DB2 z/OS Text Search at Westfield Insurance
About Westfield Insurance

- Property and casualty insurance for over 167 years
- One of the largest non-public companies in Ohio
- Property and Casualty Insurance and Banking
- Over 3.9 billion in assets/1.4 billion in written premium
- Network of more than 1,200 leading independent agencies
- 2,500 employees in 42 Service Offices servicing 49 states
About Westfield Insurance

For Additional Information Please Visit Our Websites at:

www.westfieldinsurance.com

www.westfield-bank.com
System Info

• z/OS 1.13 operating system on IBM BC12 hardware
• DB2 z/OS V10
  • 4 subsystems
  • No data sharing
• Red Hat Enterprise Linux server release 6.6
  • 2 CPU – 2.70GHz
  • 8GB memory
  • 60GB Disk
Objectives

• Understand the capabilities of Text Search
• Understand what is required to install and maintain Text Search
• Learn about Westfield’s experiences and usage of Text Search
Topics

1. Text Search Overview
2. Install on linux/Windows servers
3. Install additional software on DB2 z/OS
4. Configure on DB2 z/OS
5. The Search Application
6. Define searchable columns
7. Index build and update
8. Westfield’s Production Use – Hybrid
9. Text Search administration – DB2 z/OS
10. Text Search administration – Linux/Windows
Overview

TOPIC 1
Documentation

• “IBM Text Search for DB2 for z/OS Installation, Administration, and Reference” manual
• In the same list as the other DB2 V10 manuals
What is DB2 Text Search?

- DB2 SQL functions to provide advanced search capabilities
- Search Text (CHAR & VARCHAR), XML, and other file types
- Indexing and search engine on Linux or Windows server(s)
- Flexible search criteria
  - AND, OR, NOT, Wildcards, Synonyms
  - Scoring (some hits are better than others)
- Integrated into DB2 optimizer
CONTAINS Function

SELECT <column list>
FROM <table>
WHERE CONTAINS(<search column>,
    ' <search terms> ',
    ' <optional parms> ' ) = 1

SEARCH TERMS (the common ones)
Terms: One or more words (not case sensitive)
Operators: AND, OR, NOT
Wildcards: * matches any char(s) at beginning, middle, or end
Phrases: enclose phrase in double quotes
Grouping: normal use of paren’s

OPTIONAL PARMS
RESULTLIMIT=n limits the number of rows returned
SYNONYM=ON|OFF
CONTAINS – More cool stuff

**Fuzzy search**
Match similar terms

Ex:
```
analyze~
```
matches: analyze, analytics, analysis, etc

**Proximity search**
Find terms within a specified number of words

Ex:
```
“rock roll*”~3
```
will find:
```
“rock and roll”
“rolling rock”
```
but not:
```
“to be a rock and not to roll”
```

**Single Wildcard**
```
?
```
Sample CONTAINS

SELECT *
FROM MYTABLE
WHERE CONTAINS
  (MYCOLUMN,
   'abcd* 1299 xyzz',
   'RESULTLIMIT = 100' ) = 1
SCORE

• Same syntax and terms as CONTAINS
• Measure how well a row matches the search terms
• Returns a value between 0 and 1
• Can be used to ORDER BY
• Can weight certain terms (see manual)
Sample SCORE

SELECT
    \texttt{SCORE} (MYCOLUMN, 'abcd 1299 xyzz')
    ,MYCOLUMN
FROM MYTABLE
WHERE \texttt{CONTAINS} (MYCOLUMN, 'abcd 1299 xyzz') = 1
ORDER BY 1 DESC
CALL SYSPROC.
SYSTS_UPDATE (<parms>)

DB2 z/OS

Linux/Windows server

ROWID, search text

Collection file BLOB

Collection file

Search Table

ROWID, search text

Text search

Collection file

Text Search

Admin DB

Collection file master table
SELECT ... FROM ... WHERE CONTAINS(srch_col, search terms) = 1
Install on Linux or Windows
Install on LINUX or Windows

- Download from IBM website
- Install on linux VM
- AND/OR
- Install on zlinux ZVM
  - Uses hipersockets (Avoids network)
  - Westfield’s “Primary” server
- OR install on Windows
Initialize the server

cd /opt/IBM/ECMTextSearch/bin

sh configTool.sh configureHTTPListener -configPath ../config -adminHTTPPort 8191

Get authentication token and encryption key

sh configTool.sh printToken -configPath ../config

Set option to autostart

See manual for details.
Install on DB2 z/OS

Topic 3
Required DB2 objects are (or could be) already defined. If not, run DB2 install job specified in the manual.

Databases / tables

SYSIBMTA (DB for Admin tables)
- SYSIBMTS.SYSTEXTCOLUMNNS
- SYSIBMTS.SYSTEXTCONFIGURATION
- SYSIBMTS.SYSTEXTCONNECTINFO
- SYSIBMTS.SYSTEXTDEFAULTS
- SYSIBMTS.SYSTEXTINDEXES
- SYSIBMTS.SYSTEXTLOCKS
- SYSIBMTS.SYSTEXTSERVERHISTORY
- SYSIBMTS.SYSTEXTSERVERS
- SYSIBMTS.SYSTEXTSTATUS
- SYSIBMTS.SYSTLOB1

SYSIBMTS (DB for Text Index tables when defined)
Stored Procedures

SYSPROC. SYSTS_ALTER
SYSPROC.SYSTS_CREATE
SYSPROC.SYSTS_DROP
SYSPROC.SYSTS_RESTORE
SYSPROC.SYSTS_START
SYSPROC.SYSTS_STOP
SYSPROC.SYSTS_TAKEOVER
SYSPROC.SYSTS_UPDATE

Function

SYSFUN. SYSTS_ENCRYPT
Install on DB2 z/OS

- Install DB2 Accessory Suite - FMID H2AF210 (plus J2AG210)
  - New load modules in SDSNLOAD
- Create java runtime dataset
  - Use DB2 install job in SDSNSAMP
  - TSO ISHELL to find USS directories
    - /usr/lpp/db2/*
    - /usr/lpp/java/*
  - Used by Java encryption function
Install on DB2 z/OS

• Create WLM environment for Java Function
  • Alter SYSFUN. SYSTS_ENCRYPT to use it
• Create WLM environment for Admin Stored Procedures
  • Alter SYSROC.SYSTS* to use it
Java WLM started task example

//DB2PWLMJ PROC RGN=0K,APPLENV=XXXXXXXX,DB2SSN=DB2P
//IEFPROC EXEC PGM=DSNX9WLM,REGION=&RGN,TIME=NOLIMIT,
//      PARM='&DB2SSN,1,&APPLENV'
//STEPLIB DD DISP=SHR,DSN=CEE.SCEERUN
//      DD DISP=SHR,DSN=<hlq>.SDSNEXIT
//      DD DISP=SHR,DSN=<hlq>.SDSNLOAD
//      DD DISP=SHR,DSN=<hlq>.SDSNLOD2
//JAVAENV DD DISP=SHR,DSN=<hlq>.JSPENV
//SYSPRINT DD SYSOUT=* 
//CEEDUMP DD SYSOUT=*

Admin Stored Procedure WLM started task example

//DB2PWLM1 PROC RGN=0K,APPLENV=XXXXXXXXX,DB2SSN=DB2P,NUMMTCB=8
//IEFPROC EXEC PGM=DSNX9WLM,REGION=&RGN,TIME=NOLIMIT,
   // PARM='&DB2SSN,&NUMMTCB,&APPLENV'
//STEPLIB DD DISP=SHR,DSN=<hlq>.SDSNLOAD
//       DD DISP=SHR,DSN=<hlq>.SDSNLOD2
• Create surrogate RACF/ACF2 UserID
  • Performs DB2 z/OS tasks initiated from the linux/windows server(s)
• Define security
  • Must be able to SELECT, INSERT, UPDATE, DELETE Text Search ‘catalog’ tables (SYSIBMTS.*)
Configure on DB2 z/OS

Topic 4
Configuration & Administration

- Done by using DB2 stored procedures
- Some stored procedures need parms, so need a way to execute a stored procedure w/parms
  - DB2 Connect Command Editor
  - Custom REXX code
- Refer to manual for more options and details
Define the Text Search servers

- Use the Authentication Token and Encryption Key created during the server install(s)

**zLinux definition**

```sql
INSERT INTO SYSIBMTS.SYSTEXTSERVERS
  (SERVERNAME, SERVERPORT, SERVERAUTHTOKEN, SERVERMASTERKEY)
VALUES
  ('<ip_addr of hipersockets>', 8191, '<auth token>', '<encryption_key>');
```

**VM Linux definition**

```sql
INSERT INTO SYSIBMTS.SYSTEXTSERVERS
  (SERVERNAME, SERVERPORT, SERVERAUTHTOKEN, SERVERMASTERKEY)
VALUES
  ('<server name>', 8191, '<auth token>', '<encryption_key>');
```
Define “connect back” Information

- Use surrogate ID created earlier

```
INSERT INTO SYSIBM.SYSTXTCONNECTINFO
(DB2HOSTNAME,DB2SERVICEPORT,DB2UID)
VALUES
(‘<mainframe name>’, ’446’, ‘<surrogate ID>’);
```
Encrypt Password for surrogate ID

- Use stored procedure defined earlier
- Run once for each server

```sql
UPDATE SYSIBMTS.SYSTEXTSERVERS
SET DB2ENCRYPTEDPW = SYSFUN.SYSTS_ENCRYPT('<password>', SERVERMASTERKEY)
WHERE SERVERNAME = '<server name>'
```
DB2 Configuration

- CALL SYSPROC.SYSTS_START();
  - Initial connection to server(s) and populates remaining columns in SYSTEXTSERVERS
  - Assigns SERVERID in SYSIBMTS.SYSTEXTSERVERS
Westfield’s Search Application - a twisted path

Topic 5
Search Application

- New Claims Management System
- Policy data replicated from Legacy system to DB2 z/OS tables
- Policy information needed for claims (coverages, limits, deductibles, etc.)
- Claimant may have only partial policy identification information when claim is initiated
- Need ability to search to find the correct policy to start a claim
Policy Search Requirements

• Policy data going back 7 years
• Search policies based on names and addresses as stored in the legacy system
• Search policy holder names or ‘other insured’ names (like drivers)
• Search business names
• Search any part of the address
• Anything can include a wildcard after 2 or 3 characters
• 22 million rows
Sample Search Criteria

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Name</td>
<td>______________</td>
</tr>
<tr>
<td>Last Name</td>
<td>ADD* __________</td>
</tr>
<tr>
<td>Business Name</td>
<td>___________</td>
</tr>
<tr>
<td>Street</td>
<td>___________</td>
</tr>
<tr>
<td>City</td>
<td>CLE* __________</td>
</tr>
<tr>
<td>State</td>
<td>___________</td>
</tr>
<tr>
<td>ZIP</td>
<td>___________</td>
</tr>
<tr>
<td>Agency</td>
<td>___________</td>
</tr>
<tr>
<td>POLICY_ID</td>
<td>Policy Document Data</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------</td>
</tr>
<tr>
<td>A001</td>
<td>Gomez and Morticia Addams 0001 Cemetery Lane Cleveland Oh 44251 Lurch Insurance Agency</td>
</tr>
<tr>
<td>A001</td>
<td>Pugsley Addams 0001 Cemetery Lane Cleveland Oh 44251 Lurch Insurance Agency</td>
</tr>
<tr>
<td>A001</td>
<td>Wednesday Addams 0001 Cemetery Lane Cleveland Oh 44251 Lurch Insurance Agency</td>
</tr>
<tr>
<td>A075</td>
<td>Robert McGee c/o Bobbys Plumbing LLC 215 Main St Salinas Ca Joplin Co Insurance</td>
</tr>
<tr>
<td>A567</td>
<td>County of Summit Board of Confusing Names 1777 Cleveland Ave Akron Oh Lurch Insurance Agency</td>
</tr>
<tr>
<td>A700</td>
<td>Stephen G Cleveland Whitehouse Pennsylvania Ave Washington DC Capitol Insurance Agency</td>
</tr>
<tr>
<td>A800</td>
<td>Andrew Johnson 1865 Woodshed Ave Columbia TN Lurch Insurance Agency</td>
</tr>
<tr>
<td>.......</td>
<td>&lt; 22 million more rows &gt;</td>
</tr>
</tbody>
</table>
Policy Search – Option 1

- Each searchable piece of data in its own column and lots of regular DB2 Indexes
  - First name, last name, city, state, zip, street name, address number, company name, agency name, etc.
  - Many indexes
<table>
<thead>
<tr>
<th>Field</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>POLICY_ID</td>
<td>CHAR(10)</td>
</tr>
<tr>
<td>FIRST_NAME</td>
<td>CHAR(30)</td>
</tr>
<tr>
<td>LAST_NAME</td>
<td>CHAR(30)</td>
</tr>
<tr>
<td>STREET</td>
<td>CHAR(30)</td>
</tr>
<tr>
<td>CITY</td>
<td>CHAR(20)</td>
</tr>
<tr>
<td>STATE</td>
<td>CHAR(2)</td>
</tr>
<tr>
<td>ZIP</td>
<td>CHAR(10)</td>
</tr>
<tr>
<td>BUSINESS_NAME</td>
<td>VARCHAR(250)</td>
</tr>
<tr>
<td>AGENCY_NAME</td>
<td>CHAR(100)</td>
</tr>
</tbody>
</table>

etc.

etc.

etc.
CREATE INDEX
    LAST_NAME, FIRST_NAME, STREET, CITY
CREATE INDEX
    LAST_NAME, STREET, CITY, FIRST_NAME
CREATE INDEX
    STREET, CITY, LAST_NAME, FIRST_NAME
CREATE INDEX
    CITY, LAST_NAME, STREET
Etc.
Etc.
Etc.

CREATE_INDEX
    BUSINESS_NAME
<table>
<thead>
<tr>
<th>POLICY_ID</th>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>STREET</th>
<th>CITY</th>
<th>BUSINESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A001</td>
<td>Addams</td>
<td>Gomez</td>
<td>Cemetery</td>
<td>Cleveland</td>
<td></td>
</tr>
<tr>
<td>A001</td>
<td>Addams</td>
<td>Morticia</td>
<td>Cemetery</td>
<td>Cleveland</td>
<td></td>
</tr>
<tr>
<td>A001</td>
<td>Addams</td>
<td>Pugsley</td>
<td>Cemetery</td>
<td>Cleveland</td>
<td></td>
</tr>
<tr>
<td>A001</td>
<td>Addams</td>
<td>Wednesday</td>
<td>Cemetery</td>
<td>Cleveland</td>
<td></td>
</tr>
<tr>
<td>A075</td>
<td>McGee</td>
<td>Robert</td>
<td>Main</td>
<td>Salinas</td>
<td>Robert McGee c/o Bobbys Plumbing....</td>
</tr>
<tr>
<td>A567</td>
<td></td>
<td></td>
<td>Cleveland</td>
<td>Akron</td>
<td>County of Summit Board of .....</td>
</tr>
<tr>
<td>A700</td>
<td>Cleveland</td>
<td>Stephen</td>
<td>Pennsyl...</td>
<td>Washing...</td>
<td></td>
</tr>
<tr>
<td>A800</td>
<td>Johnson</td>
<td>Andrew</td>
<td>Woodshed</td>
<td>Columbia</td>
<td></td>
</tr>
<tr>
<td><strong>First Name</strong></td>
<td>____________________</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Last Name</strong></td>
<td>JOHN*______________</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Business Name</strong></td>
<td>____________________</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Street</strong></td>
<td>WOOD*______________</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>City</strong></td>
<td>COL*______________</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>State</strong></td>
<td>____________________</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ZIP</strong></td>
<td>____________________</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Agency</strong></td>
<td>____________________</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SQL

```
SELECT * FROM POLICY_SEARCH
WHERE LAST_NAME LIKE 'JOHN%'
AND CITY LIKE 'COL%'
AND STREET LIKE 'WOOD%'
```

-----------------------------

STATS

```
<table>
<thead>
<tr>
<th>TABLE</th>
<th>ROWS</th>
</tr>
</thead>
<tbody>
<tr>
<td>POLICY_SEARCH</td>
<td>22 million</td>
</tr>
<tr>
<td>LAST_NAME</td>
<td>101K rows</td>
</tr>
<tr>
<td>CITY</td>
<td>555K rows</td>
</tr>
<tr>
<td>STREET</td>
<td>237K rows</td>
</tr>
</tbody>
</table>
```
• How would the DB2 optimizer choose to do this?
  • Multiple Indexes with huge RID lists?
  • Index scans?
  • Index filtering?
• BUT... that is not an unreasonable search request
  • 30 row result set
Nightmare #2

- Any search of company name
- Company Name on policy is not always the common name
- Policies are legal documents and include strange stuff
First Name
Last Name
Business Name  Bobbys Plumbing
Street
City
State
ZIP
Agency
<table>
<thead>
<tr>
<th>POLICY_ID</th>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>STREET</th>
<th>CITY</th>
<th>BUSINESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A001</td>
<td>Addams</td>
<td>Gomez</td>
<td>Cemetery</td>
<td>Cleveland</td>
<td></td>
</tr>
<tr>
<td>A001</td>
<td>Addams</td>
<td>Morticia</td>
<td>Cemetery</td>
<td>Cleveland</td>
<td></td>
</tr>
<tr>
<td>A001</td>
<td>Addams</td>
<td>Pugsley</td>
<td>Cemetery</td>
<td>Cleveland</td>
<td></td>
</tr>
<tr>
<td>A001</td>
<td>Addams</td>
<td>Wednes</td>
<td>Cemetery</td>
<td>Cleveland</td>
<td></td>
</tr>
<tr>
<td>A075</td>
<td>McGee</td>
<td>Robert</td>
<td>Main</td>
<td>Salinas</td>
<td>Robert McGee c/o</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bobbys Plumbing......</td>
</tr>
<tr>
<td>A567</td>
<td>Cleveland</td>
<td>Cleveland</td>
<td>Akron</td>
<td></td>
<td>County of Summit .....</td>
</tr>
<tr>
<td>A700</td>
<td>Cleveland</td>
<td>Stephen</td>
<td>Pennsyl...</td>
<td>Washing...</td>
<td></td>
</tr>
</tbody>
</table>
SQL:

SELECT * FROM POLICY_SEARCH
  WHERE COMPANY_NAME like '%BOBBYS PLUMBING%'

RESULT:

Index Scan of 22 million indexes entries
<table>
<thead>
<tr>
<th>POLICY_ID</th>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>STREET</th>
<th>CITY</th>
<th>BUSINESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A001</td>
<td>Addams</td>
<td>Gomez</td>
<td>Cemetery</td>
<td>Cleveland</td>
<td></td>
</tr>
<tr>
<td>A001</td>
<td>Addams</td>
<td>Morticia</td>
<td>Cemetery</td>
<td>Cleveland</td>
<td></td>
</tr>
<tr>
<td>A001</td>
<td>Addams</td>
<td>Pugsley</td>
<td>Cemetery</td>
<td>Cleveland</td>
<td></td>
</tr>
<tr>
<td>A001</td>
<td>Addams</td>
<td>Wednes</td>
<td>Cemetery</td>
<td>Cleveland</td>
<td></td>
</tr>
<tr>
<td>A075</td>
<td>McGee</td>
<td>Robert</td>
<td>Main</td>
<td>Salinas</td>
<td>Robert McGee c/o Bobbys Plumbing</td>
</tr>
<tr>
<td>A567</td>
<td>Cleveland</td>
<td></td>
<td></td>
<td>Akron</td>
<td>County of Summit</td>
</tr>
<tr>
<td>A700</td>
<td>Cleveland</td>
<td>Stephen</td>
<td>Pennsylv...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SELECT * FROM POLICY_SEARCH WHERE COMPANY_NAME like '%SUMMIT COUNTY BOARD%'

RESULT:

Index Scan of 22 million indexes entries

And no rows returned
Policy Search – Option 1

• Business view of searching <> DBA view
• Need lots of rules on search criteria entered by users
• Poor performance for some searches
• Policies not found for some searches
Policy Search – Option 2

• DB2 Text Search
  • Heard of it, but can’t find anyone using it
  • Skeptical
    • Does it work?
    • Is it for document searches only?
    • How does an application use it?
    • Response time for large searches?
    • Availability?
    • Reliability?
    • Data updates?
    • Administration?
CREATE TABLE POLICY_SEARCH
    POLICY_ID CHAR(10)
    ...
    SEARCH_TEXT VARCHAR(2000)
    ...
    ROWID_COL ROWID
<table>
<thead>
<tr>
<th>POLICY_ID</th>
<th>SEARCH_TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A001</td>
<td>Gomez and Morticia Addams 0001 Cemetery Lane Cleveland Oh 44251 Lurch Insurance Agency</td>
</tr>
<tr>
<td>A001</td>
<td>Pugsley Addams 0001 Cemetery Lane Cleveland Oh 44251 Lurch Insurance Agency</td>
</tr>
<tr>
<td>A001</td>
<td>Wednesday Addams 0001 Cemetery Lane Cleveland Oh 44251 Lurch Insurance Agency</td>
</tr>
<tr>
<td>A075</td>
<td>Robert McGee c/o Bobbys Plumbing LLC 215 Main St Salinas Ca Joplin Co Insurance</td>
</tr>
<tr>
<td>A567</td>
<td>County of Summit Board of Confusing Names 1777 Cleveland Ave Akron Oh Lurch Insurance Agency</td>
</tr>
<tr>
<td>A700</td>
<td>Stephen G Cleveland Whitehouse Pennsylvania Ave Washington DC Capitol Insurance Agency</td>
</tr>
<tr>
<td>A800</td>
<td>Andrew Johnson 1865 Woodshed Ave Columbia TN Lurch Insurance Agency</td>
</tr>
<tr>
<td>.......</td>
<td>&lt; 22 million more rows &gt;</td>
</tr>
<tr>
<td>POLICY_ID</td>
<td>SEARCH_TEXT</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>A001</td>
<td>Gomez Morticia <strong>Addams 0001 Cemetery</strong> Lane Cleveland Oh 44251 Lurch Insurance Agency</td>
</tr>
<tr>
<td>A001</td>
<td>Pugsley <strong>Addams 0001 Cemetery</strong> Lane Cleveland Oh 44251 Lurch Insurance Agency</td>
</tr>
<tr>
<td>A001</td>
<td>Wednesday <strong>Addams 0001 Cemetery</strong> Lane Cleveland Oh 44251 Lurch Insurance Agency</td>
</tr>
</tbody>
</table>
All these run in < 0.25 second

```sql
SELECT *
FROM POLICY_SEARCH
WHERE CONTAINS(SEARCH_TEXT,
    'JOHN* WOOD* COL*'
    'RESULTLIMIT = 100' ) = 1

SELECT *
FROM POLICY_SEARCH
WHERE CONTAINS(SEARCH_TEXT,
    'BOBBYS PLUMBING'
    'RESULTLIMIT = 100' ) = 1

SELECT *
FROM POLICY_SEARCH
WHERE CONTAINS(SEARCH_TEXT,
    'SUMMIT COUNTY BOARD'
    'RESULTLIMIT = 100' ) = 1
```
Define Searchable Columns

Topic 6
Security Requirements

• Text Search Administration (Create, Update, Drop, Restore)
  • DBADM on database SYSIBMTS
  • SEL,INS,UPD,DEL on all tables in SYSIBMTA
  • DBADM on the database with the table containing the text
  • EXECUTE on SYSIBMTS.* Packages
• Connect-Back ID
  • SEL,INS,UPD,DEL on all tables in SYSIBMTA & SYSIBMTS
TIP

• If you use non-native DB2 security (ACF2), execute a native GRANT DBADM ON DATABASE <db with Search Text Table> TO <administrator-id>

• Text Search stored procedures query the DB2 catalog to see if the userid has native DBADM
Define Searchable Columns

Example:

Table **MYSCHEMA.POLICY_SEARCH**

- **POLICY_ID** CHAR(10)
- **SEARCH_TEXT** VARCHAR(2000)
- **ROWID_COL** ROWID

- Column **SEARCH_TEXT** contains policy holder names, addresses, additional insured names, organization names, policy number, etc

- The table must contain a ROWID column, and there must be an index on ROWID
Define the Text Search Index

CALL SYSPROC.SYSTS_CREATE
    (‘MYSCHEMA’, 'POLICY_SEARCH_TX1',
     ‘MYSCHEMA.POLICY_SEARCH(SEARCH_TEXT)',
     'INDEX CONFIGURATION(SERVER 21)');
The Stored Procedure does this:

Inserts into **SYSIBM.SYSTEINDEXES**

<table>
<thead>
<tr>
<th>INDEXID</th>
<th>INDEXSCHEMA</th>
<th>INDEXNAME</th>
<th>COLLECTIONNAME</th>
<th>SERVERID</th>
</tr>
</thead>
<tbody>
<tr>
<td>141</td>
<td>schema1</td>
<td>POL_BAS_SRCH_TX1</td>
<td>MVSDB2T_141_2014_02_</td>
<td>21</td>
</tr>
<tr>
<td>101</td>
<td>schema1</td>
<td>POL_BAS_SRCH2_TX1</td>
<td>MVSDB2T_101_2013_10_</td>
<td>1</td>
</tr>
<tr>
<td>122</td>
<td>MYSHEMA</td>
<td>POLICY_SEARCH_TX1</td>
<td>MVSDB2T_122_2013_12_</td>
<td>21</td>
</tr>
<tr>
<td>103</td>
<td>Schema2</td>
<td>POL_BAS_SRCH3_TX1</td>
<td>MVSDB2T_103_2013_11_</td>
<td>21</td>
</tr>
</tbody>
</table>
Call the stored procedure now and you’ll also get ...

These *NEW TABLES*

- `SYSIBM.EVENTS_122` (event log)
- `SYSIBM.INDEX_122` (text ‘index’ BLOB)
- `SYSIBM.STAGING_122` (ROWIDS with changed data)

Plus *NEW ROWS* in these tables

- `SYSIBM.SYSTEXTCOLUMNS`
- `SYSIBM.SYSTEXTCONFIGURATION`
- `SYSIBM.SYSTEXTLOCKS`
- `SYSIBM.SYSTEXTSERVERHISTORY`
But wait ... there's more

CREATE TRIGGER
  DSNIBMTS.ISTAGING_122
  AFTER INSERT
ON MYSCHEMA.POLICY_SEARCH
  REFERENCING
  NEW AS N
  FOR EACH ROW
  MODE DB2SQL
  BEGIN ATOMIC INSERT INTO SYSIBMTS.STAGING_122 (OPERATION, SEQID, RID) VALUES ('I', GENERATE_UNIQUE (), CAST (N."ROWID_COL" AS CHAR (40) FOR BIT DATA) ) ; END

AND ...
  CREATE TRIGGER AFTER UPDATE ...

AND ...
  CREATE TRIGGER AFTER DELETE ...
Search index initial build and update

Topic 7
INITIAL LOAD and TEXT INDEX CREATION

1. LOAD data into POLICY_SEARCH

2. CALL SYSPROC.SYSTS_UPDATE
   (‘MYSCHEMA’, ‘POLICY_SEARCH_TX1’, ‘’)


Initial build of the Text Search “index”

- What SYSPROC.SYSTS_UPDATE does:
  1. Fetches a row from POLICY_SEARCH
  2. Sends it to the ‘owning’ linux server
  3. Adds to the index structure in the server collection file
  4. Repeats 1-3 until all rows are fetched
  5. Sends the file back to DB2 a chunk at a time
  6. DB2 inserts into the BLOB in SYSIBMTS.INDEX_[n]
CALL SYSPROC. SYSTS_UPDATE (<parms>)

DB2 z/OS

Text Search Admin DB

Collection file
master table

Search Table

Collection file
BLOB

ROWID, search text

Collection file

Text search
Collection file

Linux/Windows server

ROWID, search text
TIP

The stored procedure contains this SQL (in DB2 V10):

```
SELECT U. "ROWID_COL" ,
'I' ,
U. "SEARCH_TEXT"
FROM "MYSCHEMA"."POLICY_SEARCH" U
ORDER BY U. "ROWID_COL" ASC
SKIP LOCKED DATA
FOR READ ONLY
```

This may be a huge amount of data to sort, and may result in -904 Abends for sort file space in DSNDB07

*** Define table as VOLITILE and do not execute RUNSTATS
(will use the index on ROWID_COL and avoid the sort)
Build the Text Search Index

• Calling the stored procedure can take a long time, depending on the amount of data in the source table
  • This runs 6-10 hours for 22 million rows in POLICY_SEARCH
• A native call to a stored procedure ties up your session until it is complete
  • Cannot end your session and go home
  • So ...
Build Text Search Index

• Use home-grown REXX exec for updating indexes
  • Batch job can run unattended
  • Therefore can be scheduled
  • Provides additional diagnostics
Subsequent data updates to the DB2 Search Table

• Text Search indexes are NOT updated automatically at SQL INSERT, UPDATE, and DELETE like regular DB2 Indexes
• Inserts, updates, and deletes are staged via the triggers and staging table
• Searches are operational while inserts, updates and deletes are pending
  • But be aware of what will happen....
SELECT ...
FROM ...
WHERE CONTAINS(srch_col, search terms) = 1

DB2 z/OS

Text Search Admin DB

Search terms

List of ROWID's

Search terms

Search Table

Collection file
master table

ROWID, data

Linux/Windows server

Text search Collection file
Staged Deletes

- The delete has already taken place in the DB2 table
- The Text Search server may return the ROWID of the deleted row
- That ROWID will not be found, but the CONTAINS function does not care
  - Returns the rows of other ROWID’s
- The delete ‘happens’ immediately to Text Search
Staged Inserts

• The insert has already taken place in the DB2 table (new ROWID)
• The Text Search file does not have the ROWID, so will not return it in any search
• Inserts do not ‘happen’ until the Text Search Index is updated
Staged Updates of the Search column

- The update has already occurred in the DB2 table
- The Text Search file does not have the updates, so will return ROWID’s based on the original values
- Inaccurate or misleading results until the Text Search Index update occurs
Incremental Index Updates

• CALL SYSPROC.SYSTS_UPDATE('MYSCHEMA', 'POLICY_SEARCH_TX1', ' ');

• What is it doing:
  1. Fetches a row from SYSIBMTS.STAGING_[n]
  2. Sends it to the ‘owning’ linux server
  3. Alters the index structure in the linux file
  4. Repeats 1-3 until all rows are fetched
  5. Sends the file back to DB2 a chunk at a time
  6. DB2 inserts into the BLOB in SYSIBMTS.INDEX_[n]
CALL SYSPROC. SYSTS_UPDATE (<parms>)

Collection file master table

Search Table

Text Search Admin DB

DB2 z/OS

ROWID, search text

Collection file BLOB

ROWID, search text

Text search Collection file

Linux/Windows server
Incremental Index Updates

- The Text Search Index experiences no outage during the incremental update
- SQL using the CONTAINS function are unaffected
- Can also set up automated incremental updates (see manual)
Production ‘Hybrid’ Implementation

Topic 8
Text Search Challenge for Policy Search

- Text search works great for searching for words within text
  - All words are equal
- For normal document searches, that is all we need
- For Westfield policy searches there is a real problem with context
The Grover Cleveland Dilemma

- I want to search for Grover Cleveland’s policy
- Grover is not his first name and is not on the document
- Cleveland is not a common last name, so this should be a good search
<table>
<thead>
<tr>
<th>POLICY_ID</th>
<th>SEARCH_TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A001</td>
<td>Gomez Morticia Addams 0001 Cemetery Lane Cleveland Oh 44251 Lurch Insurance Agency</td>
</tr>
<tr>
<td>A001</td>
<td>Pugsley Addams 0001 Cemetery Lane Cleveland Oh 44251 Lurch Insurance Agency</td>
</tr>
<tr>
<td>A001</td>
<td>Wednesday Addams 0001 Cemetery Lane Cleveland Oh 44251 Lurch Insurance Agency</td>
</tr>
<tr>
<td>A075</td>
<td>Robert McGee c/o Bobbys Plumbing LLC 215 Main St Salinas Ca Joplin Co Insurance</td>
</tr>
<tr>
<td>A567</td>
<td>County of Summit Board of Confusing Names 1777 Cleveland Ave Akron Oh Lurch Insurance Agency</td>
</tr>
<tr>
<td>A700</td>
<td>Stephen G Cleveland Whitehouse Pennsylvania Ave Washington DC Capitol Insurance Agency</td>
</tr>
<tr>
<td>A800</td>
<td>Andrew Johnson 1865 Woodshed Ave Columbia TN Lurch Insurance Agency</td>
</tr>
<tr>
<td>.......</td>
<td>&lt; 22 million MORE rows &gt;</td>
</tr>
</tbody>
</table>
SELECT *
FROM POLICY_SEARCH
WHERE CONTAINS(SEARCH_TEXT, 'CLEVELAND', 'RESULTLIMIT = 5000') = 1
<table>
<thead>
<tr>
<th>POLICY_ID</th>
<th>SEARCH_TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A001</td>
<td>Gomez Morticia Addams 0001 Cemetery Lane Cleveland Oh 44251 Lurch Insurance Agency</td>
</tr>
<tr>
<td>A001</td>
<td>Pugsley Addams 0001 Cemetery Lane Cleveland Oh 44251 Lurch Insurance Agency</td>
</tr>
<tr>
<td>A001</td>
<td>Wednesday Addams 0001 Cemetery Lane Cleveland Oh 44251 Lurch Insurance Agency</td>
</tr>
<tr>
<td>A075</td>
<td>Robert McGee c/o Bobbys Plumbing LLC 215 Main St Salinas Ca Joplin Co Insurance</td>
</tr>
<tr>
<td>A567</td>
<td>County of Summit Board of Confusing Names 1777 Cleveland Ave Akron Oh Lurch Insurance Agency</td>
</tr>
<tr>
<td>A700</td>
<td>Stephen G Cleveland Whitehouse Pennsylvania Ave Washington DC Capitol Insurance Agency</td>
</tr>
<tr>
<td>A800</td>
<td>Andrew Johnson 1865 Woodshed Ave Columbia TN Lurch Insurance Agency</td>
</tr>
<tr>
<td>..........</td>
<td>&lt; 22 million MORE rows &gt;</td>
</tr>
</tbody>
</table>
• All text is searched equally
• I get flooded with policies from the city of Cleveland, from Cleveland Ave, businesses with Cleveland in the name, etc
• False hits
• I can’t find the policy I asked for
• This is an extreme example, but very prevalent
• I want the best of both options:
  • The speed and flexibility of Text Search
  • The context provided with standard DB2 indexes on separate columns
• Three major approaches to resolve were tested
  • Nice improvements
  • But not good enough
Approach #4 - Tags

• Add 2 char prefix to all words
  • cx (city)
  • lx (last name)
  • fx (first name)
  • ax (address – all parts)
  • Etc
• Search now has context
• Build separate ‘indexes’ within the text
<table>
<thead>
<tr>
<th>POLICY_ID</th>
<th>SEARCH_TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A001</td>
<td>fxGomez</td>
</tr>
<tr>
<td></td>
<td>fxMorticia</td>
</tr>
<tr>
<td></td>
<td>lxAddams</td>
</tr>
<tr>
<td></td>
<td>ax0001</td>
</tr>
<tr>
<td></td>
<td>axCemetery</td>
</tr>
<tr>
<td></td>
<td>axLane</td>
</tr>
<tr>
<td></td>
<td><strong>cxCleveland</strong></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>A001</td>
<td>fxPugsley</td>
</tr>
<tr>
<td></td>
<td>lxAddams</td>
</tr>
<tr>
<td></td>
<td>ax0001</td>
</tr>
<tr>
<td></td>
<td>axCemetery</td>
</tr>
<tr>
<td></td>
<td>axLane</td>
</tr>
<tr>
<td></td>
<td><strong>cxCleveland</strong></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>A001</td>
<td>fxWednesday</td>
</tr>
<tr>
<td></td>
<td>lxAddams</td>
</tr>
<tr>
<td></td>
<td><strong>cxCleveland</strong></td>
</tr>
<tr>
<td>A075</td>
<td>....</td>
</tr>
<tr>
<td></td>
<td>oxBobbys</td>
</tr>
<tr>
<td></td>
<td>oxPlumbing</td>
</tr>
<tr>
<td></td>
<td>oxLLC</td>
</tr>
<tr>
<td>A567</td>
<td>oxCounty</td>
</tr>
<tr>
<td></td>
<td>oxof</td>
</tr>
<tr>
<td></td>
<td>oxSummit</td>
</tr>
<tr>
<td></td>
<td>oxBoard</td>
</tr>
<tr>
<td></td>
<td>ax1777</td>
</tr>
<tr>
<td></td>
<td><strong>cxCleveland</strong></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>A700</td>
<td>fxStephen</td>
</tr>
<tr>
<td></td>
<td>mxG</td>
</tr>
<tr>
<td></td>
<td><strong>lxCleveland</strong></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>A800</td>
<td>fxAndrew</td>
</tr>
<tr>
<td></td>
<td>lxJohnson</td>
</tr>
<tr>
<td></td>
<td>ax1865</td>
</tr>
<tr>
<td></td>
<td>axWoodshed</td>
</tr>
<tr>
<td>..........</td>
<td>&lt; 22 million MORE rows &gt;</td>
</tr>
</tbody>
</table>
First Name
Last Name  CLEVELAND
Business Name
Street
City
State
ZIP
Agency

SELECT *
FROM POLICY_SEARCH
WHERE CONTAINS(SEARCH_TEXT,
   'lxCLEVELAND',
   'RESULTLIMIT = 100' ) = 1
<table>
<thead>
<tr>
<th>POLICY_ID</th>
<th>SEARCH_TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A700</td>
<td>fxStephen  mxG  lxCleveland  axWhitehouse  axPennsylvania  axAve  cxWashington  cxDC ...</td>
</tr>
</tbody>
</table>
Policy Search Misc Info

• Westfield Policy Search SQL has 13 tables joined along with the CONTAINS, including table expressions
• Explain the SQL and validate that search is first (not inside a loop)
• Use RESULTLIMIT (5000 for Policy Search) to limit output and govern performance
• Only a few Westfield Policy Search queries a day run over 1 second elapsed time – normally very large policies or ambiguous search terms
Future testing at Westfield

- XML searches
- Synonyms
  - XML File on the Linux/Windows server
Text Search
Administration –
DB2 z/OS

Topic 9
Restore Search Index

Copies search file from SYSIBMTS.INDEX_[n] on DB2 to the linux host server collection file. The collection on the original server is not deleted. This has three purposes:

1) After a DB2 recover of both the source table and the associated SYSIBMTS.INDEX* table.
2) To move the hosting from one linux server to another. Search is fully functional during this process.
3) Recover a corrupted or lost collection file on the linux server

CALL SYSPROC.SYSTS_RESTORE('<TS schema>', '<TS Name>', '<Server ID>');</p>

EXAMPLE:

CALL SYSPROC.SYSTS_RESTORE('MYSCHEMA', 'POLICY_SEARCH_TX1', '2');
CALL SYSPROC.
SYSTS_RESTORE (<parms>)

DB2 z/OS

Collection file
master table

Search Table

Copy of Collection File

Collection file BLOB

New Server

Original Server

normal searches

Text search Collection file

Text Search Admin DB
Drop Search Index

Delete associated rows from SYSIBMTS.* tables
Drop associated SYSIBMTS.* tables
Drop triggers from source table
Delete collection file from current linux server.

*** Does not delete from non-current server(s) if any.

CALL SYSPROC.SYSTS_DROP('"<TS schema>"', '"<TS name>"');

EXAMPLE:

CALL SYSPROC.SYSTS_DROP('MYSCHEMA', 'POLICY_SEARCH_TX1');
Takeover

Similar to RESTORE, but DB2 will find the server to move to automatically. There are circumstances where this occurs automatically, for example when the host server is not available during UPDATE.

CALL SYSPROC.SYSTS_TAKEOVER(‘<TS server>', ‘<TS name>');</n
EXAMPLE

CALL SYSPROC.SYSTS_TAKEOVER(‘MYSCHEMA', ‘POLICY_SEARCH_TX1');
Stop and Start

These stored procedures have no parms, so can also be run in SPUFI, DSNTEP*, etc.

The START is required after a new linux/windows server is added.

CALL SYSPROC.SYSTS_STOP();

CALL SYSPROC.SYSTS_START();
Text Search Administration - Linux
Administration - Linux

- Check for free space
  - Collection files can be large
- Check status
- Delete unnecessary collections
- Start and stop Text Search processes
- Refer to manual for more options and details
TIP

• To avoid the requirement to be ROOT
give user r-x to:
  ../config/authentication.xml
  ../config/key.txt
# Check Dasd Utilization

```bash
df -h
```

<table>
<thead>
<tr>
<th>Filesystem</th>
<th>Size</th>
<th>Used</th>
<th>Avail</th>
<th>Use%</th>
<th>Mounted on</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/dasdc1</td>
<td>504M</td>
<td>258M</td>
<td>221M</td>
<td>54%</td>
<td>/</td>
</tr>
<tr>
<td>tmpfs</td>
<td>3.9G</td>
<td>0</td>
<td>3.9G</td>
<td>0%</td>
<td>/dev/shm</td>
</tr>
<tr>
<td>/dev/mapper/system_vg-home_lv</td>
<td>1008M</td>
<td>34M</td>
<td>924M</td>
<td>4%</td>
<td>/home</td>
</tr>
<tr>
<td>/dev/mapper/system_vg-opt_lv</td>
<td>6.0G</td>
<td>555M</td>
<td>5.1G</td>
<td>10%</td>
<td>/opt</td>
</tr>
<tr>
<td>/dev/mapper/system_vg-tmp_lv</td>
<td>3.0G</td>
<td>68M</td>
<td>2.8G</td>
<td>3%</td>
<td>/tmp</td>
</tr>
<tr>
<td>/dev/mapper/system_vg-usr_lv</td>
<td>6.0G</td>
<td>1.7G</td>
<td>4.0G</td>
<td>30%</td>
<td>/usr</td>
</tr>
<tr>
<td>/dev/mapper/system_vg-var_lv</td>
<td>2.0G</td>
<td>227M</td>
<td>1.7G</td>
<td>12%</td>
<td>/var</td>
</tr>
<tr>
<td>/dev/mapper/system_vg-varlog_lv</td>
<td>3.0G</td>
<td>183M</td>
<td>2.7G</td>
<td>7%</td>
<td>/var/log</td>
</tr>
<tr>
<td>/dev/mapper/app_vg-app_lv</td>
<td>75G</td>
<td>12G</td>
<td>60G</td>
<td>16%</td>
<td>/opt/IBM/ECMTextSearch</td>
</tr>
</tbody>
</table>
Check Status

```
sh adminTool.sh status -configPath ../config
```

<table>
<thead>
<tr>
<th>CollectionName</th>
<th>IndexSize</th>
<th>NumOfDocuments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>18,069B</td>
<td>0</td>
</tr>
<tr>
<td>MVSDB2T_103_2013_11_01_10_28_53_790971</td>
<td>4,128.061M</td>
<td>21,143,757</td>
</tr>
<tr>
<td>MVSDB2S_101_2014_01_27_14_29_23_687286</td>
<td>316,405.345K</td>
<td>1,862,444</td>
</tr>
<tr>
<td>MVSDB2T_141_2014_02_10_16_05_53_622496</td>
<td>70,737.354K</td>
<td>351,263</td>
</tr>
<tr>
<td>MVSDB2S_83_2013_12_23_13_38_32_679888</td>
<td>344,130.403K</td>
<td>1,806,168</td>
</tr>
<tr>
<td>MVSDB2S_82_2013_12_20_16_48_57_446100</td>
<td>366,729B</td>
<td>2,136</td>
</tr>
<tr>
<td>MVSDB2T_122_2013_12_05_16_46_53_159071</td>
<td>4,247.799M</td>
<td>21,179,586</td>
</tr>
<tr>
<td>MVSDB2S_23_2013_01_03_10_42_43_451278</td>
<td>273,449.291K</td>
<td>1,862,444</td>
</tr>
<tr>
<td>MVSDB2S_42_2013_01_10_11_18_29_609767</td>
<td>1,580.502M</td>
<td>9,522,998</td>
</tr>
</tbody>
</table>
Delete Collections

sh adminTool.sh delete -configPath ../config -collectionName MVSDB2T_103_2013_11_01_10_28_53_790971

But if that fails (IQQD0060 message) ...

cd ../config/collections

ls -l

drwxr-xr-x 3 root root 4096 Jan 10 07:47 Base
drwxr-xr-x 3 root root 4096 Jan 27 14:30 MVSDB2S_101_2014_01_27_14_29_23_687286
drwxr-xr-x 3 root root 4096 Jan 27 13:43 MVSDB2S_23_2013_01_03_10_42_43_451278
drwxr-xr-x 3 root root 4096 Jan 27 13:45 MVSDB2S_42_2013_01_10_11_18_29_609767

drwxr-xr-x 3 root root 4096 Jan 24 10:35 MVSDB2S_82_2013_12_20_16_48_57_446100
drwxr-xr-x 3 root root 4096 Jan 24 11:19 MVSDB2S_83_2013_12_23_13_38_32_679888

drwxr-xr-x 3 root root 4096 Jan 27 17:43 MVSDB2T_103_2013_11_01_10_28_53_790971

drwxr-xr-x 3 root root 4096 Jan 27 17:18 MVSDB2T_122_2013_12_05_16_46_53_159071

drwxr-xr-x 3 root root 4096 Feb 10 16:06 MVSDB2T_141_2014_02_10_16_05_53_622496

rm -rf MVSDB2T_103_2013_11_01_10_28_53_790971
Start and Stop Text Search

sh startup.sh

sh shutdown.sh