DOI:10.13656/j.cnki.gxkx.1995.03.014
广西科学 Guangxi Sciences 1995，2（3）：57～61

超媒体机 HyperBase
A Hypermedia Engine HyperBase

黄 瑜
Huang Yu

（广西计算中心广西软件新技术实验室 南宁市星湖路 32 号 530022）
（Guangxi New Software Technology Lab.，Guangxi Computing Center，
32 Xinghu Road，Nanning，Guangxi，530022）

摘要 主要介绍的是一个通用的超文本数据模型，这个模型是通过一个数据库系统来辅助实施的。这个利用关系数据库管理系统来开发的具有通用性的多媒体机被称之为 HyperBase。

关键词 超文本数据模型 超媒体机

Abstract In this paper，we will introduce a general hypertext data model，this model is based on a database system. This hypermedia engine is called HyperBase.

Key words hypertext database model，hypermedia engine

1 介绍

超文本系统是一个生成、构造、表达信息的重要工具。直至现在为止，还很少有做出能反映这一基本数据模型的工具，和支持这一数据模型的系统。这就引出了用户界面级的非自然组织结构与简单文件系统中实际存贮的一大堆对象之间的识别匹配的问题。

超媒体系统是一个允许生成和维护（多媒体）对象信息的网络。这些网络能用于对现有的知识介绍，也同样适用于新知识的生成。

HyperBase 是利用关系数据库管理系统来开发的，具有通用性的超媒体机。它不依赖于任何应用系统，是超媒体应用系统的核心。HyperBase 利用关系数据库所具有的特性，负责基于它所开发的整个超媒体系统的数据传输（包括 I/O 管理、通信、多用户管理）、存贮、查询、跟踪、版本管理等功能。利用 HyperBase，可以迅速开发出高水平、高质量的超媒体系统[1]。

HyperBase 是德国的 IPIS 研究所的 WiBAS 研究部的一个科研小组在开发 SEPIA 系统时，所提供的一个新概念。目前的 HyperBase 仍在完善之中，而他们在开发 SEPIA 系统时已经用上了一这种技术，在本文第 3 节中，将详细介绍 HyperBase 在 SEPIA 系统中的作用。在第 4 节中将详细介绍 HyperBase 的数据模型。

1995-07-01 收稿。

2 超媒体机

在这一节中，将讨论的是这个科研小组所开发 HyperBase 的目的以及弄清楚“应用独立”的意义。

2.1 开发超媒体系统的目的与途径

一个正常的超媒体系统应该是独立于某个特定的应用系统。即它应具有如下的特征：

- 将对象存贮管理同对象浏览体分开。
- 为超文本定义一个与应用无关的数据模型。然后，这样的数学模型既能可用于超文本调用模型，也能作为标准的超文本文档（hypertext document）的交换格式。
- 建立一个与实际应用存贮结构无关的模型。
- 开发一种能反映数据模型的查询语言。
- 建立可以由不同用户可以访问的超文本对象。

目前的 HyperBase 完全满足这些要求。作为超文本机所需要的很多功能（如多用户管理、通信管理等）都能由商业性的数据库系统提供。因此，这个开发小组选择了将 HyperBase 建立在商业数据库管理系统上。

他们选择了一个关系数据库系统作为 HyperBase 的平台。另外上面提到的性能和操作要适用于任何关系系统，Sybase 提供了一个扩展的标准 SQL，即 Transact-SQL，允许使用面向对象设计的数据模型技术。因而也就能直接用于面向对象的数据库系

统。

2.2 对超媒体机的限制

HyperBase 的一个目标就是应用独立（与应用无关）。HyperBase 的一个特点就是不应包括对象的编辑器，这是必要的，因为超文本要负责向以后的编辑器开放（如图象编辑、语音、动画等）。

通常，一个编辑器不直接显示不经加工的数据，而是对它们进行处理能进行显示的结构化的数据。也就是，作为对数据作注解的信息，如结构、字体、文档的层次等，对这些信息的转换通常由编辑器来完成。

在通常的情况下，一般开发人员所能知道的只是极有限的几种编辑器内部数据格式。对于大多数的编辑器，其数据组织结构和它的内部转换机制都不能用于其他程序。因此编辑器之间的数据格式交换常常要通过文件的形式来完成。HyperBase 不提供对象结构的支持，而是脱离这种管理，并只是对要应用的节点内容进行翻译。

3 HyperBase 在 SEPIA 中的作用

HyperBase 可以独立于应用进行查看，它先对 SEPIA 进行仿真设计。

SEPIA 系统是一个创作支撑环境。一个功能强大的超文本系统应该独立于特定的应用系统。因此也就能用于实现在应用级与信息存贮级之间的中间媒介。

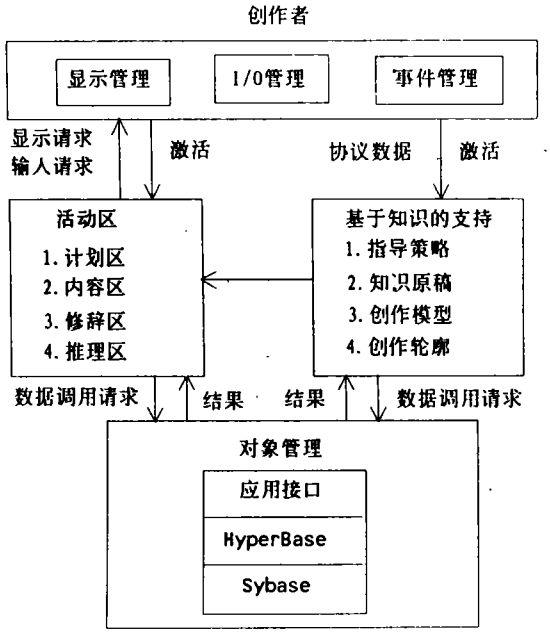

图 1 作为 SEPIA 系统一部分的 HyperBase

Fig. 1 HyperBase as a part of SEPIA

图 1 给出了一个 SEPIA 模型以及它们之间的互相关系[2]。I/O 管理提供一个面向窗口和基于菜单来访问超文本对象。可用的操作定义在一个所谓的活动窗中。并在不同的窗有不同的操作。

所有这些成分使用目标管理模块来保存超文本对象。这一模块基于数据库管理系统（Sybase）。数据库管理系统完成实际的存贮操作和多用户共享数据。在 HyperBase 之上，是与应用独立的超媒机。在应用界面层定义的是应用指定的类型的系统维护信息。

除了在活动区可用的通用超文本功能，系统还通过提供反馈和生成结构的一致性检查给定活动支持。这种基于知识的支持需要各种基于知识的成分。

所有这些成分都使用对象管理模型来存贮超文本对象。这个模型是建立在数据库管理系统（在这里就是 Sybase）。这个数据库管理系统完成实际中的存贮操作以及多用户访问共享数据的同步性操作。在这个模型的顶层是 HyperBase，即与应用无关的超媒体机。指定应用的数据类型和系统维护信息定义在应用接口。

在超媒体机对所有持续不断的对象处理（响应）存贮需求过程中，这个应用就是定义那些对象的应用有关语义。举例如下：

节点内容的翻译：由于不提供对节点内容的编辑和修改，因此只要保存应用界面所识别的一串简单正文（字符串），也就是一个命令集，如 PostScript 图象等。

点对点的链接：允许用户定义链（这个链不是把所有对象都作为其起点和终点，而只是取其其中的一部分），这里所指一部分可以是某个范围的正文、矩形（圆形、椭圆形）区域内的图象，图形目标的子目标（如一个国家地图）。

为完成这样的链，需要有关被调用对象的内部格式的信息。正如上面所讨论的一样，这些链必须被定义成在应用界面的内部，而不是在多媒体机这一级上。

版本管理：由于存贮有各历史时期以来的各种对象数据，然而，所涉及到的可供选择版本，或版本树选择必须定义在应用接口级上，这是由于它依赖于应用系统（如果在一个对象上增加一种属性、或生成一个链指到它之上：要生成的是新版本的对象吗？如果对生成的对象选择：要把它链接到哪里去，新版本的对象还是老版本的对象？）。

这个开发小组经过仔细研究在用户界面级应该建什么样的版本机制，并使它如何进入到应用界面后，决定基于历史信息由 HyperBase 来提供。

4 HyperBase 的数据模型

这一节是对数据模型的概念性描述，以及如何在提出的数据结构中反映出来。然后解释用何种数据结构能用于超文本系统上，并指出数据结构有什么特性。最后，将这一数据模型同其他文献上提出的相比较。

4.1 HyperBase 中的对象类

HyperBase 的数据模型使用一个面向对象浏览窗来对超文本应用的大范围查找。最基本的类是 HB-Object。每一个 HB-Object 都有很多任意的属性，这些属性的结构在以下有详细的描述。他们的数据模型的心脏由以下三个子类组成 HB-Object 、HB-Nodes 、HB-Links 、HB-Composite-Objects。

HB-Nodes 是 HB-Object 的内容与历史数据。它对应于超文本系统上所说的节点，例如，HyperCard 或 NoteCard 上的卡片[3]、KMS 中的帧[4]、Intermedia 中的文档[5]、Hyperty 中的文章[6]。

HB-Links 是用来连接两个已存在的 HB-Object。HB-Composite-Objects 是收集已存在的 HB-Object。这种收集器由用户定义。一个复合对象可以包括多个子对象。例如，对于 HB-Composite-Objects，可包含：数据库查询结果、选中的 HB-Object 集合、超文本网络中的使用指南、活动窗中的对象集合、传统书籍的章、节等。

4.2 HyperBase 的数据结构

下面使用一个预定义和一些记号法来说明 HyperBase 数据模型的形式表达方式。

记法：

设有 A、B，$A\times B$ 表示 A 和 B 的笛卡儿乘积，$A\cup B$ 表示 A 和 B 的并集，A^* 表示 A 中元素的有限元序列组成的集合，$A^+ := A\times B^*$，A^{fin} 表示 A 中所有子集的集合。

预定义域：

OID 表示对象标识符的集合，SOID 表示子对象标识符的集合，SOID 在整个系统中不是唯一的，但对于给定的 HB-Composite-Object 来说是唯一的。

Ref 表示调用对象的集合。作为一个数据结构，Ref=OID。然而，要保证任意时间任意点的完整性限制，每一个 HB-Node、HB-Link 或 HB-Composite-Object 的 Ref 的值都存在有它作为 OID 的值。

Time 表示时间值的集合，String：=ASCII*。SV 表示存贮值的集合。OI 表示一个代表有序信息的数据结构。

HyperBase 支持节点、链、复合节点和用户自定义的属性的生成、操作和存贮，并维持对象的历史数据。因此，应定义下面的数据结构。

HB-Object：数据库中不能被用户修改的、由对象标识符定义成唯一的每一个对象。除了这一点，每个对象都有作者名和生成日期，并且可以增加很多任意类型的属性到它之中。

$$\text{HB-Object}: = \text{OID}\times\text{String}\times\text{Time}\times\text{HB-Attributesfin}$$

在这里 $\text{HB-Attributes}: = \text{String}\times\text{SV}\times\text{HB-A-History}$

并且 $\text{HB-A-History}: = (\text{String}\times\text{Time})^+$。

属性有一个名字，这个名字在要加入的对象的内部是唯一的，属性值、历史。历史的属性是一个清单，这个清单包含修改这一属性的人的名字和编辑的时间。这个清单以生成这一属性的人的名字和时间开头，因此不可能是一个空表。

HB-Note：节点就是 HB-Object 加入内容以及历史数据的 HB-Object。

$$\text{HB-Notes}: = \text{HB-Objects}\times\text{SV}\times\text{HB-N-History}$$

这里 $\text{HB-N-History}: = (\text{String}\times\text{Time})^*$

节点的历史也是一个清单，这个清单包含修改过这个节点的人名和时间。这个清单仅仅是对修改内容操作的操作作应答，而不管对属性的添加、删除或修改。后面的信息是维护 HB-Object 中每一个属性中的历史属性。

HB-Link：链是连接两个已存在对象的 HB-Object。它们有调用节点、链、或复合对象的起点和终点。

$$\text{HL-Link}: = \text{HB-Objects}\times\text{Ref}\times\text{Ref}$$

HB-Composite-Objects 复合对象是具有子对象和历史的 HB-Objects。

$$\text{HB-Composite-Objects}: =$$
$$\text{HB-Object}\times\text{HB-Subobjects}\times\text{HB-Co-History}$$

这里：$\text{HB-Subobjects}: = (\text{SOID}\times\text{OI}\times\text{Ref})^{fin}$ 并且 $\text{HB-CO-History}: = (\text{String}\times\text{Time}\times\text{HB-Subobjects})^*$

一个子对象就是一个调用节点、链、或其他复合对象并且用子对象标识符 SOID、用有序信息 OI、Ref 调用来表示。

在一个复合对象，可能有几个（对象）调用同一个对象。因此，每一个调用都通过复合对象内部的唯一子对象来唯一标识，并且是不能修改的。

子对象集可能是排序的。站在用户这一面来看，他并不关心内部数据结构是否有序。它只关心是否能

访问到其他子对象。

复合对象的历史不仅仅记录修改对象的人名和修改的操作时间，而且还保存操作前给定的复合对象中子对象集合的状态。换句话说，对于一个复合对象有当前状态、有效版本、无效版本的清单。

由于有序信息表现的一致性，因而在复合对象中通常都加入很多操作，而不管其使用目的如何。

4.3 数据模型的基本原理

在这一节中，解析上面提到的对象类在超文本系统中的用途。

模型的依赖性：

·链能在某些对象上提供附加的信息（例如：内容）。这一额外信息仅依附于它所代表的对象。在他们的模型中，用所依附对象中的 HB-Attributes 表示。HB-Attributes 不能独立于它所依附的对象存在。

这些信息可以是系统维护（例如：屏幕上一个窗口的大小和位置都是节点的属性）或用户维护（例如：在节点上设计关键字）。

·链能在两个对象之间建立从属关系（例如：链接一个脚注到对应的全文）。如果其中一个被删除，则链也必须解除。（在他们的例子中，如果从数据库删除对一被篇调用的文章，脚注仍然完整地保存，这时指向全文的就不再有任何意义）这就是用在他们模型中的 HB-Link。

·链能收集对象到通用的结构中（例如：收集某些文章以形成一本书）。在他们的模型中，结构由 HB-Composite-Objects 来表示。HB-Composite-Objects 允许当作一个对象贮存器实体来看待，它可以被编辑，也就是可以增加或删除子对象，也可以安排它们的顺序。对 HB-Composite-Objects 的操作不会影响子对象本身。

在 HyperBase 的模型中，链的所有这些各种不同功能都是通过 HB-Attributes、HB-Links 和 HB-Composite-Objects 这些概念来表示。他们更趋向于在相应的相关级上组织超文本信息网，而不是平易的节点－链模型。

在很多应用系统中，复合对象都有两个目的：一方面它提供一个能够用于系统支持的工作区中的数据结构。这里有几个但不是全部的节点和链被收集并显示在屏幕上的一个窗口中。在这个特定的区域可以进行操作。例如，很容易就可以生成一个节点在这个特定的激活的区域中。

另一方面，要对那些相对来说是信息碎片归集起来管理，用户就可以依据这些来生成复合对象。（例如：它想管理维护一些局部的文章片断）。例如，一个查询的结果是一个具有某些共享特性的对象集合，也可以说是查询条件。

在这两种情况中，用户希望应用对给定的对象集合操作，而不是对数据库中全部对象。例如，用户只希望在一个指定中工作区中或在更早的查询中检索一个节点。因此 HyperBase 提供两组操作，一组是针对全部数据库的，另一类只针对给定的复合对象中子对象的操作。

除了这些，还有将一类复合对象同另一类联系起来的功能。例如，一个人想通过把查询结果添加到工作中已有的内容中的办法把查询结果包含到工作中里面去。对于所有这些情况，复合对象被用作（担当）一致性的工具以提供所需数据结构和功能。

在 HyperBase 数据模型中，链是第一级对象。一个链可以有指向它的属性，还有其他的链作为它的源（起点）的目的（终点）对象。

HyperBase 不仅为多媒体对象提供数据结构，还要维护下面所说的完整性限制：

·空悬调用被禁止。这些调用如下有：

—没有起点或终点对象的链

—子对象进入一个复合对象中用一个不存在的子对象

—属性不是对象的一部分，或

—链被包含进一个复合对象，而它的起点对象和终点对象不进入这个复合对象。

·一个复合对象不能是自已的子对象。

4.4 HyperBase 同其他超媒体系统的结构比较

从文献中，提到的几种超文本数据模型知道。由于版面的限制，仅讨论那些与 HyperBase 类似的结构。

从 HAM 系统中[7]他们得到了很多思想。在这两种不同的结构中最主要的差异是对待复合对象方法。在 HAM 中，所谓的上下文是指收集在一个对象集中。然而，那些上下文的最主要思想是维持对象历史版本。例如，每一个上下文必须有一个双亲，而且主要的操作是结合这两个上下文。上下文不能同时从很多对象得到（例如，对整个数据库的查询结果）并且不是第一级对象（例如，不能被链调用）。

对于 DHRM 系统[8]，这个系统类似 HyperBase 的，这个模型也涉及存贮层和脱离内容/内部节点结构（在结合层内部）和超文本的表示（运行时层）在模型的外面。对于存贮层，HyperBase 同他们模型的主要差异是，它们允许多种方法的链（或多元链），而 HyperBase 只使用二元链，并且他们不提供历史信息的格式。他们的复合结构的思想非常类似于 Hyper-

Base 的 HB-Composite-Object，就是不清楚他们能用些什么操作。

在概念级，主要的差异还在于，他们是在某些域的顶层建立他们模型，而这些域不指定，而 HyperBase 是指定所有的域。

5 HyperBase 的实现

从其他的数据模型语义域定义可清楚地看到，HyperBase 最好实现为一个面向对象数据库系统的现实的复杂对象。然而，由于可资利用的原因，他们决定采用商业性的关系数据库系统 Sybase 4.0 来作为 HyperBase 实现的首选软件。

在目前实现（的系统）中，HyperBase 利用以下数据类型：OID：=N 和 SOID：=N，这里 N 表示正整数集，他们仅支持能进行存贮的 SV 值的集合中的标准数据类型，它的域定义为：SV：=String∪Byte * ∪Time∪N∪R ∪ {NOVALUE}，这里 R 表示实数集合，NOVALUE 一个空值，即不属于哪个域，是用于作为节点的，并且还没有内容和属性，是一个还没有给值的空值。

在所实现的 IO（即输入输出）中，IO 所代表的有序信息在 HB-Composite-Object 中，他们支持（局部）分级排序，在子对象级别（按级别组成集合）内部，子对象的位置是通过其父亲的子对象标识符和其兄弟中的前任标识符来表示的。

对于关系数据库系统，他们必须将域分解成对应的关系。其分解方案他们使用标准数据模型技术。

6 结论

目前的超媒体系统大多是为某个领域定制的，因此开发难度大，时间长，而且没有通用性。HyperBase 作为与应用无关的超媒体机，在 SEPIA 系统中已经证明是很有用。目前的 HyperBase 来作为超媒体系统的顶层，并且所有对象都由 HyperBase 来管理，则建立系统时，不辟重新开发实际存贮、定位、跟踪、通信、多用户管理等 HyperBase 所能完成的功能。应用系统的开发只需对界面、浏览　创作（创意）下功夫，工作将会变得更容易。

参考文献

1 Helge A. Schutt, Norbert A. Streitz. HyperBase. A Hypermedia Engine Based on a Relational Database Management System, Hypertext: Concepts Systems and Applications, INIRA, France, November 1990, 95.

2 Streitz et al. From Ideas and Arguments to Hyperdocuments: Travelling Through Activity Spaces. Proceedings Hypertext 1989, 343～364.

3 Halasz et al. NotesCards in a Nutshell. Proceedings ChI '87, 45～52.

4 Akscyn et al. KMS: A Distributed Hypermedia System for Managing Knowledge in Organizatians, CACM, July 1988, 31 (7): 820～835

5 YanKelovich et al. Intermedia: The Concept and the Construction of a Seamless Information Environment, IEEE Computer, January 1988, 21 (1): 81～96.

6 Marchioni, Shneiderman. Finding Facts vs. Browsing Knowledge in Hypertext Systems, IEEE Conputer, January 1988, 21 (1): 70～79.

7 Campbell, Goodman. Ham: A Genceral Purpose Hypertext Abstract Machine, CACM, July 1988, 31 (7): 856～861.

8 Halasz, Schwartz. The Dexter Hypertext Reference Model. Proceedings of the NIST Hypertext Standardization Workshop, Gaithersbury. MD. January 1990, 16～18.