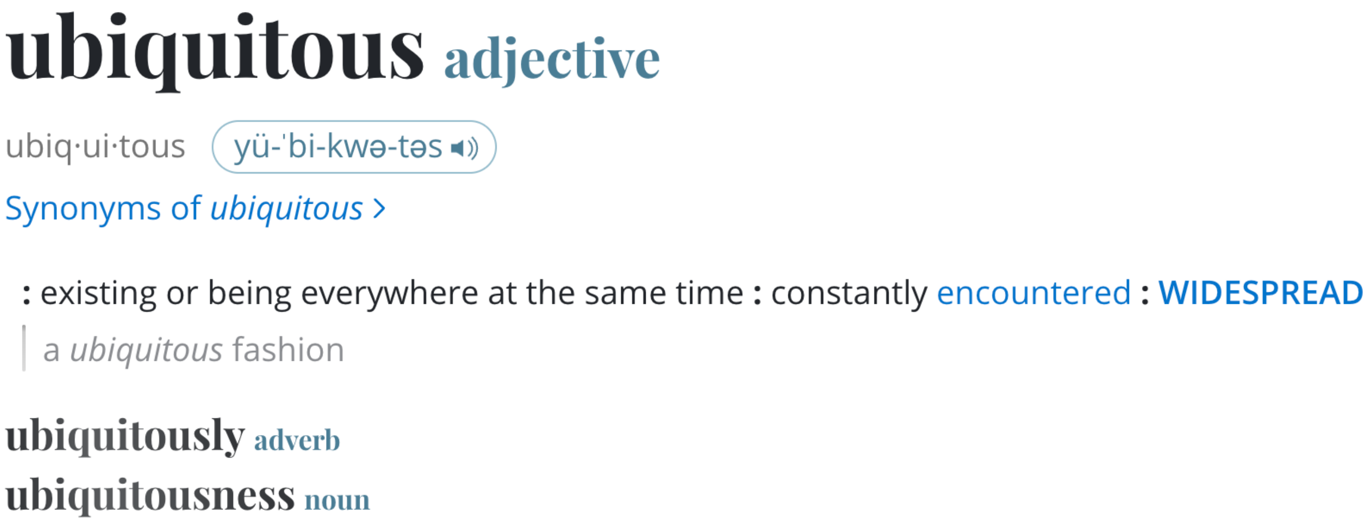
```
a_snippets ctx_doc.highlight_tab; -- declaration
begin
 ctx_doc.set_key_type (ctx_doc.type_rowid);
 ctx_doc.highlight (index_name =>’TXT_JUDGMENT_DOCUMENTS$3’,
    textkey      => rec_loop.rid,
    text_query   => upper(trim(v_word)),
    restab       => a_snippets ,  -- snippets are placed here
    plaintext    => TRUE);
```

***then loop through** a_snippets **to get all occurrences.***

# Cloud DEMO – Snippets as an APEX Classic Report

long acre
solutions today

# 23c Ubiquitous Search



**ubiquitous** adjective

ubiq·ui·tous yü-ˈbi-kwə-təs 🔊

Synonyms of *ubiquitous* ›

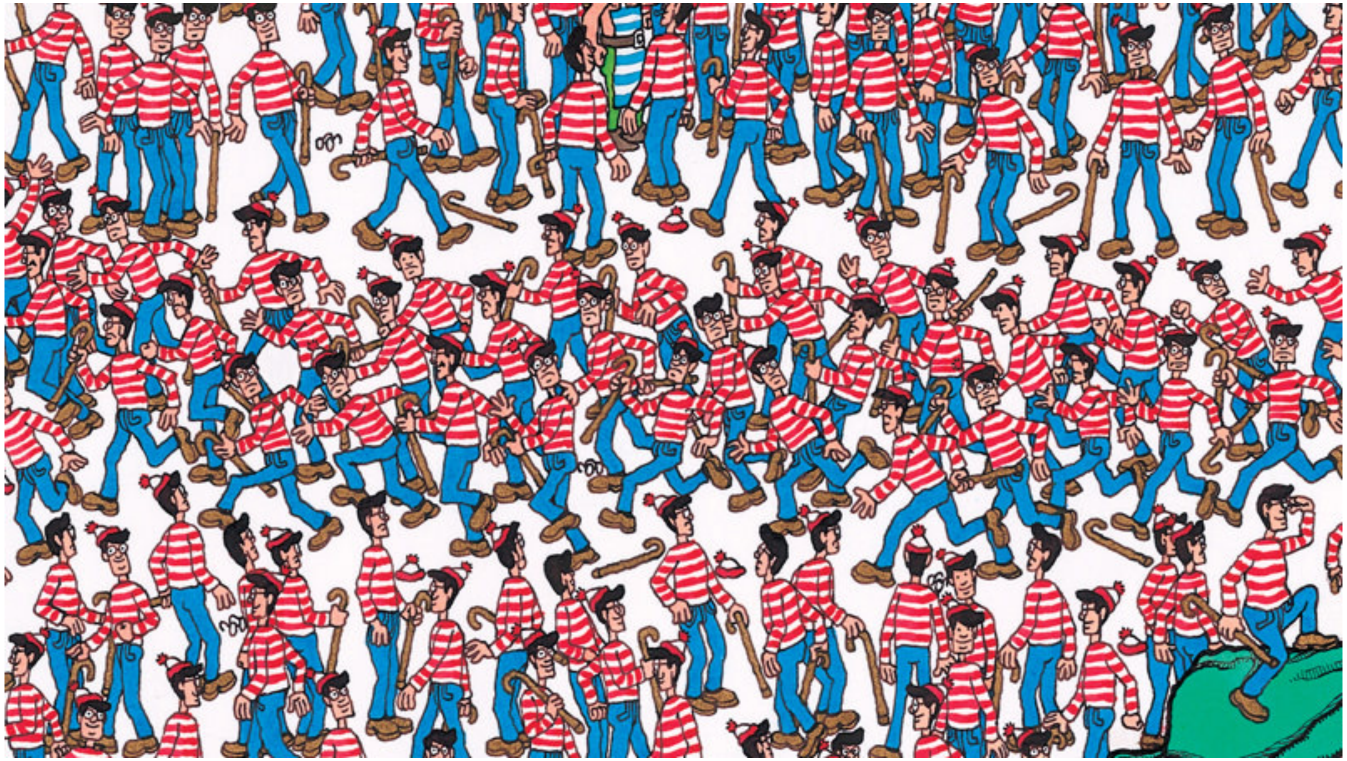
: existing or being everywhere at the same time : constantly encountered : WIDESPREAD

> a *ubiquitous* fashion

**ubiquitously** adverb

**ubiquitousness** noun

# 23c - Ubiquitous Search



# 23c Ubiquitous Search

## *DBMS_SEARCH*

# 23c Ubiquitous Search – Creating an index

First, we'll create the index:

***dbms_search.create_index('UB1SEARCH');***

# 23c Ubiquitous Search – Adding Data Sources

Now, let’s add two completely unrelated data sources:

```
dbms_search.add_source('UB1SEARCH', 'JUDGMENTS’);

dbms_search.add_source('UB1SEARCH', 'HIST_EVENTS');
```

# 23c Ubiquitous Search – table created

```
SQL> desc ub1search

Name     Null?    Type
-------- -------- --------------
METADATA NOT NULL JSON
DATA              JSON
OWNER             VARCHAR2(128)
SOURCE            VARCHAR2(128)
KEY               VARCHAR2(1024)
```

# 23c Ubiquitous Search – table created

*Let’s take a look at the metadata*

```
select metadata from ub1search
 where contains(data, 'observatory') > 0;
```

# 23c Ubiquitous Search – table created

## *Let’s take a look at the metadata*

```
{"OWNER":"NIALL","SOURCE":"HIST_EVENTS","KEY":{"ID":10000}}
{"OWNER":"NIALL","SOURCE":"HIST_EVENTS","KEY":{"ID":10001}}
{"OWNER":"NIALL","SOURCE":"HIST_EVENTS","KEY":{"ID":10002}}
{"OWNER":"NIALL","SOURCE":"HIST_EVENTS","KEY":{"ID":10003}}
....
{"OWNER":"NIALL","SOURCE":"JUDGMENTS","KEY":{"JUDGMENT_NO":995}}
{"OWNER":"NIALL","SOURCE":"JUDGMENTS","KEY":{"JUDGMENT_NO":994}}
{"OWNER":"NIALL","SOURCE":"JUDGMENTS","KEY":{"JUDGMENT_NO":993}}
```

# 23c Ubiquitous Search – getting to the data

## *We can extract the keys using JSON_TABLE*

```
...
  from ub1search us
 cross join json_table(us.metadata, '$.KEY[*]'
               columns (judgment_no   number path '$.JUDGMENT_NO',
                        hist_event_id number path '$.ID')) j
...
```

# 23c Ubiquitous Search – getting to the data

## *We can extract the keys using JSON_TABLE*

```
...
  from ublsearch us
 cross join json_table(us.metadata, '$.KEY[*]'
               columns (judgment_no   number path '$.JUDGMENT_NO',
                        hist_event_id number path '$.ID')) j
...
```

# 23c Ubiquitous Search – getting to the data

## *Join with the underlying tables...*

```
...
  from ub1search us
 cross join json_table(us.metadata, '$.KEY[*]'
               columns (judgment_no   number path '$.JUDGMENT_NO',
                        hist_event_id number path '$.ID')) j
  left outer join hist_events h on (h.id = j.hist_event_id)
  left outer join judgments ju  on (ju.judgment_no = j.judgment_no)
...
```

# 23c Ubiquitous Search – showing the data

***Join with the underlying tables...***

```
...
select case
         when j.judgment_no is not null then 'Judgment'
                                        else 'Historical Event'
       end as result_type,
       coalesce(j.judgment_no, j.hist_event_id) as theId,
       coalesce(h.theDate, to_char(ju.publication_date,'YYYY-MM-DD')) as theDate,
       coalesce(h.description, ju.short_summary_en, ju.decision_en) as theText,
       us.data
  from ub1search us
...
```

# 23c Ubiquitous Search – showing the data

## *Creating a view for other developers*

```
create or replace view vw_ubsearch as
select case
         when j.judgment_no is not null then 'Judgment'
                                        else 'Historical Event'
       end as result_type,
       coalesce(j.judgment_no, j.hist_event_id) as theId,
       coalesce(h.theDate, to_char(ju.publication_date,'YYYY-MM-DD')) as theDate,
       coalesce(h.description, ju.short_summary_en, ju.decision_en) as theText,
       us.data
  from ub1search us
 cross join json_table(us.metadata, '$.KEY[*]'
               columns (judgment_no   number path '$.JUDGMENT_NO',
                        hist_event_id number path '$.ID')) j
  left outer join hist_events h on (h.id = j.hist_event_id)
  left outer join judgments ju on (ju.judgment_no = j.judgment_no)
```

long acre
solutions today

## 23c Ubiquitous Search – showing the data

# *Demo with APEX*

long acre
solutions today

Any Questions?