OneStepOffer 系统设计
第七讲

设计Search engine

Design Search Engine - Scenario

- 需要设计哪些功能？

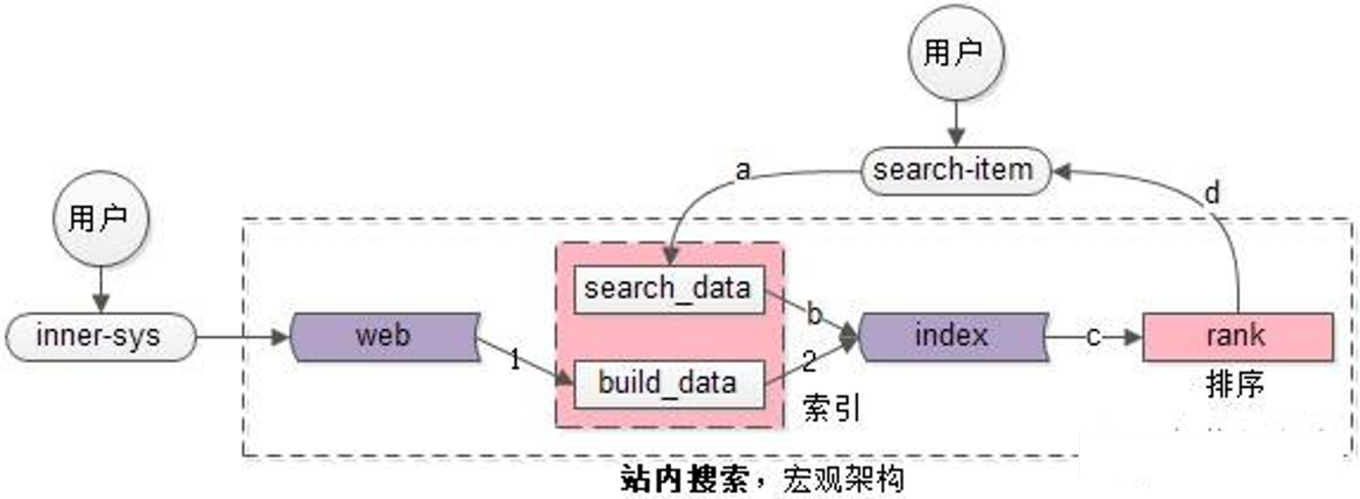
  搜索一份发言 post

- 新贴子要被加入到待搜索的内容中
- 用户可以根据关键词搜索贴子

Design Search Engine - Service

主要服务：

- Index service: 索引服务，讲新内容加入数据库
- Search service：搜索服务，根据输入关键词返回搜索结果
- Rank Service：排序服务，讲搜索结果按照相关性排序



Design Search Engine - 原理

- 什么是正排索引？
- 什么是倒排索引？
- 搜索的过程是什么样的？

什么是正排索引（forward index）？

- 用key查询实体文档的过程，是正排索引

  用户表：t_user(uid, name, passwd, age, sex)，由uid查询整行的过程，就是正排索引查询。

  网页库：t_web_page(url, page_content)，由url查询整个网页的过程，也是正排索引查询。

什么是正排索引（forward index）？

举个例子，假设有3个网页：

url1 -> “我爱北京”

url2 -> “我爱到家”

url3 -> “到家美好”

这是一个**正排索引**Map<url, page_content>。

分词之后：

url1 -> {我，爱，北京}

url2 -> {我，爱，到家}

url3 -> {到家，美好}

这是一个**分词后的正排索引**Map<url, list<item>>。

什么是倒排索引（inverted index）？

根据文档内容查询 key，是倒排索引

对于网页搜索，倒排索引可以理解为Map<word, list<url>>，能够由查询词快速

找到包含这个查询词的网页的数据结构。

什么是倒排索引（inverted index）？

分词后倒排索引：

我 -> {url1, url2}

爱 -> {url1, url2}

北京 -> {url1}

到家 -> {url2, url3}

美好 -> {url3}

由检索词item快速找到包含这个查询词的网页Map<item, list<url>>就是倒排索引

。

搜索过程是怎样的？

假设搜索词是“我爱”，用户会得到什么网页呢？

1）分词，“我爱”会分词为{我，爱}

2）每个分词后的item，从倒排索引查询包含这个item的网页list<url>

我 -> {url1, url2}

爱 -> {url1, url2}

3）求list<url>的交集，就是符合所有查询词的结果网页

对于这个例子，{url1, url2}就是最终的查询结果

Design Search Engine - Storage

原始阶段-LIKE

数据在数据库中可能是这么存储的：

post(pid, title, content)

满足标题、内容的检索需求可以通过LIKE实现：

select pid from post where content like ‘%天通苑%’

Design Search Engine - Storage

原始阶段-LIKE

能够快速满足业务需求

（1）效率低，每次需要全表扫描，计算量大，并发高时cpu容易100%

（2）不支持分词

Design Search Engine - Storage

初级阶段-全文索引

建立全文索引

alter table t_tiezi add fulltext(title,content)

使用match和against实现索引字段上的查询需求。

fulltext 对文本 分词后倒排，至少不要全表扫描了

Design Search Engine - Storage

初级阶段-全文索引

优点：

快速提高效率，支持分词，并对原有系统架构影响很小

缺点：

数据量达到百万级别，性能还是会显著降低，查询返回时间很长，业务难以接受

比较难水平扩展

Design Search Engine - Storage

中级阶段-开源外置索引

为了解决全文索的局限性，当数据量增加到大几百万，千万级别时，就要考虑外置索引了

ES（ElasticSearch）

ES完全能满足10亿数据量，5k吞吐量的常见搜索业务需求

ES能够支持很大数据量的信息存储，支持很高并发的搜索请求

ES支持集群，向使用者屏蔽高可用/可扩展/负载均衡等复杂特性

Design Search Engine - Storage

中级阶段-开源外置索引

索引数据与原始数据分离

索引数据存在ElasticSearch中，原始数据依然存在 MySQL中

索引满足搜索需求，原始数据满足CURD需求

通过通知，定期重建来保证两种数据的一致性