What's new in Eloquence B.08.20 - Part 1 -

Welcome

Welcome to the Eloquence Webinar series

- Start with two webinars on the upcoming Eloquence B.08.20
  - Database (June 11)
  - Language and tools (June 25)

- After the summer holiday season we will give periodic webinars that highlight specific topics in depth

Eloquence releases

- Eloquence B.07.00
  - Regular support ended in February 2008
  - Itanium support, image3k support, forward logging and recovery
- Eloquence B.07.10
  - Regular support ended in December 2010
  - Database auditing, client side caching, indexing enhancements, image3k enhancements, query3k, bulk import
- Eloquence B.08.00
  - Regular support ends November 2013
  - Improved scalability, 64 bit support, replication, indexing enhancements, incremental recovery, new JDLG release, installation procedure
- Eloquence B.08.10
  - Regular support ends June 2015
  - Database encryption, Item masking

Eloquence B.08.20 status

- Eloquence B.08.20 development is almost completed
  - Currently in beta test, beta 2 release by June 15
  - Production release scheduled for July/August 2012

- Supported platforms (32/64 bit) include
  - HP-UX Itanium and PA-RISC
  - Linux x86, x64 and Itanium
  - Windows x86 and x64

- Installed in a separate location and may be used concurrently with previous Eloquence versions
- Fully backward/forward compatible (as long as new functions are not used)

New functionality

Eloquence B.08.20 offers new functionality and substantial enhancements for various components, including

- Database full text search functionality
- Major language enhancements
- PCL to PDF conversion
- Improved WebDLG
- Improved JDLG

FTS Overview

- What is FTS
- What is it good for
- How does it work
- How to make use of it

What is FTS

- FTS (“Full Text Search”) is a new indexing subsystem for the Eloquence database
- Independent of search items or index items

- FTS allows to search the database by words, partial words, ranges or combination of words
- FTS adds the equivalent of a search engine to the database

- Easy to integrate in existing applications
- Fast search times, limited overhead
- Basic functionality included in the base product
- Extended FTS functionality requires additional license option

What is FTS
FTS vs. search and index items

eloquence

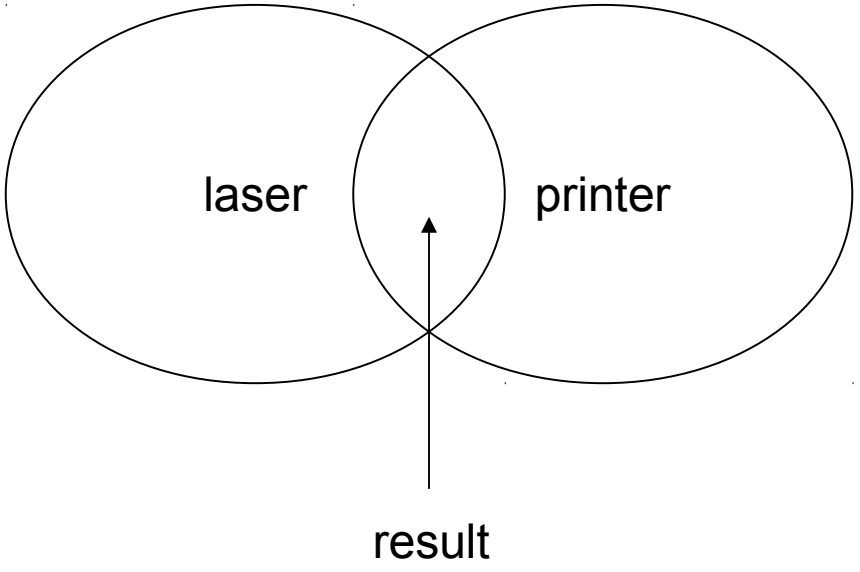
- Search items allow obtaining record with unique key
  - CUST-NO = 12345
- Index items allow records to be ordered and obtaining records by partial key
  - MATCHCODE=TEST@
  - DATE >= 2012-06-11
- FTS indexes are based in the set theory and operate on subsets or intersections of subsets that satisfy the search criteria
  - Customers with name “Software” in “Wuppertal”
  - Orders with status shipped

What is FTS
Subsets and intersections

eloquence

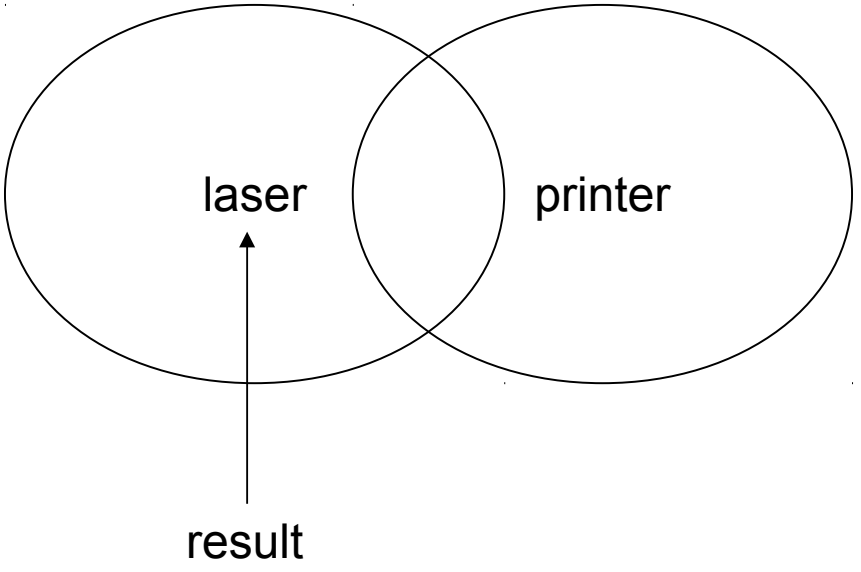
Laser and printer

1. locate list of “laser” references
2. locate list of “printer” references
3. intersect results

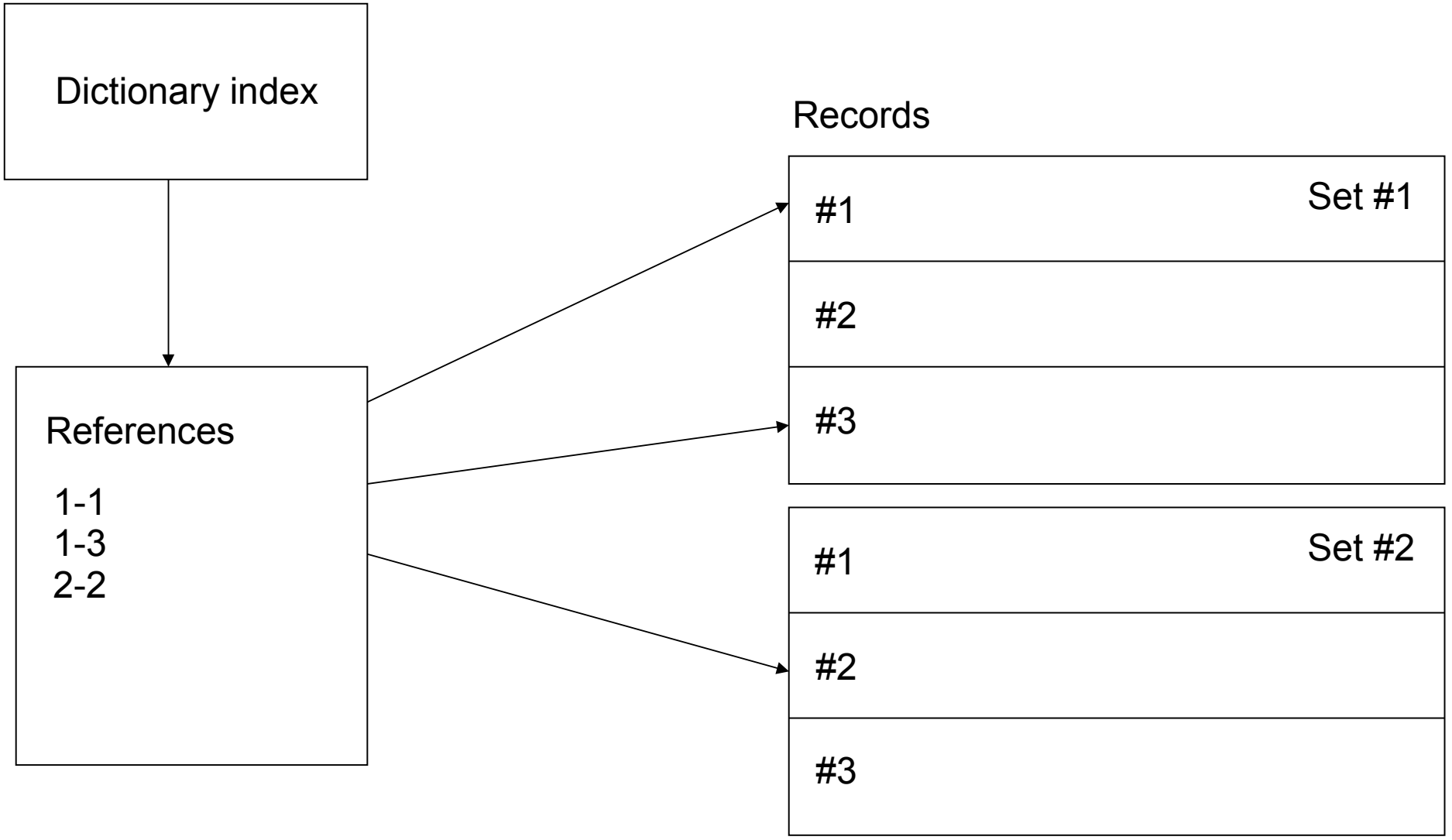


Laser and NOT printer

1. locate list of “laser” references
2. locate list of “printer” references
3. intersect results



What is FTS


Dictionary index
References
1-1
1-3
2-2
Records
Set #1
#1
#2
#3
Set #2
#1
#2
#3

What is FTS

- Any item may be marked for FTS indexing
- A separate name may be specified for the FTS index to distinguish it from search items or index items
- Text fields are parsed, words are extracted from text fields and are indexed individually
- FTS fields may be placed in a group to allow searches spanning a multiple items
  - Customer Name, Address, City
- FTS indexes may be setup to reference individual records, associated master entries or both

What is FTS

- DBFIND is used to search FTS index, DBGET is used to fetch records from result set
- A single FTS result set is maintained per database open
- Previous FTS results may be refined
  - search for “laser and printer”
  - then search for specific price range
- Previous FTS results may be applied to another set
  - Qualify customers by region
  - Apply selection to invoice data set

What is FTS

- Search for (partial) words, numbers, ranges or boolean expressions
  - keywords (“word”)
  - partial words (“word@”)
  - ranges (A@:C@)
  - boolean expression (laser and not printer)
  - relational operator (>=42)

What is FTS

- FTS indexes are specified by dbutil and maintained by the database server
- Integrated in the database kernel and compatible with all Eloquence database features
  - replication
  - network transparency
  - database encryption
  - backup and restore

What is it good for

Example1 – sampledb
Enhance customer search using FTS

eloquence

- Previously, the application supported looking up customers using customer number or customer matchcode

- Add FTS index and add all interesting customer fields in a group
- Code does not need to be changed but is more flexible
- The previous index item on matchcode is no longer needed

Original database structure

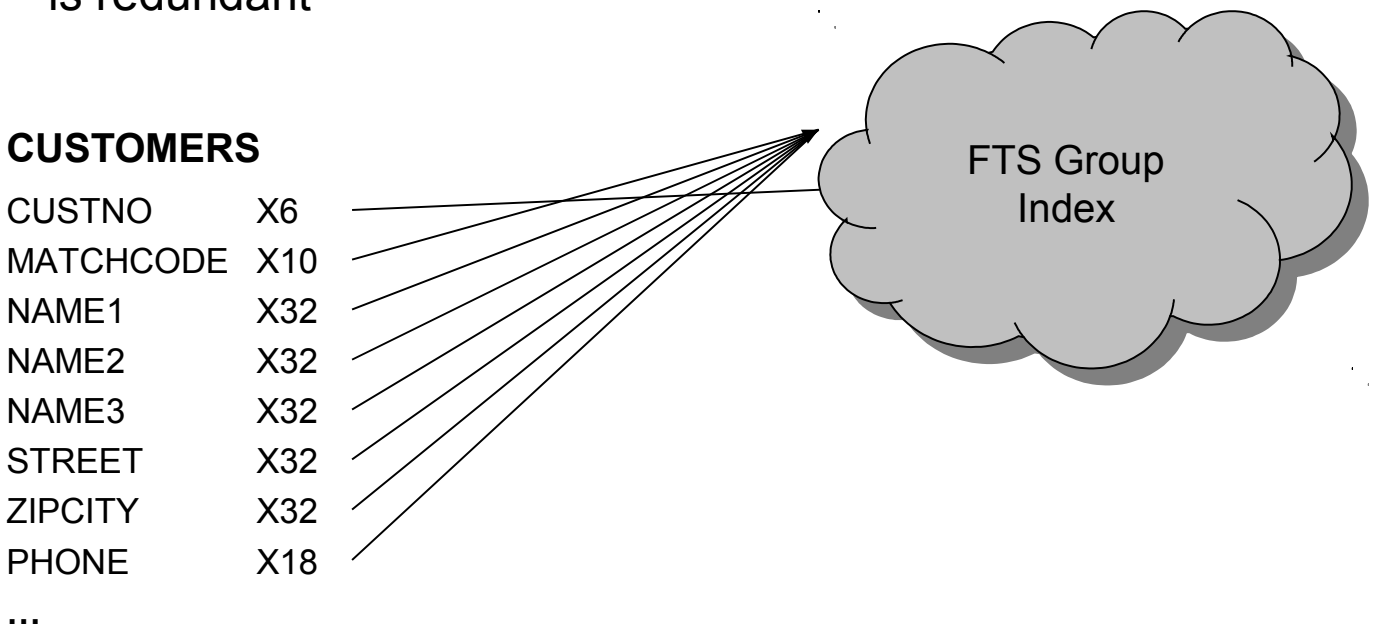
CUSTOMERS

CUSTNO	X6	(Search Item)
MATCHCODE	X10	(Index Item)
NAME1	X32	
NAME2	X32	
NAME3	X32	
STREET	X32	
ZIPCITY	X32	
PHONE	X18	
...		

DBFIND mode 1 on MATCHCODE
DBGET mode 5

New database structure

- Add FTS fields in a group
- No code changes
- The index item on matchcode
  is redundant



DBFIND mode 1 on MATCHCODE
DBGET mode 5

Demo

- Customer search
- Customer search with multiple fields
- Customer search text version

dbutil script

```
database "sample";
create fts index CUSTNO-FTS in CUSTOMERS on CUSTNO
 option GROUP=1;
create fts index in CUSTOMERS on MATCHCODE option GROUP=1;
create fts index in CUSTOMERS on NAME1 option GROUP=1;
create fts index in CUSTOMERS on NAME2 option GROUP=1;
create fts index in CUSTOMERS on NAME3 option GROUP=1;
create fts index in CUSTOMERS on STREET option GROUP=1;
create fts index in CUSTOMERS on ZIPCITY option GROUP=1;
create fts index in CUSTOMERS on PHONE option GROUP=1;
```

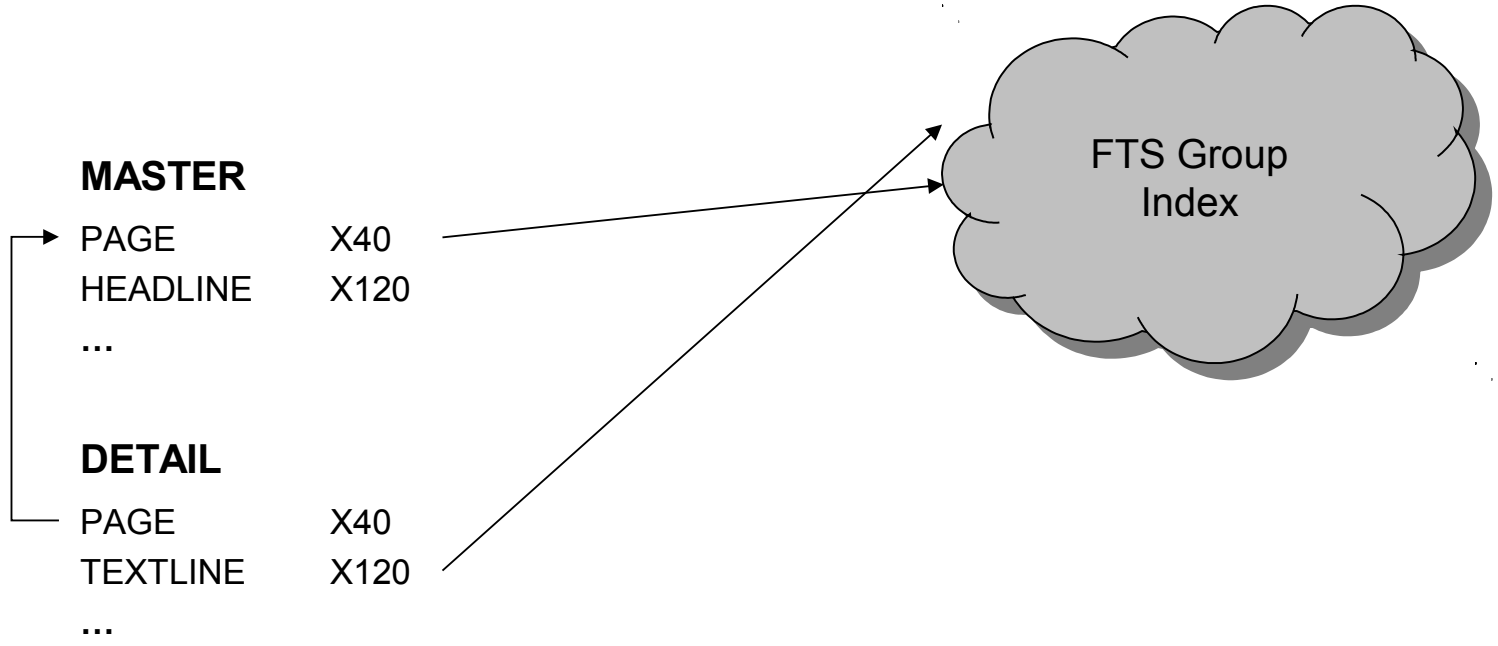
Example2 – mandb
Full text search

eloquence

- Linux man pages were put in a database
  - Master = man page title
  - Detail = man page text lines
- FTS index on text fields
- Allows search on arbitrary words
- Demonstrates advanced feature using aggregated FTS index

Database structure

FTS index is aggregated by master record



To locate qualifying man pages by keywords:

1. DBFIND on master, mode 1
   qualifies master entries based detail content
2. DBGET mode 5

To display a man page:

1. DBFIND on detail, mode 1 on PAGE
2. DBGET mode 5

mandb example

- The mandb example database holds a subset of the Linux man pages

```
SET NAME                 RECLEN CAPACITY ENTRIES
---------------- --- - ------ -------- --------
HEADINGS         001 M    180     7119     6917
CONTENTS         002 D    180  1199582  1199414
```

- The master “HEADINGS” holds the header line, one entry for each man page
- Actual text is added to the “CONTENTS” data set, each entry reflects a text line

Demo

- Mandb search

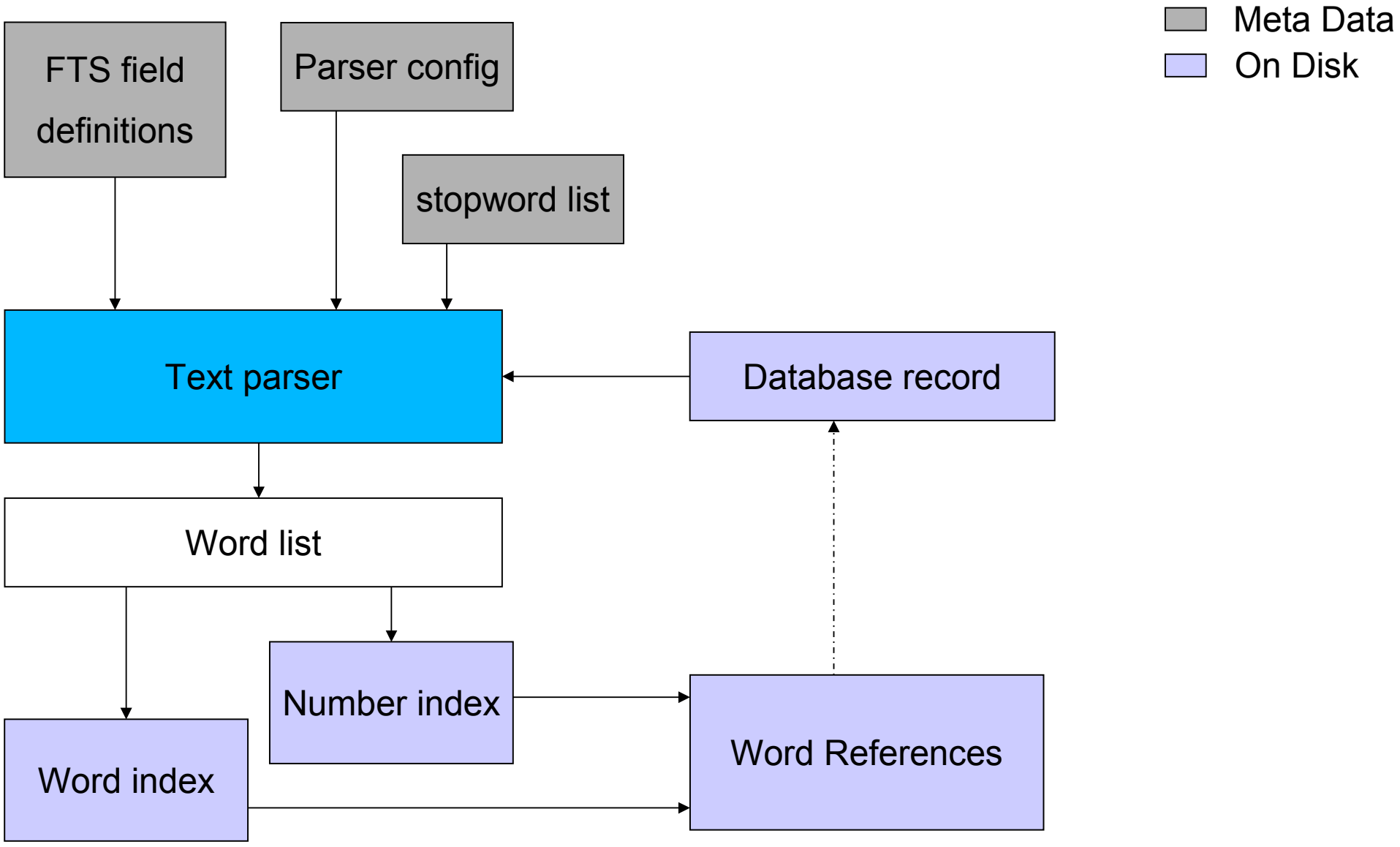
dbutil script

```
database "mandb";
create fts index WORDS in HEADINGS on PAGE
 option GROUP=1;
create fts index in CONTENTS on TEXTLINE
 option GROUP=1, path=PAGE, agg;
```

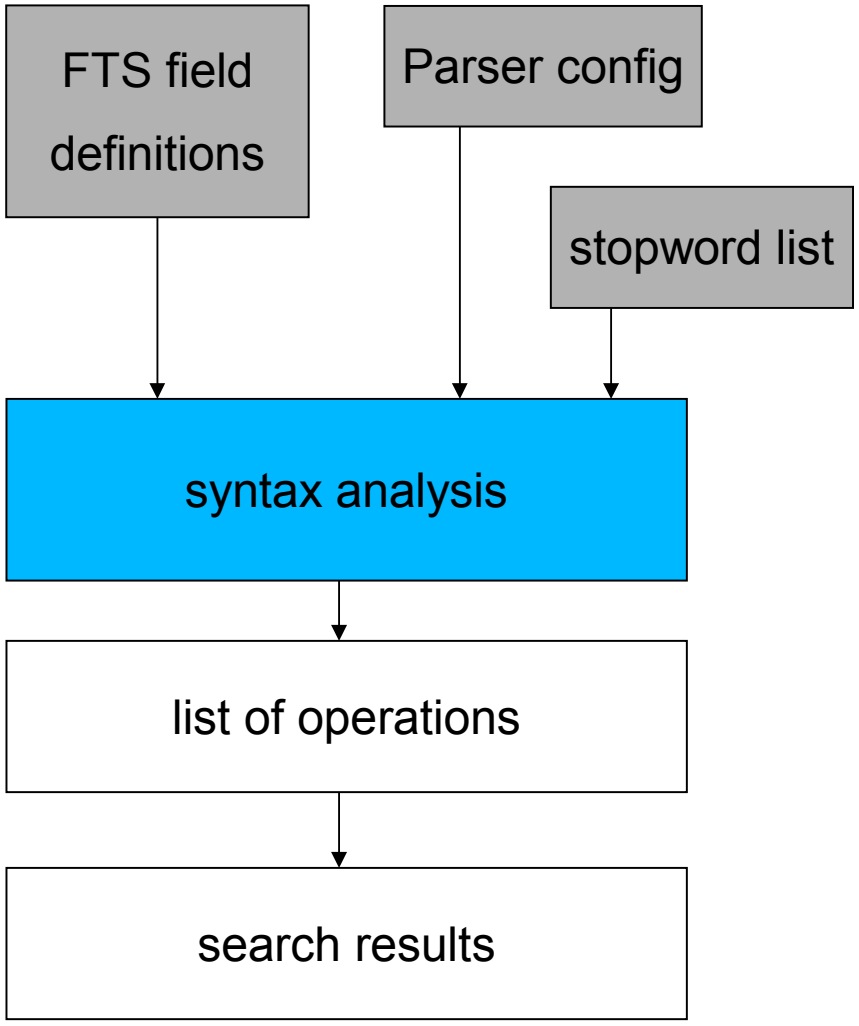
How does it work

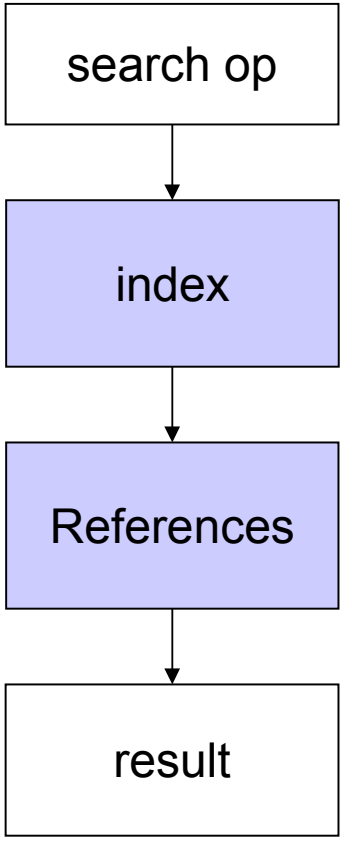
eloquence

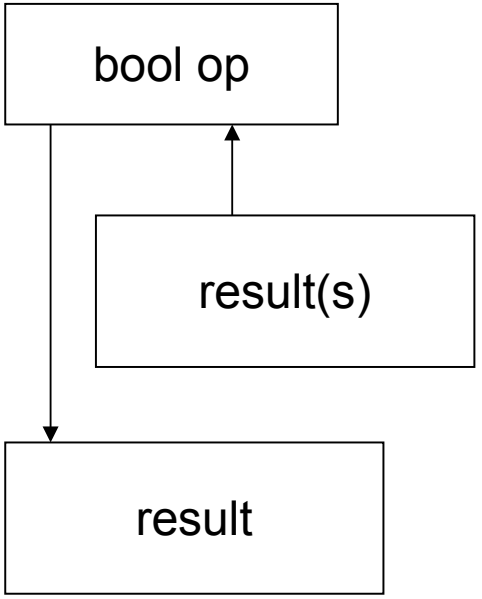
FTS overview (index update)



FTS overview (search)







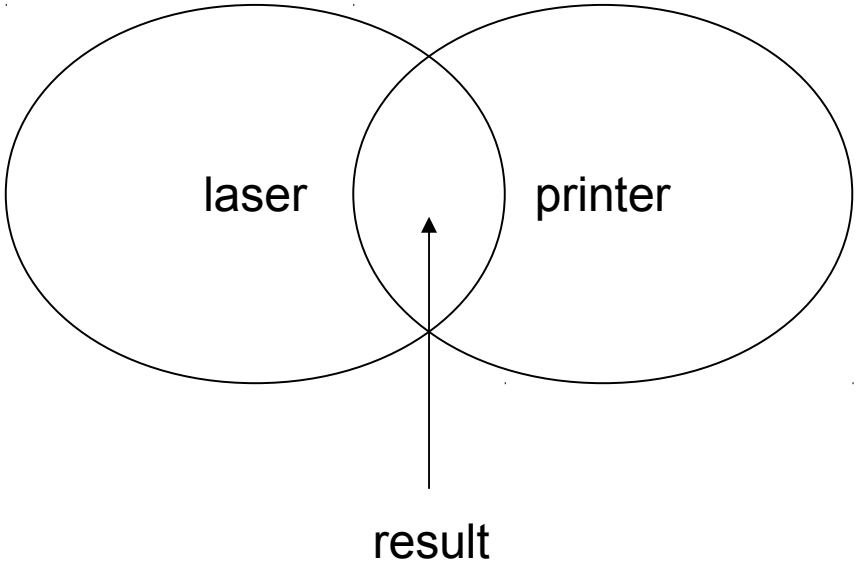
laser AND printer

1: SRCH laser
2: SRCH printer
3: AND #1 #2

FTS search expressions

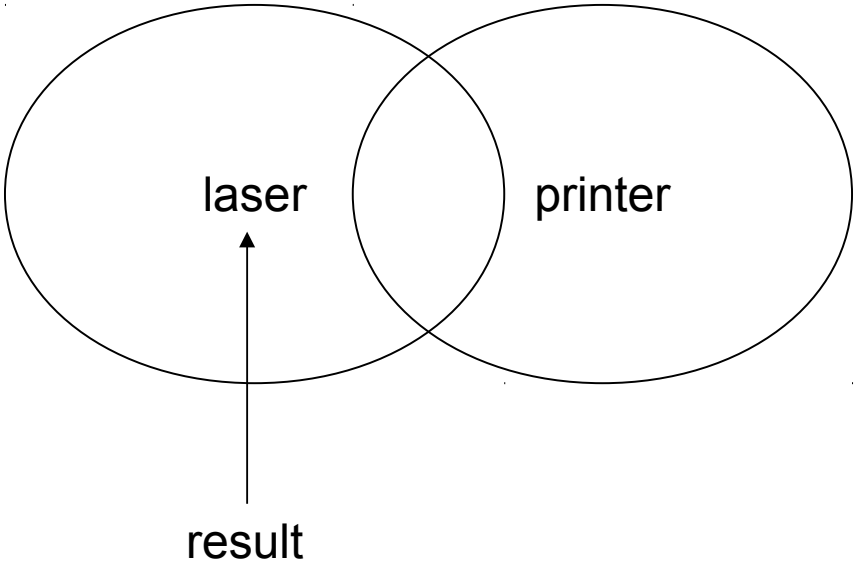
Laser and printer

1. locate list of “laser” references
2. locate list of “printer” references
3. intersect results



Laser and NOT printer

1. locate list of “laser” references
2. locate list of “printer” references
3. intersect results



FTS fields

- FTS fields specify the items to be indexed for a data set
- Typically, text items are indexed
- Numeric items (eg. integer, packed) may also be indexed
- Entire item or parts of items may be indexed
- Composite FTS fields are supported
- All elements of an array are indexed separately
- FTS fields may use a separate name

FTS search

- Locate records by keywords, boolean operators and ranges on arbitrary words
  - Boolean operators (AND, OR, NOT)
  - Relational operators (=, <, >, <=, >=)
  - Partial words (A@)
  - String ranges (A@:C@)
  - Numeric ranges (01:10)
  - SOUNDEX/Fuzzy search

- For example:
  - laser and not printer
  - (hewlett and packard or hp) and not tablet
  - >=42
  - (a:b or d) and (f or g)

Refine searches

- A single search result set is maintained per database open
- Previous searches may be refined using different items in the same set (or set group)
- Allows to perform complex queries covering multiple fields and even multiple sets

- For example:
  - search name = “hewlett and packard”
  - refine previous search results by country = “AND germany”

First DBFIND on “name” using “hewlett AND packard”

then DBFIND on “country” using “AND germany”

Field groups

- Search may span multiple fields
- A search on any of these fields will qualify records with matching words in any field in the same field group

- For example:
  - The database items “Name”, “Name2”, “Street”, “City” are placed in the same field group
  - A search on “marxmeier and wuppertal” will then qualify all entries that have both the words marxmeier and wuppertal in any of the four fields

    For example, Michael Marxmeier in Wuppertal or Marxmeier Street in Wuppertal

FTS reference types

- FTS indexes may be setup to reference individual records, associated master set entries or both

- Individual records
  - Reference points to records

- Associated master sets
  - References are aggregated per master set entry
  - Only the master set entry is referenced, including all words in any details

- Both
  - FTS maintains both a reference to the detail record and its associated master set entry

FTS parser

- The parser operates on text fields and splits them into separate words
- The parser characteristics may be customized, multiple parser configurations per database are supported

FTS parser (words)

- By default, each character that is neither a letter nor a number is considered a word separator
- Usually word separators are discarded
- Detects hyphenated words and creates multiple keywords from it
- The maximum length of a word in an index depends on the field configuration, extra characters are truncated

- For example (using default min/max length):

  “Note: A valid password MUST be entered for (successful) user-authentication”

  results:

  NOTE VALID PASSWORD MUST ENTERED SUCCESSFUL USER-AUTHENT USER AUTHENTICATI

FTS parser (numbers)

eloquence

- Parser recognizes numbers in text fields and adds them to the numeric index (in addition)
- Decimal numbers are recognized if the decimal point character is defined
- Thousands grouping is recognized (and removed) when the grouping character is defined and the grouping happens on a 3 digit position

- For example:

  “Your balance is $4,500.90 (incl. 15% VAT)”

  results:

  YOUR BALANCE 4500.90 INCL 15% VAT

  4500.9 15

FTS exclusion List

- A stop word list (aka exclusion list) may be used to exclude specific words from indexing
- Common words (e.g. “the”) impact performance and add little to search quality
- A min. word length may be specified for a FTS field
- Multiple exclusion lists per database are supported
- Exclusion lists are maintained by dbutil and may be created from a text file

dbutil syntax (field)

```
CREATE FTS INDEX [field_name] IN set_name
ON item [, item ...]
[OPTION op [,...]];

CHANGE FTS INDEX field_name IN set_name
[NAME field_name]
[ON item [, item ...]]
[OPTION op [,...]];

DELETE FTS INDEX field_name IN set_name;
```

Where item may specify

- item_name
- item_name:len
- item_name:ofs.len

dbutil syntax (field options)

- NP - no parse
- NT - no transform
- NE - no exclusion
- NM - no multi
- SX - SOUNDEX support
- MAX=# - max word length (default 12)
- MIN=# - min word length (default 2)
- EXCL=id - exclusion list
- PCFG=id - parser config
- GROUP=id - field group
- PATH=ditem - linked detail in set group
- AGG - specify aggregation by master set (linked detail)

How to use FTS

eloquence

How to use it

- DBFIND is used to perform an FTS search
  - The item name may specify an FTS field
  - The argument specifies the FTS search term
  - Subsequent DBFIND calls may be used to refine the search
- DBGET is used to obtain the resulting records

- DBFIND mode 1 provides compatibility with existing applications
- New DBFIND modes provide additional functionality

- DBGET mode 5, 6 provide compatibility with existing applications
- New DBGET modes provide additional functionality

Simple use

- FTS may be used without code changes
  - DBFIND mode 1 is used to perform FTS search
  - DBGET mode 5 is used to obtain results

- A FTS index is configured for the item specified in DBFIND
- In DBFIND mode 1 a FTS index has precendence over a search item or index

image3k library enhancements

- TPI DBFIND modes (12,13,21,23) to search for FTS fields
- DBFIND mode 1 may be used to ensure compatibility with existing applications

- TPI DBGET modes (21,22,23,25,26) to obtain FTS results
- DBGET mode 5, 6 may be used to ensure compatibility with existing applications

- TPI DBINFO modes 8xx were enhanced to support FTS indexes

- DBCONTROL mode 800 and 801 to specify FTS DBFIND behavior if no records qualify

New database utilities

eloquence

- Add support to repack a data set
  - regain disk space occupied by deleted records
  - offline utility, only useful in exceptional cases

- New utility to extract database settings
  - prschema utility may be used to reconstruct the database schema
  - additional database settings (such as security) are not easily obtainable
  - outputs dbutil syntax

More information

Detailed information is available on the Eloquence web site

http://eloquence.marxmeier.com

Get in contact

info@marxmeier.com