基于 E－R 模型的 NTFS 文件系统分析研究

Analysis Research of NTFS File System based on E－R Model

戚 岳　李 曦　龚育昌　（中国科学技术大学计算机科学技术系 合肥 230027）

摘要：本文从数据库 E－R 模型的角度分析了 NTFS 文件系统的特点，给出其对文件/目录的表示方法和逻辑组织结构，并讨论了 NTFS 的属性和索引两大技术关键点，以帮助开发人员深入理解 NTFS 的内部结构。同时对 WinFS 及文件系统的发展方向进行了探讨，以拓展文件系统设计人员的思路。

关键词：E－R 模型　NTFS　WinFS

1　引言

从文件系统的演变过程中可以看到，数据库技术正在逐渐被应用到文件系统的设计与实现当中。本文从数据库 E－R 模型的角度对 NTFS 进行分析，不仅给出其文件/目录的表示方法和磁盘逻辑组织结构，而且研究了 NTFS 特点对 WinFS 的支持以及文件管理方式的新思路，用以帮助系统开发人员更好的理解 NTFS 内部结构，同时为文件系统设计人员拓展新思路。文章第 2、3、4 部分将分别对 NTFS 的 E－R 模型、磁盘逻辑布局和索引技术进行详细分析，第 5 部分讨论了未来文件系统 WinFS 的设计思想，并在最后给出结论。

2　NTFS 文件系统的 E－R 模型

实体－关系数据模型（简称 E－R 模型）是采用实体、关系、属性三者来描述数据库结构的模型，其中，实体是可以被区分的某个抽象事物，关系是两个或多个实体之间的关联，而属性是实体所具有的性质。卷、文件、目录是文件系统的三个基本结构，NTFS 也不例外，因此当我们使用 E－R 模型对 NTFS 建模时，如图 1 所示，这三者被当作三个独立实体，模型着眼于描述它们的关系和各自的属性表示。

2.1　NTFS 文件系统的 E－R 模型描述

从实体关系上看，图 1 中的二元关系 location 表示两个实体的位置关系，即文件/目录在卷上，文件在目录中；箭头表明惟一性，即每个文件/目录只在一个卷上。显然，二元关系的集合可以描述文件系统的树型结构。三元关系 path 利用三个实体的位置关系描述了文件系统中路径的概念。卷、文件、目录实体的关系表明 NTFS 的基于文件/目录模型的管理方法并没有本质的改变。

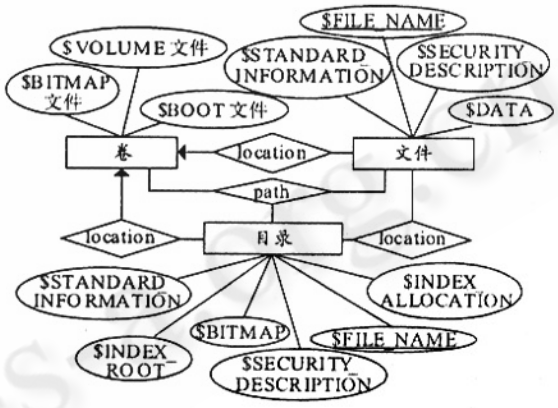

图 1　NTFS 的 E－R 模型图

从实体本身来看，文件和目录实体都是由多个属性来描述的。图 1 列出的属性都是文件和目录的基本属性。$ STANDARD_INFORMATION 属性主要描述文件/目录创建、修改、最近访问的时间（统称为时间戳）以及兼容 DOS 操作系统的文件信息。$ FILE_NAME 属性主要描述文件/目录名、文件大小、时间戳等。$ SECURITY_DESCRIPTOR 属性存放文件/目录安全描述数据。对于文件实体来说，$ DATA 属性存放文件的真正数据。对于目录实体来说，$ INDEX_ROOT 属性存储用于索引的 B＋树根节点，$ INDEX_ALLOCATION 属性存储 B＋树的所有子节点，而 $ BITMAP 属性描述索引块

的使用位图。

"卷"实体由三个元文件 $ VOLUME、$ BITMAP 和 $ BOOT 定义。$ VOLUME 和 $ BITMAP 文件就是 NTFS 文件实体的具体实例，但是普通文件所具有的属性还不能完全描述"卷"实体的信息，$ VOLUME 文件就增加 $ VOLUME_INFORMATION 和 $ VOLUME_NAME 两个属性，用来描述卷的版本、状态信息和名称。$ BITMAP 文件使用 $ DATA 属性存储簇的使用位图。$ BOOT 文件在加载 NT 系统前使用，与传统启动扇区格式兼容。

2.2 属性对文件信息描述能力的扩展

文件包含的信息可以用为两大类，即用户数据和元数据。元数据是对用户数据的补充，但并不是用户数据本身。比如，数码照片的真正数据是图像的编码串，其元数据可以包括照片上人物的名字，拍照的时间等等。文件系统中，很多信息都是以元数据的形式给出的，而且信息之间也多以元数据方式关联。比如，数码照片的人物元数据与 E－MAIL 的发件人元数据可以关联起来。所以，信息描述能力的扩展很大一部分来自于对元数据支持能力的扩展。

对于元数据的支持，在传统文件系统中，一种方法是采用与文件/目录相关的数据块（例如，EXT2 文件系统的 inode 节点和 FAT 文件系统的目录项）来存储元数据集合，每个元数据就是从该数据块中按照固定字段提取出来的。另一种方法是在用户数据的头部存放通用的元数据信息，比如数码照片的 EXIF（Exchangable Image File Format）元数据头在照片拍摄时就存储在该照片数据的头部。可以设想，如果要扩展元数据信息，无论是元数据块，还是元数据头信息，都需要添加新的字段，这使得上层的解析软件甚至底层的磁盘逻辑组织都要随之调整，而且添加字段时是还要考虑到兼容性问题。

根据 NTFS 的 E－R 模型分析可知，文件和目录相当于存储在关系数据库中的元组，元组内部存储多个属性，其中，$ DATA 属性存储用户数据，其它属性存储元数据。这种属性描述的方法使得信息描述粒度更小、管理能力更强。相比较之下，如果要扩展元数据信息，NTFS 只要为文件/目录增加新属性即可，因为 NTFS 是以属性为单位管理元数据的；同时提供给上层的 API 也支持属性操作，因此用户软件的改动是有限的，这更适用于现在越来越复杂的数据存储要求。不仅如此，在文章后面我们可以看到，NTFS 可以根据某属性建立索引结构，提供快速检索能力，同时，在新一代文件系统 WinFS 中，使用属性可以使得不同文件之间的可能关联的信息越来越多。

3 NTFS 逻辑结构组织

根据数据库设计思想，模型设计最终要映射到物理设计。NTFS 的文件/目录采用属性表示方式，同时 NTFS 利用数据库索引技术完成对属性快速检索的支持。因此 NTFS 底层设计的两个关键的问题就是属性存储和索引结构设计。

3.1 NTFS 的逻辑结构

如图 2 所示，NTFS 的磁盘布局主要分为四个部分：主文件表 MFT（Master File Table）、MFT Zone、元文件备份区和文件存储区。MFT 使用连续磁盘空间，并且被分成大小固定的记录项（称 MFT 记录项）。每个文件（目录被看作特殊的文件）占用一个或多个记录项，组成一个完整的文件记录，存储了文件所有的属性信息。每个文件由 MFT 记录项编号（称为 MFT 记录号，从 0 开始）唯一指定。上文提到的 $ BOOT、$ VOLUME、$ BITMAP 文件等元文件，是操作系统运转和维护文件系统所需要的文件，对用户透明，在 MFT 中占用最前面的十六个记录项。为 MFT 预留的连续空间 MFT Zone，根据配置，在磁盘初始化的时最大可以占到磁盘容量的 50%。每当缺少磁盘空间时，MFT Zone 就会减半。元文件备份区设置于剩余磁盘的中间位置。文件存储区用来存放属性值。

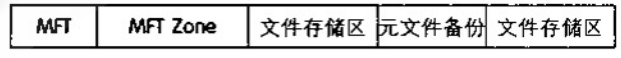

图 2 NTFS 磁盘布局图

3.2 NTFS 的属性存储方法

NTFS 文件系统的每个属性由属性类型和属性名字共同标识。属性类型是枚举整型值，属性名字是 Unicode 字符串或空。比如，文件的 $ DATA 属性的属性类型是 0x80，属性名字为空；目录的 $ ROOT_INDEX 属性的类型是 0x90；属性名字 $ I30。每个属性是属性头和属性值的二元组。属性存储采用可变长记录格式，即属性头和属性值的长度都可以变化。属性头存

放属性本身特性的描述，包括属性类型、名称等信息，相当于属性的元数据；属性值中存储属性的真正数据。按照属性值存储位置的不同，属性又可以分成驻留属性和非驻留属性。驻留属性是存放在 MFT 记录项中的，紧跟在对应属性头的后面；非驻留属性的属性值存储在文件存储区中，在记录项中只保存 RUN 列表。在 RUN 列表中，第一个 RUN 记录了开始簇的位置和长度，其它的 RUN 中记录相对前一个 RUN 开始簇位置的偏移和长度。这样可以得到存储属性值使用的所有磁盘簇。当读取属性值时，驻留属性可以直接读取，对于非驻留属性，还要翻译 RUN 列表，再进行一次磁盘访问。

MFT 记录结构（如图 3 所示）由 MFT 记录头和多个属性组成。MFT 记录头包含记录项基本信息（比如：记录号等）以及相对第一个属性位置的偏移量。而其后每个属性的属性头中除了属性基本信息外还包含两个量，一个是相对属性值位置的偏移，一个是整个属性的长度。这样就可以串联起所有的属性。MFT 记录中的属性排列是有序的，先按照属性类型的整数值升序排列，如果属性类型相同再按属性名字的字符串升序排列，这样便于在 MFT 记录项中查找属性。

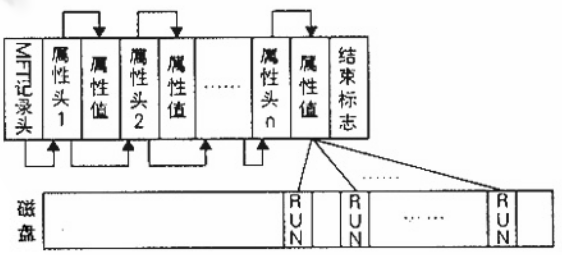

图 3　MFT 记录结构

当单个 MFT 记录项不能存储所有属性信息时，NTFS 首先会将属性中可以变作非驻留的属性全变成非驻留的，倘若记录项空间还是不够，就需要将属性信息分散到多个不一定连续的 MFT 记录项中。如图 4 所示，第一个 MFT 记录项（主 MFT 记录项）中使用非驻留的 $ ATTRIBUTE_LIST 属性（简称为属性列表）存储属性分布情况，它的每个单元记录了属性类型、属性名字、开始 VCN（Virtual Cluster Number）和属性所在 MFT 记录项的编号。在属性列表中查找到目标属性后，根据其 MFT 记录号定位到目标 MFT 记录项。当 RUN 列表被分散到多个 MFT 记录项中时，每个 RUN 列表片段在属性列表中都有对应的单元，如图中属性 i 的 RUN 列表存储在两个 MFT 记录项中。VCN 是对属性值以簇为单位划分后的数据块的编号（从 0 开始），开始 VCN 就标识出当前 RUN 列表片段指明的数据在整个属性值中的开始位置，以此来定位多个 RUN 片段的顺序。这样，使用多个 MFT 记录项的属性存储问题就完全解决了。

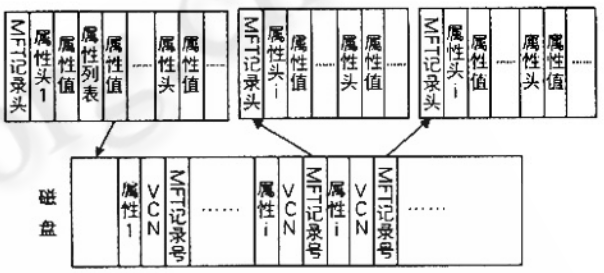

图 4　带有属性列表的文件记录结构

4　NTFS 的索引技术

索引技术使得 NTFS 可以快速查找属性信息。最好的例子就是目录的索引结构，它以存储在 $ FILE_NAME 属性中的文件名为关键字构建 B + 树结构。B + 树的节点就是索引块，每个索引块由多个索引项组成。NTFS 使用三个属性来存储 B + 树，$ INDEX_ROOT 是驻留属性，当索引项少到可以全部放到该 MFT 记录项中时，B + 树就直接存放其中，相当于 B + 树只有一个根节点。当索引项在 MFT 记录项中存储不下时，就引入 $ INDEX_ALLOCATION 和 $ BITMAP 属性来存储索引结构。非驻留的 $ INDEX_ALLOCATION 属性，以簇或簇的整数倍大小为单位存储索引块，可以使用 VCN 指出索引块的位置。而 $ BTIMAP 属性记录索引块的使用位图，以便在操作 B + 树时使用。每个索引项中包含四个主要部分，分别是文件记录的主 MFT 记录号、文件记录中的 $ FILE_NAME 属性、构建 B + 树所需的叶子节点标识以及叶子节点的 VCN 号。其中，叶子结点标识指出当前索引项有没有孩子节点，如果有孩子节点，那么 VCN 号就给出该孩子节点（索引块）的位置。这样，如图 5 所示，对于一个索引项的查找，按照文件名，首先对 B + 树的索引块进行检索，找到索引块后，再在块内

顺序查找索引项。虽然对 B + 树的操作比较复杂，但基于 B + 树的目录项检索是非常快的，而且 $ FILE_NAME 属性包括了文件/目录的基本信息，不需要访问文件/目录的 MFT 记录项，减少磁盘访问次数。

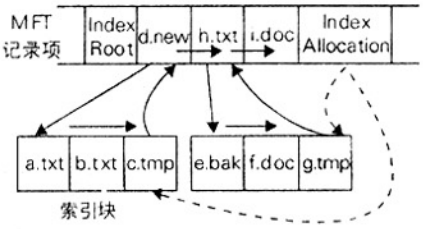

图 5　目录索引结构图

索引技术的应用不仅局限于目录访问，其它典型的例子还有文件/目录的安全描述子。安全描述子描述了文件/目录的访问控制信息。在 NTFS3.0 以前每个文件/目录都用 $ SECURITY_DESCRIPTOR 属性来存储各自的安全描述子，如果相同的安全设置被应用到整个目录树上，树上的每个文件和目录可能有相同的安全描述子，全部存储会造成大量的磁盘浪费。NTFS3.0 引入系统元文件 $ Secure，在属性名为 $ SDS 的 $ DATA 属性（其后简称 $ SDS 属性）中存储全部的安全描述子信息，同时建立两个索引结构，一个是 $ SDH 索引，它将安全描述子关键字的哈希值按 B + 树存储，每个索引项中记录该安全描述子在 $ SDS 属性值中的偏移和长度，这样可以在 $ SDH 索引结构中查找一个安全描述子是否存在。另一个是 $ SII 索引，它将分配给每个文件/目录的安全 ID 号以 B + 树存储，同样每个索引项中记录该安全描述子在 $ SDS 属性值中的偏移和长度，这样文件/目录使用各自的安全 ID 号，就可以得到对应的安全属性。总之，索引技术提供的快速定位能力使得系统更加紧凑而高效。

5　未来文件系统 WinFS

WinFS（Windows Future System）是微软下一代文件系统平台，为用户提供强大的搜索和查询功能，并且可以定义不同数据文件之间的关联，在此基础上，还可以定义规则，利用数据之间的关系驱动信息处理动作，以提高系统自动化。

如图 6 所示，WinFS 主要由四部分组成，分别是 NTFS、关系数据库引擎、WinFS 应用框架和应用编程接口（API）模型。应用框架又分为四个模块，WinFS 核心提供基本的文件系统操作和服务，如安全服务、卷管理等；数据模型提供信息的基本结构类型、关系类型及其扩展；模式基于数据模型来描述日常所需信息的数据组织形态；服务模块提供系统信息高层服务，包括：与其它存储平台的同步操作、用户数据视图个性化 InfoAgent 服务等。API 模型除了提供丰富的 API 集合外，还为用户提供对象、T/SQL 和 XML 三种接口。

图 6　WinFS 结构示意图

6　结论

内嵌数据库的未来文件系统，可以在高效管理信息数据以及信息之间关系的同时，为用户提供友好而简单的应用环境，满足日益增强的信息处理要求。

参考文献

1 Microsoft Corporation. Microsoft Extensible Firmware Initiative FAT32 File System Specification V 1.03. Dec. 6, 2000. http://www.microsoft.com/whdc/system/platform/firmware/fatgen.mspx.

2 R. Card, T. Ts'o, and S. Tweedie. Design and Implementation of the Second Extended Filesystem. In 1st Dutch Int. Symp. On Linux, 1994.

3 Daniel P. Bovet, Marco Cesati. Understanding The Linux Kernel 2nd. O'Reilly. 2002.