# 图书馆借阅信息数据仓库的设计与构建

**林晋霆 网络工程 2014051795**

目录

[图书馆借阅信息数据仓库的设计与构建 1](#_Toc485557654)

[1. 简介 1](#_Toc485557655)

[2. 数据仓库需求分析与模型设计 2](#_Toc485557656)

[2.1 需求分析 2](#_Toc485557657)

[2.2 概念模型设计 2](#_Toc485557658)

[2.3 逻辑模型设计 4](#_Toc485557659)

[2.4 物理模型设计 5](#_Toc485557660)

[3. 原始数据的 ETL 与数据仓库的实现 5](#_Toc485557661)

[3.1 数据挖掘算法的选择 5](#_Toc485557662)

[3.2 原始数据的 ETL 6](#_Toc485557663)

[3.3 数据挖掘系统的实现 7](#_Toc485557664)

[4. 挖掘结果的展示与未来应用的探讨 7](#_Toc485557665)

## 简介

随着图书馆借阅数据的不断积累，使用人工方式对借阅数据进行管理已经无法满足实际需求。在结合发达的计算机技术后，通过数据仓库对积累的海量图书借阅数据进行检索与分析已经成为了当前管理海量数据的唯一方式。

通过对图书馆海量借阅数据的挖掘，我们可以获取隐藏在其中的信息，并分析结果来对未来的决策进行支持。如通过 FP-Growth 算法来发现书籍之间的频繁项，进而在用户检索可借阅书籍的时候进行针对性的推荐，以提高图书的流通率

## 数据仓库需求分析与模型设计

在建立一个数据仓库之前，我们首先要完成需求分析以及数据仓库的概念模型、逻辑模型和物理模型的设计。

### 需求分析

显然，构建此数据仓库的目的在于管理和分析已有的海量借阅数据，并在用户发起查询请求的时候即时提供计算结果。那么，从这个起点出发，我们就可以得到以下需求点：

1. 通过分析提供的原始数据来计算其中包含的频繁项信息；
2. 需要对提供的原始数据进行高效的ETL操作；
3. 在调用者发出查询请求的1秒以内完成与本次查询有关的分析计算工作，并以适当的形式把计算结果返回到调用者一方；
4. 提供信息全面、形式得当的报表供管理员确认系统的工作状态；
5. 其他可能的合理需求。

### 概念模型设计

本节使用E-R图对所获得的本数据仓库进行建模。



**图2-1** 借阅记录数据仓库中的概念模型

图2-1所示的概念模型所代表的操作为：

1. 调用者（管理员或应用服务器）向OLAP服务器提交包含期望结果类别的查询要求；
2. OLAP服务器根据请求中的类别从数据仓库中抽取对应类别的数据，并计算频繁项集；
3. OLAP服务器把计算结果以约定的形式返回到调用者一方。

因此，我们可以从此概念模型中抽象出“用户查询请求”、“计算结果”和“原始数据”三个主题。在本数据仓库的预想使用场景中，“用户查询请求”来自于外部数据，故不需要在数据仓库中进行装载；“计算结果”来自于对原始数据的计算，也不需要在数据仓库中专门划分出空间储存；故本数据仓库需要装载的主题为“原始数据”。基于原始数据，我们就能很方便地根据调用者的请求而计算出所需结果。

在“原始数据”这一主题中，每一项均包含了以下属性：



**图2-2** 数据项E-R图

在原始数据中，每一册图书均有不重复的“图书记录号（册）”属性，而每一种（即所有ISBN相同的书籍）均有不重复的“图书记录号（种）”属性。故我们选定“图书记录号（册）”作为原始数据的主键。同时把主键和“图书分类号”添加到索引中，以加快查找速度。

### 逻辑模型设计

2.2节已经分析出本数据仓库需要装载的主题为“原始数据”，那么我们从所获的样例数据集出发，那么很自然地，我们就可以得到以下三种对数据进行分类计算的粒度以及在此粒度下的数据预计使用场景：

**表2-1** 数据粒度分类详细信息表

|  |  |  |
| --- | --- | --- |
| **粒度** | **用途** | **大小** |
| 大粒度 | 以年为单位的报告性分析所用 | 每单位约27万 |
| 中粒度 | 以月为单位的即时计算所用 | 每单位约2万 |
| 小粒度 | 以项为单位的详细报告所用 | 每单位为1项 |

在数据分割方面，我们采用与表2-1相同的粒度对数据进行分割。

在底层记录模式方面，本数据仓库不对数据记录增加导出记录；同时，底层数据存储模式采用与原始数据相同的关系型表模式。

### 物理模型设计

本节主要涉及本数据仓库的索引策略和数据存储策略。

常用的索引有B-Tree索引和位图索引两种。在本数据仓库中，要索引的属性有“图书分类号（种）”和“图书记录号”两个。由于这两个属性的取值十分多样化，故取值区间较为单一的位图索引不适用于此。同时，这两个属性的值在全数据库中，每一类书都有独特的取值，故本数据仓库采用B-Tree索引。

在数据存储策略上，本数据仓库所涉及的数据量较大。同时，原始数据中有许多字段在数据挖掘过程（通过PF-Growth计算频繁项）中是不需要使用的。故使用分割表策略。即从原始数据中抽取出“计算输入”表，其中存放ETL过程所需要的“读者记录号”、“图书记录号（种）”以及“图书分类号”三项。其余的数据存放在“原始数据”表中，以加快ETL过程。

## 原始数据的 ETL 与数据仓库的实现

本节主要讨论原始数据的ETL过程（即原始数据的提取、转换、清洗和加载四个过程）以及数据仓库的具体实现方式。

### 数据挖掘算法的选择

由于本数据仓库的主要输出结果为输入数据中的频繁项信息，而常用的频繁项挖掘算法有Apriori算法和FP-Growth算法。但我们所需要计算的数据量过大，与FP-Growth算法相比，Apriori算法又需要对内存或者数据库进行多次扫描才能得出结果，算法时空复杂度较大，故不予采用。虽然FP-Growth算法在实现上较为麻烦，但考虑到数据量的问题，这种编写时候的付出还是值得的。

### 原始数据的 ETL

2.2节和2.4节已经讨论了原始数据的属性集以及数据的抽取。那么，要实现原始数据到计算输入这一ETL过程，就需要把原始数据“重整”为FP-Growth算法所能识别的数据输入。

* + 1. **数据的提取与转换**

在数据的提取上，由于本数据仓库的原始数据存放在数据库中，我们可以直接从“原始数据”表中提取“读者记录号”、“图书记录号（种）”以及“图书分类号”三列到“计算输入”表中，然后进行ETL过程的其余步骤。同时，由于数据库能直接以查询结果集的形式提供原始数据，那么也不需要对原始数据进行格式转换。

* + 1. **数据的清洗与加载**

由于FP-Growth算法的输入为形如“Item1 Item2 Item3”这样以行为单位的一个结果集，那么我们就需要把数据重整为这样的一种格式。具体的算法为：

**for** (i = 0; i < inputItemCount; i++) {

merged.add(result.get(i).get(1));

**for** (j = i; j < inputItemCount - 1; j++) {

**if** (Objects.equals(result.get(j).get(0), result.get(j + 1).get(0))) {

merged.add(result.get(j + 1).get(1));

} **else** {

**break**;

}

}

i = j;

**if** (merged.size() == 1) {

merged.clear();

**continue**;

}

Collections.sort(merged);

FPInput.add(**new** ArrayList<>(merged));

merged.clear();

}

上面这个算法把原始数据（即result）从上之下扫描。在扫描的过程中，如果遇到“读者记录号”相同的几项，就会把他们合并到同一行中，即算法中的merged；如果合并后的merged大小为1，则认为这一行只有一个结果，便舍弃之（FP-Growth算法要求有意义的输入数据每一行至少有两条记录）。然后把FPInput直接输入到算法中即可开始计算过程。

### 数据挖掘系统的实现

在实现上，我们把这个数据仓库对原始数据的处理大致分成三个阶段：

1. **预处理阶段**

本阶段主要实现对原始数据的ETL过程。即对原始数据进行抽取、清洗以及加载，使之成为符合FP-Growth算法要求的输入。

1. **频繁项计算阶段**

本阶段主要使用FP-Growth算法对输入数据中的频繁项的计算。

1. **重整输出阶段**

本阶段把FP-Growth算法的单项结果尽可能地扩大。如频繁项集A含“1，2，3”共三项，B含“1，2，3，4”共四项。那么，要使得结果集最大化，为用户提供最多样的结果，就需要把A和B两项合并。同时，本阶段还需要把结果集中的“图书记录号（种）”替换为相应的书名，以供显示之用；同时还需要保留原始的“图书记录号（种）”，以供程序查询之用。

## 挖掘结果的展示与未来应用的探讨