第 IV 部分

DICOM 媒体和安全

Chapter 10

DICOM 媒体：文件、文件夹和 DICOMDIR

本书的重点是 DICOM 网络以及通过 TCP/IP 连接的 DICOM AE 之间传送的数据和消息。这的确是最常见的和高效的交换医学影像数据的方法。计算机网络提供非常卓越的灵活性、范围和生产力，允许你在任何时间、任何地点与任何伙伴进行协作。

然而，许多时候我们仍然需要从独立的临床网络中向其他媒体导出 DICOM 数据，比如闪盘、CD/DVD、MOD 或者其他硬盘。我们不是在说工业级的 PACS 归档存储：那是 PACS 厂商应该管的事情（这将会在之后的 10.5 中介绍）。DICOM 媒体存储（DICOM Media Storage）偶尔使用便携媒体来从 PACS 向外部存储导出 DICOM 数据，用来浏览或者传输到其他 DICOM 应用程序中。

经典的实例是使用 DICOM CD（现在通常用 DVD 代替，因为 DVD 能容纳更多数据）。一个病人可能会在某医院进行 CT 扫描。医院可能并没有 PACS 或者没有一套能与医院信息系统集成的 PACS 系统。因此，医院向病人提供刻有他自己 DICOM CT 图像的光盘，病人可以将光盘带到任何其他对他进行治疗的医疗机构。

这超出了一个典型的场景；如你在第 7 章中所见，病人实际上担当了 DICOM 网络（C-Store DIMSE）的角色，而他口袋里的 CD 担当了相关的 DICOM 数据对象。

DICOM电子邮件是另一个流行的无PACS网络的替代方法。原始DICOM文件可以作为电子邮件的附件发送到别人那里（比如咨询医师）。现在，电子邮件程序充当了DICOM网络“C-Store”并且使用电子邮件的附件用来传输DICOM数据，然而在幕后，这些数据都会使用与电子邮件兼容的多用途互联网邮件扩展（Multipurpose Internet Mail Extensions (MIME)）格式[1]。

如前所述，在这些方法中，没有哪个方法能与 DICOM 网络的生产力去比拼，但是在一些很小的或者罕见项目中，他们确实提着关键作用。DICOM 标准的 PS3.10、PS3.11 和 PS3.12 涉及所有 DICOM 文件和媒体的规格，并且其中一些将在此章节中出现。

10.1
DICOM 文件格式

当谈论 DICOM 媒体时，我们其实就是在谈论 DICOM 文件。DICOM 文件存储在 DICOM 数据对象（也称作数据集）中，最常见的形式为 DICOM 图像。数据对象写入 DICOM 文件的方法与 DICOM 网络中的编码规则完全一致，比如显式或隐式 VR 编码（见 5.5）。唯一的不同是二者的 DICOM 文件头。它位于数据对象之前，见图 72。DICOM 头起到重建丢失的 DICOM 连接的作用：它向文件读取程序描述文件中存储的 DICOM 数据的 SOP 类型；以及传输语法格式。DICOM 头包括一个报头以及一个 DICM 前缀，以及少量 DICOM 文件的属性（文件元信息元素）。

10.1.1
报头和 DICM 前缀

报头是一个 128 字节的字符串，它负责引导开启 DICOM 文件。报头广泛用在许多成像和数据格式中，DICOM 也采用了这种方法。然而，DICOM 标准没有定义任何特定的报头结构或内容。它完全取决于每个 DICOM 应用程序，这些 DICOM 应用程序会按照自己的想法来使用这 128 个字节的报头。这就使得报头会跟着应用程序变化；不同的应用程序可能使用完全不同的报头。因此，DICOM 中的报头通常都会被忽略或者填入 0 字节；在 DICOM 中这就

[1] MIME 是一个互联网标准，它扩展了电子邮件的格式，支持非文本附件，包括 DICOM 二进制对象（文件）

表示“未使用的报头”。

DICM 前缀（体现 DICOM 文件格式）紧跟在 128 字节的报头之后。它其实就是有四个简单的大写字母“D I C M”写在第 129-132 个字节中。格式化前缀（通常称作魔法数字）的用法在其它的文件格式中也非常普遍而 DICOM 也遵循这一习惯。

报头和 DICM 前缀都没有使用 DICOM VR 编码规则。只不过是简单地存储在最前面的 128＋4＝132 个字节中。如果你正在编写识别 DICOM 文件的程序，那么请略过前 128 个字节，并且校验一下 DICM 前缀。

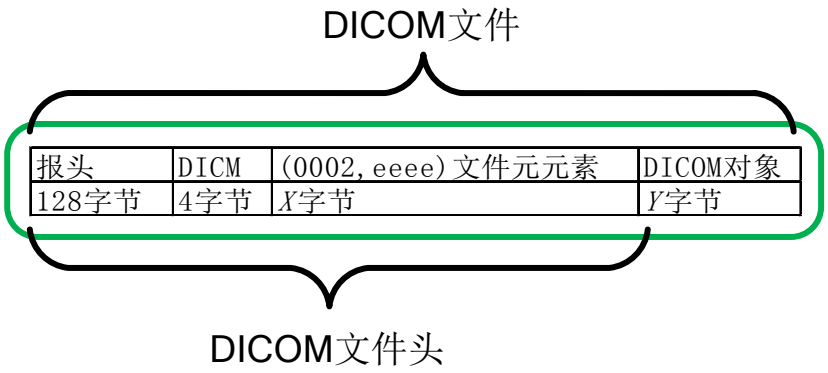

图 72 DICOM 文件结构

10.1.2
0002 组：DICOM 文件元信息（File Meta Information）

就在 DICM 前缀之后，从第 133 个字节开始，我们会看到 DICOM 文件原信息（File Meta Information）。与报头和 DICM 前缀不同，元信息使用显式 VR 编码成一套 DICOM 属性。文件元信息中的所有属性属于 DICOM 的 0002 组，如表 49 所示。表格中的最后四个元素是可选的，因此可以略去。必须的元素，要么是来自 DICOM 数据对象属性（SOP，传输语法）的，要么是负责对文件创建应用程序的属性进行记录的。整个 0002 组最重要的元素大概就是“传输语法 UID（Transfer Syntax UID）” (0002,0010)了，它定义了之后的 DICOM 数据对象是如何编码的。如果你记得（见 9.4），在 DICOM 网络中传输语法是在 DICOM 连接建立时进行沟通的；现在被记录在了 DICOM 文件头中。

在之后的章节（10.2.1）我们讨论 DICOMDIR 结构的时候，你会看到 0002 组用法的另一个实例。那时还会展示 0004 组的应用实例，此处我们就不赘述了，但它确实和 0002 组的用法非常相近。

表 49 文件元信息

属性名	标签	VR	属性描述
组长度	(0002,0000)	UL	在 0002 组中，从文件元元素开始（值字段的结尾）到最后一个文件元元素截至的字节数。

文件元信息版本	(0002,0001)	OB	这是一个 2 字节的字段，每一个字段都标识文件元信息头的一个版本。在当前的 DICOM 版本中，第一个字节值为 00H，第二个为 01H。
媒体存储 SOP 类 UID	(0002,0002)	UI	唯一标识与 DICOM 数据对象关联的 SOP 类。影像设备类型决定了这个元素的值；换句话说，它会包含影像设备的 C-Store SOP（见 7.3） *实例：1.2.840.10008.5.1.4.1.1.1（CT 图像存储）。*
媒体存储 SOP 实例 UID	(0002,0003)	UI	唯一标识与 DICOM 数据对象关联的 SOP 实例，此 DICOM 数据对象存在文件中，文件元信息之后。这个值来自图像 SOP 实例 UID 属性(0008,0018) *实例：1.2.840.10008.2008.03.25.12.33.55.7*
传输语法 UID	(0002,0010)	UI	唯一标识用来编码后续数据对象的传输语法。此传输语法不用于文件元信息 *实例：1.2.840.10008.1.2.1（显式 VR 小端字节序）*
实现类 UID	(0002,0012)	UI	唯一标识撰写文件和文件内容的实现。它提供一个清晰的实现识别方法，该实现在交互问题的事件中曾最后编写过文件。 *相同的值在连接建立时也用在实现标识项中（见 9.7）*
实现版本名	(0002,0013)	SH	标识一个实现类 UID (0002,0012)的版本，最多用 16 个字符 *相同的值在连接建立是也用在实现标识项中（见 9.7）*
源 AET	(0002,0016)	AE	书写该文件内容（或最后更新文件）的 AE 所使用的 DICOM AET。如果使用，它允许在媒体交换问题的事件中跟踪错误的源头 相同的值在连接建立时也用在发起呼啸的 AET 中（见 9.8.1）
私有信息创建者 UID	(0002,0100)	UI	私有信息(0002, 0102)的创建者 UID
私有信息	(0002,0102)	OB	包含在文件元信息中设置过的私有信息

10.1.3
数据对象

DICOM 数据对象紧跟在组 0002 后面，负责存储实际的 DICOM 数据；即我们常见的 DICOM 数据对象，在 DICOM 网络中我们已经很多次用到过它。DICOM 对象组从 0008 组开始编号（见 DICOM 数据字典），因此应用程序很容易区分哪里是文件元信息（0002 组）的结尾，哪里是数据对象（0008 组）的起始。

如果你正在编写 DICOM 应用程序，此处需要非常小心。比如，你必须用显式 VR 语法来编码文件元信息（0002 组），但是有可能并不必要将这种编码用在 DICOM 数据上（0008 组或更高的组别）。如果你的 DICOM 应用程序不能转换到数据对象编码所使用的 VR 语法上（如“传输语法”(0002,0010)字段所指出），那么数据读取一定会失败。就因此，为了支持 DICOM 压缩格式（使用显式 VR 编码），DICOM 建议在整个 DICOM 文件中始终使用显式 VR 编码；即也还是为了能够正常处理 DICOM 数据部分。

最终，DICOM 文件的最后可以追加“数据集后缀”(FFFC,FFFC)元素，其目的是为了达到某长度。真的需要吗？DICOM 标准说不用。如果你的应用程序进入到了这些后缀元素中，那么由于你一定到达了 DICOM 文件的结尾，因此完全可以忽略它并且友好地离开。大多数 DICOM 文件和软件并不适用(FFFC,FFFC)元素。

可以很好地给 DICOM 文件格式回顾下个结论了。很简单，你一直在使用的 99.9%的 DICOM 文件会符合这个排布，但还有 0.1%不太一样的文件很可能是因为文件损坏造成的。

10.1.4
DICOM 文件 ID 和文件名

我们曾几次提到过，根本没法用 DICOM 文件名称来分辨他们。然而，PS3.10 部分指出了几个简单的、应该服从的标准 DICOM 文件名称规则。

根据这些规则，所有 DICOM 文件都要标有 DICOM 文件 ID：唯一文件标识符，其实就相当于文件名。文件 ID 由多达 8 部分组成。在传统做法中，最后的部分相当于一个短文件名而前一个部分相当于保存文件的文件夹路径（见图 73）。因此，文件 ID 相当于文件的全名。每个部分职能包含以下字符：大写字母从 A 到 Z，数字从 0 到 9，或者下划线符号。

DICOM 文件 ID 命名规则看起来像是文件名和 DICOM UID 的嫁接产物。具有 UI VR 格式的 DICOM UID 是独一无二的字符串，它的负责标识 DICOM 对象，但是它们并不使用字母或下划线；它们只能是由句点分隔开的数字，比如 1.2.840.10008.1.1。相反，DICOM 文件 ID，是通过分隔符（通常是斜线符，虽然 DICOM 并没有制定任何字符）描绘地，它的 8 个 8 字符组件会用 7 个分隔符隔开，并且可以超过 UID 的 64 字符长度限制。

最后（这和 DICOM 软件开发人员息息相关），如果你记得，典型的文件 ID 分隔符是反斜线（\），已经成为了 DICOM 指定的通配符（见 5.3.2），它表示“逻辑或”。换句话说，按照 DICOM 的规定，文件 ID“SUBDIR1\SUBDIR2\ SUBDIR3\FNAME”在别处的 DICOM 字符串中可能表示“SUBDIR1 或 SUBDIR2 或 SUBDIR3 或 FNAME”，与文件 ID 相比，这显示表达了一种完全不同的含义。因此如果用反斜线分隔文件名称，则得到的会是几个关系为“或”的分块，这是一种 DICOM 软件中常见的 bug。唯一简单的解决办法是在所有文件名中使用前斜线（/）。

下面是关于传奇的“.dcm”扩展名的问题，通常它都会被数不尽的 DICOM 程序追加在文件名的后面。在操作系统中，每个文件扩展名可以与特定的文件处理程序关联，因此.dcm 用起来会非常顺手：在你的文件管理器中点击一个.dcm 文件，操作系统会立即启动你的 DICOM 程序。另一方面，根据图 73 中的文件 ID 语法，DICOM 文件命中是没有.dcm 扩展名的。此外，在 DICOM 标准的各个章节中，.dcm 扩展名既不会禁止也不会要求。我猜关于.dcm 扩展名的使用问题，DICOM 工作组的高人们可能还在讨论中吧。

所有上述内容的结果就是，在现实世界，谈论 DICOM 文件 ID 规则时要有遇到“惊喜”的心理准备。许多 DICOM 程序仅会简单地使用 DICOM SOP UID（唯一地识别背个 DICOM 对象实例）作为文件名，所以你会经常见到这些：

1.2.840.10008.234.2354.437345.79086/1.2.840.10008.8.4568.243.09.dcm

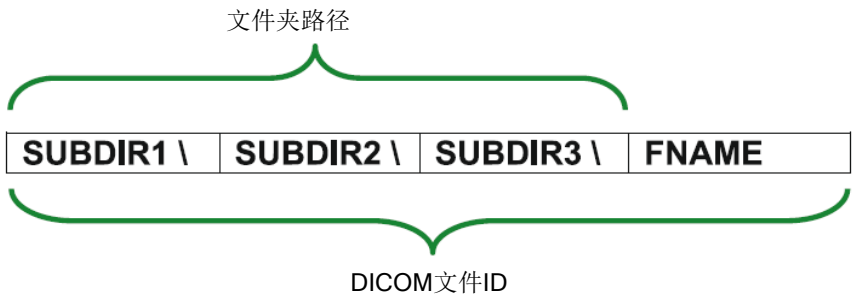

图 73 DICOM 文件 ID

而不是 SUBDIR1\SUBDIR2\SUBDIR3\FNAME。然而，在文件名称中使用 SOP 同样会产生问题：SOP 字符串是长型的（最多 64 字符），如果使用许多文件名组件，这将会非常容易滴超出你

软件所能处理的极限。做好准备吧。

难怪为什么通过 DICOM 文件名没法识别 DICOM 文件。他可能是任何东西：带有或不带“.dcm”扩展名；使用 UID 命名语法或一个标准的 8 个字符的大写字母名。此外，不同的操作系统和媒体在文件有效命名方面可能有其自己的要求。也就是说，从 DICOM 文件头中查找 DICM 前缀，是分辨 DICOM 文件有效性的唯一可靠方法。

10.2 特殊 DICOM 文件格式

10.2.1 DICOMDIR

DICOMDIR 是一种非常特殊的 DICOM 文件。当所有其它 DICOM 文件负责存储自己的 DICOM 数据对象时，DICOMDIR 则负责存储给定文件目录（文件夹）下 DICOM 文件的信息。因此，DICOMDIR 相当于一个小型的 DICOM 数据库，或者一个 DICOM 文件的目录，并放置在媒体的根目录下。

文件索引

文件索引——创建一个特殊的文件包含特定文件夹下其他文件的信息——是许多软件程序的一个非常普遍的做法。操作系统、电子邮件和多媒体软件都会试图对计算机中的相关文件进行索引。文件索引可以提供访问文件数据的素缀，并且提高以用户定义关键词查询文件的速度。特别是，如果文件查询键值存储在索引文件中，那么只需要查询索引文件就够了。相反观点认为，创建和维护索引文件会占用资源，并且被索引文件夹中有任何改动，索引文件都必须进行相应修改。

总之，就像任何 DICOM 数据库一样，DICOMDIR 会将所有的目录数据组织成 4 个主要的 DICOM 层次：病人、检查、序列和图像，如图 74 所示。因此，对于 DICOMDIR 文件夹中的每个文件，DICOMDIR 都会为其记录 4 个条目信息——病人、检查、序列和图像信息。

表 50（改编自 DICOM PS3.10）给出了一个带有样例数据的 DICOMDIR 文件实例。这个实例看起来很长，但却非常接近实际。你将会看到所有 DICOMDIR 项（病人、检查、序列和图像）的列表，以及列表是如何作为一个 SQ 序列元素(0004,1220)被放入 DICOMDIR 文件的。对 DICOMDIR 序列(0004,1220)中的每个条目而言，DICOMDIR 对象存储了两类数据：

1. 条目特定的选择键。这个数据类型是为了方便在 DICOMDIR 中进行项目查询而设计的。比如，在对序列进行查询时，序列影像设备(0008,0060)就是最常使用的选择准则，因此在 DICOMDIR 索引中一定要用它作为选择键；这样我们将在给定的目录中了解到此处涉及到哪些影像设备。几乎所有的 DICOM 软件都会有一些类似 DICOMDIR 浏览器的工具提供给用户，其中的内容就是从 DICOMDIR 中获得的这个条目列表。比如，如果你将一个 DICOM CD 盘插入 PACS 工作站，那么工作站通常会向你显示一个 CD 上的病人和检查的列表，这就是从 DICOMDIR 条目中提取出来的。病人姓名、检查日期、影像设备以及其他有用的数据都来自选择键。
2. 基本目录信息对象。第二种类型的 DICOMDIR 条目数据是存储在 0004 组的元素。我们还没介绍过这个组，因此现在进行一下简短介绍。0004 组是基本目录信息对象的预留组，并且如你在表 50 中所见，存储着关于 DICOMDIR 条目的目录相关信息：文件 ID、与文件的关系等等。其实，任何 DICOMDIR 对象都是一个基本目录 IOD 的实例，就像任何 CT 图像对象都是一个 CT 图像 IOD 的实例一样。基本目录信息对象打算要成为各种媒体目录的一个摘要表述，比如一组在某处的 DICOM 文件。

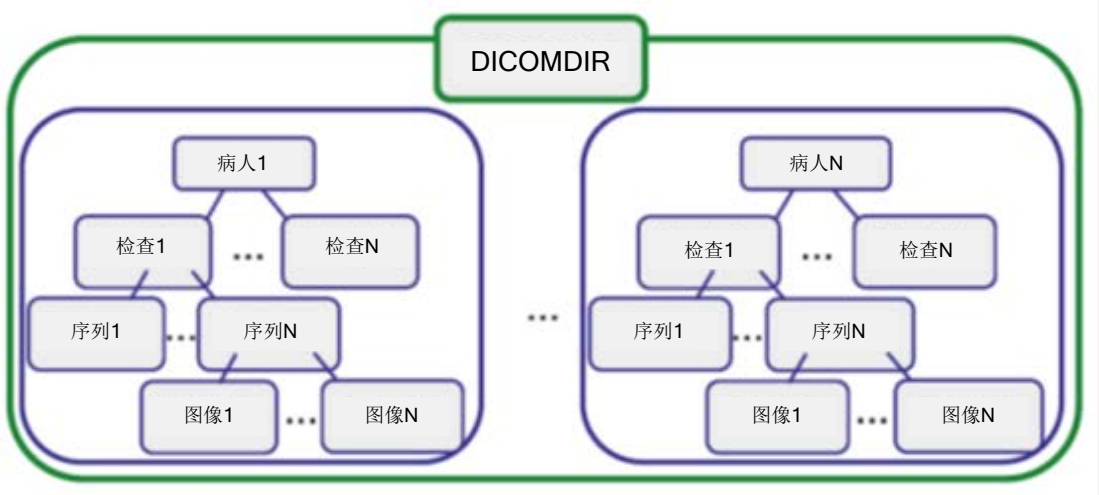

图74 DICOMDIR结构。每个图像相当于DICOMDIR目录中的一个DICOM文件

在 DICOMDIR 文件中所列出的条目中，并不限制必须为 DICOM 图像，就像 DICOM 文件并不是只存储 DICOM 图像一样。任何其他 DICOM IOD 都是允许的，因此 DICOMDIR 也可以列出 SR、波形、图像覆盖信息等许多其他的东西都会出现在这个目录中。此外，至少在理论上 DICOMDIR 可以包含非 DICOM 文件的条目；即使 DICOM 不能提供处理这些非 DICOM 数据的功能，DICOM 标准也并不介意让 DICOMDIR 中包含这些数据。

你可以在 DICOM 标准 PS3.3 的附录 F 中找到基本目录 IOD 的更多细节。我们会宅稍后的章节才来介绍 DICOMDIR 的实用性。

表 50 DICOMDIR 实例

条目类型	属性标签	属性描述	值（实例）
文件元信息（在任何 DICOM 文件中都必须存在）	128 字节	DICOM 文件报头	所有 128 字节都设为 0
	4 字节	DICOM 前缀	DICM（必须）
	(0002,0000)	组长度	
	(0002,0001)	文件元信息版本	0001（必须）
	(0002,0002)	媒体存储 SOP 类 UID	1.2.840.10008.1.3.10
	(0002,0003)	媒体存储 SOP 实例 UID	1.2.840.12345.6435.4
	(0002,0010)	传输语法 UID	1.2.840.10008.1.1（必须）
	(0002,0012)	实现类 UID	1.2.840.4578.34.2345
	…	…	…
文件集 ID	(0004,1130)	文件集 ID	MYDICOMDIR01
	…	…	…
通用目录信息	(0004,1200)	根目录实体第一个记录的偏移量	1829
	(0004,1202)	根目录实体第一个记录的偏移量	6F18
	(0004,1212)	文件集一致性标志	0000
	…	…	…
DICOMDIR 记录列表的开头	(0004,1220)	目录记录序列。这个 SQ 元素包含 DICOMDIR 元素的实际序列，见后。	

第一个 DICOMDIR 记录（从第 1829 个 字节开始）			
SQ 项标签	(FFFE, E000)	SQ 项标签数据元素（见 5.5.5），DICOMDIR 实例的开端	
检查 1	(0004,1400)	下一个目录记录的偏移量	
	(0004,1410)	记录在用标志（设置为 FFFF 表示存在的文件，而 0000 表示已删除的文件）	FFFF
	(0004,1420)	参考的较低层目录实体偏移量	2299
	…	…	…
	(0004,1430)	目录记录类型	STUDY
	(0020,000D)	检查实例 UID	1.2.840.1234.0125.5
检查 1 选择键	(0020,0010)	检查 ID	MyStudyID01
SQ 项标签	(FFFE,E00D)	该项界定标签：第一个 DICOMDIR 条目的结尾	
第二个 DICOMDIR 记录（从第 2299 个字节开始）			
SQ 项标签	(FFFE, E000)	SQ 项数据元素：之后的 DICOMDIR 条目的开始	
序列 1	(0004,1400)	下一个目录记录的偏移量	
	(0004,1410)	记录在用标志	FFFF
	(0004,1420)	参考的较低层目录实体的偏移量	2681
	…	…	…
	(0004,1430)	目录记录类型	SERIES
序列 1 选择键	(0008,0060)	影像设备	NM
	(0020,0011)	序列号	2
SQ 项标签	(FFFE,E00D)	项界定标签：当前 DICOMDIR 条目的结尾	
第三个 DICOMDIR 记录（从第 2681 个字节开始）			
SQ 项标签	(FFFE, E000)	SQ 项数据元素:之后 DICOMDIR 条目的的开端	
图像 1	(0004,1400)	下一个目录记录的偏移量	3414
	(0004,1410)	记录在用标志	FFFF
	(0004,1420)	参考的较低层目录实体的偏移量	00000000
	…	…	…
	(0004,1430)	目录记录类型	IMAGE
	(0004,1500)	参考的文件 ID	DIR\SUBDIR\ABC123
	(0004,1410)	参考的文件中 SOP 类 UID	1.2.840.10008.5.1.4.1.1.5
	(0004,1511)	参考的文件中 SOP 实例 UID	1.2.840.943.2345.54.778
	(0004,1512)	参考的文件中传输语法 UID	1.2.840.10008.1.2.1
Image 1 selection keys	(0008,0018)	图像 SOP 实例 UID	1.2.840.943.2345.54.778
	(0020,0013)	图像号	1

图像 1 选择键			
SQ item tag SQ 项标签	(FFFE,E00D)	项界定标签：当前 DICOMDIR 条目的结尾	
第四个 DICOMDIR 记录（从第 3419 个字节开始）			
SQ 项标签	(FFFE, E000)	SQ 项数据元素：后面的 DICOMDIR 条目的开端	
Image 2	(0004,1400)	下一个目录记录的偏移量	
	(0004,1410)	记录在用标志	FFFF
	(0004,1420)	参考的较低层目录实体的偏移量	00000000
	…	…	…
	(0004,1430)	目录记录类型	IMAGE
	(0004,1500)	参考的文件 ID	DIR\SUBDIR\ABC124
	(0004,1410)	参考的文件中 SOP 类 UID	1.2.840.10008.5.1.4.1.1.5
	(0004,1511)	参考的文件中 SOP 实例 UID	1.2.840.943.2345.54.779
	(0004,1512)	参考的文件中传输语法 UID	文件中参考的传输语法 UID
图像 2 选择键	(0008,0018)	图像 SOP 实例 UID	1.2.840.943.2345.54.779
	(0020,0013)	图像号	2
SQ 项标签	(FFFE,E00D)	项界定标签：当前 DICOMDIR 条目的结尾	
第五个 DICOMDIR 记录（从第 6F18 个字节开始）			
SQ 项标签	(FFFE, E000)	SQ 项数据元素：之后的 DICOMDIR 条目的开始	
病人 A	(0004,1400)	下一个目录记录的偏移量	00000000
	(0004,1410)	记录在用标志	FFFF
	(0004,1430)	目录记录类型	PATIENT
	…	…	…
病人 A 选择键	(0010,0010)	病人姓名	A
	(0010,0020)	病人 ID	123-4567
SQ 项标签	(FFFE,E00D)	项界定标签：当前 DICOMDIR 条目的结尾	
DICOMDIR 列表的结尾	(FFFE,E0DD)	序列界定标签：此处关闭了(0004，1220)中的 DICOMDIR 元素序列	

10.2.2
安全 DICOM 文件格式

在网络上发送 DICOM 对象和用文件交换 DICOM 对象的最大不同在于潜在安全风险的范围，这些安全风险主要涉及文件打开和访问。拦截网络上的消息需要特殊的技术，复制、删除或修改文件就容易多了。

即使 DICOM 文件是允许公开访问的，现代数据安全算法也能够保护 DICOM 文件信息。

DICOM 标注提供 DICOM 文件加密的方法，让文件中的数据从大众眼前消失。通过加密，整个 DICOM 文件（作为一个整体，而不是打散成 DICOM 属性）会被改写，没有安全文件密

匙就不能读取。第 11 章提供了一个在 DICOM 安全性上更好的检查方法，但现在，我要介绍一下安全 DICOM 文件会提供以下特性：

1. 数据保密（依靠加密）。你的数据会从大众眼前消失。如果某人偷了你的 DICOM 文件，他/她无法读取那些文件。
2. 数据源验证（依靠数字签章证书）。你可以了解这个文件属于谁，谁曾经更改过它。比如，如果文件包含一个已签章报告，你可以通过电子签章对报告进行安全保护，由此可以完全识别是谁签发的报告。
3. 数据完整性（依靠数字签章和总和检查）。没有人能够在你不知情的情况下修改你的数据。比如，修改病人和医生的名字，或者修改原始的报告日期和时间，这都是不可能成功的。数据总是会保持他自己的原始格式。

有了这些特性，你就可以放心地将你的 DICOM 文件公开了，而不用担心会带来任何安钻风险。不幸的是，在目前的 PACS 软件中很少会支持安全的 DICOM 文件，而且在 DICOM 标准中，这些内容也定义的非常浅薄，还需要在这方面做更多的工作才能使其更加普及。如果想要了解安全性方面的细节，可以阅读第 11 章。

10.3
DICOM 文件服务

本着 DICOM 精神，如果存在数据，我们就必须具有数据处理服务。我们的数据是文件，因此我们需要文件处理服务。这就是关键所在——我们相当传统的文件将换变换成了一个 DICOM SOP 语言，听我娓娓道来。

10.3.1
DICOM File Set

在传统的媒体上，我们是通过文件夹来组织文件的。想想吧，一个文件夹（在你的 U 盘或硬盘上）不过是一个文件的集合（可能还有其他子文件夹）。DICOM 文件集也是一样的，即一个 DICOM 文件的集合；且在集合中，可以通过唯一文件 ID 来识别每个文件。

文件集 UID 是用来标识文件集的，它符合 UID 格式（因此，符合 64 字符的数字 UI VR 格式，以句点隔开，比如 1.2.840.1008.67.7890.2）。这看起来有点古怪，因为文件 ID 名并不符合这种格式（它们是用的是大写字母，数字和下划线）。如果我们用文件夹来表示文件集，而用文件表示文件 ID，那么就具有了非常好的 DICOM 标准一致性。可能就是因为这个原因，所以 DICOM 允许使用其他方法来标识文件集：使用文件集 ID，要求其最多有 16 个字符，字符来自与文件 ID 中一致的字符集：A 到 Z、0 到 9 以及下划线（_）。很不幸，这并不符合文件 ID 命名的习惯：文件 ID 通常会有最多 7 到 8 个字符，充当文件路径（文件夹名），并且用任意字符分隔符隔开。一个 16 个字符的文件 ID 并不符合这些要求。总之，看起来 DICOM 需要一组更好的文件命名规则了。

尽管如此，文件集依然存在而且我们需要通过他们来存储信息。该怎么做？当然还是通过 DICOMDIR 文件来实现。说白了，DICOMDIR 其实就是具有文件集功能的文件，对集合内的所有内容进行索引。当一个 DICOM 应用程序在根目录找到一个 DICOMDIR 文件时，它会认为文件夹是一个文件集，是一个需要进行处理的 DICOM 数据文件摘选。

我们通常认为文件集就是文件夹，但是并不都是这样。一个 DICOM 文件集可以是一个磁盘卷、分驱或其它什么；甚至是一个只能由 DICOMDIR 文件中的文件列表才能识别的摘要容器。

10.3.2
文件管理角色和服务

如前所述，DICOM 就是一套数据处理服务。其中有五个媒体存储服务：

1. M-WRITE：在文件集中创建一个新文件，并为其分配文件 ID。
2. M-READ：基于其文件 ID 来读取现有文件。
3. M-DELETE：基于其文件 ID 来删除现有文件。
4. M-INQUIRE FILE-SET：查看是否还有足够的空间在文件集中创建新文件。
5. M-INQUIRE FILE：对于文件集中的任何文件，查看其创建的日期和时间（或者最后更新的日期和时间）。

如你所见，在这个列表中的每件事情都很明了：媒体存储 M-services 负责管理文件和文件空间。因此，任何 DICOM AE 都可能会符合以下三个角色概念中的一个。

1. 文件集创建者（File Set Creator FSC）：该 AE 使用 M-WRITE 来创建一个 DICOMDIR 文件以及 0 或 N 个 DICOM 文件（即使没有 DICOM 文件，也可以只有一个 DICOMDIR 以及一个空文件列表）。
2. 文件集读取者（File Set Reader FSR）：该 AE 使用 M-READ 在一个文件集中存取一个或多个文件。FSR 是绝对不允许修改文件集中的任何文件的。

3. 文件集更新者（File Set Updater FSU）：该 AE 使用 M-READ、M-WRITE 以及 M-DELETE。对于文件集中 DICOM 文件的内容，文件集更新者可以读取，但是不能修改。但它可以修改 DICOMDIR 文件。它可以使用 M-WRITE 来创建附加的文件，或者使用 M-DELETE 删除已经存在于文件集中的文件。

很简单吧，不是吗？三个角色中的一个或者多个在一起形成了 7 种 DICOM 软件可以支持的组合。

需要注意的是，虽然 FSU 通常不允许修改文件内容，但是它通常会作为“删除-写入”搭档。即使两个文件只有一个字符不同，你也只能删除整个旧文件并写入它的新版本。删除-写入组合该线上非常简单，但是总是很好用。一些文件媒体会不让你修改文件内容，所以虽然修改极其微小，你也必须整个覆盖那个需要修改的文件。这就是说如果不包括用来删除旧文件版本的 M-DELETE 部分，FSU 通常相当于 FSC＋FSR。

对比 DICOM 媒体存储服务和 DICOM 网络，我们可以发现他们都遵循相同的框架。见图 75。在网络和文件管理场景中，如果具有匹配应用的能力，那么 DICOM 应用程序就会进行通讯。

1. DICOM 网络上的 AE 是基于具有相应功能的 SOP 类来通讯的（比如我们在第 7 章学过的校验 Verification 或存储 Storage）。如果两个 AE 支持同一个 SOP 并且它们的 SCU-SCP 角色匹配，那么当他们的配置匹配时，AE 间就可以通讯了。假如，两个 AE 都支持校验 SOP，其中一个作为校验 SCU 另一个作为校验 SCP，当他们的配置匹配时 AE 就可以通讯了（见 7.10）。在网络方面，应用程序配置实在连接联机进程中进行协商的：角色、SOP、传输语法和其他选项会在两个参与通讯的 AE 间进行比较，并且选出可行的最佳方案。

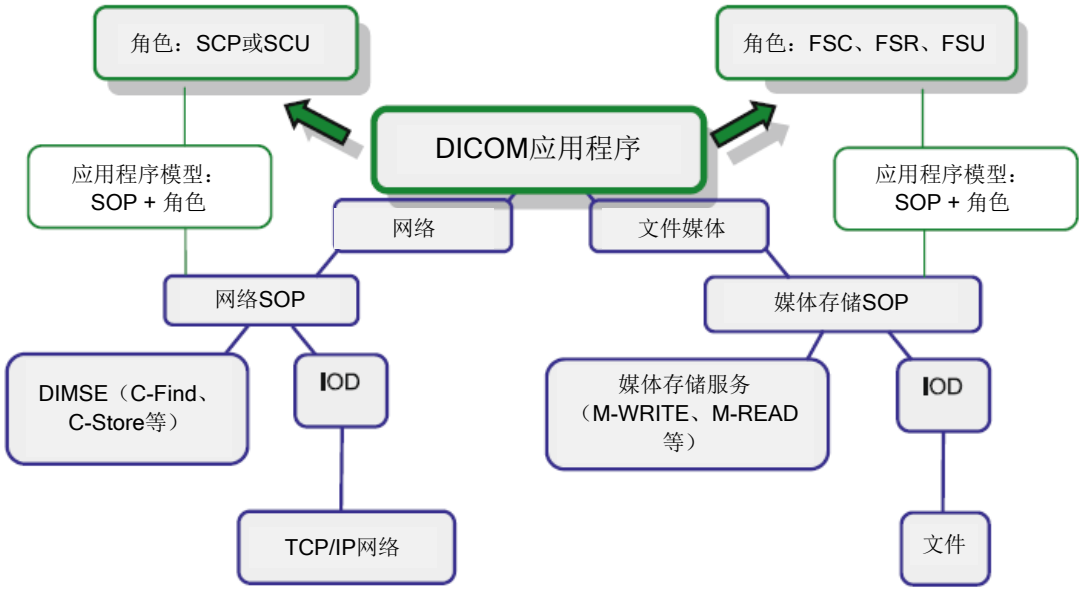

图75 DICOM媒体存储对比DICOM网络：大图像。DICOM网络在左侧，DICOM媒体在右侧

2. 文件处理 AE 也是采用类似的连接方法。也就是说，AE 会支持相同的媒体存储 SOP 类（Media Storage SOP），还会支持适合的角色。比如，一个 AE 作为 FSC（创建文件）而另一个作为 FSR（读取文件）。然而，与网络的情况不一样的是，文件处理 AE 不能协商；整个连接建立的的处理过程是缺少协商部分的，至少在现在的 DICOM 3.0 版本

中是这样的。因此必须使用一些附加的措施来保证两个媒体存储 AE 在相互间可以间理解对方。如果一个正在写 MR 图像，那么另一个就不应该在等待 CT 图像。

潜在的做法或许会采用以下两种之一：在应用程序层处理、或在标准层处理。在 DICOM 网络上，配置应用程序匹配的主要工作是由 DICOM 应用程序来实现的。对于文件存储，DICOM 标准则采用了另一路线，将应用程序配置的细节提高到一个可实现的最高高度。因此具有相同配置的 AE 之间将必须在无法沟通的情况下实现匹配。这个做法也延伸到许多其他的媒体存储应用配置（Media Storage Application Profiles）中，见 PS3.11。让我们看看一些实例吧。“基本心脏 X 线血管造影（Basic Cardiac X-Ray Angiographic）”的配置只负责处理图像大小小于 512×512 的 X 线血管造影图像；对于更大的 1024×1024 图像，“1024 X 线血管造影 CD-R（1024 X-Ray Angiographic CD-R）”配置负责 CD 部分，而“1024 X 线血管造影 DVD（1024 X-Ray angiographic DVD）”配置则负责 DVD 部分。看到了吧，和 DICOM 网络相比，这种做法非常细致。为了保证应用程序的兼容性，一些非常细节的内容也需要考虑到。若应用程序支持某个媒体存储（Media Storage）配置，那么就必须在其 DICOM 一致性声明中列出。

借用与 DICOM 网络对比学习的方法，我们能够更好地回顾标准的 DICOM 媒体存储（DICOM Media Storage）。如果你仍对某些细节感兴趣，可以参看 DICOM 标准的以下部分 10、11 和 12。DICOM 媒体交换的实践仍然是一件非常多样化且总是不一致的，但这就是现实不能无视它。在我看来，我们可以把一部分责任推给 DICOM 自己。其实 DICOM 完全可以改进标准，对于媒体变得更加宽容和高效。这是下一步的事情了。

10.4 几粒盐

DICOM 网络和 DICOM 媒体都提供相同的功能，访问和操作 DICOM 数据，但是他们只得用不同的方法。DICOM 网络建立在标准 TCP/IP 网络协议之上。当任意两个计算机通过 TCP/IP 网络连接时，人为的影响因素就基本排除了，没有人会搅乱 TCP/IP 包比如篡改或打乱他们。

文件媒体恰恰相反。我们人类在操作文件时，根本不会去关心那些愚蠢的老版本有什么要求。所谓的对 TCP/IP 术语的鲁莽篡改是司空见惯的和常规例行的处理方式。一个好的文件处理标准应该将这些全都考虑在内，但是 DICOM 没有做到。

10.4.1 DICOMDIR

就我自己而言，我算是 DICOMDIR 的忠实反对者。我的确知道它存在的逻辑，但是基于一些理由，我发现他们都不切实际。首先，DICOMDIR 几乎一点用都没有。任何设计精良的 DICOM 程序都可以做到在给定的文件夹中扫描所有文件，识别是否为 DICOM 格式，并作出相应处理。当然，对于大文件夹来说 DICOMDIR 也许可以节省时间，但是多大才算大呢？即使是塞满 DICOM 数据的 DVD 也可以请送快捷地进行扫描，DICOMDIR 也不会用到更多的工业场景下，比如 PACS 层次的存储（这时 DICOMDIR 会被数据库所替代）。大多数的多媒体程序，面对可支持文件的识别任务时，都会采用相同的扫描方式；这也就是你播放视频和音乐时所用的方法。这个方法提供了最完整和最新的账目，其中清晰记载着特定媒体中到底存着什么，不是依靠那些错误的或过时的目录文件来提供媒体中的文件信息。

DICOM 媒体（文件）有两个主要的用法：要么是导入 PACS，要么是显示。无论是哪种用途，媒体中的所有 DICOM 文件全都会被加载和处理。无论在哪种情况下都不需要 DICOMDIR 来提升那些无足轻重的效率。

开发者小贴士：索引 DICOM 文件

假如你的应用程序足够聪明来减少文件扫描的开销，那么在文件夹中索引 DICOM 文件将会是很快速的。比如，所有常用来索引 DICOM 文件的必要 DICOM 信息（病人姓名和 ID、检查日期和时间等）都可以在 DICOM 数据组(0008-0020)中找到。这些信息通常保存在 DICOM 文件的第一个千字节中。

为了提取这些数据，你的 DICOM 软件并不需要读取整个文件，也不需要做 DICOM 文件处理中最费事的工作：解压图像像素。如果与读取 DICOMDIR 相比，从第一个 DICOM 数据组中预读取数据会更加快捷也更加简单。

第二、DICOMDIR 靠不住的。当我们向移动媒体上的 DICOM 文件中输出 DICOM 数据时，就像打开潘多拉盒子。媒体的拥有者可以随便复制其内容（比如从闪盘复制到其他计算机）；重命名一些内容（比如重命名遵从 DICOM 的八字符文件夹 ABCD1234，软件会将其自动改为更长、但可读性更高的“MyInterestingMRCase”）；删除一些内容；以及一切我们常常会对文件所做的处理。可是，这些动作中的任何一种都建造成 DICOMDIR 文件内容的不可用，但是其仍旧会和一堆其他内容一同复制给其他设备，最终通过 DICOM 应用程序来实现数据导入。如果应用程序依靠 DICOMDIR 来导入文件，那么它只能处理那些内容正确的部分。

从此观点出发，DICOMDIR 又让我想起了可恶的 DICOM 点对点架构：在早期 PACS 中很有用，但是现在已经弃用了。DICOMDIR 基于以下假设：输出的文件将会直接进入另一个文件输入应用程序。这个假设现在看来完全是错误的。

第三，DICOMDIR 太难了。我们在文件夹中更新或修改了任何 DICOM 文件都需要更新 DICOMDIR。这不是任何时候都可行的。如果媒体是进行了更新保护的，或者你只能在媒体上进行一次写操作（如 CD-R），那么 DICOMDIR 将是最后被记录在媒体上的文件，只有这样才能保证其中保存着最准确的内容。

简而言之，维护和更新DICOMDIR文件都会实际上产生开销，很少带来额外的优势，成为大家所唾弃的东西。比如，你甚至能找到专门负责修复无效DICOMDIR文件的软件（可以在 www.tritech.com中找到），就是因为DICOMDIR中记录的文件条目在媒体中有所修改或已经删除。

当然，我对于 DICOMDIR 的强烈反对并没有实质上影响 DICOM 标准，DICOM 标准仍然需要 DICOMDIR 文件来提供媒体中的 DICOM 数据。然而，我对 DICOMDIR 的言论来自于实践经验。如果你的 DICOM 软件可以基于实际的数据和内容导入 DICOM 文件，而不是必须依靠 DICOMDIR 数据，那么你会有效地改善你的工作流程。

DICOM 委员会小贴士：灵活点更好

一个避开 DICOMDIR 的实际做法是增加一个新的 M-INQUIRE-DICOM FILE 服务。若有此服务，M-INQUIRE FILE 可以仅查询文件日期和时间，而一个扩展的 M-INQUIRE-DICOM FILE 能够支持一些传统上由 DICOMDIR 提供的一些 DICOM 查询键值（比如病人姓名、检查日期等）。

如我们所知，这个方法实现起来很容易，只需要通过部分读取 DICOM 数据组就可实现，即只查找一些靠前的关键元素，而不需要读取整个文件。事实上，几年前，我的公司必须要通过 FTP 协议来实现 DICOM，就是通过这个方法，我们实现了在已建立 FTP 连接的任何媒体上进行 DICOM 数据查询的功能。这使我们可以在任何跨平台和跨空间的东西上实现 DICOM 查询。现实中选用了 DICOMDIR 且又在应用程序层面建立它，这将会大大增加 DICOM 软件的复杂性，并且必须要支持各种类型的本地或远程媒体。

此外，如果我们将 DIMSE 服务（C-Find、C-Store、C-Move 等）扩展到 DICOM 文件层面，那么这个做法还可以向更深层次发展。其实，我们假设的 M-INQUIRE-DICOM FILE 已经实现了 C-Find，其它的服务也很好搞定。无论是网络上的 DICOM 内容还是从 DICOM 文件中读取到的内容，这都将为处理 DICOM 数据提供一种非常统一和一致的方法。

10.4.2
媒体存储

很多情况下，DICOM 媒体存储模型（尤其是 DICOM 标准中的 PS3.12）试图做一些 DICOM 目前不会做的事情：对于 DICOM 文件数据记录在各类媒体上时应该如何处理给出了实施细节。从实践的立场来看，要求提供 DICOM 媒体存储说明书确实有点多余。将媒体存储于 DICOM 数据编码模型对比可以看到，编码模型是基于简单和统一的规则的并且持续使用了 20 多年，这是因为它总是置身具体实施之外。相反的，DICOM 媒体存储规则却需要每一两年重写一次，因为它依赖于媒体种类。它们试图控制许多不必要的细节并且偶尔会蹦出一些模棱两可的或自相矛盾的声明。此外，它们为人为错误留下了太多空间，因此几乎每次人们在进行涉及 DICOM 文件的操作时都会出现问题。

若要取代这种对于特定实施细节的过分控制（比如控制引导区和文件名句号分隔），DICOM 标准的部分 10、11、12 需要一些关注些清新的变化，这样才能变得更抽象以及更加重视功能性。

在这之前，许多医疗实践将与非法 DICOM 文件名、散乱的文件集、以及不正确或遗漏的 DICOMDIR 继续战斗。这些中的很多其实都没有必要。

10.4.3
请发给我门胶片吧！

我们聊过 DICOM FSC 和 FSR；我们聊过复杂的媒体格式，但是我们遗漏了在整个媒体交换过程中的一类非常重要的角色：常说的接诊医师和临床医师。这其中还包括那些在办公室里、计算机上、手术室里或任何地方想要查看 DICOM 检查的人。

他们会怎样？

在多数情况下，他们可能没有 PACS 可以用，即使他们可以找到 PACS，他们也不会用。他们想要将 CD 放到计算机中即可浏览图像。他们中的多数对 DICOM 一点概念都没有，甚至可能会试图将 DICOM 文件放到幻灯片中。其他人则希望 CD 可以自动播放，然而计算机的 CD 自动播放选项可能被禁用了，无论他将 CD 放到光驱中多少次都不会有任何反应。这时他们不试了，改为给你打电话，让你把胶片给他们而不是这些“不好使的破烂 CD”。

你能做什么？

打印胶片可不行。不只是违背了整个数字医疗的宗旨，而且会找你口袋要钱。在当前 DICOM 媒体的使用方面，对于用户的平均使用水平来说都已经算是一种挑战了。底线：显示文件，你的用户可能需要一些 DICOM 软件，他们不得不学会如何使用这些软件。如果他们会一直作为图像浏览的客户端，我建议在他们的计算机上安装软件并且教会他们。不要只是依赖 DICOM 图像 CD 浏览器。

10.4.4
导出和导入

几乎可以确定的是，从 PACS 导出媒体就意味着会在别处将这个媒体导入 PACS。像 CD/DVD 数据导入这样的媒体存储任务必须在同时期的 PACS 中才能实现。当一个带有病人数据的 CD 送到医院时，放射科的技术人员可以轻松地将所有 DICOM 数据从 CD 传入医院的 PACS。如果你正在购买 PACS，那么一定要确保它支持文件媒体导出和导入。

然而，没有听起来那么简单。比如，在影像中心记录的病人 ID 将肯定与医院 PACS 中的病人 ID 不同。许多其他参数也有可能不匹配或记录的方法不同。这些问题不可避免地揭示出此时需要实施数据合并和调整步骤。

当代的 PACS 实践中，媒体导入已经成为了一个薄弱环节。当任何人用他们最方便的方法轻松地导出了数据时，在各类不能协调一致的数据源中导入和调整数据会花费许多时间和资源。请记住外部媒体可交换性非常贫乏，根本无法满足大型项目的需求。远程放射诊断常

用的应用程序服务提供商（Application service provider ASP）模式，就非常时候大型机构之间的数据叫喊。在 ASP 模式中，数据可在某处保存和处理，并在另一处显示，整个过程不需要导入或导出。总之：要想解决为转介医师刻录 CD 的问题，最好还是能够让他们远程浏览图像。

10.5
在 PACS 中存储 DICOM 数据

之前提过，PACS 厂商更喜欢使用自己私有的数据存储模式来存储内部的 PACS 数据；并且内部的 PACS 存储即不是这章的主题，也不是 DICOM 媒体存储协议的主题。但是有必要了解当前 PACS 厂商在存储方面都做了什么。

首先，对于巨大的数据卷，PACS绝不会使用DICOMDIR来索引它内部的DICOM文件。除了我们之前提到其他问题以外，DICOMDIR在处理成千上万的文件时效率实在太低了。所以PACS会使用关系数据库：商业数据库（SQL、Oracle、Cache、DB2 等）或自己编写的。如果你不太了解关系型数据库，那么就想象一下常用的微软Access或Excel吧。任何关系型数据库都是一个相互之间具有关联性的表格集合。实际上，任何PACS数据库都会反映DICOM数据登记，将所有数据放在四个首要的DICOM表格结构中，病人（Patient）、检查（Study）、序列（Series）和图像（Image），它们之间关系非常明确（在Patient表中的每个病人都会有一些检查在Study表中，每个检查都会有一个或多个序列在Series表中，每个序列都会有图像在Image表中）。[1]图 76 描绘了基本的PACS数据库。

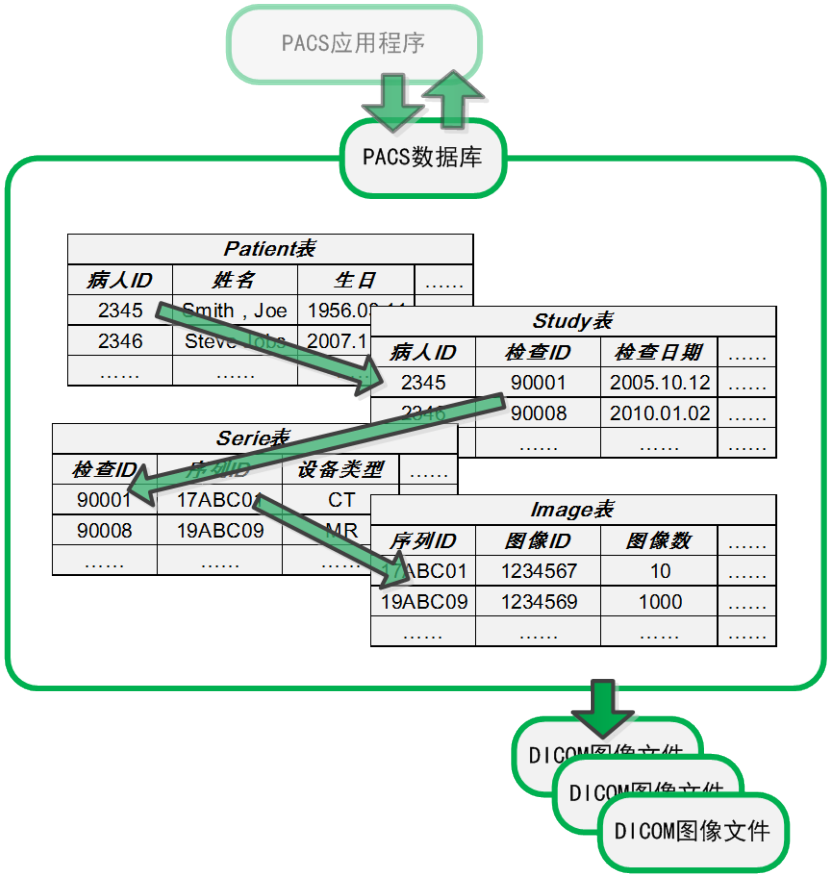

图76 PACS中的DICOM数据库，基本布局图，有四个数据库表：病人、检查、序列和图像

[1] 在 PACS 数据库中的其他表通常会包括 PACS 用户的表，审计表（记录什么人和何时访问过图像），AE 表（存储可连接的 AE 属性），等。

在关系数据库中的表格会通过后写共用字段关联在一起，比如PatID，[2]它是在Patient表中唯一标识病人的字段，同时会在Study表中标识这个病人的检查。在表中如此布置信息会使整个数据库运行更高效以及组织更完善。数据库可以很快地在已经排好序的表格中定位数据，而不用在DICOM文件中或DICOMDIR中去查找。为了提高效率，查询只允许在一些相关表格中进行。

所以，这种优化的数据库设计对内部 DICOM 数据存储产生了根本影响。PACS 会选择三种方法中的一种来存储 DICOM 图像数据，其中任何一个都还需要深入考虑一下细节。

10.5.1
基于文件的PACS存储

传统的、基于文件的存储模型中，DICOM图像是以普通DICOM文件形式存储的。这种情况下，在PACS数据库中的Image表会保存这些文件的名称。当PACS需要为某位病人提取图像时，就可以从PACS数据库中找到病人的检查、序列和图像记录，这些图像记录将会包含病人图像的文件名。之后，即可在硬盘上定位这些文件，并且将其作为DICOM对象调取到PACS中。

以上方法最明显的优点在于简单。其次，不是很明显的优点是所有DICOM数据（文件）都是在PACS以外分开存储的。PACS只是指向文件并负责日常处理部分（删除、更新、相互关联文件记录）。应用程序和数据的这种分隔在许多数据迁移项目中会显得极其有用；比如，当一个新PACS要替换掉旧PACS时，此场景下，你不需要完全依靠旧PACS软件。你只需要定位到存储这些DICOM文件的硬盘，然后将它们导入到新系统即可。这种情况下的数据迁移就转化成了一种数据导入任务。如果DICOM能够正确导入，那么新系统的实施过程将变得更简单些。

还有一个阴暗面：直接访问应用程序数据（文件），绕过应用程序的这种做法可能会被滥用。文件可能会被非法复制、删除或更改。基于文件存储的PACS应该具有一些安全机制来避免这些问题的发生。

机构内部的应用程序

许多机构利用文件分开存储的方法来开发它们的机构内部的DICOM处理软件。如果，你需要编写一个核医学图像后处理应用程序，并且你知道在图像服务器中的什么地方能够找到这个文件夹，那么你的应用程序可能就直接从图像服务器中读取图像了，避免在DICOM网络研发过程中遇到麻烦。但是一定要将这些文件置成只读，而且不要将这个方案用到任何大型项目中。

10.5.2
基于数据库的PACS存储

现在的数据库允许直接在数据库表中存储大容量的二进制数据，就像我们存储病人姓名或图像ID一样。这些大型二进制字段就是所谓的“二进制大对象”（blobs）。数据库架构师Jim Starkey发明了blobs，他将blobs描述为“可以吞下辛辛那提、克利夫兰或任何事物的东西”；那么无论DICOM对象有多大，PACS blobs都可以称作“可以吞下DICOM对象的东西”。一个正常的、未压缩超声电影回放，虽然只是存储在一个单一的多帧DICOM对象/文件中，但是也可以很轻松地达到好几百兆的数据量。为了替代在DICOM文件中存储DICOM对象，PACS也可能将他们直接扔到数据库中；比如前面提到过，在Image表中的Pixel Data（像素数据）列。

这种一体化模式的主要优点是可以利用这个数据库所能提供的所有工具。比如，带有DICOM数据的blobs能够利用数据库加密工具进行数据的加密，可以立即为你的PACS应用程

[2] 事实上，PACS 厂商很少依靠从 DICOM 数据中提取的 ID，而是自己去解决问题（如表的主键），这是为了保证格式的统一性以及数据的唯一性。

序增加一层强有力的数据保障（而基于文件的DICOM安全性仍然保持在低水平）。由于有了数据加密，所有当前的数据库都会向你提供审计工具，用这些工具你可以清楚地了解何时、如何以及是谁进行了DICOM对象的访问和修改。

和数据加密类似，blobs可以在压缩后优化存储；因此，它可以代替我们对于DICOM图像压缩算法（只能压缩图像数据）的依赖，一个数据库通常会提供它自己内嵌的压缩机制。这种机制可以无损压缩整个blob，无论里面存的是什么东西。

一些数据库厂商（比如Oracle）可能会做的更好，这有可能走向更有意思领域。他们在他们的数据库中加入了对基本DICOM对象的支持。因此，相比去管理那些缺少定义的blob来说，这些数据库能够认得DICOM对象，读取它们、解析它们的数据、处理最常见的DICOM字段、隐去DICOM数据、将DICOM数据映射成XML、甚至将DICOM图像从对象转化为预览小图，这个小图可以用在你的PACS应用程序上（Oracle 2007）。同样，还应记住当前这些表现很好的数据库访问blob的速度要比读取普通DICOM文件更快，它们可以在不同平台和操作系统上运行；可以自动备份；而且很容易就能处理大量的DICOM图像。简而言之，你可以从中获得一套免费又好用的工具，任何PACS所有者都会梦想拥有它，并且期望获得这些工具将会进一步开发和提升的承诺。

基于数据库数据存储的主要问题是无论用的什么数据库，它都会依赖数据封装和特定的数据库格式。对于所有于DICOM存储相关的任务，你不得不完全依靠数据库。对于基于文件的PACS存储，假如你的PACS或PACS数据库出现问题，你仍然保留着完整的DICOM文件——神圣的数据高于一切。如果使用基于数据库的存储，那么你相当于将所有鸡蛋放到了一个篮子里，因此你最好确定你的篮子是可以百分之百可靠的。

10.5.3 混合的PACS存储模型

我见过许多混合的PACS存储模型，但没有一个能真的让我印象深刻的。所谓的“混合”表示在DICOM对象（文件）和DICOM管理（数据库）之间的界限是模糊的或者是错误界定的，刺穿对象和记录，并且对于逻辑方面不太重视。比如，一个早期的著名PACS软件模型将DICOM文件逐个切片。只有数据库知道这些切片如何连接在一起，并且每个切片对它自己来说都是无意义的。当这个PACS过时了，快要散架了并且需要被别的系统替换时，这个数据迁移过程就是一场冗长的噩梦。从那时到现在，我也用过几个那个时代的DICOM设备，这些设备也将它们的DICOM数据存储成了类似的“切片状”。请记住在存储方面，DICOM标准已经设计的很好了，它可以将数字设备获取的数据存储为结构优异的DICOM对象；真的没有必要分割或分发任何东西。

另一个实例有些相反并且和一个致命的PACS公司有关。这个公司选择只在PACS数据库中存储某些DICOM数据；从不更新DICOM文件（对象）。既然这样也就不更新放射医师在图像上标记的注释。虽然DICOM对在DICOM对象中使用注释提供了丰富的支持，但是这根本无法实现。注释仅仅被保存到的PACS数据库，并且保存到了一个非常模糊的特有格式。结果，放射医师事倍功半：首先在PACS工作站的图像上放置注释，然后再手工在放射科数据库中再次录入这些注释（追踪注释测量数据的另一个工具）。这样就可以方便地导出注释数据了。不用说，这个过程是费时的、恼火的和充满人为错误的。我已经反复强调很多次了，我将一直重申下去：当选取DICOM软件时，请确认它是否能够导出由它收集的所有数据，并且这些数据的格式必须是标准的和明确定义的。这会是你的生活和整个项目都变得轻松惬意！

10.5.4 内部文件格式的选择

对你的应用程序在效率和交互方面的表现来说，文件格式的选择是最基本的。尤其是如果你的PACS使用的是基于文件的DICOM存储，就更应该重视文件格式的选择。理想上，我们会选择DICOM。此外，如果你不得不压缩它（对于现今放射科的数字数据卷来说，你一定会这么做），请一定要使用标准的无损DICOM压缩（6.2.1），比如8比特和12比特无损JPEG压缩。对于其他DICOM应用程序来说，这将会使你PACS所存储的文件兼容性更好。

此外，为了避免过分存储，在常见的医疗成像系统中，通常会将他的成像数据用两种格式进行存储：原始DICOM文件（为兼容性考虑），以及私有格式的图像文件（为获得最佳处理考虑）。不用说，这肯定会增加一倍存储量，并且为了实现在两种格式之间进行转换，而增加了实际的处理开销。当这种系统来到一个业务繁忙的医院时，他自己的低效存储方案只能带来缓慢和不可靠的表现，这都源于在一份拷贝能够搞定的前提下还需要管理两倍的数据。开始只会使用一个小服务器的系统，很快就会占据整个服务器机架。PACS应该足够聪明来处理单一内部数据格式，显然DICOM更适合成为这种格式。