FOM Hochschule für Oekonomie & Management

university location Bonn

Bachelor Thesis

in the study course Wirtschaftsinformatik

to obtain the degree of

Bachelor of Science (B.Sc.)

on the subject

Development of a Query Language for Full-Text Search in Relational Databases

by

Sebastian Bunge

Advisor:	Prof. Dr. Peter Steininger
Matriculation Number:	539441
Submission:	August 11, 2022

Contents

List of Figures

List of Tables

List of Abbreviations

DDL Data Definition Language
HTML Hypertext Markup Language
MS Microsoft
PDF Portable Document Format
SQL Structured Query Language
XML Extensible Markup Language

List of Symbols

1 Abstract

Abstract

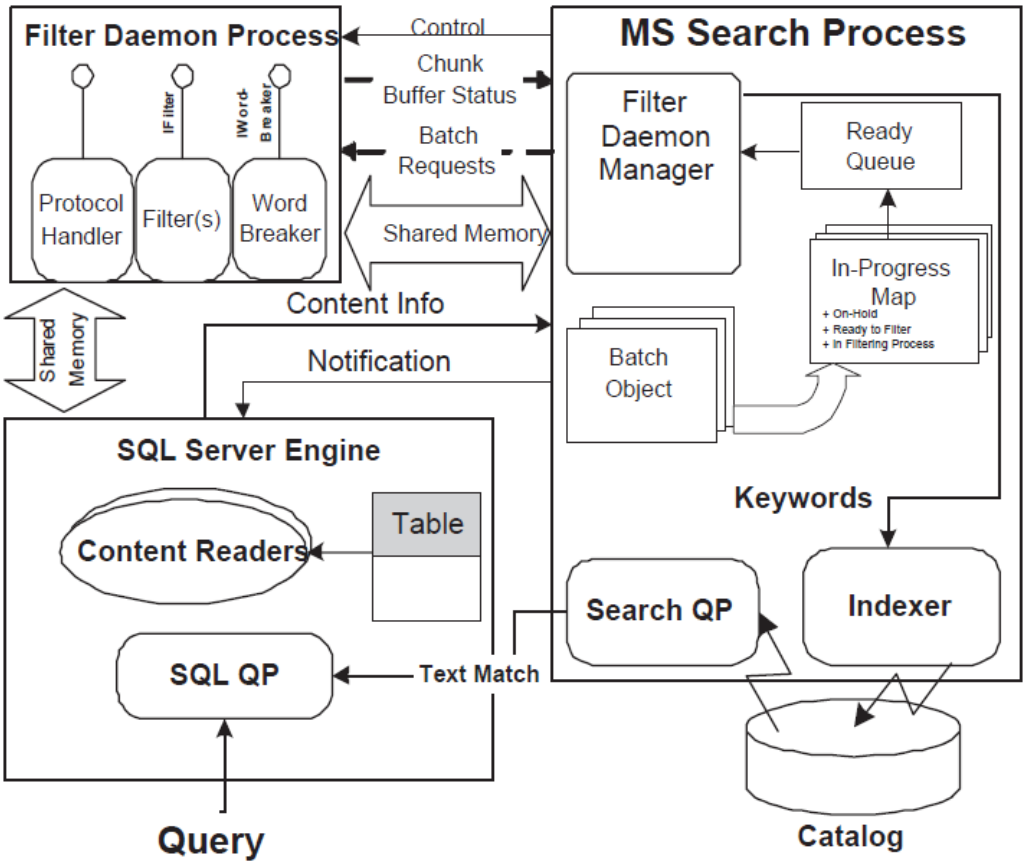
2 Full-Text Search

Commercial database management has long focused on structured data and the industry requirements have matched those of structured storage applications quite well. The problem is that only a small part of the data stored is completely structured, while most of it is completely unstructured or only semi-structured, in the form of documents, emails, web pages, etc. (cf. Hamilton, Nayak 2001, p. 7)

2.1 MS SQL Server Search Architecture

Structured Query Language (SQL) Server uses the same access method and infrastructure for full-text search as other Microsoft (MS) products and the Index Service for file systems. This decision enables standardized semantics for full-text search of data in relational databases, web-hosted data, and data stored in the file system and mail systems. On SQL servers, not only simple strings can be indexed, but also data structures, such as Hypertext Markup Language (HTML) and Extensible Markup Language (XML), and even complex documents, such as Portable Document Format (PDF), Word, PowerPoint, Excel and other custom document formats. (cf. Hamilton, Nayak 2001, p. 7)
The architecture can be divided into five modules, which interact with each other to perform a full-text search. (See Figure 1)
The **content reader** scans indexed data stored in SQL Server tables to assemble data and its associated metadata packets. These packets are then injected into the main search engine, which triggers the search engine filter daemon to consume the data.
Depending on the content, the **filter daemon** calls different filters, which parse the content and output so-called chunks of the processed text. A chunk is a related section with relevant information about this section like the language-id of the text. These chunks are output separately for any properties, which can be elements like the title, an author or other content-specific elements.

Figure 1: Architecture of MS SQL Server Full-Text Search



Source: Hamilton, Nayak 2001, p. 8

Word breakers split the chunks into keywords and additionally provide alternative keywords and the corresponding position in the text. Word breakers can recognize human languages and on SQL Server several word breakers for different languages are installed by default. The generated keywords and metadata are passed on to the MS Search process, which processes the data with an indexer.

The **indexer** generates an inverted keyword list with a batch containing all keywords of one or more items. These indexes are compressed to use memory efficiently, this may lead to high costs for updates of these indexes. Therefore a stack of indexes is maintained. New documents first create their small indexes, which are regularly merged into a larger index, which in turn is merged into the base index. This stack can be deeper than three, but the concept remains and allows a strongly compressed index without driving the update costs too high. If a keyword is searched, all indexes are accessed, so the depth should still be kept reasonable.

A **query processor** manages the insertion and merge operations and collects statistics on distribution and frequency for ranking purposes and query execution. (cf. Hamilton, Nayak 2001, pp. 8-9)

2.2 MS SQL Server Full-Text Query Features

Full-text indexes can be created on SQL Servers with the Data Definition Language (DDL) statement `CREATE INDEX` and can make use of other SQL Server utilities; these include backup and restore and attachment of databases. There are three options to create and manage indexes on SQL Servers. **Full Crawl** always rebuilds the whole full-text index by scanning the entire table. **Incremental Crawl** logs the timestamp of the last re-index and retains changes by storing them in a column. **Change Tracking** enables a near real-time validity between the full-text index and the table by tracking changes to the indexed data using the SQL Server Query Processor. (cf. Hamilton, Nayak 2001, p. 9)
Full-text search is represented in SQL with three possible constructs: (cf. Hamilton, Nayak 2001, p. 9)

1. Contains Predicate: A contains predicate is true if one of the specified columns contains terms that satisfy the specified search condition. E.g. `Contains(author, ('Ag* or "Marc Miller"'))` will match entries where the column author contains words like 'Ag', 'Agatha', or 'Marc Miller'.

2. Freetext Predicate: Freetext predicates are true if one of the specified columns contains terms that stem from the terms in the specified search condition. E.g. `Freetext(content, 'fishing')` will match entries where content contains words like 'fishing', 'fish', or 'fisher'.

3. ContainsTable and FreetextTable: ContainsTable and FreetextTable are functions that match entries similar to their corresponding function, but additionally return multiple matches including a ranking for each entry and the entire corpus.

The search conditions of these constructs can be of various types to find the intended results: (cf. Hamilton, Nayak 2001, p. 9)

1. Keyword, phrase, prefix: E.g. 'fishing', 'Marc Miller', 'Ag*'

2. Inflections and Thesaurus: E.g. `Contains(*, 'FORMSOF(INFLECTIONAL, fishing)AND FORMSOF(THESAURUS, boat)')` will find all entries containing words that stem from 'fishing' and all words sharing the meaning with 'boat' (Thesaurus support).

3. Weighted terms: Keywords and phrases can be assigned a relative weight to impact the rank of entries. E.g. `ContainsTable(*, 'ISABOUT(generator weight (.7), full-text weight (.3))')` will rank entries higher in the result corpus which mention 'generator' over 'full-text'.

4. Proximity: E.g. `Contains(*, 'corn NEAR salad')` contains the proximity term 'NEAR' to match entries where 'corn' appears close to 'salad'.

5. Composition: E.g. `Contains(*, 'full-text AND NOT database')` uses two search query components that are composed using a term like 'AND', 'OR', or 'AND NOT'.

3 My Language

My Language

4 Summary

Summary

Appendix

Appendix 1: Appendix

Appendix

Bibliography

HAMILTON, James R.; NAYAK, Tapas K.: Microsoft SQL server full-text search. In: *IEEE Data Eng. Bull.* 24 (2001) Nr. 4. Publisher: Citeseer, pp. 7–10

Declaration in lieu of oath

I hereby declare that I produced the submitted paper with no assistance from any other party and without the use of any unauthorized aids and, in particular, that I have marked as quotations all passages which are reproduced verbatim or near-verbatim from publications. Also, I declare that the submitted print version of this thesis is identical with its digital version. Further, I declare that this thesis has never been submitted before to any examination board in either its present form or in any other similar version. I herewith agree that this thesis may be published. I herewith consent that this thesis may be uploaded to the server of external contractors for the purpose of submitting it to the contractors' plagiarism detection systems. Uploading this thesis for the purpose of submitting it to plagiarism detection systems is not a form of publication.

Bonn, 11.8.2022

(Location, Date)

(handwritten signature)