(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 101751469 B
(45)授权公告日 2014.07.02

(21)申请号 200910258389.9
(22)申请日 2004.07.29
(30)优先权数据
10/837,929 2004.05.03 US
10/775,282 2004.02.10 US
(62)分案原申请数据
200480001723.8 2004.07.29
(73)专利权人 微软公司
地址 美国华盛顿州
(72)发明人 B·赛兹金 D·Y·阿尔图多夫 J·A·布莱克雷 R·凡卡特施 W·俞
(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 顾嘉运
(51)Int.Cl.
G06F 17/30(2006.01)

(56)对比文件
US 5826077 A,1998.10.20, 全文.
US 5696961 A,1997.12.09, 全文.
US 6564205 B2,2003.05.13, 全文.
WO 9715004 A1,1997.04.24, 全文.

审查员 于白

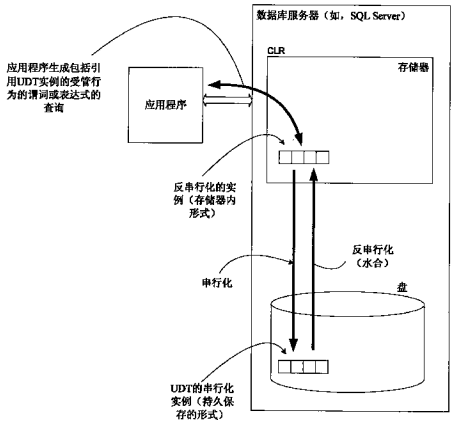
权利要求书6页 说明书14页 附图6页

(54)发明名称

用以支持用户定义类型的继承的系统和方法

(57)摘要

本发明的各种实施例针对具有用于至少一种用户定义类型的可扩展类型系统的数据库，该用户定义类型包含用于描述其结构的信息及描述其与另一用户定义类型（子类型或超类型/基类型）间的继承关系的信息。对于某些实施例，该用户定义类型是用除顺序查询语言(SQL)语句以外的语句，例如公共语言运行时间(CLR)语句、C、C++及C#(“C升半音号”)语句、和/或visual basic语句来定义的。在任一情况下，若干实施例还针对一种系统，其中通过向所述数据库明确地注册该用户定义类型，使数据库知道两个用户定义类型间的继承关系。



CN 101751469 B

1. 一种用于通过在数据库系统中创建用户定义类型来允许用户扩展数据存储的标量类型系统的计算机实现方法，包括：

接收代码，所述代码实现所述用户定义类型的结构的类以及可在所述用户定义类型的实例上调用的方法；

对所述类实施一合同以确保所述用户定义类型担当所述数据库系统的任意标量类型的角色，所述合同包括：

第一要求：所述类指定用于在数据库存储中持久保存用户定义类型的实例的多个不同格式之一，所述多个不同格式包括：用户定义类型的实例根据数据库系统的本机格式来自动串行化的第一格式以及用户定义类型的实例按由所述类定义的方式串行化的第二格式；

第二要求：所述类能够为该用户定义类型返回空值；以及

第三要求：所述类提供一种用于将用户定义类型转换为另一类型的方法；

存储关于所述用户定义类型的元数据，并向所述数据库系统注册所述用户定义类型，以供所述数据库系统接下来创建该用户定义类型的实例时使用；以及

仅当所述类满足所述合同的要求时，创建所述用户定义类型的实例；

接收所述数据库系统的查询语言的表达式，其中所述表达式的求值需要对该用户定义类型的实例的方法进行调用；

将该表达式翻译成一程序代码指令序列以调用所述用户定义类型的实例上所需的方法；

根据程序代码的执行来调用所述方法；以及

将所述方法调用的结果作为查询语言表达式的求值返回。

2. 如权利要求 1 所述的计算机实现方法，其特征在于，所述用于在数据库存储中持久保存用户定义类型的实例的多个不同格式进一步包括：用户定义类型的实例按 MICROSOFT.NET 框架提供的方法进行串行化的第三格式。

3. 如权利要求 1 所述的计算机实现方法，其特征在于，进一步包括：将所述用户定义类型例示为表中的列值、变量、例程的参数、或例程的返回值之一。

4. 如权利要求 1 所述的计算机实现方法，其特征在于，定义所述用户定义类型的类包括一增变器方法，当被调用时，使得所述用户定义类型的值被改变，并且其中，所述方法进一步包括在所述用户定义类型的实例上调用该增变器方法以改变所述实例的值。

5. 如权利要求 1 所述的计算机实现方法，其特征在于，定义所述结构和用户定义类型的方法的所述类还包括：属性，该属性指定该用户定义类型的实例串行化的二进制表示将按二进制排序。

6. 如权利要求 5 所述的计算机实现方法，其特征在于，进一步包括：

串行化所述用户定义类型的实例，以便所述实例的二进制表示按二进制排序；

接收要求将所述用户定义类型的第一实例与所述用户定义类型的第二实例进行比较的所述数据库系统的查询语言的表达式；

将所述用户定义类型的第一实例和第二实例的串行化的二进制表示进行比较以对该表达式求值，而无须反串行化任一实例。

7. 如权利要求 5 所述的计算机实现方法，其特征在于，进一步包括：

在数据库存储中创建一表，在所述表中，所述表的列的类型被定义为所述用户定义类型；以及

在所述列上创建索引。

8. 如权利要求 5 所述的计算机实现方法，其特征在于，进一步包括：

串行化所述用户定义类型的实例，以便所述实例的二进制表示按二进制排序；

接收要求调用所述用户定义类型的一方法的求值的所述数据库系统的查询语言的表达式；

生成所述表达式上的计算的列；以及

创建所述计算的列上的索引。

9. 一种用于通过在数据库系统中创建用户定义类型来允许用户扩展数据存储的标量类型系统的计算机实现的系统，包括：

用于接收代码的装置，所述代码实现定义所述用户定义类型的结构的类以及可在所述用户定义类型的实例上调用的方法，

用于对所述类实施一合同以确保所述用户定义类型担当所述数据库系统的任意标量类型的角色的装置，所述合同包括：

第一要求：所述类指定用于在数据库存储中持久保存用户定义类型的实例的多个不同格式之一，所述多个不同格式包括：用户定义类型的实例根据数据库系统的本机格式来自动串行化的第一格式以及用户定义类型的实例按由所述类定义的方式串行化的第二格式；

第二要求：所述类能够为该用户定义类型返回空值；以及

第三要求：所述类提供一种用于将用户定义类型转换为另一类型的方法；

用于存储关于所述用户定义类型的元数据，并向所述数据库系统注册所述用户定义类型，以供所述数据库系统接下来创建该用户定义类型的实例时使用的装置；以及

用于仅当所述类满足所述合同的要求时，创建所述用户定义类型的实例的装置；

用于接收所述数据库系统的查询语言的表达式的装置，其中所述表达式的求值需要对该用户定义类型的实例的方法进行调用，

用于将该表达式翻译成一程序代码指令序列以调用所述用户定义类型的实例上所需的方法的装置；

用于根据程序代码的执行来调用所述方法的装置；以及

用于将所述方法调用的结果作为查询语言表达式的求值返回的装置。

10. 如权利要求 9 所述的计算机实现的系统，其特征在于，所述用于在数据库存储中持久保存用户定义类型的实例的多个不同格式进一步包括：用户定义类型的实例按 MICROSOFT.NET 框架提供的方法进行串行化的第三格式。

11. 如权利要求 9 所述的计算机实现的系统，其特征在于，进一步包括：用于将所述用户定义类型例示为表中的列值、变量、例程的参数或例程的返回值之一的装置。

12. 如权利要求 9 所述的计算机实现的系统，其特征在于，定义所述用户定义类型的类包括一增变器方法，当被调用时，使得所述用户定义类型的值被改变，并且其中，所述数据库服务器在所述用户定义类型的实例上调用该增变器方法以改变所述实例的值。

13. 如权利要求 9 所述的计算机实现的系统，其特征在于，定义所述结构和用户定义类

型的方法的所述类还包括：属性，该属性指定该用户定义类型的实例串行化的二进制表示将按二进制排序。

14. 如权利要求 13 所述的计算机实现的系统，其特征在于，进一步包括：

用于串行化所述用户定义类型的实例的装置，以便所述实例的二进制表示按二进制排序，

用于接收要求将所述用户定义类型的第一实例与所述用户定义类型的第二实例进行比较的所述数据库系统的查询语言的表达式的装置，

用于将所述用户定义类型的第一实例和第二实例的串行化的二进制表示进行比较以对该表达式求值，而无须反串行化任一实例的装置。

15. 如权利要求 13 所述的计算机实现的系统，其特征在于，进一步包括：

用于在数据库存储中创建一表的装置，在所述表中，所述表的列的类型被定义为所述用户定义类型，以及

用于在所述列上创建索引的装置。

16. 如权利要求 13 所述的计算机实现的系统，其特征在于，进一步包括：

用于串行化所述用户定义类型的实例的装置，以便所述实例的二进制表示按二进制排序，

用于接收要求调用所述用户定义类型的一方法的求值的所述数据库系统的查询语言的表达式的装置，

用于生成所述表达式上的计算的列的装置；以及

用于创建所述计算的列上的索引的装置。

17. 一种通过在数据库系统中创建用户定义类型来允许用户扩展数据存储的标量类型系统的计算机实现的系统，所述计算机实现的系统包括：

用于在存储器中存储由受管代码创建的类的装置，所述类包括描述所述至少一个用户定义类型的结构的信息，其中所述类被编译到组件中；

用于在存储器中存储用于描述所述至少一个用户定义类型和所述至少一个用户定义类型中的第二用户定义类型之间的继承关系的信息的装置，其中所述第二用户定义类型相对于所述至少一个用户定义类型的继承关系为所述至少一个用户定义类型的至少一个子类型或超类型；

用于通过执行所述组件向存储在存储器中的所述数据库的明确注册来扩展标量类型系统，以便所述用户定义类型担当结构查询语言（SQL）类型系统中的标量类型的角色的装置，所述明确注册指示了所述至少一种用户定义类型与所述第二用户定义类型之间的所述继承关系；

用于接收所述数据库系统的查询语言的表达式的装置，其中所述表达式的求值需要对该用户定义类型的实例的方法进行调用，

用于将该表达式翻译成一程序代码指令序列以调用所述用户定义类型的实例上所需的方法的装置；

用于根据程序代码的执行来调用所述方法的装置；以及

用于将所述方法调用的结果作为查询语言表达式的求值返回的装置。

18. 如权利要求 17 所述的计算机实现的系统，其特征在于，进一步包括：用于创建作为

所述至少一个用户定义类型的子类型的所述第二用户定义类型的装置。

19. 如权利要求 17 所述的计算机实现的系统，其特征在于，进一步包括：用于创建作为所述第二用户定义类型的子类型的所述至少一个用户定义类型的装置。

20. 如权利要求 17 所述的计算机实现的系统，其特征在于，所述至少一种用户定义类型由公共语言运行库（CLR）语句来定义。

21. 如权利要求 17 所述的计算机实现的系统，其特征在于，所述受管代码以下编程语言组中的至少一种编程语言来定义或得到：visual basic、visual C++、C、C++ 及 C#。

22. 如权利要求 17 所述的计算机实现的系统，其特征在于，还包括：

用于将类型编译到所述组件的装置；

用于向所述数据库注册所述组件的装置；

用于向所述数据库注册所述组件的所述类型的装置；以及

用于使用所述类型的装置。

23. 一种用于通过在数据库系统中创建用户定义类型来允许用户扩展数据存储的标量类型系统的计算机实现方法，所述计算机实现方法包括：

在存储器中存储由受管代码创建的类，所述类包括描述所述至少一个用户定义类型的结构的信息，其中，所述类被编译到组件中；

在存储器中存储用于描述所述至少一个用户定义类型和所述至少一个用户定义类型中的第二用户定义类型之间的继承关系的信息，其中所述第二用户定义类型相对于所述至少一个用户定义类型的继承关系为所述至少一个用户定义类型的至少一个子类型或超类型；

通过执行所述组件向存储在存储器中的数据库的明确注册来扩展标量类型系统，以便所述用户定义类型担当结构查询语言（SQL）类型系统中的标量类型的角色，所述明确注册使得所述数据库知道所述至少一种用户定义类型与所述第二用户定义类型之间的所述继承关系；

接收所述数据库系统的查询语言的表达式，其中所述表达式的求值需要对该用户定义类型的实例的方法进行调用；

将该表达式翻译成一程序代码指令序列以调用所述用户定义类型的实例上所需的方法；

根据程序代码的执行来调用所述方法；以及

将所述方法调用的结果作为查询语言表达式的求值返回。

24. 如权利要求 23 所述的计算机实现方法，其特征在于，进一步包括：创建为所述至少一个用户定义类型的子类型的所述第二用户定义类型。

25. 如权利要求 23 所述的计算机实现方法，其特征在于，进一步包括：创建为所述第二用户定义类型的子类型的所述至少一个用户定义类型。

26. 如权利要求 23 所述的计算机实现方法，其特征在于，进一步包括：使用公共语言运行库（CLR）语句来定义所述至少一个用户定义类型。

27. 如权利要求 23 所述的计算机实现方法，其特征在于，其中所述受管代码以下编程语言组中的至少一种编程语言来定义或得到：visual basic、visual C++、C、C++ 及 C#。

28. 如权利要求 27 所述的计算机实现方法，其特征在于，进一步包括：

将类型编译到所述组件；

向所述数据库注册所述组件；

向所述数据库注册所述组件的所述类型；以及

使用所述类型。

29. 如权利要求 23 所述的计算机实现方法，其特征在于，所述至少一种用户定义类型是用除结构查询语言（SQL）语句以外的语句来定义的。

30. 如权利要求 23 所述的计算机实现方法，其特征在于，进一步包括：使用除结构查询语言（SQL）语句以外的语句来定义所述至少一种用户定义类型。

31. 一种至少部分由计算系统实现的用于通过提供用于至少一个用户定义类型的元数据的可查询目录视图来扩展标量类型系统的计算机实现方法，所述计算机实现方法包括：

在存储器中存储用于描述所述至少一个用户定义类型的结构的信息；

在存储器中存储用于描述所述至少一个用户定义类型和所述至少一个用户定义类型中的第二用户定义类型之间的继承关系的信息，其中所述第二用户定义类型相对于所述至少一个用户定义类型的继承关系为所述至少一个用户定义类型的至少一个子类型或超类型；

在用于所述至少一个用户定义类型的元数据中记录所述至少一个用户定义类型和第二用户定义类型之间的继承关系，其中所述元数据可通过被查询的可查询目录视图而被搜索；以及

在存储器中存储所述元数据并向数据库系统注册所述用户定义类型；

接收所述数据库系统的查询语言的表达式，其中所述表达式的求值需要对该用户定义类型的实例的方法进行调用；

将该表达式翻译成一程序代码指令序列以调用所述用户定义类型的实例上所需的方法；

根据程序代码的执行来调用所述方法；以及

将所述方法调用的结果作为查询语言表达式的求值返回。

32. 如权利要求 31 所述的计算机实现方法，其特征在于，还包括：

查询所述存储器中所存储的所述用户定义类型的元数据以识别所述至少一个用户定义类型以及存储的描述与第二用户定义类型的继承关系的信息；

识别所述至少一个用户定义类型，其中所描述的继承关系为所述第二用户定义类型是所述至少一个用户定义类型的子类型；

识别所述至少一个用户定义类型的站点的属性，其中所述站点具有相关联类型定义以及规定哪里的子类型的值可替换所述至少一个用户定义类型的值的一属性；

将所述第二用户定义类型的值替换所述至少一个用户定义类型的值。

33. 如权利要求 31 所述的计算机实现方法，其特征在于，还包括：

在所述存储器中存储用于描述所述至少一个用户定义类型的结构的信息，其中描述的继承关系为所述至少一个用户定义类型是超类型，而第二用户定义类型是所述超类型的子类型；

识别所述超类型的一个值的运行时类型，其中所述值的运行时类型可以与所述超类型的编译时的值不同；

在所述值的运行时类型上创建索引；

通过基于所述值的运行时类型在预测中使用所述索引。

用以支持用户定义类型的继承的系统和方法

[0001] 本申请是申请日：2004.07.29，申请号为200480001723.8（国际申请号为PCT//US2004/024442），名称为“用以支持用户定义类型的继承的系统和方法”的申请的分案申请。

[0002] 相关申请的参照

[0003] 本申请要求对于2004年5月3日提交的美国专利申请第10/837,929号（代理卷号3843）的优先权，后者要求对于2004年2月10日提交的美国专利申请第10/775,282号（代理卷号MSFT-3029）的部分延续的优先权，这些申请所揭示的内容通过整体引用包括在此。

[0004] 本申请的主题涉及在以下共同转让的申请中所揭示的发明，其内容整体包括在此：于2003年8月21日提交的美国专利申请第10/647,058号（代理卷号MSFT-1748），题为“SYSTEMS AND METHODS FOR REPRESENTING UNITS OF INFORMATIONMANAGEABLE BY A HARDWARE/SOFTWARE INTERFACE SYSTEM BUT INDEPENDENT OFPHYSICAL REPRESENTATION”（表示可由硬件/软件接口系统管理、但与物理表示无关的信息单元的系统和方法）；于2003年8月21日提交的美国专利第10/646,941号（代理卷号MSFT-1749），题为“SYSTEMS AND METHODS FOR SEPARATING UNITSOF INFORMATION MANAGEABLE BY A HARDWARE/SOFTWARE INTERFACE SYSTEM FROMTHEIR PHYSICAL ORGANIZATION”（将可由硬件/软件系统接口管理的信息单元从其物理组织分离的系统和方法）；于2003年8月21日提交的美国专利申请第10/646,940号（代理卷号MSFT-1750），题为“SYSTEMS AND METHODS FOR THEIMPLEMENTATION OF A BASE SCHEMA FOR ORGANIZING UNITS OF INFORMATIONMANAGEABLE BY A HARD WARE/SOFTWARE INTERFACE SYSTEM”（实现组织可由硬件/软件接口系统管理的信息单元的基本模式的系统和方法）；于2003年8月21日提交的美国专利申请第10/646,632号（代理卷号MSFT-1751），题为“SYSTEMS ANDMETHODS FOR THE IMPLEMENTATION OF A CORE SCHEMA FOR PROVIDING A TOP-LEVELSTRUCTURE FOR ORGANIZING UNITS OF INFORMATION MANAGEABLE BY AHARDWARE/SOFTWARE INTERFACE SYSTEM”（实现提供用于组织可由硬件/软件接口系统管理的信息单元的顶层结构的核心模式的系统和方法）；于2003年8月21日提交的美国专利第10/646,645号（代理卷号MSFT-1752），题为“SYSTEMS ANDMETHOD FOR REPRESENTING RELATIONSHIP BETWEEN UNITS OF INFORMATIONMANAGEABLE BY A HARDWARE/SOFTWARE INTERFACE SYSTEM”（表示可由硬件/软件接口系统管理的信息单元间的关系的系统和方法）；于2003年8月21日提交的美国专利申请第10/646,575号（代理卷号MSFT-2733），题为“SYSTEMS AND METHODSFOR INTERFACING APPLICATION PROGRAMS WITH AN ITEM-BASED STORAGE PLATFORM”（使应用程序与基于项目的存储平台接口的系统和方法）；于2003年8月21日提交的美国专利申请第10/646,646号（代理卷号MSFT-2734），题为“STORAGEPLATFORM FOR ORGANIZING, SEARCHING,AND SHARING DATA”（用于组织、搜索及共享数据的存储平台）；于2003年8月21日提交的美国专利申请第10/646,580号（代理卷号MSFT-2735），题为“SYSTEMS AND METHODS FOR DATA MODELING INAN ITEM-BASED STORAGE PLATFORM”（用于基于项目的存

储平台中的数据建模系统和方法）。

技术领域

[0005] 本发明涉及计算机系统内的数据存储，尤其涉及在一数据库系统内提供用户定义类型（UDT）的继承的系统和方法，在该数据库系统中，UDT 是连同关系数据库引擎和文件系统使用的可扩展性机制，用于通过注册实现一特定合同的受管类型来扩展数据存储的标量类型系统。

[0006] 背景

[0007] MICROSOFT SQL SERVER 是一种综合性数据库管理平台，它提供了广泛的管理和开发工具、强大的提取、转换及加载（ETL）工具、商业智能和分析服务以及其他性能。除其他改进之外，MICROSOFT WINDOWS.NET 框架公共语言运行库（CLR）最近被集成到 SQL SERVER 数据库中。

[0008] CLR 是 MICROSOFT.NET 框架的中心内容，它为所有 .NET 代码提供执行环境。因而，在 CLR 内运行的代码被称为“受管代码”。CLR 提供程序执行所需的各种函数及服务，包括即时（JIT）编译、分配和管理存储器、强制类型安全、异常处理、线程管理和安全。如今，在首次调用 .NET 例程时，CLR 由 SQL SERVER 加载。

[0009] 在 SQL SERVER 早先的版本中，数据库程序员在编写服务器方代码时仅限于使用 Transact-SQL。Transact-SQL 是由国际标准组织（ISO）和美国国家标准化组织（ANSI）定义的结构化查询语言（“SQL”）的扩展。使用 Transact-SQL，数据库开发者能够创建、修改及删除数据库和表，也能够插入、检索、修改及删除储存在数据库内的数据。Transact-SQL 是专为直接结构性数据访问和操纵而设计的。尽管 Transact-SQL 擅长结构性数据访问和管理，它并不是象 VISUAL BASIC.NET 和 C# 那样成熟的编程语言。例如，Transact-SQL 不支持数组、集合、对每个循环、位移位或类。

[0010] 随着 CLR 被集成到 SQL SERVER 数据库中，数据库开发者能够完成单用 Transact-SQL 不可能或很难完成的任务。VISUAL BASIC.NET 和 C# 都是提供对数组、结构化异常处理和汇编的完全支持的现代编程语言。开发者可充分利用 CLR 集成来编写具有更复杂逻辑且更合适使用例如 VISUAL BASIC.NET 和 C# 语言的计算任务代码。这些编程语言提供面向对象的性能，例如封装、继承及多态。相关的代码能被容易地组织成类和名字空间。

[0011] 受管代码比 Transact-SQL 更适合数字密集运算和复杂的执行逻辑，且具有对包括字符串处理及正则表达式的许多复杂任务的广泛支持的特征。使用在 .NET 框架基类库（BCL）中找到的功能，数据库开发者可访问数千个预构建的类和例程，它们可从任意地储存的过程、触发器或用户定义的函数容易地访问。

[0012] 受管代码的另一好处是类型安全。在执行受管代码前，CLR 验证该代码是安全的。此过程被称为“验证”。在验证中，CLR 执行若干次核查，以确保该代码可安全运行。例如，核查该代码以确保未被写的存储器不被读。CLR 还将防止缓冲区溢出。

[0013] 在编写受管代码时，部署单元被称为程序集（assembly）。程序集被打包成动态链接库（DLL）。受管 DLL 程序集可被加载到 SQL SERVER 中并由其主存。CREATEASSEMBLY 语句用来在服务器中注册程序集。这里是一个例子：

[0014] CREATE ASSEMBLY YukonCLR

[0015] FROM' C:\MyDBApp\YukonCLR.dll'

[0016] 在此例中，FROM 子句指明要加载的程序集的路径名。

[0017] SQL SERVER 传统上支持“内建”标量类型，例如整数、浮点数、日期、时间及字符串。这些内建类型总是伴随着一组内建运算，例如 +、-、*、/，以及这些类型上的内建函数。这些类型、运算及函数在它们是由产品实现并打包的，且用户不能定义自己的类型的意义上是“内建”的。

[0018] 期望诸如SQL SERVER等数据库系统允许用户扩展数据库系统的类型系统，从而用户能够创建新的类型，它们在该数据库系统中担当标量类型的角色，但包含更复杂结构和行为，例如，创建由 X 和 Y 坐标组成的“Point（点）”类型。在这点上，SQL 标准及一些数据库管理系统 (DBMS) 产品使用了术语“用户定义类型”来描述几种形式的可扩展性类型。例如，SQL-99 标准描述了一种“独特类型”，这是可由用户定义的类型，它有一内部表示，该内部表示是一现有 SQL 内建数据类型的值。“独特类型”可任选地与现有的 SQL 内建标量类型共享比较和算术运算符、类型转换及合计（列）函数（例，最大值、最小值、平均值）。该独特类型可允许对其值定义各种约束。另外，通过经用户定义函数来定义独特类型专用的新函数，该独特类型可展示超越现有标量类型的行为。对于类型核查，独特类型和现有标量类型被视作不同类型。

[0019] 独特类型的主要优点是容易定义。如果新类型的内部表示只有可用现有内建类型描述的单个数据成员，且该内建类型已实现该新类型所需的大部分行为，那么独特类型是有吸引力的选择。用户不必担心如何实现管理该类型的盘上存储所需的行为、构造函数、比较运算符（用于排序和索引），算术运算符及类型转换运算符。用户只需选择需要在该独特类型上展示底层内建类型的哪个功能，并可任选地对各个值添加约束或对该新类型添加附加函数。独特类型另一优点是对内建类型可用的所有查询处理，例如直方图计算，可容易地用于独特类型的列。然而，独特类型的一个缺点是它们不能被容易地用来创建更复杂的类型。

[0020] SQL-99 标准还描述了一种“结构化类型”，这是一种可由用户定义的类型，且具有它是数据成员的集合的内部表示，其每个数据成员可以是一种不同的 SQL 内建或用户定义类型。这类似于 C 和 C++ 中结构的概念。SQL-99 描述了一种用于定义结构化类型的样式，通过这种样式，用户只需按照其内部结构来定义类型。系统自动生成其数据成员的取值器 (accessor) 和赋值器 (mutator) 函数、构造函数、及管理该类型实例的盘上表示的函数。

[0021] SQL 中所定义的结构化类型的主要优点是 (a) 容易定义类型的基本行为，及 (b) 定义更复杂类型的灵活性。然而，结构化类型的一个显著缺点是定义类型专用方法的复杂性，这些方法通常经以如 C 或 C++ 等通用编程语言编写的外部函数来定义。为了完整地定义结构化类型，类型的定义者必须跨立在 SQL 和某种其他的编程语言之间。

[0022] 尽管 SQL-99 的独特类型和结构化类型的特征向用户提供了使他们能扩展 SQL 数据库的现有标量类型系统的某些优点，但仍存在对使用户能通过用户定义类型来扩展数据库系统的标量类型系统的改良系统和方法的需求，这些用户定义类型担当标量类型的角色，但包含更复杂的结构和行为。

[0023] UDT 专利申请中所揭示的发明针对一种允许用户通过创建用户定义类型来扩展数据库系统的标量类型系统的系统和方法，该用户定义类型担当标量类型的角色，但包含更

复杂的结构和行为。根据该发明，用户以实现类的高级编程语言编写程序代码，该类定义了用户定义类型的结构，及可在该用户定义类型的实例上调用的方法。如在其中（及此文中）所使用的，当术语“结构”指用户定义类型时，它包含实现该类型的字段或属性组。在 UDT 专利申请中每个字段的类型可以是标量 SQL 类型或任何之前定义的用户定义类型。定义用户定义类型的类随即被编译并向数据库系统注册。特别地，定义了用户定义类型的 CLR 类可被编译为程序集，该程序集随即经 CREATE ASSEMBLY 数据定义语句向数据库系统注册。注册该程序集后，用户可使用 CREATE TYPE 数据定义语句在程序集内注册定义用户定义类型的类。

[0024] 对于该 UDT 专利申请的发明，数据库系统实施一种特定合同，即类必须实现为使用户定义类型能够担当 SQL 类型系统中标量的角色。如在该申请中（及此文中）所使用的术语“合同”指在面向对象编程环境中在运行时使用的一种技术，该技术用于核查要执行的代码是否满足某些先决条件或要求，从而确保它能正确地执行。根据该发明，与定义用户定义类型的类进行比较的合同包含若干要求。首先，类必须指定用于在数据库存储中持久保存用户定义类型的实例的多个不同格式之一。第二，类必须能够为该用户定义类型返回空值。第三，类必须提供一种用于在使用户定义类型和另一类型（例如字符串类型）间来回转换的方法。一旦满足了这些要求，数据库系统使用户定义类型的实例能被创建。在该申请的一实施例中，用户定义类型可被例示为表中的列值、变量、例程的参数、或例程的返回值。数据库系统储存关于定义用户定义类型的类的元数据，以供接下来创建该类型的实例时使用。在本发明的另一实施例中，使用描述定义该类型的类的元数据来执行对该用户定义类型的合同的验证。

[0025] 用于持久保存用户定义类型的实例的多个格式包含用户定义类型的实例根据数据库系统的本机格式来自动串行化的第一格式，以及用户定义类型的实例按由用户所创建的类定义的方式串行化的第二格式。此外，当该 UDT 专利申请的发明在集成了 MICROSOFT.NET CLR 的 MICROSOFT SQL SERVER 内实施时，可有用户定义类型的实例按 MICROSOFT.NET 框架提供的方法串行化的第三格式。

[0026] 对于 UDT 专利申请的发明，数据库系统查询语言中的表达式可包括对用户定义类型实例的一个或多个引用，从而表达式的求值需要对该用户定义类型的实例调用一方法。当数据库系统收到这样一个查询语言表达式时，它将该表达式翻译成一程序代码指令序列，当这些指令被执行时，将对该用户定义类型的实例调用所需方法。该数据库系统随即返回方法调用的结果，作为查询语言表达式求值的结果。在一实施例中，在对用户定义类型的实例调用方法前，该类型先被反串行化。

[0027] 该发明的另一个特征是，通过调用一赋值器方法改变用户定义类型的实例的值的能力。特别地，定义用户定义类型的类的作者包括一增变器方法，作为该类的一部分。当对该用户定义类型的实例调用此赋值器方法，它使该用户定义类型的值被改变。这个过程可包括将该用户定义类型的实例反串行化、调用赋值器方法来改变该实例的反串行化数据的值、然后将修改过的用户定义类型的实例串行化以持久保存改变。

[0028] 根据该发明的另一特征，定义户定义类型的类还可包含一属性，该属性指定该用户定义类型实例串行化的二进制表示将按二进制排序。这允许对该类型的实例做二进制比较，并能对该类型的实例执行索引。特别地，对于二进制排序的用户定义类型的实例，当

数据库系统收到要求对该类型的两个实例进行某种比较的查询语言表达式时（例如，＞、＜或＝），这两个实例的串行化二进制表示可用来对该表达式求值，而无须反串行化任一实例。此外，对于二进制排序的用户定义类型，可在数据库存储中创建一具有被定义为用户定义类型的列的表。然后可在该列上创建索引。也可对引用用户定义类型的查询语言表达式创建索引。在这种情形下，首先对该表达式生成一计算的列，然后对该计算的列创建索引。

[0029] 概述

[0030] 本发明的各种实施例针对扩展 UDT 框架以支持继承。继承是面向对象编程的关键原则之一，也是最现代的类型系统（例如 CLR 类型系统或 XSD 类型系统）的基本构建块。有了这一扩展，存储的对象数据建模性能将大为改善。它允许从公共语言运行库 (CLR) 类型系统到数据库类型系统的自然映射，而无需麻烦而缓慢的对象到关系映射技术。在查询语言中支持涉及继承的概念大大简化了继承的 SQL 编程建模。

[0031] 尽管在其他数据库产品中已实现了用户定义类型的继承，然而本发明的各种实施例针对 UDT 继承解决方案，该方案拥有现有技术未提供的以下性能中的一项或多项：

[0032] • 与 CLR 类型系统的无缝集成。本发明的若干实施例针对 UDT 继承系统与方法，其中基本 UDT 合同被指定为受管类型的一组必需的自定义属性和接口，且其中继承扩展了同一概念，并使用受管类型定义来推进对 SQL 类型系统的继承的规范。此解决方案使开发者能以其选择的任何遵从 CLS 的编程语言，包括但不限于 C#、C++ 及 VB.NET，来创建继承类型。

[0033] • 对可替换性的完全支持。对于本发明的若干实施例，如果一地点 (site)（变量、参数、列定义等等）被定义为是一特定的 UDT 则，该 UDT 的任一子类型的值可被储存于该地点。这个可替换性的概念扩展到该类型的所有使用，包括类型转换、转换、赋值及方法调用，并且以此方式结果组由客户端应用程序处理。

[0034] • 与索引和查询子系统的深度集成。对于本发明的许多实施例，继承添加了值的确切运行时类型的概念，该类型可以异于声明的类型。例如，SQLServer 支持对这类信息创建索引，并在基于该值的类型的谓词中使用该索引。继承还添加了虚拟行为的概念，虚拟行为可在子类型中被重定义（覆盖），且此概念可被合并到对一特定 UDT 函数的可索引性的计算中，并在类型创建时被验证。

[0035] • 可查询元数据及目录服务。对于本发明的各种实施例，类型及其子类型间的关系在类型创建时记录于元数据中，且可通过可查询目录视图获得。这使应用程序组成了查询，来确定一特定类型的可替换范围。

[0036] 在这点上，本发明的一实施例针对一种具有用于至少一个用户定义类型的可扩展类型系统的数据库，该用户定义类型包含描述其结构的信息，以及描述其与另一用户定义类型（或是子类型，或是超类型 / 基类型）间的继承关系的信息。对于某些实施例，用户定义类型是用除顺序查询语言 (SQL) 语句以外的某种语句定义的，例如公共语言运行库 (CLR) 语句、C、C++ 及 C#（“C 升半音号”）语句、和 / 或 visual basic 语句。在任一情况下，若干实施例还针对这样的系统，其中数据库知道两个用户定义类型间的继承关系（一为子类型，另一为超类型 / 基类型）

[0037] 对于某些实施例，通过明确地向所述数据库注册用户定义类型，来使数据库知道两个用户定义类型间的继承关系。在这点上，本发明用于若干实施例的延及全部的方法包

含 :(a) 将类型编译为程序集 ;(b) 向所述数据库注册所述程序集 ;(c) 向所述数据库注册所述程序集的所述类型 ;及 (d) 使用所述类型。

[0038] 附图简述

[0039] 上述总结及以下对较佳实施例的详细描述在连同附图一起读时能更好地被理解。为了阐明本发明,图中示出了本发明的示例性构造 ;然而,本发明并不限于所揭示的特定方法及手段。附图中 :

[0040] 图 1 是表示其中可包括本发明的各个方面的计算机系统的框图。

[0041] 图 2 是表示具有其中可实现本发明的多种计算设备的示例性网络环境的框图。

[0042] 图 3 是示出了在受管代码中例示的用户定义类型的实例的串行化和反串行化的框图。

[0043] 图 4 是示出 SqlUserDefinedType(SQL 用户定义类型) 的属性的非穷尽列表的表格。

[0044] 图 5 是伪代码,示出如果 Address 是一类型,而 USAddress 是 Address 的子类型,那么 Address 可以是具有类型为 Address 的列 addrcol 的表。

[0045] 图 6 是示出一客户机 / 服务器反串行化器的伪代码,它使用被反串行化的值的确切类型来确定其应该被反串行化成的 CLR 类型。

[0046] 图 7 是示出了 CONVERT(转换)、CAST(类型转换) 及 TREAT(处理) 命令的行为及语义的表。

[0047] 图 8 是示出了 UDT 方法可包含的例程属性的表。

[0048] 详细描述

[0049] 本发明的主题用符合法定要求的细节进行描述。然而,该描述本身并不旨在限制本专利的范围。相反,发明人构想所要求保护的本发明也可结合其他现有或未来技术以其它方式来实现,以包括类似于本文所描述的、但有所不同的步骤或元素。并且,尽管可在此文中用术语“步骤”意味着所用方法的不同方面,然而不应将此术语解释为暗示此文所揭示的各个步骤间的任何特定的顺序,除非明确地描述各个步骤的次序。

[0050] 在下文描述的实施例中,本发明的上述特征被描述为在 MICROSOFT SQLSERVER 数据库系统中实现。如上面提及的, SQL SERVER 结合了 MICROSOFT.NET 公共语言运行库 (CLR),以使受管代码能被编写及执行,以在 SQL SERVER 数据库的数据存储上操作。尽管下文描述的实施例运行于此环境中,然而应当理解,本发明绝非仅限于在 SQL SERVER 产品中实现。相反,本发明可在支持面向对象编程代码的实现以在数据库存储上操作的任一数据库系统中实现,例如面向对象数据库系统和具有对象关系扩展的关系数据库系统。因此应当理解,本发明不局限于下文描述的特定实施例,而是旨在覆盖所附权利要求书所定义的本发明精神和范畴之内的所有修改。

[0051] 计算机环境

[0052] 本发明的许多实施例可在计算机上执行。图 1 及以下讨论旨在提供对可实现本发明的合适计算机环境的简要概括描述。尽管不是必需,但本发明将在诸如由客户机工作站或服务器等计算机执行的程序模块等计算机可执行指令的通用环境中描述。一般而言,程序模块包括例程、程序、对象、组件、数据结构等等,它们执行特定任务或实现特定的抽象数据类型。此外,本领域技术人员应当理解,本发明可用其他计算机系统配置来实施,包括手

提设备、多处理器系统、基于微处理器的或可编程消费者电子设备、网络 PC、小型机、大型主机等等。本发明也可在分布式计算环境中实施，其中任务由通过通信网络连接的远程处理设备执行。在分布式计算环境中，程序模块可位于本地及远程记忆存储设备中。

[0053] 如图 1 中所示，一示例性通用计算系统包括常规个人计算机 20 或类似设备，包括处理单元 21、系统存储器 22 及系统总线 23，该总线将包括系统存储器在内的各种系统组件耦合到处理单元 21。系统总线 23 可以是若干种总线结构之一，包括存储器总线或存储器控制器、外设总线、以及使用多种总线体系结构中的任一种的局部总线。系统存储器包括只读存储器 (ROM)24 和随机存取存储器 (RAM)25。包含诸如在启动时帮助在个人计算机 20 内部各元件间传递信息的基本例程的基本输入 / 输出系统 26(BIOS) 储存在 ROM 24 中。个人计算机 20 还可包括用来读或写硬盘 (未示出) 的硬盘驱动器 27、用来读或写可移动磁盘 29 的磁盘驱动器 28、及用来读或写诸如 CD ROM 或其他光介质等可移动光盘 31 的光盘驱动器 30。硬盘驱动器 27、磁盘驱动器 28 和光盘驱动器 30 分别经硬盘驱动器接口 32、磁盘驱动器接口 33 和光盘驱动器接口 34 连到系统总线 23。这些驱动器及其相关的计算机可读介质为个人计算机 20 提供计算机可读指令、数据结构、程序模块及其他数据的非易失存储。尽管本文描述的示例性环境使用硬盘、可移动磁盘 29 和可移动光盘 31，然而本领域技术人员应当理解，可储存能由计算机访问的数据的其他类型的计算机可读介质，诸如磁带盒、闪存卡、数字视频盘、贝努利盒式磁带、随机存取存储器 (RAM)、只读存储器 (ROM) 等等，也可用于该示例性操作环境中。

[0054] 若干程序模块可储存在硬盘、磁盘 29、光盘 31、ROM 24 或 RAM 25 上，包括操作系统 35、一个或多个应用程序 36、其他程序模块 37 及程序数据 38。用户可通过诸如键盘 40 和定位设备 42 等输入设备来输入命令和信息到个人计算机 20 中。其他输入设备 (未示出) 可包括话筒、操纵杆、游戏垫、圆盘式卫星天线、扫描仪等等。这些及其他输入设备常通过耦合到系统总线的串行端口接口 46 连到处理单元 21，但也可用其他接口连接，诸如并行端口、游戏端口或通用串行总线 (USB)。监视器 47 或其他类型的显示设备也通过诸如视频适配器 48 等接口连到系统总线 23。除监视器 47 之外，个人计算机通常包括诸如扬声器及打印机等其他外围输出设备 (未示出)。图 1 的示例性系统还包括主机适配器 55、小型计算机系统接口 (SCSI) 总线 56、及连到 SCSI 总线 56 的外部存储设备 62。

[0055] 个人计算机 20 可用到一个或多个远程计算机，诸如远程计算机 49 的逻辑连接在联网环境中运行。远程计算机 49 可以是另一个人计算机、服务器、路由器、网络 PC、对等设备或其他普通网络节点，且通常包括上述相对于个人计算机 20 所描述的许多或全部元件，尽管在图 1 中只示出了记忆存储设备 50。图 1 中描述的逻辑连接包括局域网 (LAN)51 及广域网 (WAN)52。诸如此类的网络环境常见于办公室、企业范围计算机网络、内联网及因特网。

[0056] 当用于 LAN 网络环境中时，个人计算机 20 通过网络接口或适配器 53 连到 LAN 51。当用于 WAN 网络环境中时，个人计算机 20 通常包括调制解调器 54 或用于通过诸如因特网等广域网 52 建立通信的其他装置。调制解调器 54 可以是内置或外置的，它通过串行端口接口 46 连到系统总线 23。在联网环境中，相对于个人计算机 20 所描述的程序模块或其部分可储存在远程记忆存储设备中。应当理解，示出的网络连接是示例性的，也可使用在计算机间建立通信链路的其他装置。

[0057] 尽管可以想象，本发明的许多实施例尤其适用于计算机化的环境，然而本文中没有任何内容旨在将本发明限于诸如此类的实施例。相反，如本文中所用的术语“计算机系统”旨在包括能够存储并处理信息，和/或能使用所储存的信息来控制该设备本身的行为或执行的任何及所有设备，而不论这些设备本质上是电子的、机械的、逻辑的还是虚拟的。

[0058] 网络环境

[0059] 图 2 提供了示例性联网或分布式计算环境的示意图。该分布式计算环境包含计算对象 10a、10b 等，及计算对象或设备 110a、110b、100c 等等。这些对象可包含程序、方法、数据存储、可编程逻辑等等。这些对象可包含同样或不同设备的各部分，诸如 PDA、电视机、MP3 播放器、个人计算机等等。每个对象都可经由通信网络 14 与另一对象通信。此网络自身可包含向图 2 中的系统提供服务的计算对象及计算设备，且其自身可代表多个互连的网络。根据本发明的一方面，每一对象 10a、10b 等或 110a、110b、110c 等可包含一应用程序，该应用程序可利用 API 或其他对象、软件、固件和 / 或硬件，来请求对用来实现本发明方法的过程的使用。

[0060] 也应当理解，诸如 110c 等对象可主宿于另一计算设备 10a、10b 等，或 110a110b 等中。因此，尽管所描绘的物理环境将连接的设备示为计算机，然而诸如此类的图解纯粹是示例性的，且该物理环境可选地可被描绘或描述为包含诸如 PDA、电视机、MP3 播放器等的各种数字设备、诸如接口、COM 对象等软件对象，等等。

[0061] 有支持分布式计算环境的各种系统、组件及网络配置。例如，计算系统可由有线或无线系统、由本地网络或广泛分布的网络连到一起。当前，许多网络被耦合到因特网，它为广泛分布的计算提供了基础结构并包含许多不同的网络。任一基础结构可用于附到本发明的示例性通信。

[0062] 因特网通常指使用 TCP/IP 协议套件的网络和网关的集合，这些在计算机网络领域内是公知的。TCP/IP 是“传输控制协议 / 互联网协议”的缩写。因特网可被描述为地理上分布的远程计算机网络的系统，这些网络由执行各种网络协议的计算机互连，这些网络协议允许用户通过网络交互并共享信息。因为诸如此类的广泛分布的信息共享，诸如因特网等远程网络迄今一般已进化成一开放式系统，开发者可为其设计软件应用程序，来本质上不受限制地执行专门的操作或服务。

[0063] 因而，网络基础结构允许大量的网络拓朴，诸如客户机 / 服务器、对等、或混合体系结构。“客户机”是一个类或组的成员，该类或组使用与其不相关的另一个类或组的服务。因而，在计算中，客户是一进程，即概略地说是请求由另一程序所提供的服务的一组指令或任务。客户机进程无须“了解”关于其他程序或服务本身的工作细节，即可使用所请求的服务。在客户机 / 服务器体系结构中，尤其是连网系统中，客户机通常是访问由另一计算机，如服务器所提供的共享网络资源的计算机。在图 2 的例子中，可认为计算机 110a、110b 等是客户机，而计算机 10a、10b 等是服务器，尽管取决于环境，任何计算机可被认为是客户机、服务器或两者。这些计算设备的任一个可以用包含本发明的用户定义类型技术的方式来处理数据。

[0064] 服务器通常是可通过诸如因特网等远程或本地网络来访问的远程计算机系统。客户机进程可以在第一计算机系统里活动，而服务器进程可以在第二计算机系统里活动，它们通过通信媒介彼此通信，从而提供了分布式功能，并允许多个客户机利用服务器的信息

搜集能力。依照本发明所使用的任何软件对象可遍及多个计算设备而分布。

[0065] 客户机与服务器可使用由协议层提供的功能彼此通信。例如，超文本传输协议(HTTP)是与万维网(WWW)，或称“Web”联合使用的公用协议。通常，可用诸如互联网协议(IP)地址等计算机网络地址，或诸如统一资源定位器(URL)等其他引用，来相互标识服务器或客户计算机。网络地址可视作URL地址。可通过任何可用通信媒介提供通信。

[0066] 因此，图2示出了其中可使用本发明的示例性连网或分布式环境，它有一经由网络/总线与客户计算机通信的服务器。依照本发明，网络/总线14可以是LAN、WAN、内联网、因特网或其他网络媒介，它具有若干客户机或远程计算设备110a、110b、110c、110d、110e等，诸如便携式计算机、手持式计算机、瘦客户机、连网装置、或其他设备，诸如VCR、TV、烤箱、电灯、加热器等等。由此，可以构想本发明可应用于任何计算设备，期望结合该计算设备在数据库存储中创建用户定义类型。

[0067] 例如，在通信网络/总线14为因特网的网络环境中，服务器10a、10b等可以是服务器，客户机110a、110b、110c、110d、110e等经由诸如HTTP等许多已知协议的任一种与其通信。服务器10a、10b等也可担当客户机110a、110b、110c、110d、110e等，这可以是分布式计算环境的特性。

[0068] 通信在适当时可以是有线或无线的。客户机设备110a、110b、110c、110d、110e等可以经由通信网络/总线14通信或不通信，且可具有与其相关联的独立通信。例如，在TV或VCR的情况下，可以有或没有控制其的连网方面。每个客户计算机110a、110b、110c、110d、110e等，及服务器计算机10a、10b等，可配备各种应用程序模块或对象135，并配备对各种类型的存储元件或对象的连接或访问，文件或数据流可被在这些存储元件或对象上储存，或可向这些存储元件或对象下载、发送或移动部分文件或数据流。任一计算机10a、10b、110a、110b等可负责维护与更新数据库、存储器或其他存储元件20，以存储根据本发明处理的数据。因而，本发明可在具有客户计算机110a、110b等以及服务器计算机10a、10b等的计算机网络环境中使用，客户计算机可访问计算机网络/总线14并与其交互，服务器计算机可与客户计算机110a，110b等、及其他类似设备及数据库20交互。

[0069] UDT和受管代码

[0070] 图3是示出了用户定义类型的实例的串行化及反串行化的框图。如图所示，通过在存储器中串行化代表实例的对象，该用户定义类型的实例在盘上持久存留。当应用程序产生一包括引用UDT的方法或实例的谓词或表达式的查询时，该实例的持久存留的形式被反串行化(也称作“水合”的过程)，且CLR为整个对象分配存储器以接收其储存的值。CLR随即对实现应用程序或用户期望行为的对象调用适当的方法。

[0071] UDT继承

[0072] 如本文中所用的，以下术语应当具有所表明的涵义：

[0073] • 基类型：给定类型从其继承的类型，在各种继承模型中也称作超类型、超类或真先辈(proper ancestor)。

[0074] • 子类型：直接、或通过另一子类型，将当前类型作为其基类型的类型。

[0075] •地点(site)：编程语言中具有相关类型定义的任意上下文。例子包括变量定义、参数定义、列定义等等。

[0076] • 声明类型：地点的“编译时”类型，例如表中一列的类型，或函数参数。

[0077] •确切类型：特定值的“运行时”类型。在强类型的系统中，确切类型必须是声明类型，或声明类型的子类型。

[0078] • 可替换性：地点的属性，其中子类型的值可“取代”基类型的值。例如，参数可替换性意味着，如果一函数被声明为取一特定类型的参数，应当可能向该函数传递子类型的值。

[0079] 对于本发明的各种实施例，UDT 是在向 SQL Server 注册的程序集中实现的受管类型。UDT 可用于大多数能使用本机类型的上下文中，包括表定义变量和参数。在遵照 UDT 合同的 UDT 中定义的方法、属性及字段能从 T-SQL 调用。

[0080] 对于若干实施例，假设 UDT 是具有行为的简单标量。此假设体现在 UDT 支持的简单编程模型及串行化设计中。例如，文件系统可使用 UDT 抽象来创建“结构化类型”，这些结构化类型不是简单标量，而是具有复杂结构与行为的类型。

[0081] 下面的伪代码描述了本发明的若干实施例的子类型的创建：

[0082] CREATE TYPE[类型模式名 .] 子类型名

[0083] UNDER[类型模式名 .] 基类型名

[0084] EXTERNAL NAME 程序集名 :CLR 类型名

[0085] 为建立继承合同，并用于本发明的若干实施例，UDT 作者必须用例如 SqlUserDefinedType 自定义属性在类型分层结构中注释基类型，且该属性的性质指定了应用于整个类型家族（类型及其所有子类型）的不变量。图 4 中示出了 SqlUserDefinedType 的属性的非穷尽列表。然后，在类型注册时，用户必须添加“UNDER”子句来指示一必须在其下注册特定类型的基 SQL 类型，例如：

[0086] / *创建 Address 类型，没有超类型* /

[0087] create type Address

[0088] external name MyTypes::Address

[0089] / *创建其下的 USAddress 子类型* /

[0090] create type USAddress under Address

[0091] external name MyTypes::USAddress

[0092] 对于本发明的若干实施例，基类型名必须以专用模式注册为有效 UDT。此外，CLR 类型必须是通过其定义基类型名的 CLR 类型的直接子类型。该子类型不应定义 SqlUserDefinedTypeAttribute，而应从其父类型继承该属性。此外，基类型决不能是二进制排序的。类似地，基类型中指定的所有带外例程属性 (SqlMethodAttribute 注解）必须与子类型定义相一致 - 具体地，基类型中定义的所有可索引方法对于子类型中覆盖的例程必须仍可索引 - 并且可从基类型收集此信息，并通过在该子类型的所有例程上枚举来验证此信息，作为 CREATE TYPE 核查的一部分。

[0093] 对于本发明的各种实施例，可按照特征的外部表面区域及其底层实现来描述 UDT 继承的框架。此外，子类型值被认为是在列定义中；局部变量和参数的批量定义、已储存的过程及函数中；以及其中结果列可包含该列所声明的类型或其任一子类型的值的查询结果集中被认为是可替换的。

[0094] 对于本发明的若干实施例，允许子类型的实例被储存在被声明为超类型的列中有以下含义：(a) 对于基类型和子类型，串行化格式应当是相同的，及 (b) 父类型的

MaxByteSize(最大字节大小)值必须足够大,以允许储存子类型的实例。在运行时,如果空间不够,则试图在基类型的列中储存子类的值将失败,因此基类型可被定义为无限来防止此错误。

[0095] 对于本发明的各种实施例,所有的赋值可保持该值的确切类型,且将子类型的值赋给基类型的变量总是会成功,但将声明类型为基类型的值赋给子类型的变量要求一显式的转换,仅当该值的确切类型可被赋给目标类型时,该转换才会成功。例如,如果 Address 是一类型,且如果 USAddress 是 Address 的子类型,那么如图 5 中所示,Addresses 可以是具有类型为 Address 的列 addrcol 的表。

[0096] 在实现本发明的若干实施例时,关于两个类型间的继承关系的信息是数据库的标量类型系统的头等部分,且该信息用于确定可赋值性及各值之间的转换。在某些实施例中,称为 IS OF 和 TREAT 的两个新算子被引入到标量类型系统中,这些算子可用于在标量表达式求值过程中对类型谓词和原地转换建模。为允许正确的语义推理及组合公用运算(诸如查询的项目列表中的 IS OF 谓词和 TREAT),TREAT 算子在查询编译时被转换。作为进一步的优化,IS OF 算子可按照 hierarchical_type_id 来重写,以利用可用的任何类型专用索引。当碰到 UDT 表达式时,UDT 表达式编译过程被修改,来说明该分派的本质(虚拟的还是静态的),并消除基于该方法的完整签名的方法的歧义,包括参数类型。此信息从外部编译过程传入,并被用于方法定位。一旦找到该方法,就使用适当的 IL 指令来调用静态分派或虚拟方法。然后在运行时,UDT 反串行化代码使用该值的确切类型来创建正确类型的对象,并用持久存留的状态来填充它。该对象被压入执行栈,且调用该方法。

[0097] 对于若干实施例,客户机 / 服务器反串行化器也使用被反串行化的值的确切类型来确定其应当被反串行化成的 CLR 类型。如果客户机以前没有遇到过该类型,则作出对复制连接的带外请求,以按需下载关于该 CLR 的元数据,并用此信息来定位 CLR 类型(见图 6)。

[0098] 对于若干实施例,可假定所有从 T-SQL 调用的方法、属性和字段都用实例的确切类型而不是调用地点的声明类型来动态地解析。此外,允许在超类型中定义且可直接从 T-SQL 调用的所有继承的方法、属性和字段。因而,在用于调用声明方法与继承的方法的句法上没有区别。

[0099] 对于本发明的若干实施例,用于定位目标调用的方法可以用伪代码来如下表现其特征 :如果 D 是调用地点的声明类型,M 是方法名,P1..Pn 是该方法的参数,且 T1..Tn 是这些参数的类型,那么如果查询是“select<类型D的地点>.M(P1...Pn)”,则该调用方法将包含 :

[0100] startType = D ;

[0101] Method targetMethod = null ;

[0102] while(true)

[0103] {

[0104] If(startType == null)

[0105] throw new NoSuchMethodException(D,M) ;

[0106] Method[]methods = GetMethods(startType,M,{T1...Tn})

[0107] if(methods.Length == 0)

[0108] startType = startType.BaseType ;

[0109] else if(methods.Length > 1)

[0110] throw new OverloadedMethodException(M) ;

[0111] else

[0112] {

[0113] targetMethod = methods[0] ;

[0114] break ;

[0115] }

[0116] }

[0117] 对于用于本发明若干实施例的方法调用，方法名匹配使用了二进制集合。此外，参数类型核查确保每个参数的 CLR 类型可从 SQL 参数类型隐式地转换。并且，如果在同一类型中有两个合适的方法，则不允许方法重载，但如果这两个方法取不同个数的参数或它们在分层结构中以不同的类型来定义，则允许方法重载。类似地，如果方法或属性返回一未向 SQL 注册的 CLR 子类型的实例，则系统在运行时会引发一个错误。此外，实际所用的分派指令取决于所解析的方法 - 即，如果找到的方法被定义为虚拟或重载方法，就使用虚拟分派，否则就使用静态分派。

[0118] 对于本发明的各种实施例，从一类型到其超类型的转换（向上类型转换）总会成功，而对于从一类型到其子类型的转换（向下类型转换），如果其值是正确的类型就会成功，如果不是正确的类型就会失败，其确切的失败模式取决于所用的转换算子。

[0119] 对于诸如此类的实施例，有如下三个转换算子 :CONVERT，CAST 及 TREAT :

[0120] •CONVERT([模式 .] 类型，目标类型) → [模式 .] 类型的值或者如果转换失败则为错误

[0121] • CAST(值 as[模式 .] 目标类型) → [模式 .] 类型的值或者如果转换失败则为错误 .

[0122] • TREAT(值 as[模式 .] 目标类型) → [模式 .] 类型的值或者如果转换失败为 NULL

[0123] 对于这些实施例，CAST 和 CONVERT 可以是现有的算子，它们经修改以理解继承与可替代性。另一方面，TREAT 是遵从 SQL99 的句法 (SQL 标准的 §6.25)，供向下类型转换，或当超类型的表达式要被当作其子类型之一（该标准称之为子类型处理）时缩小脚本使用。此算子在两个上下文中有效 :(a) 用来访问在子类型中定义的属性 ;以及 (b) 用来更新在子类型中定义的属性。在编译时，如果值的声明类型不是目标类型的超类型，将报告一个错误，且 TREAT 将失败。例如 :

[0124] select TREAT(person as Employee).Salary from T

[0125] where person is of(Employee)

[0126] update T

[0127] set TREAT(person as Employee).Salary = 10000

[0128] where ID = x

[0129] 另一方面，该标准不支持在 NULL 实例上调用赋值器方法，且 SQL Server 也会对 TREAT 算子的结果强加这个限制。如果 TREAT 产生一 NULL 值，则在试图更新该 NULL 值时系

统将引发一错误。换言之，对于本发明的若干实施例，逻辑上 TREAT(值 as 类型 x) 可被变换为 IF(值 IS OF 类型 x)THEN 值 elseNULL，如图 7 的表中所示，它概括地示出了 CONVERT、CAST 及 TREAT 命令的行为和语义。

[0130] 对于若干实施例，并作为 UDT 扩展的一部分，如果接收者 (this) 是空值，则 SqlUserDefinedTypeAttribute 和 SqlMethodAttribute 上的新性质可用于控制是否应调用类型上的方法，尽管通过使用 SqlMethodAttribute 来将其设置为真，可对特定方法覆盖它。对于这些实例，默认值为 FALSE。此外，如果该方法的返回类型是可为空的类型，则返回该类型的特异的空值。如果返回类型不能为空，则返回该类型默认的 clr 值。引用类型的默认值为空，值类型的默认值是调用该类型的默认构造函数的结果。

[0131] 对于各种实施例，系统中所有的空值都是类型化的。然而，由于若值为空，如该标准中所指定的类型谓词返回未知，因此无法从 SQL 确定空值的具体类型。因而这些实施例可把所有空值当作是调用地点的声明类型，而不是插入到地点的值的确切类型。例如，

[0132] declare @emp Employee

[0133] set@emp = NULL

[0134] insert into Persons values(@emp)

[0135] select pcol from Persons-returns a null value, the type of

[0136] the instance is Person

[0137] 对于各种实施例，类型谓词支持以下两者 :(a) 在列中返回类型及其所有子类型的所有实例，以及 (b) 在列中返回一特定类型的所有实例 (如 SQL99 标准 §8.14 中所述)。考虑下列代码 :

[0138] udtColumnOrValue IS[NOT]OF(类型指定符)

[0139] 类型指定符 :: = [ONLY]TYPE[, 类型指定符]

[0140] 如果 udtColumnOrValue 为空，则结果是未知，否则谓词返回一布尔类型的结果。

[0141] 类型谓词可在其中准许产生布尔值的表达式的所有标量上下文中使用例如，仅以下类型的实例 :

[0142] select convert(USAddress, addrcol)as USAddrCol from Addresses

[0143] where addrcol is of(only USAddress) ;

[0144] 这与类型及其所有子类型的实例相反 :

[0145] select convert(USAddress, addrcol)as USAddrCol from Addresses

[0146] where addrcol is of(USAddress) ;

[0147] 此外，对于本发明的若干实施例，sys.assembly_types 目录视图会有如下的附加列 :

[0148]

base_type_id	int	我的基类的 user_type_id

[0149] 此外，对于本发明的许多实施例，当在子类型中覆盖虚拟方法时，被覆盖的实现对于在基方法中指定的例程属性必须是可替代的。在类型注册时核查该矩阵，且如果检测到无效组合则产生一个错误。这里的指导原则是子类型的实现不应破坏基方法的可索引性。因而，如果基方法的定义是可索引的，则覆盖的实现也应是可索引的。此外，子类型的实现

不应使为基方法定义（空接收器及自变量、sql 方面 (sqlfacet)、数据访问、赋值器方法调用）编译的表达式无效。另一方面，子类型可添加不与基类型冲突，但需要类型转换或处理以被激活的行为（可索引性）。如果“N”表示“在子类型中是新的”，即，即使基类型具有设置为假或未定义的值子类型也可将属性设置为真，且如果“S”表示“与基类型定义相同”，即，子类型中的定义与基类型中的定义相同，那么 UDT 方法可包含如图 8 中所示的例程属性。

[0150] 除前述之外，对于本发明的若干实施例，每个 UDT 值必须自始至终带着其特定的 type_id，且一旦创建了 UDT 就永不改变。由于不支持对 UDT 继承的二进制排序，因此该方法不影响对类型的二进制比较（因为所有要进行二进制比较的实例有相同的 type_id）。此外，DROP 类型应当核查没有列被定义为包含子类型值的任何超类型（从而结果，DROP 类型可能开销很大）。类似地，添加子类型会使引用链中的每个类型的超类型的任何比较方案无效。

[0151] 结论

[0152] 本文描述的各种系统、方法及技术可用硬件或软件，或在适当时以两者的组合来实现。因而，本发明的方法与装置，或其特定方面或部分，可以是包含在例如软盘、CD-ROM、硬盘或任何其他机器可读存储介质等有形介质中的程序代码（即，指令）的形式，其中，当程序代码被加载到机器，例如计算机，并由其执行时，该机器成为用来实施本发明的装置。在可编程计算机上的程序代码执行的情况下，计算机通常包括处理器、处理器可读的存储介质（包括易失性及非易失性存储器和 / 或存储元件）、至少一个输入设备及至少一个输出设备。一个或多个程序较佳地以高级过程语言或面向对象的编程语言来实现，以与计算机系统通信。然而，如有需要，程序能以汇编语言或机器语言来实现。在任何情况下，语言可以是已编译或已解释的语言，并与硬件实现结合。

[0153] 本发明的方法与装置也可以用程序代码的形式来实施，该代码通过某一传输媒介发送，例如通过电线或电缆、通过光纤或经由任何其他传输形式，其中，当程序代码由例如 EPROM、门阵列、可编程逻辑器件 (PLD)、客户计算机、录象机等等机器接收，并加载到其上由其执行时，该机器成为用来实施本发明的装置。当在通用处理器上实现时，该程序代码联合处理器来提供用于执行本发明的索引功能的唯一装置。

[0154] 虽然本发明是结合各个图中的较佳实施例来描述的，然而应当理解，可使用其他类似实施例，或可对所描述的实施例作修改和添加以执行本发明同样的功能，而不会偏离本发明。例如，尽管本发明的示例性实施例是在仿真个人计算机功能的数字设备的环境中描述的，然而本领域技术人员会意识到，本发明不限于诸如此类的数字设备，正如本申请中所描述的，本发明可应用于任何数量的现有或新兴计算设备或环境，例如游戏控制台、手持式计算机、便携式计算机等等，不管它们是有线还是无线的，并且本发明也可应用于经由通信网络连接并在网络上交互的任意数量的诸如此类的计算设备。此外，必须强调的是，各种计算机平台，包括手持设备操作系统及其他应用专用硬件 / 软件接口系统，已为此文所构想，尤其是随着无线连网设备的持续增长。因而，本发明不应限于任一单个实施例，而是应当根据所附权利要求书的广度及范围来解释。

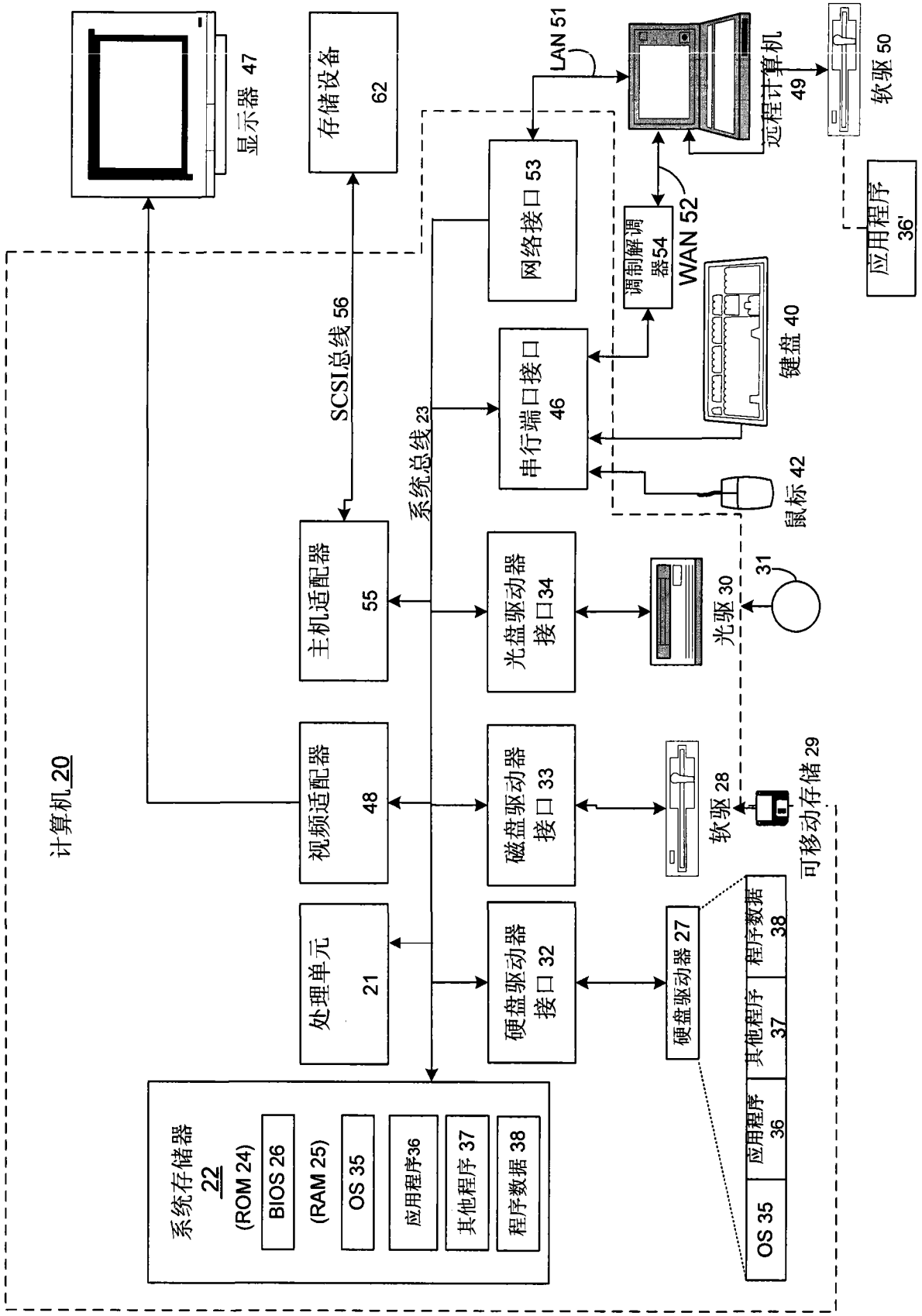

计算机 20
显示器 47
存储设备 62
LAN 51
远程计算机 49
软驱 50
应用程序 36'
网络接口 53
调制解调器 54
WAN 52
SCSI总线 56
系统总线 23
串行端口接口 46
键盘 40
鼠标 42
主机适配器 55
光盘驱动器接口 34
光驱 30
31
视频适配器 48
磁盘驱动器接口 33
软驱 28
可移动存储 29
处理单元 21
硬盘驱动器接口 32
硬盘驱动器 27
程序数据 38
其他程序 37
应用程序 36
OS 35
系统存储器 22
(ROM 24)
BIOS 26
(RAM 25)
OS 35
应用程序36
其他程序 37
程序数据 38

图 1

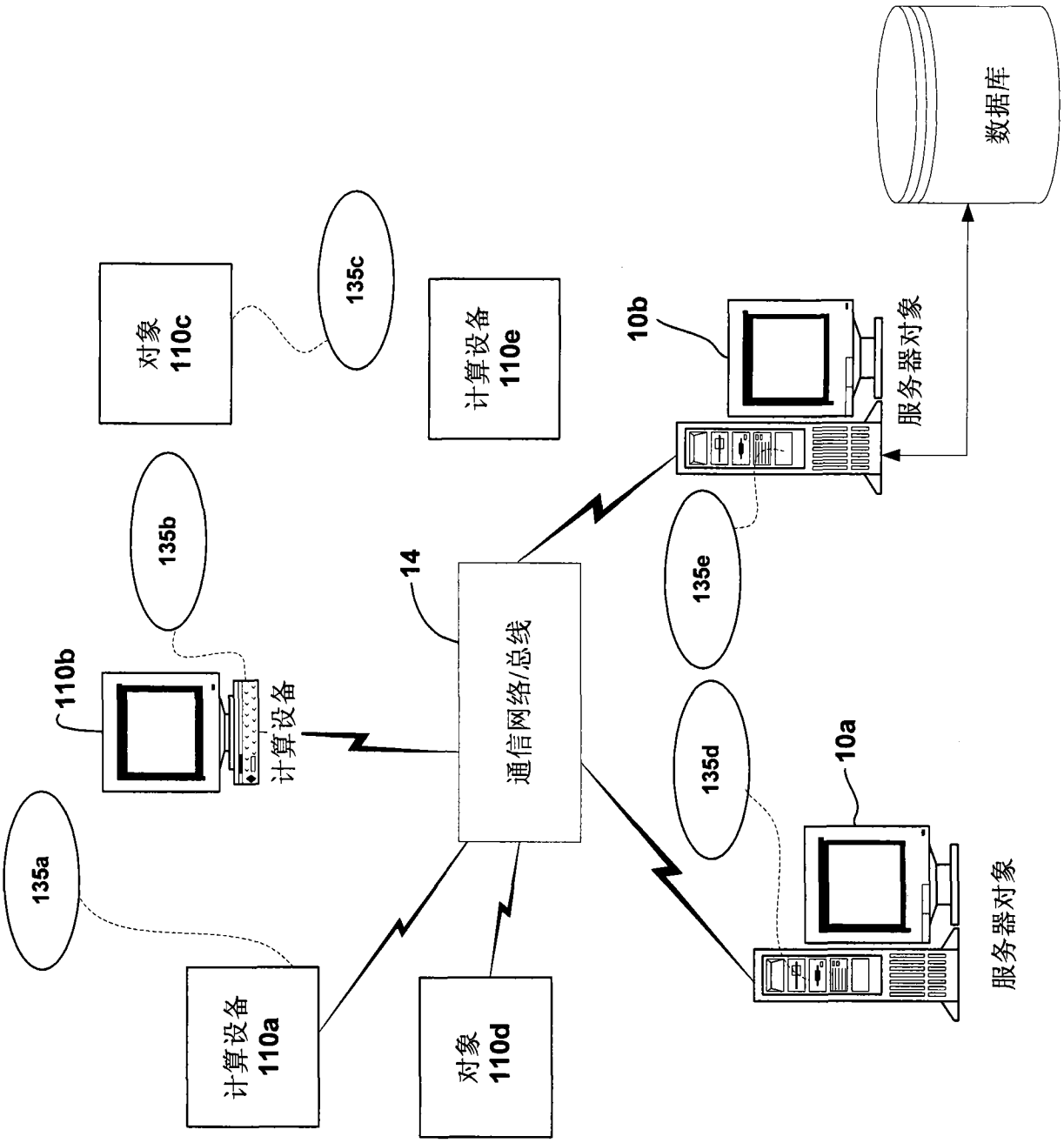

数据库
对象
110c
135c
计算设备
110e
10b
服务器对象
135b
14
通信网络/总线
135e
110b
计算设备
135d
10a
135a
计算设备
110a
对象
110d
服务器对象

图 2

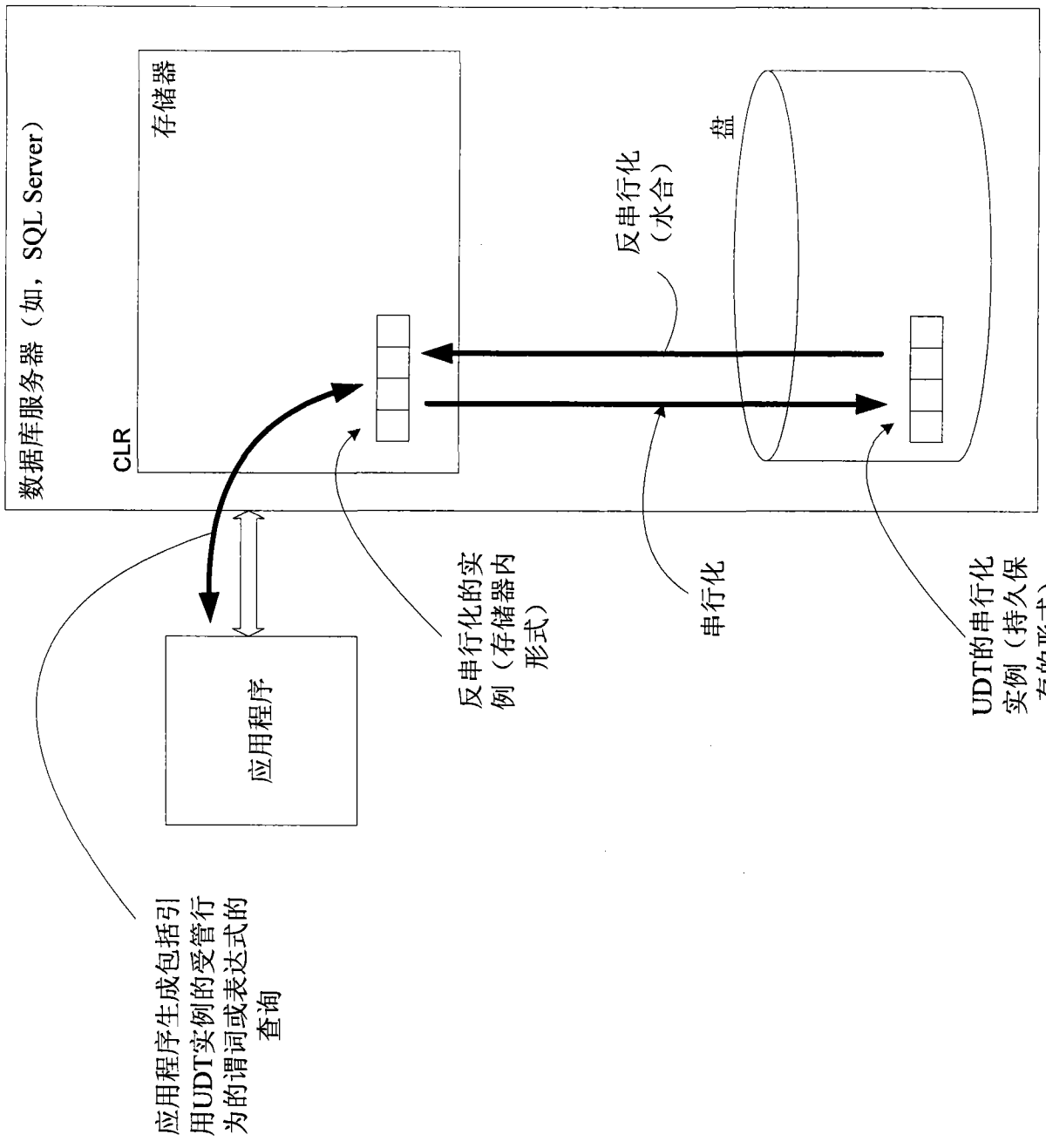

数据库服务器（如，SQL Server）
存储器
CLR
盘
反串行化（水合）
应用程序
反串行化的实例（存储器内形式）
串行化
UDT的串行化实例（持久保存的形式）
应用程序生成包括引用UDT实例的受管行为的谓词或表达式的查询

图 3

属性名	描述
Format	持久格式
MaxByteSize	最大字节数
IsByteOrdered	持久形式的二进制顺序是否和该类型的值的语义顺序相同？
IsFixedSize	所有的值都是同样大小？

图 4

```
/* 创建Address类型，无超类型 */
create type Address external name
MyTypes::Address

/*于其下创建USAddress子类型 */
create type USAddress under Address
external name MyTypes::USAddress

/*变量可替代性*/
declare @usaddr USAddress, @addr Address

/*
创建USAddress类型的值，并将其赋给USAddress类型的变量
*/
set @usaddr = convert(USAddress, 'state=WA;
city=Redmond; street=1234 NE 1st street;
zip=98052'

/* 将子类型赋给基类型的变量*/
set @addr = @usaddr
/* 将基类型变量的值转换成子类型并赋值*/
set @usaddr = convert(USAddress, @addr)

/*列可替代性*/
insert into Addresses (addrcol)
VALUES(@addr)
insert into Addresses (addrcol) VALUES
(@usaddr)
```

图 5

```
/*结果集可替代性，一些地址值是Address类型，其他是USAddress类型*/
select addrcol::ToString() from Addresses

/* 现在在地址表里有两个行，
 * 一行是Address类型，另一行是USAddress类型。
 * 这是来自这两行的值的输出，第一个是对于Address类型的值，
 * 第二个是对于USAddress类型的值。
 * 此例试图示意结果集也是可替代的。
 */
--output:
[from addr type] 1234 NW 1st Ave Redmond

[from usaddr type] 1234 NW 1st Ave Redmond state WA zip 98052
```

图 6

名	出错时的行为	语义
CONVERT	引发错误	新值
CAST	引发错误	新值
TREAT	NULL	原地

图 7

名	描述	允许的值
IsMutator	控制方法是否可 用于更新上下文	S
OnNullCall	若参数为空 则调用方法	S
InvokeIfReceiverIsNull	若接收者为空 则调用方法	S
IsDeterministic	同样的输入 返回同样的输出	S, N
IsPrecise	进行浮点计算	S, N
DataAccess	进行数据访问	S
SystemDataAccess	进行元数据访问	S
SqlFacet on return value or parameter	指定返回类型或 参数的SQL签名	S(*)该方面上的所有 属性应当相同

图 8