基于对象型层次型数据库的 IFC 数据存储研究

曾　强[1]　张其林[1,2,3]　张金辉[2,3]

（1. 同济大学　土木工程学院，上海　200092；2. 上海同磊土木工程技术有限公司，上海　200092；
3. 上海土木工程结构健康监测工程技术研究中心，上海　200092）

【摘　要】IFC 作为 BIM 技术的核心概念，是为了提高建筑工程项目中各方的协作能力所制定的通用数据交换标准。为了解决传统的基于对象关系型数据库 IFC 数据存储的定义复杂和低效的问题，本文基于对象型层次型数据库，提出了一种通过 IFC XML 格式进行数据交换的新型 IFC 数据存储模型，介绍了 IFC 模式映射和数据库设计过程中的多种技术难点和解决方案并进行了实验验证。本研究对有效管理和利用 IFC 数据具有一定参考价值。

【关键词】BIM；IFC；对象型数据库；层次型数据库；XML
【中图分类号】TU17　【文献标识码】A　【文章编号】1674－7461(2021)01－0017－07
【DOI】10.16670/j.cnki.cn11－5823/tu.2021.01.03

引言

为了解决建筑工程项目的各个参与方间信息交换的困难，国际交互操作性联盟 IAI（International Alliance for Interoperability）制定了一种用于描述、交换和共享建筑信息的国际化规范工业基础类 IFC（Industry Foundation Classes），并且 IFC4 的版本已经成为 ISO 国际标准（ISO 16739）[1]。当前 BIM 技术主要通过 IFC 标准进行数据交换，该标准基于面向对象的方法定义建筑模型的数据结构。

通常情况下，可以直接通过符合 IFC 标准的 IFC 文件进行 BIM 数据的存储与管理。然而在涉及多专业和大量复杂实体的 IFC 数据时，由于数据量庞大，应用复杂，且多专业参与进行数据模型的协同操作与处理，则需要基于数据库进行 BIM 数据的存储与管理[2]。一直以来都有学者进行基于数据库的 IFC 工程模型存储、管理和信息交换的研究。这种基于数据库的 IFC 数据存储和信息管理服务被 IFC Wiki 定义为 IFC 模型服务器（IFC model server）[3]。许多研究将复杂的查询和缓慢的性能确定为以前开发的 IFC 模型服务器的主要局限性。随着建筑物信息模型的大小增加，查询和性能问题会变得更加严重。导致查询复杂和性能降低的主要原因之一是，传统的 IFC 模型服务器是基于 E. F. Codd 提出的关系数据库（RDB）概念构建的[4]。由于 RDB 的稳定性和定义良好的查询语言 SQL（称为结构化查询语言），大多数商业数据库系统仍是基于 RDB。但是，RDB 作为 IFC 模型服务器的基础存在一个关键缺点：IFC 的定义是基于对象的，而 RDB 不是。虽然之后也有不少学者提出基于对象关系型数据库的解决方案，例如 Kang 等分析了如何利用对象关系型数据库 Cubrid 存储 IFC 数据[5]，但是仍然都无法摆脱 RDB 作为底层实现的固有缺陷。

为了解决这一难点，本文提出了一个基于对象型层次型数据库 InterSystems IRIS 的 IFC 数据存储的实现方案，可以实现复杂建筑模型的高效存储和

【基金项目】国家重点研发计划课题“乡村住宅装配式快速建造体系与被动式节能集成研究”（编号：2018YFD1100903）
【作者简介】曾强（1995－），男，硕士研究生，主要研究方向：建筑信息模型 BIM；张其林（1962－），男，博士，教授，主要研究方向：空间结构、建筑信息模型 BIM、结构健康监测等；张金辉（1974－），男，硕士，高级工程师，主要研究方向：建筑信息模型 BIM、建筑物联网。

读取,并且在数据交换过程中完全符合 IFC 标准的模式。

1 基于对象型层次型数据库的 IFC 数据存储方案设计

对象型层次型数据库是一类以基于层次模型的 MUMPS(Massachusetts General Hospital Utility Multi-Programming System)技术作为底层存储方案,扩展出支持面向对象数据模型的数据库[6]。常见的层次型数据库有 InterSystems IRIS 和 GT.M,GT.M 由于不支持面向对象的数据模型,不能称作对象型层次型数据库。同时,不同于传统的对象关系型数据库,对象型层次型数据库可以完全支持面向对象数据模型的概念,如类、对象、继承、抽象类等。

InterSystems IRIS 作为一个分布式的多维模型数据库是成熟的对象型层次型数据库的代表,其分布式架构提供了优秀的水平扩展能力。本节介绍了对象型层次型数据库在存储 IFC 模型数据时的优势,以及基于 InterSystems IRIS 的 IFC 模式映射和读写接口的实现过程。

1.1 InterSystems IRIS 数据库技术要点

InterSystems IRIS 是 InterSystems 公司基于 Caché 开发的新一代分布多维模型数据库,它的底层实现是 MUMPS(Massachusetts General Hospital Utility Multi-Programming System)技术,其核心概念是被称做 Global 的多维稀疏数组。MUMPS 技术拥有非常强大的表达能力,Caché 在其基础之上映射出了面向对象的数据库模式,并且利用关系模式映射实现了关系型数据库的结构[7]。同时集成了符合标准的 SQL 实现,还引入了 ODBC、JDBC 等接口以及 SOAP 和 RESTful 等技术,使同一数据可以同时表现为稀疏数组、表、对象、XML 格式文件和 JSON 格式文件等不同的形式,以适应完全不同的应用场景。InterSystems IRIS 沿用了 Caché 的核心存储技术,加入了包括集群、水平扩展、大数据分析和商业智能等等现代数据库特性,以用于更复杂的数据存储和处理的场景[8]。

InterSystems IRIS 通过将 Global 映射为对象和关系数据表的方式来提供面向对象模式和关系模式的数据访问。因此,可以通过三种主要方式使用 InterSystems IRIS 进行数据的存储和管理,包括直接操作底层的 Global、通过面向对象的数据模型进行操作以及通过 SQL 语言进行关系型映射的数据操作。无论使用哪种方式,底层数据均存储于 Global 数组中。由于具有支持层次型、对象型、关系型等多种数据模式的特性,InterSystems IRIS 将其数据核心称为多维数据引擎,它的工作原理如图 1 所示。

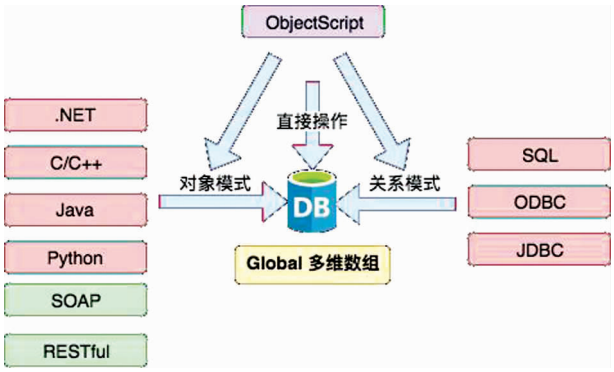


图 1 InterSystems IRIS 多维数据引擎

1.2 对象型层次型数据库的优势

IFC 的核心设计思路是基于面向对象的思想。IFC 的最初定义是包含基于 EXPRESS 语言的实体关系模型和 SPF(STEP Physical File)的数据格式,并在 IFC4 的版本中正式引入了等价的 IFC XSD 实体关系模型和 XML 的数据格式。数百个存在继承关系的实体类型被划分为了多个抽象层次。这种设计天然契合了对象型层次型数据库的数据模型和存储技术。因此,基于对象型层次型数据库的 IFC 数据存储具有以下三大优势:

第一,IFC 格式到对象型层次型数据库的模式映射难度较低。对象型数据的核心概念和面向对象的编程思想一致,IFC 标准中的类、对象、继承和抽象类等等概念可以非常便捷地在支持面向对象数据模型的数据库中实现[9]。这大大降低了 IFC 格式抽象到数据库中的实现难度,同时摒弃了传统关系型数据库中必不可少的关系表的这一抽象层。

第二,对象型层次型数据库支持 IFC 格式的可扩展性。为了在建筑行业极高复杂度的背景下实现最大的兼容,IFC 标准提供了自定义实体和自定义属性来达到较高的可扩展性。传统关系型数据库和对象型层次型数据库都具备一定的可扩展性。但比较而言,对象型层次型数据库的扩展成本较低,实体类可以轻松扩展自定义属性,父类也可以轻松派生出新的子类[10]。而关系型数据库则需要扩展定义表结构或者定义新的表,并且在设计新的

表结构时还要满足不同范式的要求。

第三,对象型层次型数据库在存储 IFC 数据时具有较高的空间利用率。实际的工程项目中,由于建筑设计的独特性,同一实体类的实例化对象往往不多。在传统关系型数据库的设计思路中,为了提高 IFC 数据的存储效率,往往基于较高层次的父类或者抽象类来定义表结构,这会导致同一表下的每一行数据差异较大,这大大提高了数据的冗余度,降低了空间利用率。而对象型层次型数据库则不会有这一问题,由于底层 MUMPS 存储技术在保证对象型模式实现时的性能同时还具有良好的空间利用率[11]。

第四,对象型层次型数据库便于实现基于 XML 格式的 SOAP 服务,具有面向云服务的 IFC 数据操作优势[12]。XML 格式作为对象型层次型数据库的数据交互格式之一,比较容易实现 SOAP 技术来支持 IFC 之类的对象型数据的云服务。

1.3 IFC 的模式映射设计

为了利用 InterSystems IRIS 的面向对象模式对 IFC 数据库进行建模,首先需要设计 IFC 到 InterSystems IRIS 的模式映射。

模式是指一种以格式化的计算机可读符号组成的数据模型,主要用于进行数据格式的定义,包含数据类型、数据结构和约束等等。IFC 模式(IFC schema)是指 IFC 数据模型的格式化定义。buildingSMART 组织参考工业数据标准 EXPRESS 和 STEP,定义了 IFC 的最初的模式定义,后来逐渐发展出以 XML Schem 为基础的模式定义[13]。因此,目前 IFC 主要提供两种形式的模式描述文件,一种是扩展名为 exp 的 EXPRESS 格式数据文件,EXPRESS 是一种结构化的且具有很强的可读性的数据定义语言,完整定义了 IFC 模式的所有内容,这是 IFC 标准的默认模式定义格式。另一种是扩展名为 xsd 的 XML Schema 模式定义文件,其中包含了 IFC 模式的最小化描述,其实体定义只包含明确属性,不包含 INVERSE、WHERE 等等关键字定义的约束规则。

根据两种 IFC 模式的定义格式可以实现 IFC 数据的两种文件序列化存储格式。利用 EXPRESS 格式的模式定义实现的序列化数据格式符合 STEP 标准,扩展名为 ifc。而利用 XML Schema 模式定义实现的序列化数据格式被称作 XML(Extensible Markup Language,可扩展标记语言),扩展名为 ifcxml。虽然 XML Schema 模式的定义仅包含 IFC 模式的最小化描述,但基于该模式所实现的 IFC XML 物理格式不会丢失任何 IFC 的模型信息。两种格式各有优点,STEP 物理格式文件数据组织更紧凑,表达同样信息量的模型时,文件大小仅是 XML 文件格式的三分之一。而 XML 格式文件可读性更强,且能够以 XML 解析程序处理,适合基于网络的数据交换,且更容易实现 SOAP 网络服务。

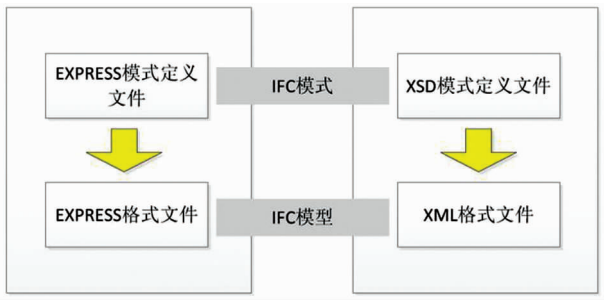


图 2 两种 IFC 模式、模型与文件

1.3.1 IFC EXPRESS 格式的模式映射

根据国际标准 ISO 10303 - 11 的定义,在 EXPRESS 语言中有五种数据类型:简单类型(Simple)、集合类型(Aggregation)、声明类型(Named)、构造类型(Constructed)和通用类型(Generalized)。因此,对于 IFC 物理格式中的各种数据类型,在 InterSystems IRIS 中以替代类型进行存储。

IFC EXPRESS 定义中的简单数据类型和 ENUMERATION 类型在 InterSystems IRIS 中都有对映的原始类型,而其它的复杂数据类型(集合类型、实体类型和选择类型)需要转换成 InterSystems IRIS 的类或者属性。下面对这些复杂类型的定义做详细说明:

(1)实体类型(ENTITY)

IFC EXPRESS 中的实体类型是 IFC 数据的主要组成部分。因此,在数据库中也是主要的存储对象。实体类型的定义非常符合面向对象的思想,可以通过 InterSystems IRIS 中的类进行定义,并且可以实现实体类型包含属性和继承等特点。在 IFC EXPRESS 的实体类型中,主要包括抽象类和非抽象类。抽象类因为具有不可实例化的特点,理应从% RegisteredObject 类派生,而非抽象类需要作为独立对象持久化存储,应从% Persistent 类派生。然而在 InterSystems IRIS 中,从% RegisteredObject 派生的类,由于不具有存储特性,也就不会存储在数据库底层多维数组中,因此也无法参与一些查询。在某

些情况下需要查询某一抽象类下所有派生类的实例,但由于抽象类从% RegisteredObject 派生而来,不在数据库中,在映射为关系模式时也找不到该类对映的表,因此以该类作为表名的 SQL 查询便会失效。解决方法是将所有 IFC 类都从% Persistent 类派生,即可执行支持继承特性的 SQL 查询,并且不会对数据的存储产生任何影响。

关于属性的规定,在 IFC EXPRESS 的定义中,OPTIONAL 作为可选属性的限制,与 InterSystems IRIS 中的 Required 限制含义正好相反。因此,在进行实体类型的属性映射时需要注意限定。以 IfcWindow 为例说明在 InterSystems IRIS 中实体的定义:

```
Class IFC4.IfcWindow Extends IFC4.IfcBuildingElement
{
    Property OverallHeight As %Double;
    Property OverallWidth As %Double;
    Property PredefinedType As %EnumString;
    Property PartitioningType As  %EnumString;
    Property UserDefinedPartitioningType As %String;
}
```

图 3　IfcWindow 的模型映射

(2)集合类型(Aggregation)

在 IFC EXPRESS 中,集合类型通常作为实体类型的属性出现。分为两种情况,简单数据类型的集合和实体类型的集合。对于这两种类型,在 InterSystems IRIS 中可以采取不同的两种定义方式。

对于简单数据类型的集合,可以直接使用 InterSystems IRIS 中的% ArrayOfDataTypes 或者% ListOfDataTypes 来定义。

对于实体类型的集合,为了直接表达主实体和属性实体之间的联系,可以映射为 InterSystems IRIS 中的 Relationship 属性。采用一对多的 Relationship 属性可以描述出实体类型直接的关联。在定义 Relationship 属性时,需要在主实体和属性实体的类定义中同时定义。Relationship 属性的定义包括三部分,关联对象的类、关联对象的基数(Cardinality 关键词)、以及该属性的反向属性(Inverse 关键词)。反向属性的意义与 IFC EXPRESS 中 INVERSE 属性的意义一致。根据 IFC 的定义规则,每个对象集合属性的内部实体类型中都存在一条与该集合属性相应的 INVERSE 属性。因此在定义 Relationship 属性时,可以利用 INVERSE 关键字找到对应的反向属性的名称。

(3)选择类型(SELECT)

在 IFC EXPRESS 中,选择类型也作为实体类型的属性出现。选择类型的底层类型可以是任意 IFC 类型,在数据库中无法进行类型明确的存储,因此需要对每个选择类型做特殊处理,定义为从% SerialObject 派生的类。该类型可以被实例化作为实体的属性,但不会进行独立存储,随使用它的类一起存储。由于 InterSystems IRIS 中所有的类型在存储时实际都以% String 来保存,故属性值定义可以为% String,只是在提取属性值时要在外部用属性的类型做强制类型转换。

1.3.2　IFC XML Schema 格式的模式映射

XML Schema 是 IFC 模式的另一种定义格式。自 IFC2x3 开始,buildingSMART 便开始提供 XML Schema 文件。IFC XML Schema 的数据类型主要包括 XSD 文件本身的 Schema 文件所定义的简单数据类型和 IFC 所扩展的复杂数据类型,它们分别可以映射为 InterSystems IRIS 中的内置简单数据类型和自定义类的复杂数据类型。

(1)XSD Schema 中的简单数据类型

XSD Schema 中的部分简单数据类型可以直接映射为 InterSystems IRIS 中的简单数据类型。但是另一些简单数据类型无法使用其系统命名空间的数据类型,但是由于 InterSystems IRIS 拥有良好的 XML 支持,其在 XSD 命名空间中扩展了所有其它的数据类型。

(2)IFC XML Schema 定义的复杂数据类型

所有由 IFC XML Schema 复杂数据类型映射生成的类均可以派生自% Persistent 和% XML. Adaptor (InterSystems IRIS 支持多类继承),其中% Persistent 用来将类的定义和实例化的数据持久化存储到底层的多维数组中,而% XML. Adaptor 类则实现了 XML 的存储和读取相关的方法,为直接使用 IFCXML 的数据提供了便利。

另外,抽象类和在自定义类中的数组类型属性需要特别注意。

对于抽象元素或类型的处理,同样为了保证在某些情况下查询抽象类下所有派生类的实例,将其映射成为从% Persistent 类派生,而不是从% RegisteredObject 派生。

对于 IFC XML Schema 中定义的数组类型,采用类似于 IFC EXPRESS 中的集合类型的处理方式,将集合从属类和元素类之间的联系映射为 InterSys-

tems IRIS 中一对多的 Relationship 属性。具体实现方式参考上一节关于集合类型的映射规则。虽然在 XML Schema 中无法实现 INVERSE 的约束类型，但是通过使用 Relationship 可以间接实现在 IFC EXPRESS 中的 INVERSE 约束。

(3)利用 XML Schema 的类生成实现 IFC XML Schema 的自动映射

InterSystems IRIS 为了简化 XML 的使用，在% XML 的命名空间中提供了 SchemaReader 类来简化 XML Schema 到类定义的自动生成。具体方法是在创建% XML. Utils. SchemaReader 的实例之后，设定该实例的属性来完成对数组类型和抽象元素的处理，还有关于 XML 生成的属性设置也非常必要，具体在下一节介绍，然后，利用 Process 的方法即可自动生成类定义。

1.4 基于 IFC XML Schema 的 IFC 数据的存储和读取方案

XML 作为一种广泛使用的网络通信数据格式，在 IFC 模型服务器中有着更大的应用优势。同时，InterSystems IRIS 对 XML 有着更加完整的支持，在类的定义、XML 数据的导入和导出方面都有比较成熟的实现方案。虽然在 IFC 的标准中，XML Schema 只是 IFC 模式的最小化实现，但是仍然可以保证 IFC 数据和 STEP 格式的一致性，建筑信息本身不会丢失。因此，基于 IFC XML Schema 实现 InterSystems IRIS 数据库对 IFC 数据的存储和读取是相对更加便利的方案。本文以下的应用系统的构建和工程实践的测试均基于 IFC XML Schema 的格式来进行。

1.4.1 IFC XML 数据的存储

在通过 IFC XML Schema 文件定义好所有的类之后，便可以将 IFC XML 数据导入 InterSystems IRIS 了。InterSystems IRIS 在 XML Tool 中提供了% XML. Reader 类实现 XML 到对象的映射，可以将 XML 文件的元素实例化成所有由% XML. Adaptor 派生的类的对象。% XML. Reader 类还可以间接完成对 XML 文件的 Schema 验证。

存储 IFC XML 数据的具体步骤是，先实例化一个% XML. Reader 的对象，用于读取和验证 IFC XML 文件数据，并解析成为 XML DOM 树，再通过 Correlate 方法建立 XML 元素和类的对应关系，然后可以从 XML DOM 树上找到对应元素，再利用元素信息实例化相关的类。最后调用对象的% Save( ) 方法将对象存储到 Global 多维数组中。下面以从 filename 变量所指的文件导入 IFC 模型数据到 IRIS 中的“IFC4. ifcXML”类(也对应为关系型模型下的一张表)为例，说明 IFC XML 的存储过程。

```
ClassMethod Read(filename As %String)
{
    // 实例化 %XML.Reader 的对象
    set reader=##class(%XML.Reader).%New()
    // 调用 OpenFile 方法读取 XML 数据并解析成 DOM 树
    // 同时根据 xsi:schemaLocation 属性进行 XSD Schema 验证
    set status=reader.OpenFile(filename)
    if $$$ISERR(status) {do $System.Status.DisplayError(status)}
    // 关联 XML 数据中的 ifcXML 根元素和数据库中 IFC4 命名空间下的 ifcXML 类
    do reader.Correlate("ifcXML","IFC4.ifcXML")
    // 递归检索 DOM 树中的 ifcXML 根元素和所有子元素并生成相关类的对象
    while reader.Next(.object,.status)
    {
        if $$$ISERR(status) {do $System.Status.DisplayError(status)}
        // 调用 %Save 方法来存储对象到数据库
        set status=object.%Save()
        if $$$ISERR(status) {do $System.Status.DisplayError(status)}
    }
    if $$$ISERR(status) {do $System.Status.DisplayError(status)}
}
```

图 4 IFC XML 的存储过程

这里的 ifcXML 是 IFCXML 文件中的根元素标签。在保存 XML 数据到 IRIS 时，reader 对象可以遍历所有的 XML 子元素来保存到对应的类。因此，只需要调用与 IFC4. ifcXML 相关的 reader 即可存储完整的 IFCXML 数据到数据库中。但需要特别注意的是，当遇到自定义类作为属性时，需要调用% XML. New 方法进行属性的构造，因为该属性的实例会保存在自己的类(表)中，属性所属类(表)中只保存它的记录 ID。

1.4.2 IFCXML 数据的读取

从 InterSystems IRIS 的对象生成 IFC XML 数据需要注意几个问题。首先是命名空间的处理，InterSystems IRIS 和 XML 的命名空间具有不同的含义，另外，每一个版本的 IFC XML Schema 均定义了不同的命名空间。其次是对 XML 空子段或者空对象的处理，以及 XML 元素的顺序问题。为了保证读取出来的 IFCXML 数据仍然符合 Schema 的定义，并且不丢失任何 IFC 建筑模型信息，需要对这些问题进行特别的处理。InterSystems IRIS 的% XML. Adaptor 为这些问题的处理提供了便利，下面以 IFC4. Entity 为例说明如何保证 XML 数据读取的一致性。

2 IFC 数据存储方案可行性验证

为了验证基于对象型层次型数据库的 IFC 数据存储方案的可行性，本节选取了三种从简单到复杂

```
// 继承自 %XML.Adaptor 来实现 XML 数据的生成功能和一些自定义设置
Class IFC4.Entity Extends (%Persistent, %XML.Adaptor) [ Abstract, ProcedureBlock ]
{
    Parameter ELEMENTQUALIFIED = 1;
    // 指定 XML 的命名空间
    Parameter NAMESPACE = "http://www.buildingsmart-tech.org/ifc/IFC4x1/final";
    // 指定 XML 中的元素名称
    Parameter XMLNAME = "Entity";
    // 指定 XML 中的元素是否需要保持属性的顺序
    Parameter XMLSEQUENCE = 0;
    // 指定是否忽略空字段
    Parameter XMLIGNORENULL = 1;
    Index ifcXMLIndex On ifcXML;
    Property href As %xsd.anyURI(XMLNAME = "href", XMLPROJECTION = "ATTRIBUTE");
    Property ref As %String(MAXLEN = "", XMLNAME = "ref", XMLPROJECTION = "ATTRIBUTE");
    Property id As %String(MAXLEN = "", XMLNAME = "id", XMLPROJECTION = "ATTRIBUTE");
    Property path As %String(MAXLEN = "", XMLNAME = "path", XMLPROJECTION = "ATTRIBUTE");
    Property pos As %String(MAXLEN = "", MINLEN = 1, XMLNAME = "pos", XMLPROJECTION = "ATTRIBUTE");
    Relationship ifcXML As IFC4.ifcXML(XMLPROJECTION = "NONE") [ Cardinality = one, Inverse = Entity ];
}
```

图 5　IFC4. Entity 类的定义

的 IFC 模型,以 IFC XML 的格式存储到 InterSystems IRIS 数据库中,并读取生成新的 IFC XML 模型数据,进行对比验证。

(1)实验数据和环境

三种 IFC 模型分别对应了实体数目百级、万级和百万级。其中最简单的模型来自于 buildingSMART 提供的 IFC 样例文件,较复杂的两个模型来自 Revit 2019 的建筑项目样例模型 rac_basic_sample_project 和 rac_advanced_sample_project。

验证实验的硬件环境为英特尔 i5 – 8400 六核处理器、16 GB DDR4 内存和 256 G 固态硬盘,InterSystems IRIS 数据库版本为 2020.1.0.215.0 社区版,单节点运行在 Ubuntu 18.04 LTS 64 位操作系统上。

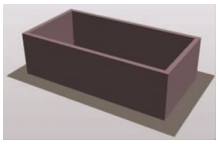
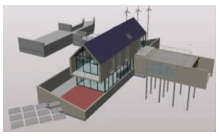
(2)验证结果

按照以上实验数据和环境可以得到如表 1 所示的实验验证结果。

表 1 中第二至四列的结果分别表示原始模型和数据库读取的模型。从结果中可以看出,IFC XML 文件大小以及 ZIP 压缩文件大小在数据库存储前后有一定的差异,但实体数目没有发生变化。为了进一步验证前后数据的一致性,利用开源的 xmldiff 工具对 XML 数据进行节点比对,仅仅只有部分与 IFC Schema 定义相关的属性信息发生变化,同时部分为空的数据在数据库导出时去除,减少了无用信息。因此,从实验结果来看,三种级别的 IFC 模型数据均未丢失,该数据库方案提供了 IFC 数据的一致性保证。

从存储和读取的时间结果来看,最复杂的百万级别模型存储时间在一分钟级别,但读取时间只有十秒左右,原因在于存储 IFC XML 数据时会进行 Schema 的验证,消耗更多的时间。由于没有进行索引和数据库参数配置的优化,实验结果所体现的存取效率已经比较令人满意。

表 1　三种 IFC 模型的实验验证结果对比

实验模型	文件大小(KB)	ZIP 压缩文件大小(KB)	实体数目	存储时间(ms)	读取时间(ms)
 4walls1floorSite. ifcxml	177/181	19/19	579/579	2532	878
 rac_basic_sample_project. ifcxml	12735/12709	2559/2549	21068/21068	19935	3603
 rac_advanced_sample_project. ifcxml	398954/398116	14238/14334	1085678/1085678	57354	12875

3　结论和展望

本文总结了 BIM 的核心技术之一 IFC 标准在数据库存储相关领域的研究进展,针对传统的基于对象关系型数据库 IFC 数据存储的缺陷,提出了一种基于对象型层次型数据库的新型 IFC 数据存储模型。同时,对比两种 IFC 模式映射思路,设计了基于 IFC XML Schema 的数据库定义方案。最后,利用三种不同实体数目级别的典型 IFC 模型进行了实验,

验证了该数据库设计方案的数据一致性和存取效率。

不过展望 BIM 技术的发展趋势,如何在兼容 IFC 标准的基础上,实现版本控制和高级查询等等更加复杂的应用场景仍然值得进一步研究。充分利用 InterSystems IRIS 或者其它新型数据库的现代特性,如商业智能、大数据处理、集群等等,结合信息技术领域的最新成果,才能将 BIM 技术的应用推向新的高度和广度。

参考文献

[1] 李云贵. 国内外 BIM 标准与技术政策[J]. 中国建设信息,2012(20):14-17.

[2] 陆宁,马智亮. 利用面向对象数据库与关系数据库管理 IFC 数据的比较[J]. 清华大学学报(自然科学版),2012,52(6):836-842.

[3] 张迪,刘华,李航. 基于对象关系型数据库的 IFC 存储模型[J]. 城市建筑,2018(2):38-40.

[4] Lee G,Jeong J, Won J, et al. Query Performance of the IFC Model Server Using an Object-Relational Database Approach and a Traditional Relational Database Approach [J]. Journal of Computing in Civil Engineering, 2012,28 (2):210-222.

[5] Kang H S, Lee G. Development of an Object-Relational IFC Server[C]//Iccem/iccpm. 2009.

[6] Kirsten W,Ihringer M, Schulte P, et al. Object-Oriented Application Development Using the Caché Postrelational Database[M]. Springer-Verlag, 2001.

[7] 张金辉,刘仲英. 后关系型数据库 Caché 的应用研究[J]. 计算机系统应用,2003(10):56-58.

[8] Lu J,Holubová I. Multi-model databases: a new journey to handle the variety of data[J]. ACM Computing Surveys (CSUR), 2019,52(3):1-38.

[9] Faraj I,Alshawi M, Aouad G, et al. An industry foundation classes Web-based collaborative construction computer environment: WISPER[J]. Automation in Construction, 2001,10(1):79-99.

[10] 陆宁,马智亮. 利用面向对象数据库与关系数据库管理 IFC 数据的比较[J]. 清华大学学报(自然科学版),2012,52(06):836-842.

[11] Webster S. Medical Query Language: Improved Access to MUMPS Databases[J]. 1987.

[12] SEELY S,FOREWORD BY - SHARKEY K. SOAP: cross platform Web service development using XML[M]. Prentice Hall PTR, 2001.

[13] 何关培. 实现 BIM 价值的三大支柱 - IFC/IDM/IFD [J]. 土木建筑工程信息技术,2011,3(1):108-116.

Research on IFC Data Storage Based on Object-oriented Hierarchical Database

Zeng Qiang[1], Zhang Qilin[1,2,3], Zhang Jinhui[2,3]

(1. *College of Civil Engineering*, *Tongji University*, *Shanghai* 200092, *China*;
2. *Shanghai Tonglei Civil Engineering Technology Co., Ltd. Shanghai* 200092, *China*;
3. *Shanghai Engineering Research Center of Structural Health Monitoring for Civil Engineering*, *Shanghai* 200092, *China*)

Abstract: As the core concept of BIM technology, IFC is a general data exchange standard developed to improve the collaborative capabilities of all parties in construction projects. In order to solve the complex definition and inefficiency of the traditional IFC data storage based on object-relational database, this paper proposes a new IFC data storage model for data exchange through IFC XML format based on the object hierarchical database and introduces various technical difficulties and solutions in the process of pattern mapping. The database design were verified by experiments. This research has certain reference value for the effective management and utilization of IFC data.

Key Words: BIM; IFC; Object Oriented Database; Hierarchical Database; XML