调研报告

小组成员

- 高楚晴（组长）
- 王幸美
- 王章瀚
- 万嘉诚
- 黄志远

项目概述

在信息大爆炸的时代，文件系统与数据库高度成熟，有很多方法可以帮助用户完成高效的文件读写，节省时间空间成本，但对几乎所有文件系统，用户在写入新文件时，都必须要考虑这个文件存储的逻辑位置以备未来使用，可人类总会遗忘，或许现在你正有着某个文件被遗忘在某个角落。从文件系统的角度来说，这无疑会损耗存储空间，降低索引效率，我们不希望出现这种情况。然后一种朴素的想法迸发出来，文件系统是否可以帮助用户找到他/她想要的？

其实现在已经存在了某种类似的解决思路，即macOS的Spotlight，其用一个索引程序搜寻磁盘上的文件并建立索引，这将消耗数百mb的空间，同时因为其与文件系统本身分离，需要频繁地搜寻有什么东西发生改变，消耗CPU资源，而且最重要的一点是，文件在不同PC之间转移时，这个索引不随之移动，所以对于另外一部PC，这个索引是无意义的。

这显然不是我们想要的，因而要做的更好，我们必须向文件系统进军。如果文件系统能够理解文件的自然属性（比如识别某部电影的名称导演主演等），并依据其与不同的文件建立联系，那么文件系统自然可以帮助用户找到自己想要的东西，这便是DBFS（Database File System）。而图索引又是处理文件间关系的好手，于是本次大作业的选题自此敲定，基于图数据库的着重人与文件系统交互的GDBFS（Graph Database File System）。

项目背景

文件系统

文件系统提供在存储介质上组织数据的一种方式方法。其功能包括：管理和调度文件的存储空间，提供文件的逻辑结构、物理结构和存储方法;实现文件从标识到实际地址的映射，实现文件的控制操作和存取操作，实现文件信息的共享并提供可靠的文件保密和保护措施，提供文件的安全措施。

DBFS

DBFS是一种新类型文件系统，最早由Onne Gorter在他的硕士论文中开发，现在是一个sf.net项目。事实上，与其称DBFS为文件系统，不如称它为文档系统。它的设计焦点集中于用户，目标是，使得用户的文件管理更加轻松。它将查找文件的责任放在了计算机上，而用户不必记忆文件储存的位置。然后DBFS实际上并不存储文件，而是在基于基础层次结构的文件系统上保存对文件的引用。 GUI部分在KDE中实现，在其中替换了所有基于层次结构的文件访问。这给人的印象是没有层次结构，但是对于应用程序而言，什么都没有改变，打开文件和保存文件对话框具有相同的API。

API

API（Application Programming Interface，应用程序接口）是一些预先定义的函数，或指软件系统不同组成部分衔接的约定。目的是提供应用程序与开发人员基于某软件或硬件得以访问一组例程的能力，而又无需访问原码，或理解内部工作机制的细节。

层次模型

数据结构

数据库系统中最早出现的数据模型，层次数据库系统采用层次模型作为数据的组织方式。它采用**树形结构**来表示各类实体以及实体间的联系。

这种结构允许两种类型的数据之间一对多的关系。这种结构非常有效地描述了现实世界中的许多关系：配方，目录，段落/段落的顺序，任何嵌套和排序的信息。

此层次结构用作存储中记录的物理顺序。记录访问是通过使用指针与顺序访问相结合的方式在数据结构中向下导航来完成的。因此，如果没有为每个记录提供完整路径（与向上链接和排序字段相对），则对于某些数据库操作而言，层次结构效率很低。通过在基本物理层次结构上施加附加的逻辑层次结构，可以在更高的IMS版本中弥补这些限制。

优点

- 数据结构比较简单清晰
- 因为记录之间的联系用有向边表示，这种联系在DBMS中通常使用指针实现，查询效率高
- 提供了良好的完整性支持。进行插入操作时，如果没有对应的双亲结点就不能插入它的子女结点值。进行删除操作时，如果删除双亲结点，则相应的子女结点值也将被同时删除

缺点

- 现实世界中很多联系是非层次性的，它不适用于结点之间具有多对多联系； 如果结点之间具有多对多联系，不再适合使用层次模型表示。如果一个子女结点确实具有多个双亲结点，使用层次结构模型表示的时候就会出现大量的冗余，且操作复杂
- 查询子女结点必须通过双亲结点
- 由于结构严密，层次命令趋于程序化。

网状模型

网状数据模型的典型代表是DBTG（CODASYL）系统。

利用它的流行DBMS产品是Cincom Systems的Total和Cullinet的IDMS。IDMS获得了可观的客户群。在1980年代，除原始工具和语言外，它还采用了关系模型和SQL。

大多数对象数据库（于1990年代发明）使用导航概念来提供跨对象网络的快速导航，通常使用对象标识符作为指向相关对象的“智能”指针。例如，Objectivity / DB实现了可以跨越数据库的一对一，一对多，多对一和多对多命名关系。许多对象数据库还支持SQL，结合了两种模型的优势。

数据结构

满足以下两个条件的基本层次联系的集合为网状模型

- 允许一个以上的结点无双亲；
- 一个结点可以有多于一个的双亲。

备注：层次模型实际上是网状模型的一个特例。

集合由循环链接列表组成，其中一种记录类型（集合所有者或父记录）在每个圈子中出现一次，而第二种记录类型（下属或子项）在每个圈子中可以出现多次。这样，可以在任何两个记录类型之间建立层次结构，例如，类型A是B的所有者。同时，可以定义另一个集合，其中B是A的所有者。因此，所有集合都包括一个通用的有向图（所有权定义方向）或网络结构。对记录的访问可以是顺序的（通常是每种记录类型），也可以通过循环链接列表中的导航来访问。

优点

- 能够更为直接地描述现实世界，如一个结点可以有多个双亲，结点直接可以有多种联系；
- 具有良好的性能，存取效率较高
- 网络模型能够比分层模型更有效地表示数据冗余，并且从祖先节点到后代的路径可能不止一个。
- 网络模型的操作具有导航性：程序保持当前位置，并通过遵循记录参与的关系从一个记录导航到另一个记录。还可以通过提供键值来定位记录
- 通常通过直接寻址磁盘上记录位置的指针来实现设置关系。这提供了出色的检索性能

缺点

- 结构比较复杂，随应用环境的扩大，数据库的结构就变得越来越复杂，不利于最终用户掌握；
- 网状模型的数据定义语言和数据管理语言复杂 ，并且要嵌入某一种高级语言（C、COBOL）中，用户不容易掌握和使用；
- 由于记录之间的联系是通过存取路径实现的，应用程序在访问数据时必须选择适当的存取路径，因此用户必须了解系统结构的细节，加重了编写应用程序的负担。
- 以数据库加载和重组之类的操作为代价。

关系模型

最重要的一种数据模型

数据结构

一些术语

- 关系：一个关系对应通常说的一张表；
- 元组：表中的一行即为一个元组；
- 属性：表中的一列即为一个属性，给每一个属性起一个名称即属性名；
- 码：也称为码键，表中的某个属性组，它可以唯一确定一个元组；
- 域：一组具有相同数据类型的值的集合。属性的取值范围来自某个域；
- 分量：元组中的一个属性值。

- 关系模式：对关系的描述，一搬表示为：关系名（属性1，属性2，...，属性n）

关系模型要求关系必须规范化的，关系必须满足一定的规范条件，这些规范条件中最基本的一条就是，关系的每一个分量必须是一个不可分的数据项，也就是说，不允许表中还有表。

优点

- 建立在严格的数学概念的基础上；
- 概念单一，无论实体还是实体之间的联系都是用关系来表示。对数据的检索和更新结构也是关系（也就是我们常说的表）所以数据结构简单清晰，用户易懂易用。
- 它的存取路径对用户透明，从而具有更高的独立性、更好的安全保密性，简化了程序员的工作个数据库开发建立的工作

缺点

- 存取路径的隐蔽导致查询效率不如格式化数据模型（即层次和网状）
- 由于存取路径对用户是透明的，查询效率往往不如格式化数据模型。
- 为了提高系统性能，数据库管理系统必须对用户的查询请求进行优化

图数据库

图数据库类似于1970年代的网络模型数据库，两者都代表通用图，但是网络模型数据库的抽象级别较低，并且缺乏在边缘链上的轻松遍历。

使用图结构进行语义查询的数据库，其中语义节点，边和属性用于表示和存储数据。系统的关键概念是图形（或边或关系）。该图将存储中的数据项与节点和边的集合相关联，边代表节点之间的关系。这些关系允许将存储中的数据直接链接在一起，并且在许多情况下，可以通过一项操作来检索它们。图形数据库将数据之间的关系作为优先级。在图形数据库中查询关系的速度很快，因为它们永久存储在数据库本身中。关系可以使用图形数据库直观地可视化，从而使其对于高度互连的数据很有用。

存储机制

有些依赖关系引擎，并将图形数据“存储”在表中（尽管表是逻辑元素，因此此方法在图形数据库，图形数据库管理系统和数据所在的物理设备之间强加了另一个抽象层次实际存储）。

其他人则使用键值存储或面向文档的数据库进行存储，从而使其固有地具有NoSQL结构。大多数基于非关系存储引擎的图形数据库还添加了标签或属性的概念，这些标签或属性本质上是具有指向另一个文档的指针的关系。这允许对数据元素进行分类，以方便整体检索。

从图形数据库检索数据需要使用SQL以外的查询语言，而SQL是为关系系统中的数据处理而设计的，因此无法“优雅地”处理遍历图形。截至2017年，还没有像SQL关系数据库那样普遍采用单一图查询语言，并且存在各种各样的系统，大多数情况下都与一种产品紧密相关。已经进行了一些标准化工作，从而导致了诸如Gremlin, SPARQL和Cypher的多供应商查询语言。除了具有查询语言接口之外，还可以通过应用程序编程接口（API）访问某些图形数据库。

立项依据

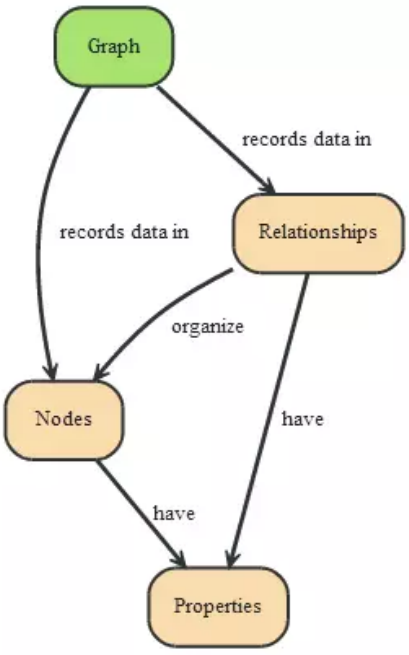
本项目旨在利用neo4j搭建图数据库，采用多种方式确立文件的自然属性构建索引

neo4j

Neo4j是一个高性能的NOSQL图形数据库，它将结构化数据存储在图上而不是表中。它是一个嵌入式的、基于磁盘的、具备完全的事务特性的Java持久化高性能引擎，该引擎具有成熟数据库的所有特性。我们计划使用neo4j完成图数据库的搭建。

存储结构

在一个图中包含两种基本的数据类型：Nodes和 Relationships。Node和Relationship 的 Property 是用一个 Key-Value 的双向列表来保存的。Nodes通过Relationships所定义的关系相连起来，形成关系型网络结构。Node 的 Relatsionship 是用一个双向列表来保存的，通过关系，可以方便的找到关系的 from-to Node. Node 节点保存第1个属性和第1个关系ID。不使用schema，因此可以满足任何形式的需求，示意图如下。



算法

Neo4j自带多种搜索算法，功能强大，能够满足GDBFS的潜在需求

中心性算法（Centrality algorithms）

中心度算法主要用来判断一个图中不同节点的重要性

社区发现算法（Community detection algorithms）

评估一个群体是如何聚集或划分的，以及其增强或分裂的趋势

路径寻找算法（Path Finding algorithms）

用于找到最短路径，或者评估路径的可用性和质量

相似度算法（Similarity algorithms）

用于计算节点间的相似度

链接预测算法（Link Prediction algorithms）

有助于确定一对节点的紧密程度。然后，我们将使用计算的分数作为链接预测解决方案的一部分

优势

与许多图数据库不同，Neo4j是一个原生的图数据库引擎，可以使用图结构的自然伸展特性来设计免索引邻近节点遍历的查询算法，即图的遍历算法设计。图的遍历并不受数据量的大小所影响，因为邻近查询始终查找的是有限的局部数据，不会对整个数据库进行搜索。所以，Neo4j具有非常高效的查询性能，相比于RDBMS可以提高数倍乃至数十倍的查询速度。而且查询速度不会因数据量的增长而下降。同时RDBMS因为不可避免地使用了一些范式设计，所以在查询时如果需要表示一些复杂的关系，势必会构造很多连接，从而形成很多复杂的运算。

文件的自然属性

能够合理地描述文件的自然属性是GDBFS的基础，找到如何描述文件的方法是必不可少的，然而研究具体如何获取自然属性在定位上并不属于操作系统的范畴，不是我们的主要工作，在工作量上也太过繁重，我们预备采用互联网上公开提供使用的或开源的内容来帮助GDBFS完成文件自然属性的识别。因为网络上相关工作是针对不同文件类型进行处理的，我们也针对不同文件类型准备了如下备选方案

文档

讯飞开放平台

讯飞开放平台是一个开放的智能交互技术服务平台，开发者可以使用语音合成、语音识别、语音唤醒、语义理解、人脸识别等多项服务。在文档处理方面具体到提供了多种形式的文字识别，词法分析，依存句法分析，语义角色标注，语义依存分析，情感分析和关键词提取的功能。

ParallelDots

ParallelDots针对中文文档处理的功能尚有欠缺，但是对于英文文档处理功能相当全面，包括情感分析（Sentiment Analysis），情绪分析（Emotion Analysis），关键词提取，意图分析，潜在语义分析，文本分类，辱骂分容分类器（Abusive Content Classifier），讽刺检测（Sarcasm Detection）。

LinguaKit

LinguaKit包括情感分析，语言分析，关键词提取(一次性支持最多5000字)，词组提取，专用名词识别等

cortical.io

cortical.io作为一个付费平台，提供了一些免费服务，包括关键词提取，文档比较，歧义条款（Disambiguate Terms）

doc2vec

Doc2Vec 或者叫做 paragraph2vec, sentence embeddings，是一种非监督式算法，可以获得 sentences/paragraphs/documents 的向量表达，是 word2vec 的拓展。学出来的向量可以通过计算距离来找 sentences/paragraphs/documents 之间的相似性，可以用于文本聚类，对于有标签的数据，还可以用监督学习的方法进行文本分类，例如经典的情感分析问题。

音频

对于音乐一类的音频，有很多公开平台做到了很好的识别正确率与识别速度，但是对于日常生活中的音频文件，自然属性的处理变得麻烦起来，我们预定对日常音频文件做声纹识别，语音转写并分析语义等多种方式综合处理

讯飞开放平台

讯飞开放平台在语音识别方面具体到提供了语音听写，语音转写，声纹识别，性别年龄识别和歌曲识别的功能。

音乐播放器的音乐识别

类似网易云音乐，QQ音乐等多款音乐播放器拥有音乐识别的功能

翻译软件的语音转写

类似有道词典，谷歌翻译，微软翻译等翻译软件因为用户需求已经形成了强大的语音转写功能。

图像与视频

对于图像，我们直接采用图像识别提取自然属性，对于视频文件，我们采取先提取关键帧再图像识别的方式，故将这两者方面一起

Clarifai

提供相似图片查找服务，能够识别图中多个要素，并按置信度排序，api可申请使用。

Waston Recognition(IBM)

提供高度可定制平台，能够分析来自场景、物体、面部、颜色、食物、品牌及其他内容的图像，可以同时识别多个要素并按置信度排序，提供免费的api。

Imagga

Imagga的图像识别工具提供了多种自动选项，用于根据类别，颜色，标签或自定义输入对图像进行分类，组织和显示。能够识别图中多个要素，并按置信度排序，提供免费的api。

百度识图

百度识图的成功率相比前面几个个能并不高，但是具有百度百科的天然优势，在识别成功后，会把相应的百度百科（如果有的话）展示出来

whatanime

whatanime拥有强大的影视作品识别能力，可以识别出图片具体出现在影视作品的第几季第几集（或者哪部电影），并给出相应的百科。

FFmpeg

FFmpeg是一套可以用来记录、转换数字音频、视频，并能将其转化为流的开源计算机程序。采用LGPL或GPL许可证。它提供了录制、转换以及流化音视频的完整解决方案。它包含了非常先进的音频/视频编解码库libavcodec，为了保证高可移植性和编解码质量，libavcodec里很多code都是从头开发的。

FFmpeg在Linux平台下开发，但它同样也可以在其它操作系统环境中编译运行，包括Windows、Mac OS X等。这个项目最早由Fabrice Bellard发起，2004年至2015年间由Michael Niedermayer主要负责维护。许多FFmpeg的开发人员都来自MPlayer项目，而且当前FFmpeg也是放在MPlayer项目组的服务器上。项目的名称来自MPEG视频编码标准，前面的"FF"代表"Fast Forward"。

因为FFmpeg在关键帧提取方面非常优秀，我们并没有提供相应的关键帧提取的备选方案。

模糊搜索

在用户主动搜索文件的过程中，我们需要对用户的需求与现有自然属性匹配，这就用到了模糊匹配，我们准备了以下备选方案

word2vec

利用word2vec提取语义向量，在多维空间内寻找最近的一个（或多个）自然属性点来实现模糊搜索。

前瞻性/重要性分析

文件系统的智能化

智能生活概念目前是科技领域的热点，涉及到智能家居、智能出行、智慧城市等一系列相关研究领域。智能这一概念出现在了生活的方方面面，然而对于文件系统本身，或者说操作系统的智能化却鲜有关注，然而一个智能的文件系统却是至关重要的，它不仅可以避免用户遗忘带来的损失，还可以极大地改善用户体验，堪比文件系统中的自动驾驶，GDBFS拥有显而易见地前瞻性和重要性

发掘利用文件的自然属性

发掘利用文件的自然属性，为构建文件索引提供了一个不同于以往添加tag的新思路，通过文件的自然属性构建的索引具有优秀的稳定性以及泛用性，是对未知领域的探索。

图数据库的开发

图天生关注对象和对象之间的关系，可以高效插入大量数据，查询关联数据，与以文件的自然属性为基础构建的GDBFS高度契合，但其应用范围并不广泛，主要应用于社交网络和推荐引擎。GDBFS创新地将图数据库与DBFS相结合，拓宽了图数据库的应用范围，推动了DBFS地发展。

高可扩展性

第三方可以自行添加关联算法与相似算法，用户可以选择使用的算法，使得GDBFS实际上可以构建基于不同逻辑的索引，也有利于GDBFS构建更加真实的自然属性索引，对于一个开源项目的开发和进步有重要意义。

相关工作

数据库文件系统

Oracle数据库

Oracle数据库通常用于存储与数据库应用密切相关的文件，包括 CAD，医学图像，发票图像，文档等。应用程序使用SQL标准数据类型 BLOB（和CLOB）将文件存储在数据库中。与传统文件系统相比，Oracle 数据库提供了更好的安全性，可用性，鲁棒性，事务和可扩展性。当文件存储在数据库中时，它们将被备份，使用Data Guard同步到灾难恢复站点，并与数据库中的关系数据一起恢复。

在Oracle Database 中，Oracle引入了Oracle SecureFiles LOB为文件提供高性能存储，与传统文件系统的性能相当。 SecureFiles LOB支持对文件进行压缩，重复数据删除和加密的高级功能。 由于SecureFiles LOB保持与BLOB（和CLOB）的向后兼容性，所以针对 BLOB编写的应用程序继续透明地使用SecureFiles LOB，即使用前面提到的功能。

sf.net项目

DBFS最先由University of Twente的Onne Gorter进行开发，现已成为sf.net项目。如今KDE的core中已经加入了DBFS，打开文件和保存文件界面已经被DBFS替代，然而其底层始终是传统的层次化文件系统，只是向用户呈现出了DBFS的表象。 sf.net的这个项目最开始是为了实现使用户能够通过文件的各种特征来找到文件的便捷性。例如，当用户需要找到一个文件，那么这个DBFS将向用户收集该文件的信息，其中包括但不限于：这个文件上个月被编辑过吗？这个文件是否是一个文字文档？这个文件是否属于某个特定的项目？有了这样一些问题，DBFS的搜索范围逐渐减小，最终确定在一些文件上，甚至能直接定位到用户想要的文件。 相较于传统的文件系统使用的目录结构，DBFS中更多地使用关键词。并且在DBFS中，这样的一些关键词更像是目录的超集。而sf.net项目中的DBFS有几个比较重要的设计。首先是，DBFS不存储系统文件，也就是它不存储libraries，或字体文件等。主要原因是这些都不是文档，不是日常生活中会去查找的，也不太方便在DBFS中存储。其次，一些应该在一起的文件，应该视为一个文档。举例来说，一个音乐文件，它的对应的歌词文件应该和这个音乐文件绑定在一起作为一个文档；或者一部电影有两个部分，这两个部分虽然是两个文件，但在DBFS中也应该被视为一个文档。

图文件系统

参考文献

http://dbfs.sourceforge.net/