一种适用于嵌入式人机界面的实时数据库内存数据组织方法

A Memory Data Manipulation Method In Real – Time Database For Embedded HMI

孙士明 董占勇 (中国石油大学(华东)计算机与通信工程学院 257061)

摘 要: 根据嵌入式人机界面的特殊数据环境将其内存数据按照不同属性分为静态域和动态域,对于静态域采用基于T-树的索引方法,动态域采用双向链表索引结构。避免了动态数据更新对静态数据的影响,节省了内存空间,降低了检索的时间复杂度,满足嵌入式人机界面的灵活性和实时性需要。

关键词: 嵌入式人机界面 静态域 动态域 T-树 双向链表

1 引言

随着工业信息化水平的提高,嵌入式人机界面(HMI)越来越多地应用在工业现场,使得人机交互更加形象化。由于工业现场要求嵌入式系统具有较强的实时性,所以合理的数据组织,高效的数据查询是提高嵌入式 HMI 性能的关键问题。

为了避免嵌入式 HMI 工作过程中内存数据与 Flash 之间频繁的 I/O 操作,克服磁盘延迟时间难于预测的弊端,嵌入式 HMI 的实时数据库宜采用主存方式。嵌入式 HMI 既要绘制 PC 端界面编辑者的画面,又要为现场的操作人员和各个数据采集点实现数据交互,所以其数据来源比较复杂。针对这种特殊的环境,设计合理的数据组织方式和索引结构从而提高数据查询处理的能力,是满足现场强实时性要求的有效途径。

本文介绍一种适用于嵌入式人机界面的分域内存数据索引机制。

2 数据分域方法

2.1 嵌入式人机界面的数据流程

嵌入式人机界面是联系着 PC 端界面设计者,工业现场操作人员和数据采集端三者的纽带。用户先通过 PC 端的界面编辑程序编辑图形界面和设定工作参数,经过编译之后形成图形界面的特征数据文件,再经过转换下载到嵌入式设备中;嵌入式人机界面端根据工业现场的需要检索特征数据文件中的数据,经过数据恢复后显示在触摸屏上或者传递给数据采集模块;同时,嵌入式人机界面将实时采集到的数据按照用户的要求实时显示或者保存在实时数据库中;现场的操作人员又随机地对人机界面设置命令和参数。整个过程如图 1 所示。

2.2 数据分域

由图 2.1 可见,由 PC 端下载到嵌入式 HMI 端的特征数据包括界面各个基本元件的数量,属性,状态参数等,还包括预设的对控制端各存储器地址的操作参数。这些数据用于在嵌入式 HMI 端按照预定的规则恢复出界面编辑者设计的图形界面以及初始化各元件的参数。这些数据的特点是一旦下载到嵌入式端在 HMI 运行过程中不随时间的变化而变化,其元组数量也是固定不变的,即数据为只读的。这类数据我们规划为静态域。另外,由数据采集模块实时采集到的数据需要不断地对实时数据库进行插入操作,由于内存资源的有限性,一段时间的历史数据要定期向 Flash 或远程数据库转移,这又需要对实时数据库进行删除操作,由现

场操作人员通过触摸屏输入的数据也是随机的。这类数据的特点是数据源组不断变化，需要随时更新。我们把这类数据划归为动态域。

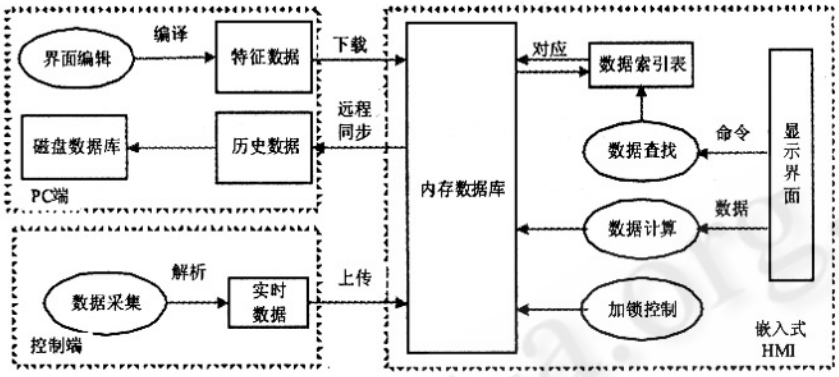

图 1　人机界面实时数据库数据流程

3　分域的内存数据索引方法

传统的内存数据库索引方式是对所有的数据统一采用一种索引结构，这显然不符合嵌入式人机界面多数据来源和数据特性鲜明的特点，所以根据不同的数据域分别采用不同的索引结构。

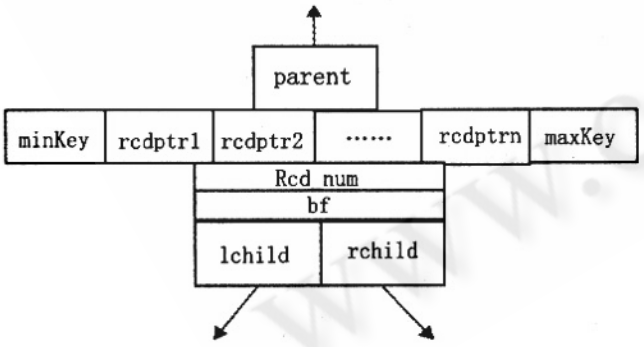

图 2　T－树结点结构图

当前经典的索引机制主要分为两种三大类：一类是基于 HASH 函数的对数据随机组织的索引机制，比如线性 HASH 等；另一类是基于查询树的对数据有序组织的索引机制，比如 B－树等；第三类是前两种索引机制的混合，如 Hybrid－TH 方法。对于静态域的数据，为满足嵌入式 HMI 实时性与空间的需求，采用基于改进的 T－树的索引方法，实验证明该方法不仅提高了空间利用率，而且降低了算法复杂度，适合内存数据库的特殊条件[1]。

对于动态域的数据由于人机界面工作过程中不断地对该域数据元组进行插入和删除操作。若采用 T－树建立索引，那么结点的插入和删除都将引起 T－树的失衡。对于插入操作导致的失衡，T－树要进行一次平衡旋转才能恢复平衡，而对于删除操作，可能经过多次平衡旋转才能恢复 T－树的平衡。这样会使 T－树不断地重构。所以，根据动态域数据的特点，应该采用直接地址访问的链式结构，以便提高插入删除操作和查询操作的效率。

3.1　内存数据的组织方式

在设计内存数据库的数据结构时，在物理上采用段页式的存储结构，把对象内存分为数据段，索引段，和缓冲段。所有的段内又分页，每个页面在段内连续分配，其结构为线性结构。数据段中属于同一域的元组对象存储于同一页面中。索引段用于存储对于不同域的索引结构。缓冲段用于数据的缓冲 。

3.2　静态数据域的索引

T－树是 Lehman 提出的适用于主存数据库系统的索引结构。它是在一个结点中有多个数据对象的二叉树，遵循 AVL 的特性，在每个结点中，数据对象按升序排序，指针 parent pointer、left child pointer 和 right child pointer 分别指向该结点的父结点、左子树和右子树。

如图 2 和图 3 所示，在每个结点上，小值在左边，大值在右边。对于每个内部结点 A，有一个叶子或半叶子结点，它上面的数据值是 A 结点最小值的前序，同样，存在一个叶子结点或半叶子结点，它上面的值是 A 结点最大值的后续。

T－树索引在每次装载 PC 端用户编译好的嵌入式人机界面的特征数据时生成，装载之后数据不再发生变化，根据工业现场人员的随机操作检索此部分静

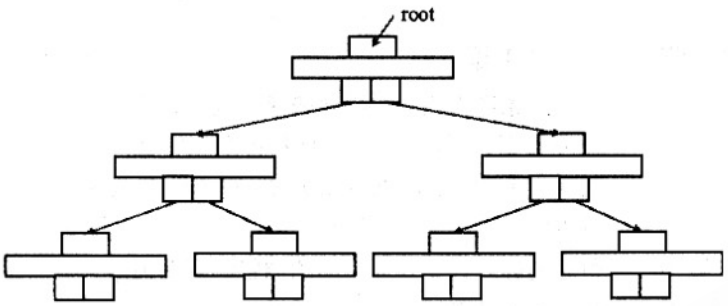

图 3　T－树结构图

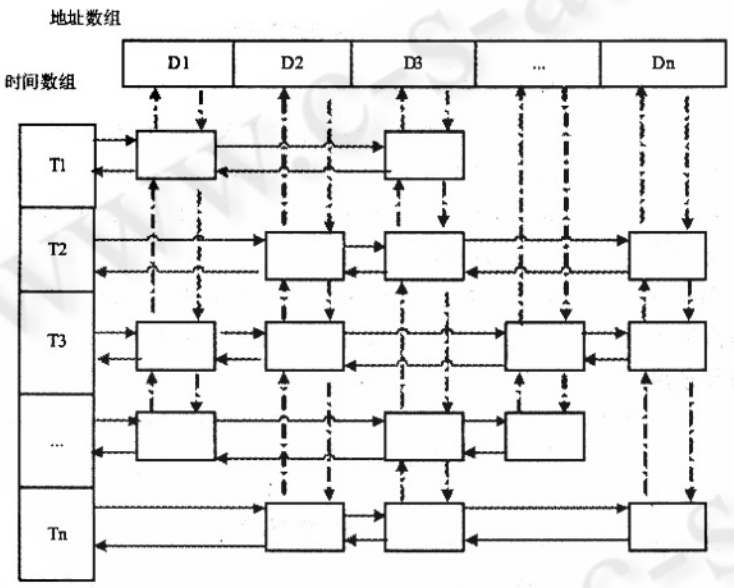

图 4　有序链表结构示意图

态数据来显示界面。

T－树结点的结构可定义为：

```
Struct Tnode
{
Tnode * parent; /* 指向父结点的指针 */
Int minKey; /* 结点中的最小键值 */
Int maxKey; /* 结点中的最大键值 */
Tnode * lchild, * rchild; /* 指向左子结点和右子结点的指针 */
Record * rcdptrs[n]; /* 结点中存放的各个记录指针 */
Int rcd_num; /* 结点中存放的记录指针数目 */
Int bf; /* T 树结点的平衡因子 */
};
```

将 Ttree 定义为指向 T－树根结点的指针类型：

```
Typedef Tnode * Ttree;
```

3.3　动态数据域的索引

嵌入式人机界面采集的数据一般有两种索引条件：按时间索引和按数据来源地址索引。这两组索引可以分别按照时间顺序和数据来源地址顺序构成线性表，线性表的每个单元为一组链式结构的头结点。每个头结点又链接着一组双向数据链表。两组索引总体结构如图 4 所示，其内部结点如图 5 所示。

时间数组的头结点的数据结构为：

```
Typedef struct{
  Int TKeyID;
  linkNode * pHead;
  }TfstNode;
```

其内部结点的数据结构为：

```
Typedef struct{
  Int TkeyID;
  Int DkeyID;
  linkNode * pTprv;
  linkNode * pTnxt;
  linkNode * pDprv;
  linkNode * pDnxt;
```

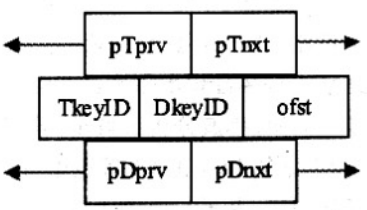

图 5　内部结点结构

```
  int ofst;
}linkNode;
```

基于链式结构的索引首先根据时间关键字 TkeyID

或来源地址关键字 DkeyID 找到相应的头结点，这个过程采用折半查找的算法。然后沿头结点顺序查找双向链表直到找到对应的 DkeyID 或 TkeyID 其中对应的 ofst 就是页偏移地址。

4 性能分析

对于静态区的一个 K 阶 T－树索引，每个结点的最大分枝数为 2，路经上每个内部结点的比较次数为 2，终结点的比较次数为 lnk，若有 N 个关键字，则总的比较次数为 lnN＋1/2ln(N/K) 。若规定 K 阶 T－树所有内部存储关键字实际个数大于等于 K－2，这样总的比较次说约为 lnN。所以对于静态区的 T－树索引的时间复杂度为 O(lnN)。

对于动态数据区的链式索引，由于在内存的空间限制，只能规定存储就近时间段的数据，历史数据将转移到 Flash 之后同步到远程商业数据库，而在现场查询中按时间关键字查询一般要显示在此时段所有的数据地址来源的数据，所以按时间查询的算法复杂度等于对头结点的算法复杂度。对头结点采用折半查询的算法，若有 N 个关键字，则其算法复杂度为 O(ln(N＋1)－1)。若按数据来源地址查询，除对头结点的折半查询外，对双向链表再进行顺序查询，所以其算法复杂度为 O(ln(N＋1)－1＋N)。

5 结束语

根据嵌入式人机界面的特殊数据环境提出了对其内存数据分域的方法，根据各自数据特征对于静态域的采用 T－树的数据索引方法，对于动态域采用双向链表索引结构。避免了动态数据域由于数据实时更新引起的静态数据存储空间的调整和查询树的失衡与重构。对于动态域直接的指针操作使得查询更加简洁。

此方法通过基于 ARM 的嵌入式触摸屏人机界面项目的验证取得了良好的效果。实验证明，分域的内存数据库数据组织方法适合嵌入式 HMI 的特殊环境。

参考文献

1 张志鸿等，面向交易实时数据库内存数据的组织方法，北京理工大学学报，2004，24(8)．683－686.

2 Leman T J. A study of index structures for main memory databases management systems. The 12th VLDB Conference, LosAltos, CA, 1986.

3 马洪连等，一个适用于内存数据库系统的多维索引结构，计算机工程与应用，2003，29．213.