第 36 卷　第 14 期　　计　算　机　工　程　　2010 年 7 月
Vol.36　No.14　　Computer Engineering　　July 2010
·软件技术与数据库·　文章编号：1000—3428(2010)14—0049—02　文献标识码：A　中图分类号：TP392

# 互动式游戏数据库的数据存储与管理

王秀坤[1]，商振东[2]，朴　勇[1,2]
(1. 大连理工大学电子与信息工程学院，大连 116023；2. 大连理工大学软件学院，大连 116620)

**摘　要：**互动游戏的实时与动态交互性要求游戏能够快速调用存储介质中的图像、语音、动作等多媒体数据，作为游戏角色参与游戏。针对该特性，设计适合于互动游戏的专用数据库管理软件，构建能够存储和管理多媒体数据的基于关系的数据库模型，从物理存储和逻辑语义 2 个方面探讨基于 Hash 的多媒体数据的存储与管理方法。

**关键词：**互动游戏；数据存储；二级 Hash

# Data Storage and Management for Interactive Game Database

WANG Xiu-kun[1], SHANG Zhen-dong[2], PIAO Yong[1,2]
(1. School of Electronic and Information Engineering, Dalian University of Technology, Dalian 116023;
2. School of Software, Dalian University of Technology, Dalian 116620)

**【Abstract】** Because of the real-time and interactivity, an interactive game needs to call for multimedia data composed by picture, sound and action script as game roles stored on the disk quickly. Aiming at this attribute, a kind of special database management software focusing on interactive games is introduced, and a relational database model, enabled to store and manage multi-media data is built. The method of storing and managing multimedia data by Hash from the two aspects of both physical storage and logic semantic is studied.

**【Key words】** interactive game; data storage; two-level Hash

由于互动游戏的实时与动态交互性，如何提高游戏多媒体数据的读取速度是游戏设计的关键。本文设计了一个适合于互动式游戏的专用数据库模型，把数据库嵌入到互动游戏中，针对存储在外存的多媒体数据，研究基于 Hash 的多媒体数据的存储和管理方法，提高了数据的读取速度，能满足游戏的实时性和互动性要求。

## 1　互动游戏数据库的研究与构建

互动游戏与传统游戏相比，最大的区别在于游戏情节的变化由游戏和游戏玩家共同决定，允许玩家最大限度地参与游戏。互动游戏需要保存一些在游戏制作过程中预先制定好的数据，因为这些数据是庞大的，不便通过网络从服务器端传输给客户端，所以只能保存到客户端以便快速查找。同时，游戏客户端运行在玩家的电脑上或游戏机上，考虑到硬件配置要求等原因，不便于采用商用数据库来实现数据的存储。

本文针对互动游戏特点，设计了结构简单、操作便捷的专用数据库模型。该模型作为游戏的底层支撑完成数据的存储和查找，不会给游戏玩家增加任何费用[1]。该模型的特点如下：

(1)该数据库模型专用于互动游戏，通过分析游戏中的动作规则、角色特点、场景特点、音效特点和情感描述方法，来定义数据库中数据文件信息表的字段[2]，从而有效地提高了数据库的查询效率。

(2)为了保证游戏的正常运行，要求内存数据库和游戏程序、资源文件一起打包成游戏的安装程序。玩家启动游戏时，数据库运行在 PC 机上支撑游戏运行，加快游戏中数据读取的速度，使游戏在玩家感觉不到数据库存在的情况下流畅运行。

(3)该模型存储数据分为图片、语音和动作 3 类，每类数据大小接近。为了便于数据管理，对于所有数据按照“目录_分类名_含义名”方式命名。

互动游戏数据库体系结构如图 1 所示。

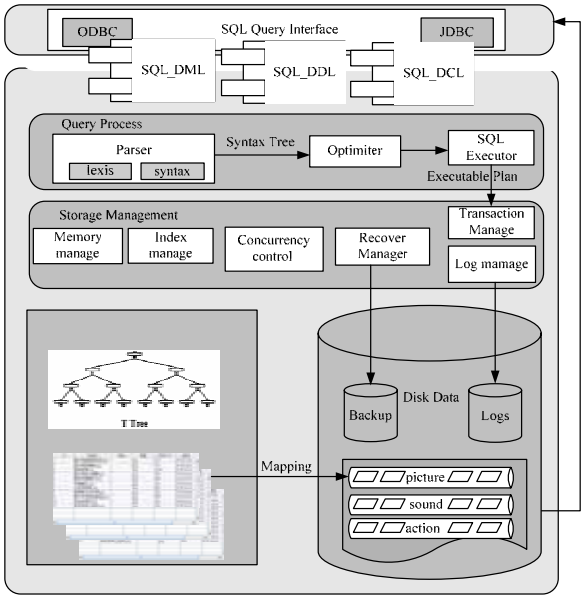


**图 1　互动游戏数据库体系结构**

**基金项目：**大连市高新园区管理委员会基金资助项目“互动游戏嵌入式数据库技术”(2008386)
**作者简介：**王秀坤(1945－)，女，教授，主研方向：数据库与决策支持系统；商振东，硕士；朴　勇，讲师
**收稿日期：**2010-02-09　**E-mail：**piao@dl.cn

## 2 基于 Hash 的数据存取方法

### 2.1 基本思想

在数据库的硬盘介质中，数据分别存储在图片、语音和动作 3 个目录中。为了提高数据的读取速度和管理效率，可以设计数据文件的存储结构，以便在读取文件时数据库尽可能保留文件的逻辑信息[3]。对于某款游戏，客户端所存数据通常不发生变化，因此，本文采用适用于静态数据快速查找的 Hash 方法[2]。设计二级 Hash 散列地址函数，定义“桶”结构封装数据文件。基本思想如下：以数据库模型中的文件名为关键字，经二次 Hash 求得数据存储地址。定义数据库文件的命名规则为“目录_分类名_含义名”的 3 级命名方式。与此相应的数据存储分为目录、桶和数据块 3 层。按照图片、语音和动作 3 个目录分类存储相应文件，每个目录下有 $M$ 个桶，每个桶中含有 $B$ 个数据块，同一目录下的数据块大小相同。设第 $i$ 个桶中含 $N$ 个数据块，通过一级 Hash 确定数据所在桶号为 $i(0\leq i<M)$，通过二级 Hash 得到数据在第 $i$ 个桶中的数据块的序号 $j(0\leq j<N)$，加上桶地址，得到所需数据的物理存储地址。

在具体实现中，以一般 RPG 打斗类游戏为例，根据游戏需要设定所需各类数据文件数目不超过 1 000。以图片数据为例，统计游戏的情节信息并考虑到数据分布的均匀性，设计文件的二级分类(具体指英雄、野怪、坐骑、猛兽等)为 25，因此，设定每个目录桶的数目 $M$=29(多余部分作为冲突缓冲空间)；为了便于数据的存储，合理利用空间资源，经过反复测试，$M$ 能控制同一目录下各桶的数据数目之比绝大部分在 1:1.2 之内，而且同一桶下数据块的数目大多在 20~35 之间。为了便于管理，设定各桶的数据块数目均为 $N$=37。桶内数据块的大小主要考虑各类数据文件的空间以及整个游戏的磁盘空间，原则上每个数据块能够封装一个数据文件[4]。根据统计相关的图片、语音和动作文件信息，同类文件的大小相差不大，设定图片、语音和动作目录下的各类数据块大小分别为 105 KB, 1.6 MB, 310 KB，估算该游戏的数据需占硬盘空间为 2 GB。

### 2.2 Hash 函数设计

借鉴 Java 语言 Hashtable 索引方法中的 hashcode 编码方式，本文利用 hashcode 实现文件名的数值转换，该方法能够保证单个字或词翻译成数值的唯一性，但是由于 hashcode 可能为负数，因此转换后的 hashcode 需要和 0x7FFFFFFF 做与操作，以保证其为正整数。在查找文件时，以图片类型文件为例，Hash 函数设计如下：

(1)确定文件类型：查找“目录_分类名_含义名”，以“_”为分隔符，根据“目录”确定查找文件类型。

(2)一级 Hash 函数设计：

$Hash_1$(分类名)=(*hashcode*(分类名) & 0x7FFFFFFF) mod $M=m$

确定该图片存储于 $m$ 号桶。

(3)二级 Hash 函数设计：

$Hash_2$(含义名)=(*hashcode*(含义名) & 0x7FFFFFFF) mod $N=n$

确定该图片存储于 $m$ 号桶的第 $n$ 号数据块。因此，该图片地址为

*Addr*(目录_分类名_含义名)=*Addr*(桶 $m$)+$n$×*Blocksize*(图片)

### 2.3 冲突解决方案

采用 Hash 方法理想的情况是“一次定位到关键词对应的地址”。但在 Hash 技术中，冲突是不可避免的，即关键词 $key1\neq key2$，但是 $H(key1)=H(key2)$，只能减少冲突的概率。本文采用局部冲突空间和全局冲突空间相结合的办法[5]。数据存储结构如图 2 所示，局部冲突空间是指某个桶中数据块发生冲突，则将冲突数据存放至和该桶相连接的局部冲突空间中。全局冲突空间是指在同一目录下，若某个局部冲突空间已满，则将冲突数据存放在全局冲突空间中。在完成游戏情节设计的前提下，通过对游戏数据合理的分类设计，该解决方案既可保证数据的合理存储，又便于查询和维护工作[6]。

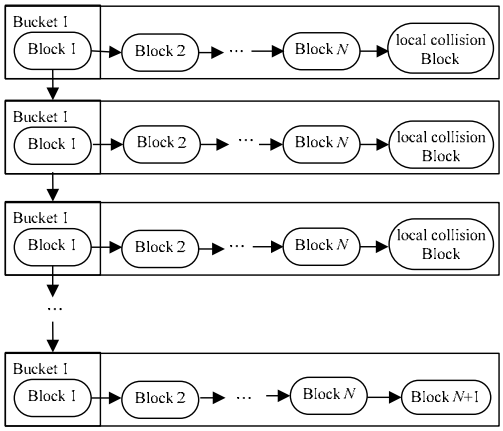


**图 2 数据存储结构**

## 3 基于游戏角色的数据管理

多媒体数据有字符数值、文本、声音、图形、图像等类型。由于多媒体数据的复杂性，一般采用面向对象的机制，将多媒体数据、展示属性、操作方法封装在一起，以降低用户使用多媒体数据的复杂性。因此，本文采用基于游戏角色的数据管理方法，以实现程序人员通过操纵游戏角色管理各类多媒体数据的目的。

在游戏设计中，图片用于角色的贴皮，即角色的“皮肤”，动作文件相当于角色的“骨骼”，而语音文件就好比角色的“血肉”，三者组合形成一个完整的游戏角色。对于某一游戏角色来讲，由图片、语音和动作文件组成，即 Role=(picture, sound, action)。

某个游戏角色的出场一般首先调用其图片文件和动作脚本，后续调用占用资源较大的语音数据。针对这一特点，本文的设计在调用角色图片或动作文件的同时，系统会根据角色表的关联规则判断所需的语音数据，提前读取语音数据进入内存等待游戏调用，实现“预判”目的，使游戏实时性效果更佳。

本文模拟了 PC 机在不同负载(以 CPU 占用率为标准)的情况下，读取不同大小多媒体数据的时间开销比较，结果如表 1 所示。其中，A 表示基于 Hash 方法的多媒体数据读取方法；B 表示基于文件系统的多媒体数据读取方法。

**表 1 多媒体数据读取时间比较**

| CPU 使用率/(%) | 多媒体数据读取时间/ms | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 300 KB | | 800 KB | | 1 MB | | 5 MB | | 10 MB | | 15 MB | | 30 MB | | 50 MB | |
| | A | B | A | B | A | B | A | B | A | B | A | B | A | B | A | B |
| 10 | 0 | 281 | 16 | 656 | 15 | 844 | 31 | 4 078 | 47 | 8 141 | 62 | 12 453 | 125 | 24 453 | 219 | 40 844 |
| 30 | 0 | 281 | 0 | 703 | 0 | 859 | 31 | 4 265 | 46 | 8 407 | 63 | 12 562 | 141 | 25 765 | 235 | 42 328 |
| 50 | 0 | 281 | 16 | 688 | 15 | 891 | 31 | 4 219 | 31 | 8 766 | 63 | 12 937 | 141 | 25 890 | 219 | 42 969 |
| 70 | 0 | 297 | 0 | 735 | 0 | 890 | 31 | 4 391 | 63 | 8 812 | 78 | 14 000 | 140 | 26 906 | 235 | 45 234 |
| 90 | 0 | 312 | 0 | 812 | 0 | 1 000 | 16 | 4 641 | 47 | 9 359 | 63 | 13 875 | 157 | 27 703 | 250 | 47 125 |
| 平均 | 0 | 290 | 6 | 719 | 6 | 897 | 28 | 4 319 | 44 | 8 697 | 66 | 13 165 | 141 | 26 143 | 232 | 43 700 |

(下转第 53 页)