# 第2章 万维网网页信息的表达及解析

2.1 万维网架构及网页表达

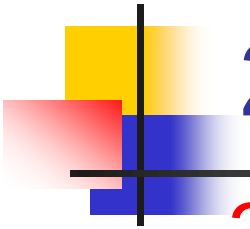
2.2 网页信息抽取

2.3 搜索引擎与分析系统

2.4 搜索引擎索引系统

2.5 搜索引擎查询系统5

《走进搜索引擎》 潘雪峰 花贵春 梁斌编著
电子工业出版社 2011年5月第2版



# 2.4 搜索引擎索引系统

## 2.4.1 全文搜索引擎基本概念

## 2.4.2 倒排索引结构与原理

## 2.4.3 倒排索引的创建

# 2.4.1 全文搜索

- 信息检索领域的一场革命
- 细化了信息检索的粒度
- 提供了多角度，多侧面且全新的信息检索体验

- 全文检索（full-text retriev
  - 通过检索关键词对文档的全部
  - 检索结果能够提供检索词出现的实际位置

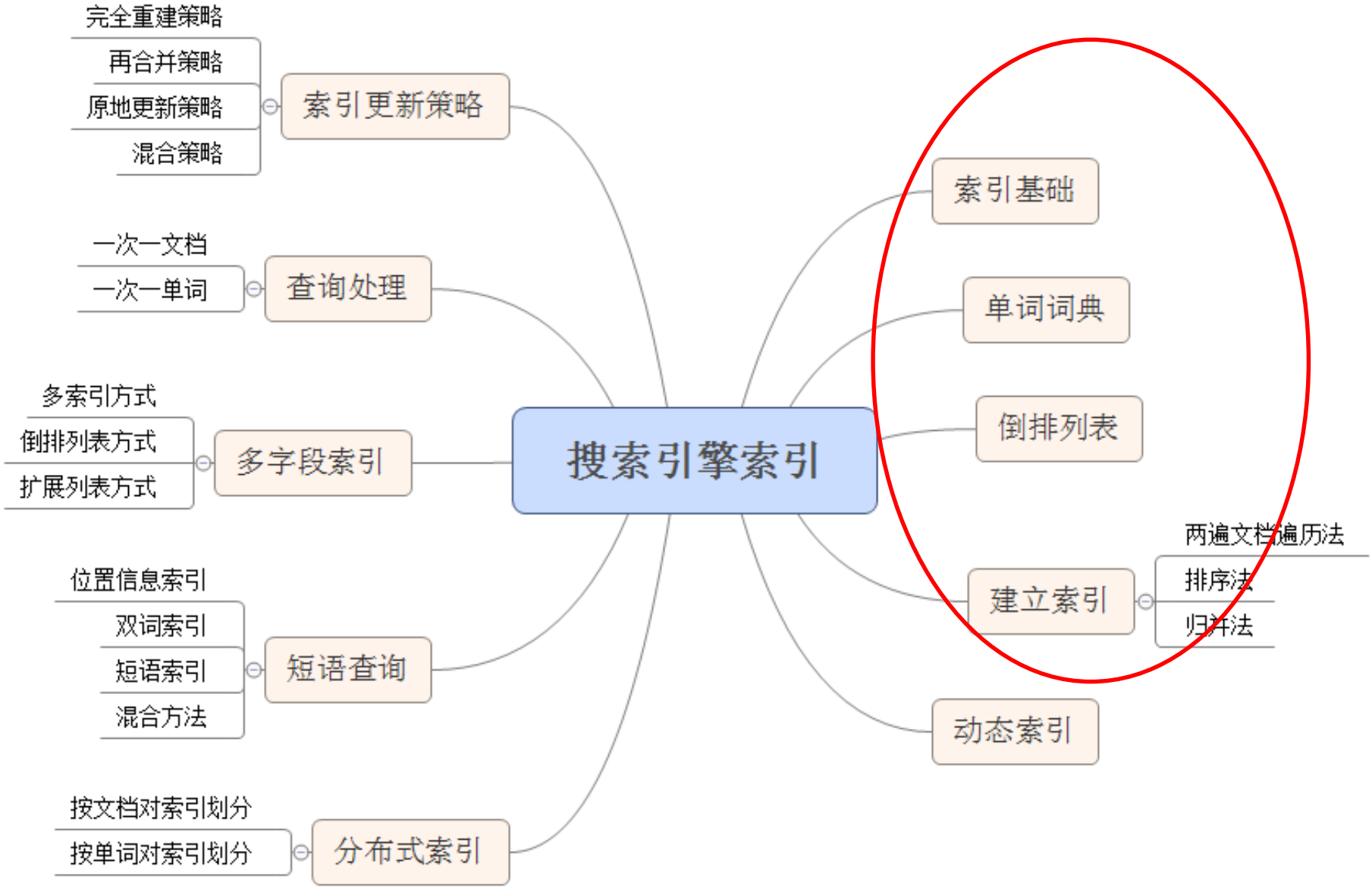
蕴涵的是大量的信息组织的智慧

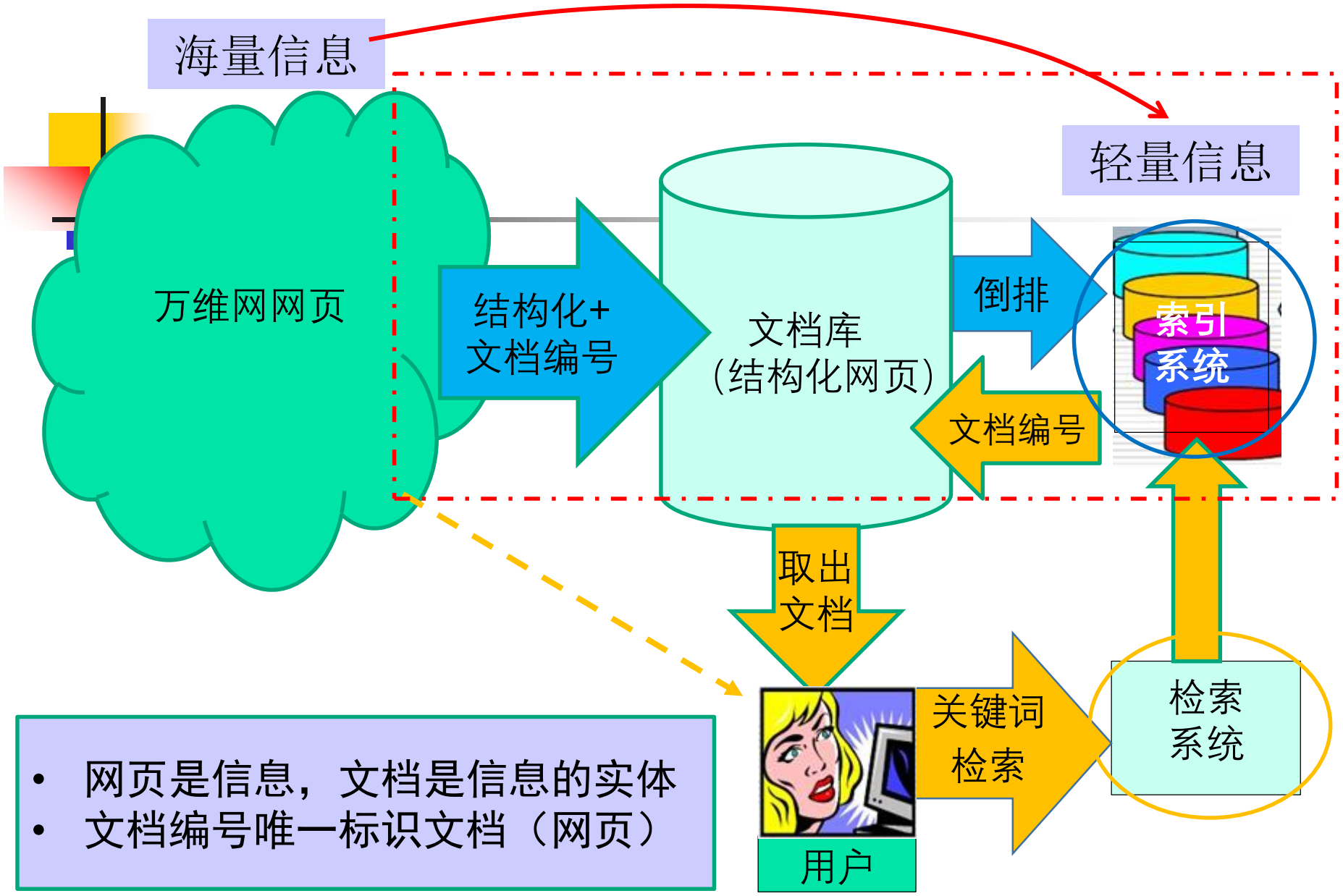

当今搜索引擎无一例外地采用了全文检索

# 2.4.1 搜索引擎基本概念


- 索引
  - 信息的信息
  - (**信息**：结构化的网页数据)
- (全文)搜索引擎的 “索引系统”
  - 负责将分析系统处理后的网页对象建索引入库
  - 在搜索引擎早期的发展中，能够索引的网页数量代表了整个行业的技术发展水平。
- 索引系统目标：
  - 存储容量大
  - 检索速度快（ 低于秒级的检索时间）

是搜索引擎的数据大本营，存储并索引了数以亿计的网页。


搜索引擎索引
索引更新策略
完全重建策略
再合并策略
原地更新策略
混合策略
查询处理
一次一文档
一次一单词
多字段索引
多索引方式
倒排列表方式
扩展列表方式
短语查询
位置信息索引
双词索引
短语索引
混合方法
分布式索引
按文档对索引划分
按单词对索引划分
索引基础
单词词典
倒排列表
建立索引
两遍文档遍历法
排序法
归并法
动态索引




- 网页是信息，文档是信息的实体
- 文档编号唯一标识文档（网页）

# 2.4.1 全文搜索引擎基本概念

## 1.文档及文档编号https://blog.csdn.net/C6k7Ch/article/details/79974411

- 文档(Document)：
  - 以文本形式存在的存储对象，相比网页来说，涵盖更多种形式，比如Word，PDF，HTML，XML等不同格式的文件都可以称之为文档。
  - 很多情况下用文档来表征文本信息。
- 文档编号(Document ID，DocID)：
  - 每个文档的内部编号
  - 在搜索引擎内部，每个文档赋予一个唯一的内部编号，以此编号来作为这个文档的唯一标识，这样方便内部处理

# 2.4.1 全文搜索引擎基本概念

## 2.文档编号的方法

- 文档编号性质：
  - （1）任何一个文档在其生命周期中仅有一个编号
  - （2）任何两个不同的文档的编号不同
  - （3）编号在计算中尽可能便于存储，越短越好
- 文档编号：
  - 网页的URL唯一地表示一个文档
- 文档标号处理：
  - 采用长整形整数
    - URL字符串运算慢，占用空间较大
    - 把字符串映射为占64比特大小的长整形整数
    - 采用URL字符串签名的方法（MD5签名）
    - 整型编号节约大量存储空间，且大大提高检索计算效率

- 一个URL只能到达唯一网页
- 文档编号和文档内容无关（否则不稳定）
- 不能直接由文档文件名代替（否则不能保证唯一）

单词是文档的基本单位

## 3.单词编号(Word ID)：

- 搜索引擎内部以唯一标识某个单词

## 4.单词词典(Lexicon)：

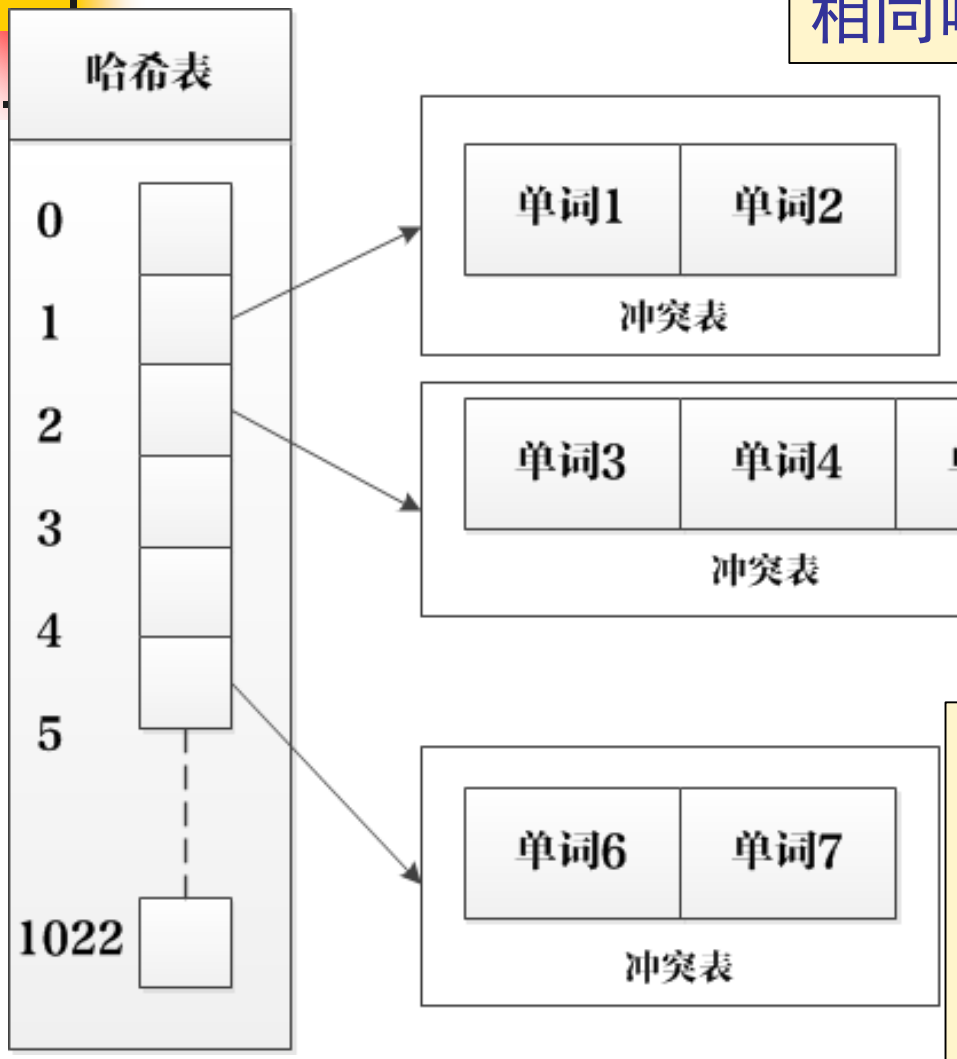
- 由文档集合中出现过的所有单词构成的字符串集合
- 单词词典内每条索引项记载单词本身的一些信息以及指向“倒排列表”的指针。

## 5.单词-文档矩阵

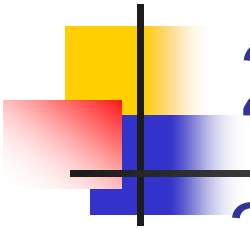
- 表达两者之间所具有的一种包含关系
- 每列代表一个文档
- 每行代表一个单词，打对勾的位置代表包含关系

- 单词词典结构1-哈希加链表词典结构

主体是哈希表，每个哈希表项保存一个指针，指针指向冲突连表，相同哈希值的单词形成链表结构。




- 构建过程：
  - 对文档进行分词；
  - 利用哈希函数获取哈希值；
  - 如果冲突链表已经存在，建冲突连表



# 2.4 搜索引擎索引系统

## 2.4.1 全文搜索引擎基本概念

## 2.4.2 倒排索引结构与原理

## 2.4.3 倒排索引的创建

# 2.4.2 倒排索引结构与原理

- 索引系统中的索引［S. Brin 1998］包含了3个概念：
  - 命中（Hit）：
    - 表示索引词在文章中的位置（position）等信息
  - 正向索引（Forward Index）：
    - 基于文档（doc）来查询关键字key
  - 倒排索引（Inverted Index）
    - 基于关键字key来查询文档（doc）
    - 关键词称为“索引词”
- 全文检索通过关键词来进行——倒排索引

# 2.4.2 倒排索引结构与原理

- **正向索引（前向索引）概念：基于文档查关键字**

| 文件 ID | 文件内容经过分词，提取出来的关键词 |
| --- | --- |
| 文件 1 | 关键词 1、关键词 2、关键词 4…关键词 A |
| 文件 2 | 关键词 3、关键词 9、关键词 27…关键词 B |
| 文件 3 | 关键词 5、关键词 25、关键词 125…关键词 C |
| 文件 4 | 关键词 6、关键词 36、关键词 216…关键词 D |
| …… | …… |
| 文件 N | 关键词 N、关键词 N*N、关键词 N*N*N…关键词 M |

- 优点：易维护，每新增一个doc只需要把每个key在该doc中出现的次数和位置进行维护即可；
- 缺点：搜索耗时太长

# 2.4.2 倒排索引结构与原理

- 正排索引（前向索引）结构：

  （1）LocalId（Lid）：文档的局部编号。

  （2）WordId：文档分词的编号（索引词编号）

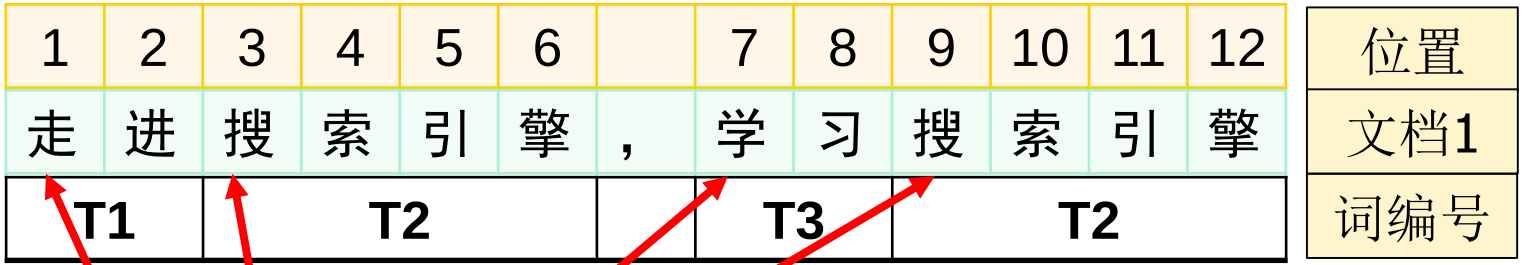
  （3）NHits：某个索引词在文档中出现的次数（命中）

  （4）HitList变长字段：某个索引词在文档中出现的位置，即相对于正文的偏移量。

| LId | WordId | NHits | HitList |
|---|---|---|---|
| Doc1 | Word1 | m | Hit1,...Hitm |
| | Word2 | n | Hit1,...Hitn |
| | ... | | |
| | WordN | | |
| | NULL | | |
| Doc2 | ... | ... | ... |
| | NULL | | |

# 2.4.2 倒排索引结构与原理

- 创建正排索引例
- 假定存在编号为1的文档




- 1.分词并编号：“走进 / 搜索引擎 / 学习 / 搜索引擎”
- 2.统计单词出现次数及位置：
  - “走进”出现1次，出现位置为1；
  - “搜索引擎”出现两次，出现位置为3和9；
  - “学习”出现1次，出现位置为7
- 3.生成正排索引
  - HitList是变长的，记录单词相对位置（偏移量）（图中的为未压缩的差分序列3和6）

| LId | WordId | NHits | HitList |
|---|---|---|---|
| 1 | T1 | 1 | 1 |
| | T2 | 2 | 3,6 |
| | T3 | 1 | 7 |
| | NULL | | |

## 倒排索引概念

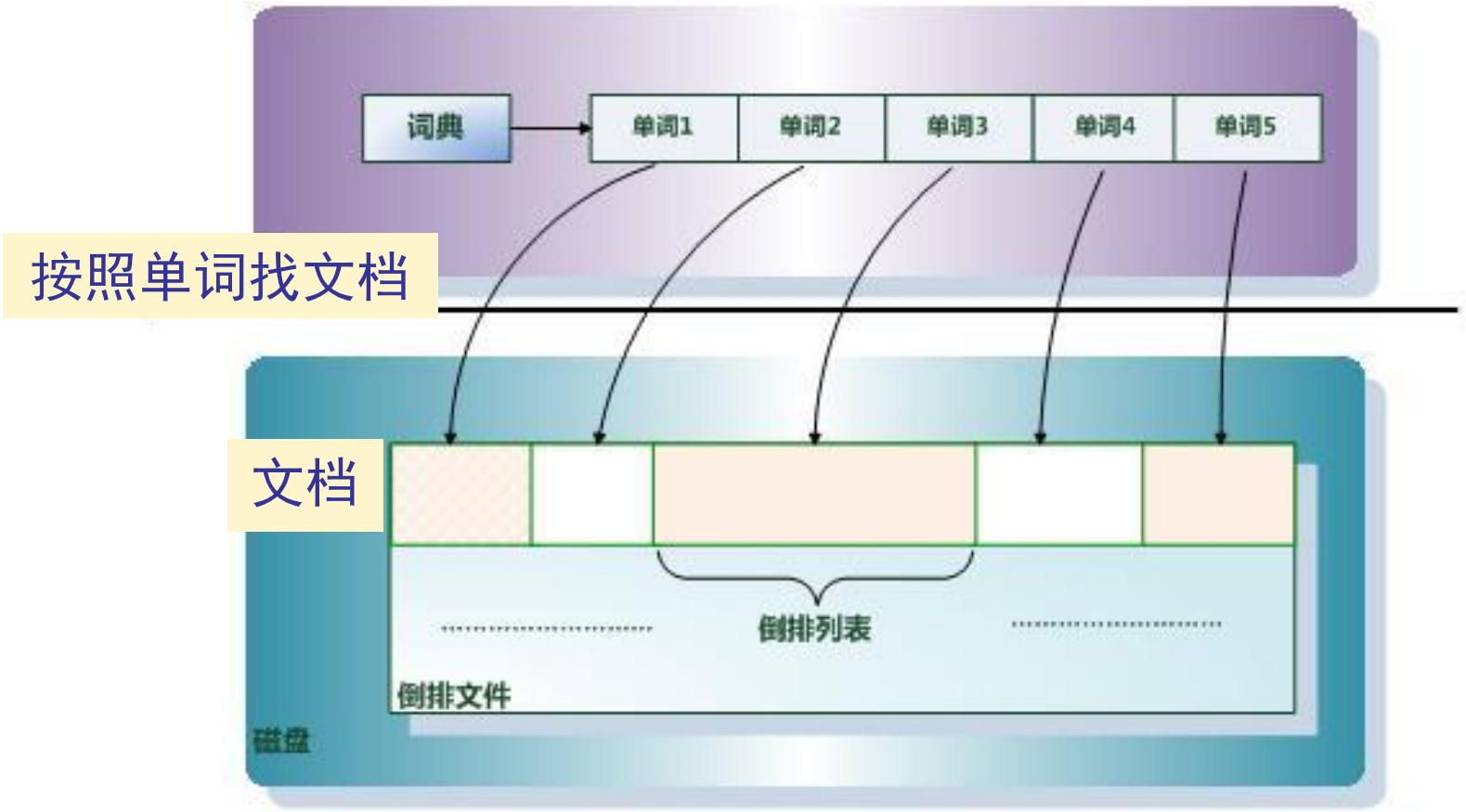
- 以关键字作为主键的索引结构
- 倒排索引并不是一种排序方式，而是一种存储方式，用来提升检索效率

| 关键词 | 文件 |
| --- | --- |
| 关键词 1 | 文件 1、文件 2、文件 4…文件 A |
| 关键词 2 | 文件 3、文件 9、文件 27…文件 B |
| 关键词 3 | 文件 5、文件 25、文件 125…文件 C |
| 关键词 4 | 文件 6、文件 36、文件 216…文件 D |
| …… | …… |
| 关键词 N | 文件 N、文件 N*N、文件 N*N*N…文件 M |

- 优点：搜索效率高；
- 缺点：维护成本高，每新增一个doc，需要维护倒排索引

# 2.4.2 倒排索引结构与原理

## 倒排索引基本概念示意



# 2.4.2 倒排索引结构与原理

- 倒排索引结构，两个部分：
  - 单词词典（lexicon）
  - 记录表（posting file）或记录列表（posting list）

- DocId：记录文档编号
- Nhits：索引词在该文档的命中个数
- HitList：以及命中位置的列表

| WordId | nDocs | 偏移量 |
|---|---|---|
| T1 | 3 | x |
| T2 | 2 | y |
| … | … | … |
| Tn | | z |

| DocId | NHits | HitList |
|---|---|---|
| Doc1 | | |
| Doc2 | | |
| Doc3 | | |
| Doc1 | 3 | 3,2,2 |
| Doc2 | | |
| … | … | … |
| Doc2 | | |
| Doc3 | | |

- Word ID：索引词Id号
- nDocs:匹配该索引词的文档数量
- 偏移量：匹配文档在记录文件内的偏移量，通过这个偏移量就可以读取记录文件对应区域的信息

# 倒排索引相关概念

- 倒排表

小

  - 存放在内存中的能够追加倒排记录的倒排索引，是迷你的倒排索引。

中

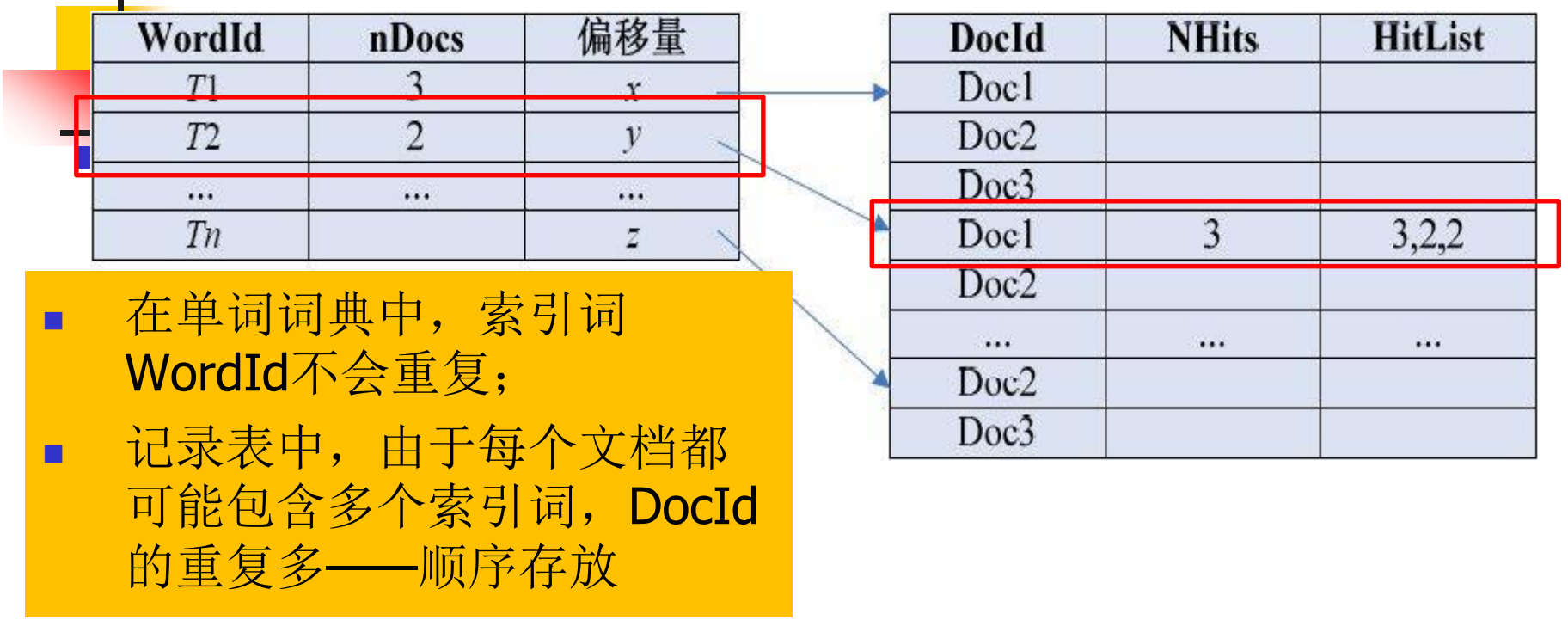
- 临时倒排文件
  - 存放在磁盘中，以文件的形式存储的不能够追加倒排记录的倒排索引。临时倒排文件是中等规模的倒排索引。

大

- 最终倒排文件
  - 由存放在磁盘中，以文件的形式存储的临时倒排文件归并得到的倒排索引。最终倒排文件是较大规模的倒排索引

- 倒排索引作为抽象概念，而倒排表、临时倒排文件、最终倒排文件是倒排索引的三种不同的表现形式

- 例.下图为倒排索引的单词词典和记录表



| WordId | nDocs | 偏移量 |
|---|---|---|
| T1 | 3 | x |
| T2 | 2 | y |
| … | … | … |
| Tn | | z |

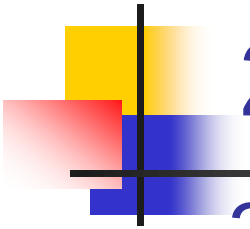
| DocId | NHits | HitList |
|---|---|---|
| Doc1 | | |
| Doc2 | | |
| Doc3 | | |
| Doc1 | 3 | 3,2,2 |
| Doc2 | | |
| … | … | … |
| Doc2 | | |
| Doc3 | | |

- 在单词词典中，索引词WordId不会重复；
- 记录表中，由于每个文档都可能包含多个索引词，DocId的重复多——顺序存放

- 以索引词T2为例，理解倒排索引
  - T2匹配Doc1和Doc2两个文档
  - 在Doc1中出现了3次，位置分别为3、5和7（图中为差分序列后的实际值，即3、5-3和7-5）

# 2.4.2 倒排索引结构与原理

- 总结：正排索引和倒排索引的关系
- 存在两个空间
  - “索引词空间”
  - “文档空间”
- 正排索引
  - 文档空间到索引词组空间的一个映射，任意一个文档对应唯一的一组索引词；
- 倒排索引
  - 索引词空间到文档组空间的一个映射
  - 任意一个索引词对应唯一的一组该索引词其命中的文档



# 2.4 搜索引擎索引系统

## 2.4.1 全文搜索引擎基本概念

## 2.4.2 倒排索引结构与原理

## 2.4.3 倒排索引的创建

# 2.4.3 倒排索引的创建

- 倒排索引文件的创建是搜索引擎的核心难点
- 建立倒排索引步骤：
  - 1.文档编号
  - 2.分词
  - 3.单词编号
  - 4.创建倒排索引
    - 1）简单倒排索引
    - 2）带单词频率信息的倒排索引
    - 3）带有单词频率、文档频率和出现位置信息的倒排索引
- 搜索引擎索引数据结构和算法
- https://www.cnblogs.com/Leo_wl/p/5470570.html

- **1）简单倒排索引实例**
  - 假设文档集合包含五个文档，任务就是对这个文档集合建立倒排索引。
  - 第1步：文档编号

- 第2步：分词
- 第3步：对每个不同的单词赋予唯一的单词编号，同时记录下哪些文档包含这个单词
- 第4步：得到最简单的倒排索引

每个单词对应的倒排列表，包含该单词的文档列表

记录每个单词的单词编号

| 单词ID | 单词 | 倒排列表（DocID） |
|---|---|---|
| 1 | 谷歌 | 1,2,3,4,5 |
| 2 | 地图 | 1,2,3,4,5 |
| 3 | 之父 | 1,2,4,5 |
| 4 | 跳槽 | 1,4 |
| 5 | Facebook | 1,2,3,4,5 |
| 6 | 加盟 | 2,3,5 |
| 7 | 创始人 | 3 |
| 8 | 拉斯 | 3,5 |
| 9 | 离开 | 3 |
| 10 | 与 | 4 |
| 11 | Wave | 4 |
| 12 | 项目 | 4 |
| 13 | 取消 | 4 |
| 14 | 有关 | 4 |
| 15 | 社交 | 5 |
| 16 | | |

- “谷歌”单词
  - 编号为1
  - 倒排列表为{1,2,3,4,5}，说明5个文档都包含了这个单词

只记录了关键词在文档的出现

- 2）带有单词频率信息的倒排索引
- 单词对应的倒排列表中记录：文档编号，单词频率信息（TF：单词在某个文档中出现的次数）

文档编号，单词频率

| 单词ID | 单词 | 倒排列表（DocID:TF） |
|---|---|---|
| 1 | 谷歌 | (1;1),(2;1),(3;2),(4;1),(5;1) |
| 2 | 地图 | (1;1),(2;1),(3;1),(4;1),(5;1) |
| 3 | 之父 | (1;1),(2;1),(4;1),(5;1) |
| 4 | 跳槽 | (1;1),(4;1) |
| 5 | Facebook | (1;1),(2;1),(3;1),(4;1),(5;1) |
| 6 | 加盟 | (2;1),(3;1),(5;1) |
| 7 | 创始人 | (3;1) |
| 8 | 拉斯 | (3;1),(5;1) |
| 9 | 离开 | (3;1) |
| 10 | 与 | (4;1) |
| 11 | Wave | (4;1) |
| 12 | 项目 | (4;1) |
| 13 | 取消 | (4;1) |
| 14 | 有关 | (4;1) |
| 15 | 社交 | (5;1) |
| 16 | 网站 | (5;1) |

- 单词“创始人”的单词编号为7
- 倒排列表内容（3:1）为：
  - 3代表文档3包含这个单词
  - 数字1代表词频信息，即这个单词在3号文档中只出现过1次

## 3）带有单词频率、文档频率和出现位置信息的倒排索引

文档频率：文档集合中有多少个文档包

文档编号，单词频率，出现位置

| 单词ID | 单词 | 文档频率 | 倒排列表（DocID;TF;<POS>） |
|---|---|---|---|
| 1 | 谷歌 | 5 | (1;1;<1>),(2;1;<1>),(3;2;<1;6>),(4;1;<1>),(5;1;<1>) |
| 2 | 地图 | 5 | (1;1;<2>),(2;1;<2>),(3;1;<2>),(4;1;<2>),(5;1;<2>) |
| 3 | 之父 | 4 | (1;1;<3>),(2;1;<3>),(4;1;<3>),(5;1;<3>) |
| 4 | 跳槽 | 2 | (1;1;<4>),(4;1;<4>) |
| 5 | Facebook | 5 | (1;1;<5>),(2;1;<5>),(3;1;<8>),(4;1;<5>),(5;1;<8>) |
| 6 | 加盟 | 3 | (2;1;<4>),(3;1;<7>),(5;1;<5>) |
| 7 | 创始人 | 1 | (3;1;<3>) |
| 8 | 拉斯 | 2 | (3;1;<4>),(5;1;<4>) |
| 9 | 离开 | 1 | (3;1;<5>) |
| 10 | 与 | 1 | (4;1;<6>) |
| 11 | Wave | 1 | (4;1;<7>) |
| 12 | 项目 | 1 | (4;1;<8>) |
| 13 | 取消 | 1 | (4;1;<9>) |

- “拉斯” 文档频率为2：代表整个文档集合中有两个文档包含该词
- 倒排列表中{(3;1;<4>)，(5;1;<4>)},其含义：
  - 在文档3和文档5出现过这个单词
  - 单词频率都为1；
  - 在两个文档中的出现位置都是4，即文档中第四个单词是“拉斯”

# 2.4.3 倒排索引的创建

- 建立索引方法：

- 两遍文档遍历法
- 排序法
- 归并法

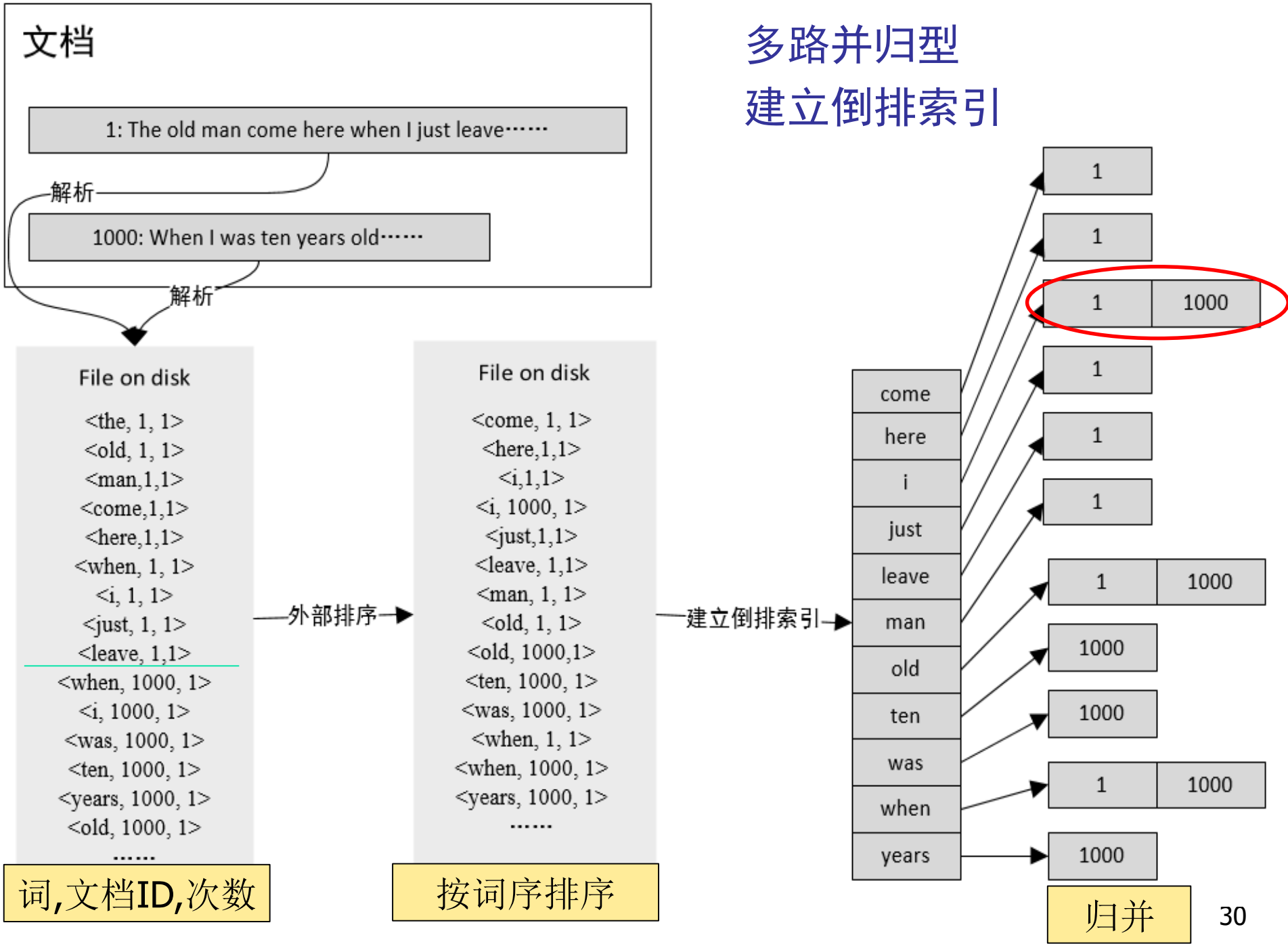
- 单遍内存型
- 多路并归型

- 主要工作：
  - 1）对每个文档，在内存或磁盘完成建立倒排表工作
  - 2）归并：扫描每个文档归并

# 2.4.3 倒排索引的创建

- 多路并归型建立过程：
  - 解析文档，把<词，文档编号，单词频率TF>写入到磁盘文件。
  - 对磁盘文件进行外部排序：
    - 按照词的字典序从小到大排序
    - 如果词相同，则按照文档编号从小到大排序
  - 顺序扫描排序后的文件，建立倒排索引。由于相同的词紧挨在一起，可以一个词一个词的建立倒排索引。
- 这种方式适合在内存小，磁盘大的情况下进行倒排索引的建立。
- 优点：内存使用量小，排序灵活
- 缺点：多次归并排序，比较慢。

# 多路并归型
# 建立倒排索引

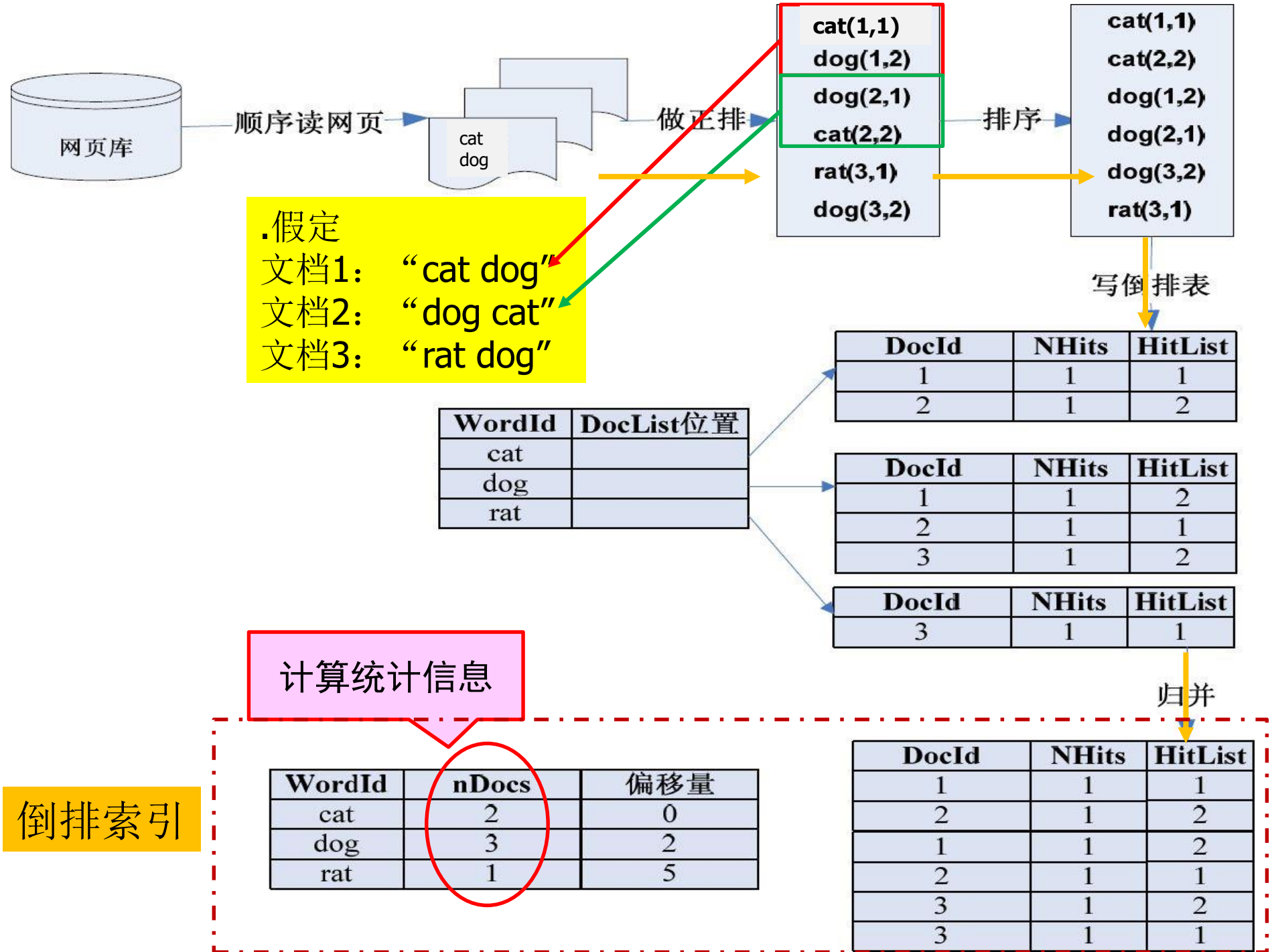


# 2.4.3 倒排索引的创建

例.假定有三个文档

文档1：“cat dog”
文档2：“dog cat”
文档3：“rat dog”

- 解析文档：
  - 通过在内存中完成正排得到索引词出现的文档和位置信息
  - 例如，cat（1，1）表示在文档1的第1个位置出现“cat”这个索引词。
- 排序：
  - 对字母排序（汉字可以按照汉字词汇编号排序）
  - 得到一个临时的按照索引词有序的结构，这有助于顺序写入各个索引词对应的记录表。
- 计算统计信息、归并——形成倒排文件：
  - 多个单词、多个文档排序表合并，计算统计信息


网页库
顺序读网页
cat
dog
做正排
cat(1,1)
dog(1,2)
dog(2,1)
cat(2,2)
rat(3,1)
dog(3,2)
排序
cat(1,1)
cat(2,2)
dog(1,2)
dog(2,1)
dog(3,2)
rat(3,1)
.假定
文档1： “cat dog”
文档2： “dog cat”
文档3： “rat dog”
写倒排表
WordId
DocList位置
cat
dog
rat
DocId
NHits
HitList
1
1
1
2
1
2
DocId
NHits
HitList
1
1
2
2
1
1
3
1
2
DocId
NHits
HitList
3
1
1
计算统计信息
归并
倒排索引
WordId
nDocs
偏移量
cat
2
0
dog
3
2
rat
1
5
DocId
NHits
HitList
1
1
1
2
1
2
1
1
2
2
1
1
3
1
2
3
1
1