基于 CWM 的关系数据库建模方法①

Relational Database Modeling Method based on CWM

朱 婵　许龙飞　周锦煌　（广州 暨南大学信息科技学院计算机系 510632）

摘要：对关系数据库进行建模的不同工具和产品都有其自己的元数据定义和格式，这使得它们之间的互操作非常困难。论文介绍了基于公共仓库元模型（CWM）的关系数据库建模方法，使用这种方法，数据库设计者能够通过标准的 CWM 格式来设计和创建数据库。

关键词：公共仓库元模型（CWM）　元数据　关系模式

1　引言

从 20 世纪 70 年代初关系数据库产生至今，已有多种数据库设计工具都能够帮助数据库设计者设计数据库并创建数据库模式，但大多数商业产品都有其各自的元数据描述来表达数据库模式，元数据是描述数据的数据或是与数据有关的信息。当设计者需要将由某种工具设计和创建的数据库模式转换到另一个工作环境中时极为不方便，因为在对源环境和目标环境的元数据进行映射时，需要对交换环境双方的每一部件都有透彻的了解。通常，具有不同元数据的工具是通过建立复杂的元数据桥（meta data bridge）来进行集成的，元数据桥是一种将某产品的元数据转换成另一个产品所需元数据格式的一种软件，构建元数据桥是一个艰巨且耗资巨大的过程，且其可重用性低。

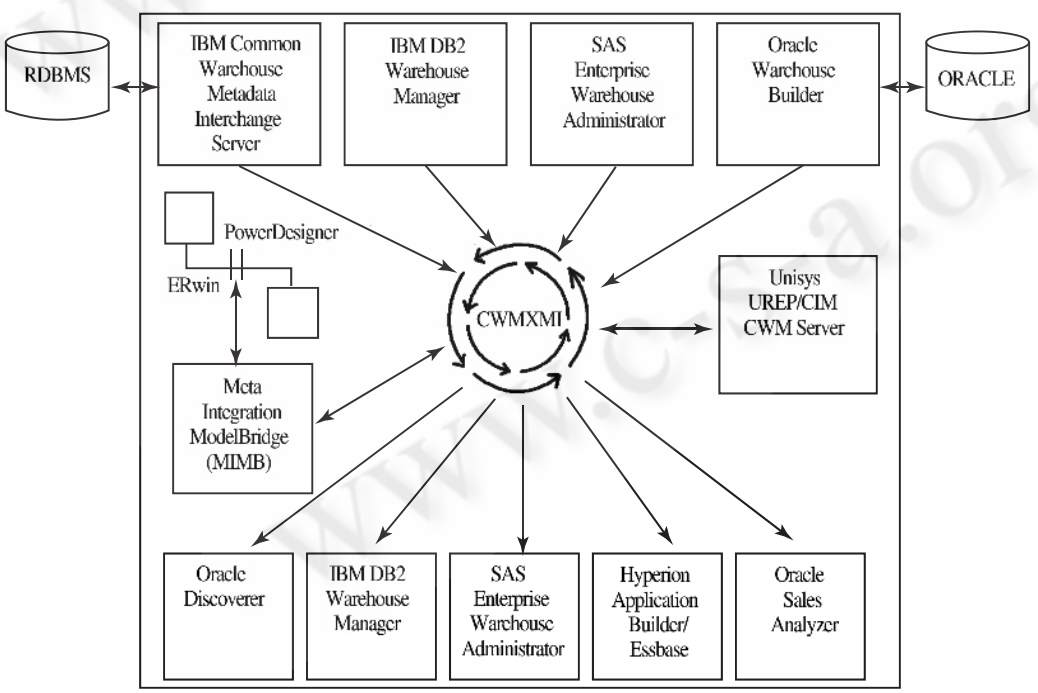

图 1　多种工具通过 CWM 进行互操作

CWM（Common Warehouse Metamodel，公共仓库元模型）是一个最近被 OMG 采纳为在数据仓库和业务分析环境中进行元数据交换的标准。它为不同厂商的产品之间的元数据交换提供了一种解决方案。如图 1 所示，通过 CWM，元数据可以在

① 基金项目：国家自然科学基金资助项目（编号：60374070），广东省自然科学基金资助项目（编号：031903）

数据存储、仓库建模工具、OLAP 产品和终端用户工具之间交换。目前,已有多个公司的产品采用了 CWM 进行元数据交换,例如 Oracle 9i、DB2 等。

本文第二部分将对 CWM 采用的相关技术进行简要的介绍,第三部分介绍了基于 CWM 的关系数据库建模方法,第四部分进行了小结。

2 CWM 及其相关技术

CWM 提供了一种经过长期研究的通用语言来描述元数据,同时 CWM 还支持用模型驱动的方法对元数据进行交换。在 OMG 的元建模体系结构中,UML 作为定义元数据模型的标准语言;XMI 作为交换元数据和元模型的标准机制。下面简要介绍 CWM 所采用的 OMG 元建模体系结构的几种基本技术。

2.1 CWM 和 UML

UML(Unified Modeling Language,统一建模语言)是一种对离散结构建模的图形语言,CWM 元模型在极大程度上是对基于 UML 的对象模型的扩展,CWM 的主要元素在语法和语义上都直接或间接继承了 UML 元模型并对其进行了扩展,CWM 还大量引用了 UML 核心元模型中的元素以避免重复的工作。例如,在 CWM 关系包的关系元模型中定义的元类(metaclass)"Table",该类表示任意关系数据库的表,这个元类就是源自对象模型的元类"Class"的。同样的,CWM 中的关系元类"Column"源自于对象模型的元类"Attribute"。此外,CWM 元模型的图形表示中使用了 UML 的符号,并采用 UML 中的 OCL(Object Constraint Language,对象约束语言)作为元模型约束语言。

2.2 CWM 和 MOF

MOF(Meta Object Facility,元对象工具)是为元模型规范定义公共抽象语言的一种 OMG 标准,CWM 元模型的设计采用了这种标准,这样 CWM 便能够使用其他依赖于 MOF 的 OMG 标准。MOF 本质上是元模型的模型,它定义了为离散系统建模要用到的元模型中的基本元素、语法和结构,使得不同的元模型可以进行互操作。

2.3 CWM 和 XMI

XMI(XML Metadata Interchange,XML 元数据交换)是 OMG 关于元数据交换的标准,它是在 W3C 的可扩展标记语言(XML)的基础上产生的。XMI 通过一种自描述和异步的方式为符合 MOF 元模型的元数据交换提供一种标准语言,它允许元数据按照 XML 的标准格式进行交换。CWM 使用 XMI 作为其交换机制,利用了 XMI 在仓库元数据交换和 CWM 元模型本身交换上所具有的功能和灵活性。

3 基于 CWM 的关系数据库建模

3.1 CWM 包

CWM 的体系结构中有 21 个包,每个包都含有与数据仓库和商业智能领域特定部分相关的类、关联和约束,这些包根据其在整个体系结构中的不同作用被分为五个层次。在通过 CWM 元数据进行关系建模的过程中,我们要用到的 CWM 包主要有以下几类:

(1) 关系型包(Relational)。关系型包位于资源层,关系数据库的模式可以使用关系型包来进行描述。关系型包的元模型支持 SQL99 标准及其面向对象扩展的关系数据库描述。创建关系模式所需要的主要关系对象有以下几种,我们用 CWM 元数据格式对其进行记录时,每个对象都会用一个 XMI ID 作为标识:

Catalog(目录)表示元数据的名称;

Schema(模式)表示模式名;

Table(表)表示一个模式中的表名;

Column(列)表示一个数据库表中的列;

PrimaryKey(主键)表示一个数据库表中的主键,它引用一个 Column 作为其 feature 值;

ForeignKey(外键)表示一个数据库表中的外键,它指向另一个表中的 Primary Key;

SQL data type(SQL 数据类型)包括了可用的数据类型,其 XMI ID 被 Column 引用。

上述各对象之间的关系如图 2 所示,其中箭头表示对象间的引用关系:

(2) 键和索引包(Keys and Indexes)。键和索引包位于基础层,键是一个或多个值的集合,用来唯一确定数据库中的某项记录;索引是一种确定数据对象可替代顺序的性能优化技术,数据库管理系统能够通过使用索引快速返回查询结果。键和索引包用 Index 类、UniqueKey 类和 KeyRelationship 类分别描述索引、主键和外键的数据结构。

(3) 数据类型包(Data Types)。数据类型包位于基础层,在数据仓库的元交换领域,基于 CWM 的工具

必须能够识别不兼容的类型系统。在 CWM 中描述数

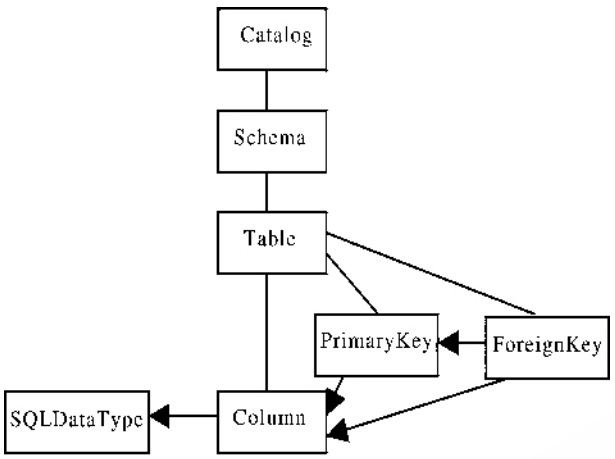

图 2　CWM/XMI 关系对象

据资源的类型系统后，数据资源便能与其他遵从 CWM 的工具进行交换。数据类型包提供了定义基本数据类型和构造数据类型所需的基础结构。

的概念，并支持类型系统间数据类型的映射。

（5）转换包（Transformation）。转换包位于分析层，它支持数据抽取、转换与载入和数据踪迹跟踪的功能。转换包中的 Transformation 类记录单个转换中的数据源和目标，相关的多个转换会被组合为较大的集合；TransformationMap 类记录抽象层之间的转移过程。

3.2 关系数据库建模

将数据库模式以统一的 CWM 元数据格式来表达，使其可以方便地在不同环境中转换。基于 CWM 的关系数据库建模步骤如下：

（1）用 CWM 关系元模型表示数据库模式。该过程包括两步：

①从概念模型抽取元数据（capture metadata）产生 CWM ER 模型。

②通过转换映射（TransformationMap）将 CWM ER 模型转换为 CWM 关系元模型。此处以设计 StudentDB 数据库中的 Student 表作为示例，如图 3 所示。

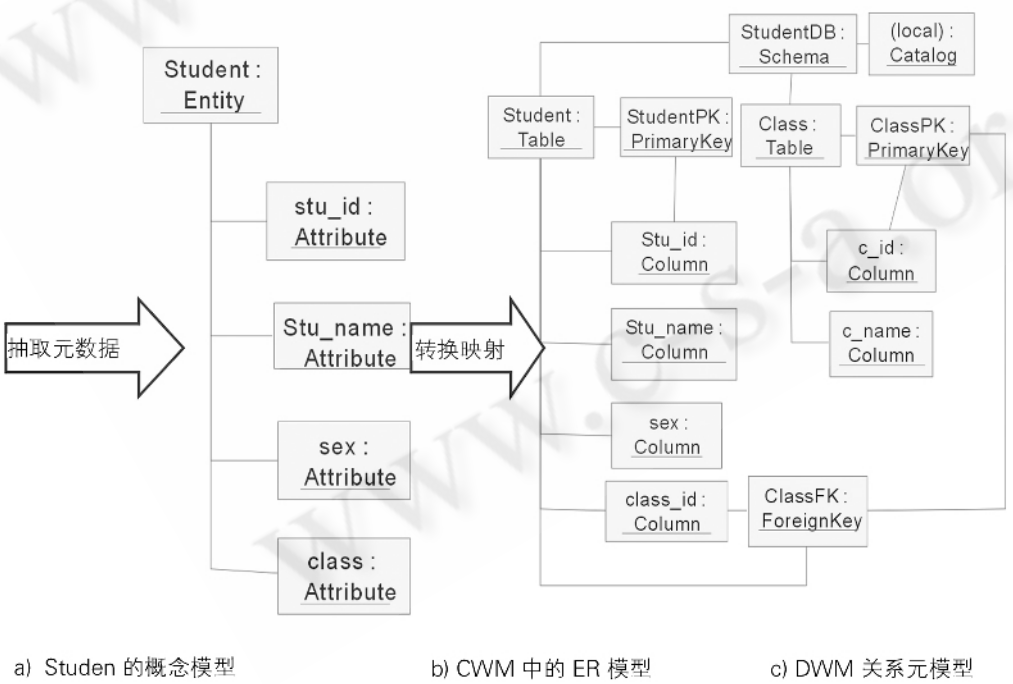

图 3　概念模型转换为 CWM 关系模型

在图 3 b）到 c）的转换过程中，主要使用了转换包中的 ClassifierMap（类元映射）类和 FeatureMap（特征映射）类，并引入 Class 表中的部分元素来说明外键的转换映射。从 ER 模型到关系元模型的转换映射中，Student 实体（Entity）通过 ClassifierMap 转换为 Student 关系表（Table）；实体的各属性（Attribute）通过 FeatureMap 转换为表中的列（Column）。Student 实体的 class（班级名称）属性的值在关系数据库中是通过与 Class 表的 c_id 相关联而得到的，因此引用 Class 表的 c_id 作为 Student 表的外键 class_id。

（4）类型映射包（Type Mapping）。类型映射包也位于基础层，数据类型包提供了定义数据类型的功能，而类型映射包则定义了作为数据类型集合的类型系统

（2）将 CWM 关系元模型转换为 CWM 规格说明（Specification），即将 UML 模型用 XMI Document 描述。StudentDB 数据库的部分 CWM 元数据文件记录如图 4 所示，该文件由 CWM 关系型包及其依赖包生成，当此文件的格式和内容完备且形式化时，可用任何支持 CWM 的工具来处理。

```
……
<XMI xmlns:CWM="org.omg.CWM1.1" xmlns:CWMRDB="org.omg.CWM1.1/Relational">
<XMI.header>
<XMI.metamodel XMI.name="CWM" XMI.version="1.0" />
</XMI.header>
<XMI.content>
<CWMRDB:Catalog xmi.id="_1" name="student.cwm" visibility="public">
<CWM:Namespace.ownedElement>
<CWMRDB:Schema xmi.id="_1.1" name="Logical View" visibility="public" namespace="_1">
<CWM:Namespace.ownedElement>
<CWMRDB:Table xmi.id="_1.1.1" name="student" IsSystem="false" visibility="public" namespace="_1.1">
<CWM:Classifier.feature>
<CWMRDB:Column xmi.id="_1.1.1.1" name="stu_id" isNullable="columnNoNulls" visibility="public" type="_11" length="10" owner="_1.1.1" />
<CWMRDB:Column xmi.id="_1.1.1.2" name="stu_name" IsNullable="columnNullable" visibility="public" type="_12" length="30" owner="_1.1.1" />
<CWMRDB:Column xmi.id="_1.1.1.3" name="sex" IsNullable="columnNullable" visibility="public" type="_13" length="1" owner="_1.1.1" />
<CWMRDB:Column xmi.id="_1.1.1.4" name="class_id" isNullable="columnNoNulls" visibility="public" type="_14" length="6" owner="_1.1.1" />
</CWM:Classifier.feature>
<CWM:Namespace.ownedElement>
<CWMRDB:PrimaryKey xmi.id="_1.1.1.5" name="pk_student_student_id" visibility="public" namespace="_3" feature="_1.1.1.1" />
<CWMRDB:ForeignKey xmi.id="_1.1.2.4" name="fk_student_class_id" visibility="public" namespace="_3" feature="_1.1.1.4" uniqueKey="_1.1.2.1" />
</CWM:Namespace.ownedElement>
</CWMRDB:Table>……
<CWMRDB:SQLSimpleType xmi.id="_11" name="SQL_char_10" visibility="public" characterMaximumLength="10" characterOctetLength="1" typeNumber="1" />
<CWMRDB:SQLSimpleType xmi.id="_12" name="SQL_char_30" visibility="public" characterMaximumLength="30" characterOctetLength="1" typeNumber="1" />
<CWMRDB:SQLSimpleType xmi.id="_13" name="SQL_char_1" visibility="public" characterMaximumLength="1" characterOctetLength="1" typeNumber="1" />
<CWMRDB:SQLSimpleType xmi.id="_14" name="SQL_char_6" visibility="public" characterMaximumLength="6" characterOctetLength="1" typeNumber="1" />……
```

图 4　StudentDB 的 CWM 元数据文件

（3）将 CWM 元数据文件导入支持 CWM 标准的 RDBMS（关系数据库管理系统）中，创建相应的关系数据库模式。得到的 Student 表如表 1 所示。

表 1　Student 表

Table	Column	Datatype	Length	PrimaryKey	ForeignKey
Student	stu_id	char	10	yes	
	stu_name	char	30		
	sex	char	1		
	class_id	char	6		yes

4　小结和展望

本文主要讨论了 CWM 在关系数据库建模方面的作用。CWM 的优势在于它完全独立于任何具体实现的元模型，任何支持 CWM 的工具都能够相互理解其元数据实例。CWM 已成为 OMG 的示范性技术，随着 CWM 不断吸收新的技术，使用 CWM 的元数据交换会变得更为简单和有效。

参考文献

1 OMG.”Common Warehouse Metamodel（CWM）Specification”[EB/OL],version 1.1. March 2003.

2 D. T. Chang.”CWM Enablement Showcase”[EB/OL],March 2001.
http://www.cwmforum.org/paperpresent.htm

3 John Poole, Dan Chang, Douglas M. Tolbert, David Mellor,彭蓉等译，公共仓库元模型：数据仓库 集成标准导论[M],机械工业出版社,2004. 84－110。