IBM DB2 Alphablox

DB2 Alphablox Cube Server 管理员指南

版本 8.4

S151-0145-03

IBM DB2 Alphablox

DB2 Alphablox Cube Server 管理员指南

版本 8.4

S151-0145-03

注意:

在使用本资料及其支持的产品之前，请阅读第 43 页的『声明』中的信息。

第四版（2006 年 9 月）

本版本适用于 IBM DB2 Alphablox for Linux, UNIX and Windows（产品号 5724-L14）的 V8R4 及所有后续发行版和修订版，直到在新版本中另有声明为止。

目录

第 1 章 构建立方体概念

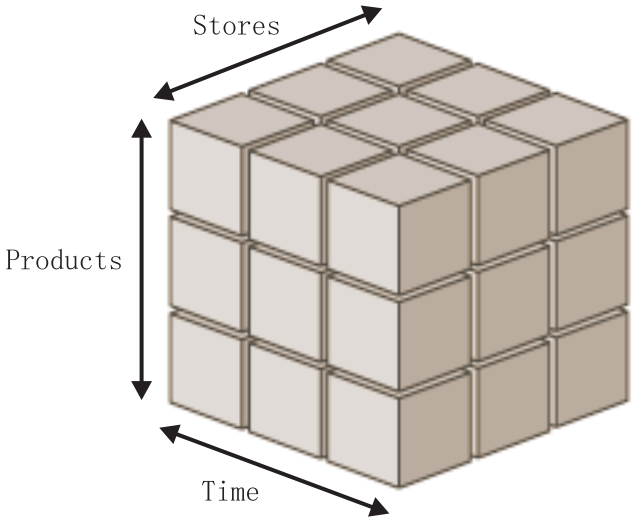
IBM DB2 Alphablox for Linux, UNIX and Windows 包含 DB2 Alphablox Cube Server。设计 DB2 Alphablox Cube Server 的目的是为存储在关系数据库中的数据提供多维视图。本主题介绍 DB2 Alphablox Cube Server、提供它所适用的应用程序类型的背景信息并描述它的使用要求。

DB2 Alphablox Cube Server 概述

IBM DB2 Alphablox for Linux, UNIX and Windows 包含 DB2 Alphablox Cube Server。设计 DB2 Alphablox Cube Server 的目的是为存储在关系数据库中的数据提供多维视图。本主题介绍 DB2 Alphablox Cube Server、提供它所适用的应用程序类型的背景信息并描述它的使用要求。

DB2 Alphablox Cube Server 允许管理员为关系数据库中的数据创建多维表示。*立方体*是在联机分析处理（OLAP）中经常使用的数据模型，它用来表示那些通常通过多个维进行分析的业务数据。*维*是用来分析业务的概念上的轴。例如，零售业务的业绩分析可以基于时间、产品和商店。对于此项业务，*Time*、*Products* 和 *Stores* 全都是维。每个维都具有一个或多个*层次*，这些层次共同定义了该维的整体层次结构。例如，*Time* 维可以具有 *Year*、*Quarter* 和 *Month* 层次。

立方体用来建立业务模型。由于三维的立方体可以被绘制成几何立方体，所以很容易形象化，但是立方体可以具有任意数目（从 1 到 *n*）的维。



在立方体的维的交点处，分析员可以查看量度。*量度*是位于给定一组维交点处的数值，它们通常是业务度量（例如商品的销售量、利润和成本）。例如，要查看给定产品在给定时间给定商店的销售情况，请检查立方体中那些维的相交位置以查找量度。

对关系数据构建立方体

许多机构希望使用“数据集市”和“数据仓库”来以可查询的格式存储它们的关系数据。通常，他们将此数据从此数据所在的事务系统移动、清理和转换到另一个为了提高查询性能而进行优化的关系数据库。

这些经过转换的数据库包含有关一个或多个主题的历史信息，并且有时称为数据仓库或数据集市。这些数据集市和数据仓库将使用 IBM® DB2™、Oracle、Microsoft® SQL Server、Sybase 或其他关系数据库来创建。这些数据库的主要用途是允许用户查询历史信息。要了解有关这些数据仓库和数据集市数据库的典型模式设计的详细信息，请参阅第 7 页的第 2 章，『维模式设计』。

由于维模型使您能够更容易地指出与特定业务流程或业务领域相关的业务问题，所以，对于用户来说，如果使用维模型，查询关系数据库就会变得更为容易。以维层次结构组织数据时，数据的结构对于用户在感觉上更熟悉且更容易理解数据类别之间的关系。根据维模型的大小和复杂性以及业务需求的不同，用户可能需要专用的 OLAP 服务器（如 IBM DB2 OLAP Server™ 或 Microsoft Analysis Services）的功能。在这些情况下，将从关系数据库中抽取数据并将它们构建为能够提供高级分析功能的专用高速立方体。在那些不需要专用 OLAP 服务器的全部功能但是要向用户提供 OLAP 分析功能的情况下，可以使用 DB2 Alphablox Cube Server 的关系立方体构建能力。

借助 DB2 Alphablox Cube Server，管理员可以根据关系数据来构建 DB2 Alphablox 立方体；也就是，使用对底层 RDBMS 执行的查询来填充 DB2 Alphablox 立方体。

DB2 Alphablox Cube Server 应用程序

DB2 Alphablox Cube Server 使您能够以 OLAP 立方体形式快速显示关系数据。它提供了功能全面的 OLAP 服务器（如 IBM DB2 OLAP Server、Hyperion Essbase 或 Microsoft Analysis Services）的一部分智能功能。设计 DB2 Alphablox 立方体是为了利用位于数据仓库和数据集市中的干净数据；而不是用来替代功能全面的 OLAP 服务器。这对于创建您没有时间和资源来为其开发功能全面的 OLAP 数据库的多维数据源来说是十分有用的，并且这对于提供相对较小的立方体来说尤其有帮助（即使那些立方体是根据非常大型的数据库构建的）。

DB2 Alphablox Cube Server 非常适合于构建立方体，这些立方体返回的数据集与填充它们的底层数据库比较而言相对较小。底层数据库可以非常大，事实表可能包含数以十亿计的行（要了解事实表的定义，请参阅第 8 页的『事实表』）。DB2 Alphablox 立方体将预先计算的结果存储在内存中，而不是存储到磁盘。任何未存储在内存中的结果都将保存在底层数据库中；立方体通过将 SQL 查询发送给数据库来根据实际需要检索结果。然后，将查询结果存储在内存中，并且该结果立即可供 DB2 Alphablox 应用程序访问。

创建原型

使用 DB2 Alphablox Cube Server 创建的关系 OLAP（ROLAP）立方体提供了一种快速测试专用 MOLAP 立方体的可能值的方式。DB2 Alphablox 立方体也可能会为用户提供一种满意的解决方案，使他们稍后不必在单独的 MOLAP 服务器上构建 MOLAP 立方体。

因为可以快速创建 DB2 Alphablox 立方体，所以您可以向用户提供对那些可以快速访问公司数据的 DB2 Alphablox 应用程序的访问权，并使业务用户能够洞察各种信息。如果您已启用了 DB2 数据库的 Cube Views 功能，则可以使用 DB2 Cube Views 元数据

来快速地定义 DB2 Alphablox 立方体并快速为用户提供对 DB2 数据的访问权。使用预定义的 Cube Views 元数据来构建 DB2 Alphablox 立方体的另一个好处是: 由具体化查询表（MQT）提供的性能增益将使 DB2 Alphablox 立方体的用户获益。或者，DB2 Alphablox 立方体可以访问位于其他受支持的关系数据库上的数据。DB2 Alphablox 立方体提供了一个极好的方法供您为开发周期中早期的大规模 OLAP 解决方案创建原型。并且，DB2 Alphablox 立方体的用途通常足够满足 DB2 Alphablox 应用程序用户的需要。

具有简单明了的维和量度的立方体

DB2 Alphablox 立方体的每个维都可以具有单一层次结构。为了表示复杂的具有多个层次结构的维，请使用功能全面的 OLAP 服务器，如 DB2 OLAP Server、Hyperion Essbase 或 Microsoft Analysis Services。然而，许多复杂的业务方案并不要求每个维具有多个层次结构。

注: 如果应用程序在一个维中需要多个层次结构，则可以创建具有相同根层次但却具有不同层次结构的多个维。

DB2 Alphablox 立方体中的量度使用对底层数据库执行的有效 SQL 表达式进行定义。为了避免指代不清而造成问题（当不同的表具有同名的列时就会发生这种问题），对指定的 SQL 表达式有一些限制。要了解更多详细信息，请参阅第 12 页的『量度』。

虽然不同的 RDBMS 供应商支持不同级别的计算，但是所有的主要 RDBMS 供应商都支持一组相当丰富的计算。如果应用程序需要进行一些无法以 SQL 表示的计算，则可能需要考虑使用功能全面的 OLAP 服务器。

DB2 Alphablox Cube Server 的优点

因为 DB2 Alphablox Cube Server 随 DB2 Alphablox 一起提供，并且不需要管理物理磁盘存储器，所以功能全面的 OLAP 服务器的许多典型管理任务都已被简化或消除。某些优点包括:

- DB2 Alphablox Cube Server 不要求管理磁盘空间。
- DB2 Alphablox Cube Server 使用 DB2 Alphablox 安全模型，从而不要求执行其他的工作来管理用户。
- DB2 Alphablox Cube Server 随 DB2 Alphablox 一起提供，从而不需要安装其他的软件。

在 DB2 Alphablox 应用程序环境中的 DB2 Alphablox Cube Server

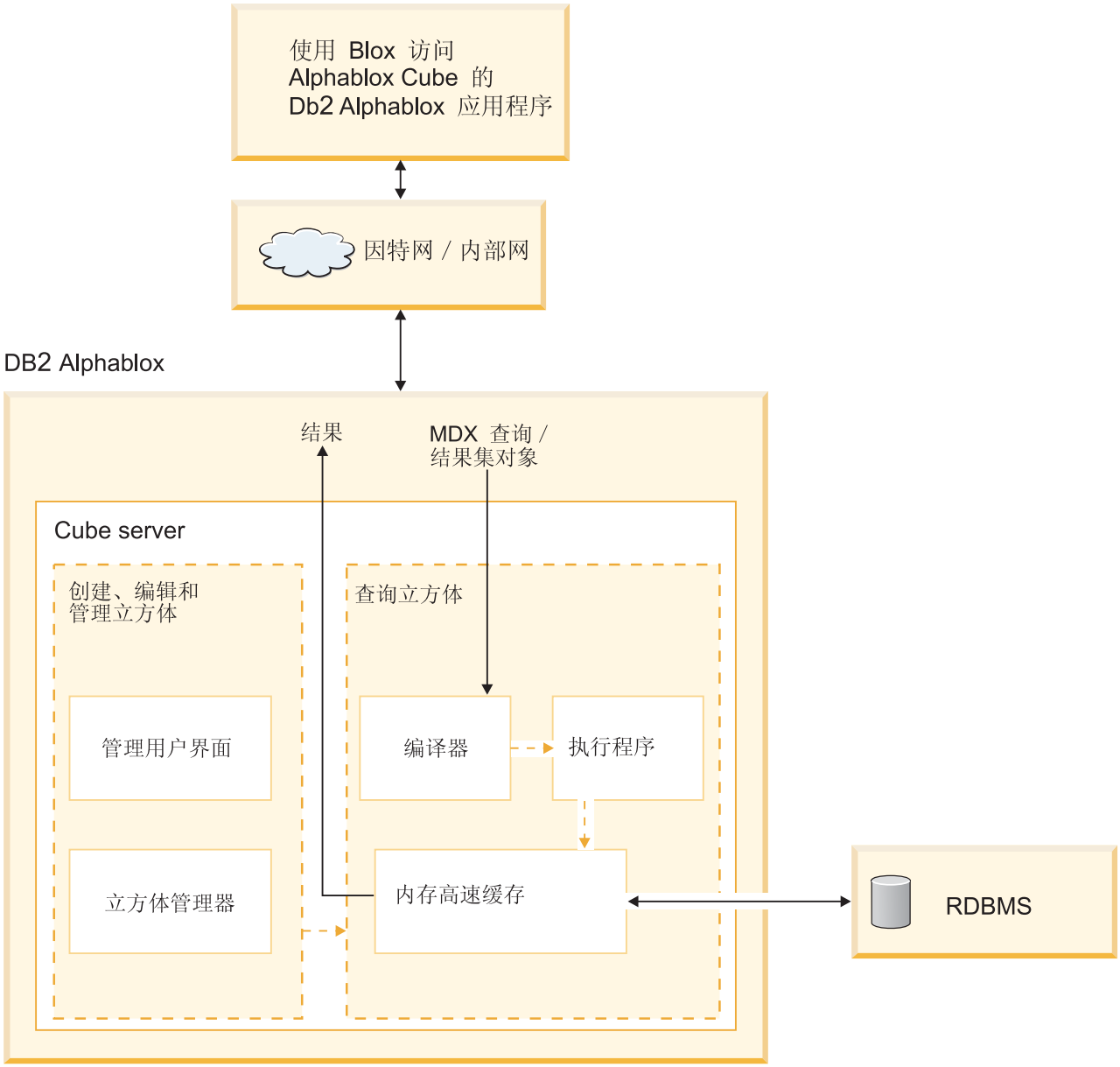
对于 DB2 Alphablox 应用程序，DB2 Alphablox 立方体仅仅是另一个数据源; 即，Blox 组件象使用任何其他受支持的数据源一样使用 DB2 Alphablox 立方体。DB2 Alphablox 立方体与其他数据源使用相同的 Blox 组件。例如，可以通过简单地更改查询值和数据源 DataBlox 参数值来将访问 DB2 Alphablox 立方体的应用程序更改为访问 DB2 OLAP Server 立方体。应用程序将按相同的方式执行; 它仅仅是访问不同的数据。可用来处理其他多维数据源和关系数据源的丰富 Blox 组件功能也同样适用于 DB2 OLAP Server 立方体。

DB2 Alphablox Cube Server 体系结构

DB2 Alphablox Cube Server 是高性能的可伸缩立方体构建引擎，设计它的目的是支持许多用户查询许多不同的立方体。它被设计成使您能够对数据仓库或数据集市数据库中存储的关系数据进行快速的多维访问。

DB2 Alphablox Cube Server 组件

DB2 Alphablox Cube Server 由数个组件组成。这些互补的组件提供了用于定义 DB2 Alphablox 立方体、管理它们以及对它们执行查询的基础结构。DB2 Alphablox Cube Server 的组件在 DB2 Alphablox 框架内工作，如下图所示。



管理用户界面

立方体管理员通过使用 DB2 Alphablox 管理页面来执行用于设置和管理 DB2 Alphablox 立方体的任务。在**管理**选项卡的**常规**部分之下，有两个用于管理 DB2 Alphablox 立方体的相关链接。选择 **DB2 Alphablox 立方体管理器**链接将打开一个对话框，以便有选择地指定要同时运行的最大立方体数并指定成员高速缓存目录的备用位置。在“运行时管理”之下，选择 **DB2 Alphablox 立方体**链接将打开一个用于启动和停止 DB2 Alphablox 立方体的对话框。

DB2 Alphablox 主页的**管理**选项卡之下的**立方体**链接可用来访问用于创建和编辑立方体定义的对话框（即定义 DB2 Alphablox 立方体的维、层次和量度的位置）。

DB2 Alphablox 用户必须是 administrators 组的成员才可以创建、查看或修改 DB2 Alphablox 立方体。有关使用 DB2 Alphablox 立方体管理用户界面的详细信息，请参阅第 13 页的第 3 章，『创建和修改立方体』和第 25 页的第 4 章，『维护立方体』。另外，DB2 Alphablox 管理页面还提供了联机帮助。

立方体管理器

立方体管理器是一个组件，它创建对象、执行验证检查、启动和停止 DB2 Alphablox 立方体以及对 DB2 Alphablox 立方体执行其他工作。DB2 Alphablox 控制台还接受由立方体管理器执行的命令。要获取立方体管理器控制台命令的描述，请参阅第 28 页的『控制台命令』。

内存高速缓存

DB2 Alphablox Cube Server 将计算结果存储在内存的高速缓存中。于是，所有访问 DB2 Alphablox 立方体的用户都共享这些存储的结果。在内部，每个立方体都被分解为较小的结果片段。这些片段中的每个片段都可能存储在立方体的内存高速缓存中。根据立方体结果所需的内存容量以及可供立方体使用的内存容量的不同，可能需要从高速缓存中除去一些条目。如果需要释放内存，则将从高速缓存中清除条目。对底层关系数据库进行的查询将填充高速缓存。如果对 DB2 Alphablox 立方体执行的查询所请求的数据尚未存储在高速缓存中，则将从底层数据库中检索该数据，并且，如果有必要的话，将从高速缓存中清除旧数据。系统自动执行所有这些高速缓存功能。

编译器

对 DB2 Alphablox 立方体发出的查询请求使用 MDX 查询语言。编译器对 MDX 查询进行语法分析、验证请求并生成方案以将结果返回给客户机应用程序。编译器利用为每个立方体存储的元数据来为每个请求生成经过优化的方案。

执行程序

执行程序运行由编译器生成的方案并从高速缓存中检索结果集。在生成结果之后，将把结果返回给请求那些结果的 DataBlox、GridBlox 或 PresentBlox。

MDX 到 SQL 查询转换

DB2 Alphablox 应用程序通过 MDX 查询来向立方体发出请求以获取结果。DB2 Alphablox Cube Server 对 MDX 查询进行处理，这将生成从 DB2 Alphablox 立方体中检索结果的方案。接着，DB2 Alphablox 立方体通过对底层关系数据库运行 SQL 查询来计算那些结果。这些 SQL 查询或者是在 MDX 查询发出之前已被运行并已存储在高速缓存中，或者是在 MDX 查询的运行时期间被执行。如果结果已存储在立方体的内存高速缓存中，则不需要再为该结果集运行该 SQL 查询。当 DB2 Alphablox 应用程序发

出 MDX 查询时，DB2 Alphablox Cube Server 自动发出任何必需的 SQL 查询。通常，需要多个 SQL 查询才能满足单个 MDX 请求。

模式要求

本主题描述 DB2 Alphablox 立方体所引用的底层数据库的要求。DB2 Alphablox 立方体必须引用受支持的关系数据库。*《DB2 Alphablox 安装指南》*描述了 DB2 Alphablox 所支持的数据库。数据库应该包含以*维模式*存储的*干净数据*。

干净数据

术语*干净数据*是指符合引用完整性规则的数据（无论 RDBMS 是否强制执行引用完整性）。干净数据也表示数据中的任何具有相同含义但却具有不同值的字段都已被转换为具有相同的值。例如，如果事务层数据中有一些记录将第二季度称为 *Q2*，而另一些记录却将将第二季度称为 *Quarter_2*，则必须对那些记录进行转换，以便使用唯一的值来标识第二季度。

维模式

在关系数据库中，维模式具有的干净数据存储结构使您能够方便地对该数据执行历史查询。通常，维模式可以具有下列其中一种形式:

- 单个表
- 星型模式
- 雪花模式
- 星型模式与雪花模式的组合

DB2 Alphablox 立方体的底层数据库必须仅包含一个事实表；不支持多个事实表模式。DB2 Alphablox 立方体中的每个维都必须具有单一层次结构。要了解有关模式的更多信息，请参阅第 7 页的『维模式』。

注: 如果数据库具有多个事实表或者不符合维模式，则可以在数据库中创建视图以便创建一个“虚拟的”单事实表维模式，从而与 DB2 Alphablox 立方体配合使用。

第 2 章 维模式设计

DB2 Alphablox Cube Server 要求底层数据库具有维模式。要正确地设置 DB2 Alphablox 立方体，管理员应该了解底层 RDBMS 中的数据。本主题说明维模式设计的概念、定义诸如星型模式和雪花模式之类的术语并说明数据库结构与立方体层次结构之间的关系。

维模式

数据库由一个或多个表组成，数据库中所有表之间的关系统称为数据库*模式*。虽然有许多不同的模式设计，但是用于查询历史数据的数据库通常被设置为具有维模式设计（通常是星型模式或雪花模式）。采用维模式既有许多历史方面的原因也有许多实践方面的原因，但是，它们在决策支持关系数据库方面的应用的增长是由两项主要的益处推动的:

- 能够形成用来应答业务问题的查询。通常，查询根据若干个业务维计算某些业绩量度。
- 在大部分 RDBMS 供应商使用的 SQL 语言中形成这些查询必需维模式。

维模式在物理上将用于量化业务的量度（也称为*事实*）与用于描述业务和对业务进行分类的描述性元素（也称为*维*）分隔开。DB2 Alphablox 立方体要求底层数据库使用维模式；即，在物理上必须将事实数据与维数据分隔开（至少位于不同的列中）。通常，维模式具有星型模式形式、雪花模式形式或者这两种模式的混合形式。尽管不是常见的情况，但维模式也可以具有单个表的形式，即事实和维仅仅是位于表的不同的列中。

注: 如果数据库不符合维模式，则可以在数据库中创建视图以创建一个“虚拟的”维模式以便与 DB2 Alphablox 立方体配合使用。

本主题描述星型模式和雪花模式以及在这些模式中表示业务层次结构的方式。

要彻底了解维模式设计及其所有分支的背景信息，请阅读由 Ralph Kimball 编著并由 John Wiley and Sons, Inc. 出版的 *The Data Warehouse Toolkit*。

星型模式和雪花模式

星型模式和雪花模式设计是用来将事实和维分隔到单独的表中的机制。雪花模式将层次结构的不同层次进一步分隔到单独的表中。在任何一种模式设计中，每个表都通过*主键*／*外键*关系与另一表相关。在关系数据库中，使用主键／外键关系来定义各个表之间的多对一关系。

主键

*主键*是表中的一个列或一组列，它们的值唯一地标识表中的一行。关系数据库设计成通过仅允许表中的一行具有给定的主键值来强制实施主键的唯一性。

外键

*外键*是表中的一个列或一组列，它们的值与另一个表中的主键值相对应。为了添加具有给定外键值的行，在相关的表中必须存在具有相同主键值的行。

在星型模式或雪花模式中，表之间的主键 / 外键关系（有时称为多对一关系）表示数据库中将相关的表连接到一起的路径。这些连接路径是形成对历史数据执行的查询的基础。要了解有关多对一关系的更多信息，请参阅第 9 页的『多对一关系』。

事实表

*事实表*是星型模式或雪花模式中的一个表，它存储用于量度业务（如销售量、商品成本或利润）的事实。事实表还包含指向维表的外键。这些外键使事实表中的每个数据行与其对应的维和层次相关。

在 DB2 Alphablox Cube Server 中，可以在新的或现有的立方体的立方体定义中指定 DB2 Alphablox 立方体的“量度”事实表，这些立方体显示在 DB2 Alphablox 管理页面的**管理**选项卡的**立方体**链接之下。要指定量度事实表，必须选择一个有效的模式和目录组合来填充可用表选项的列表，然后选择表或输入一个有效的表名。

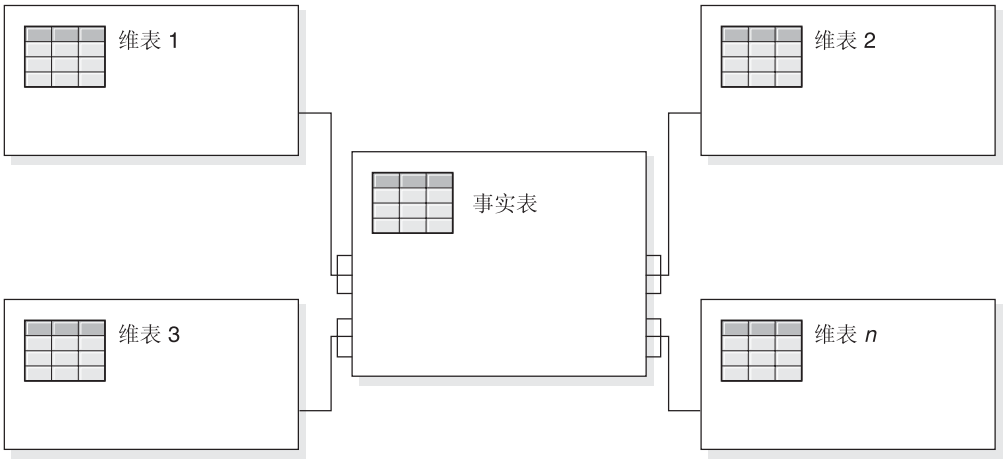
维表

*维表*是星型模式或雪花模式中的一个表，它存储用来描述维的各个方面的属性。例如，时间表存储时间的各个方面，如年份、季度、月份和天。事实表的外键引用多对一关系中的维表的主键。

星型模式

下图显示了具有单个事实表和 4 个维表的星型模式。星型模式可以具有任意数目的维表。用于连接表的链接末尾的多个分支指示了事实表与每个维表之间的多对一关系。

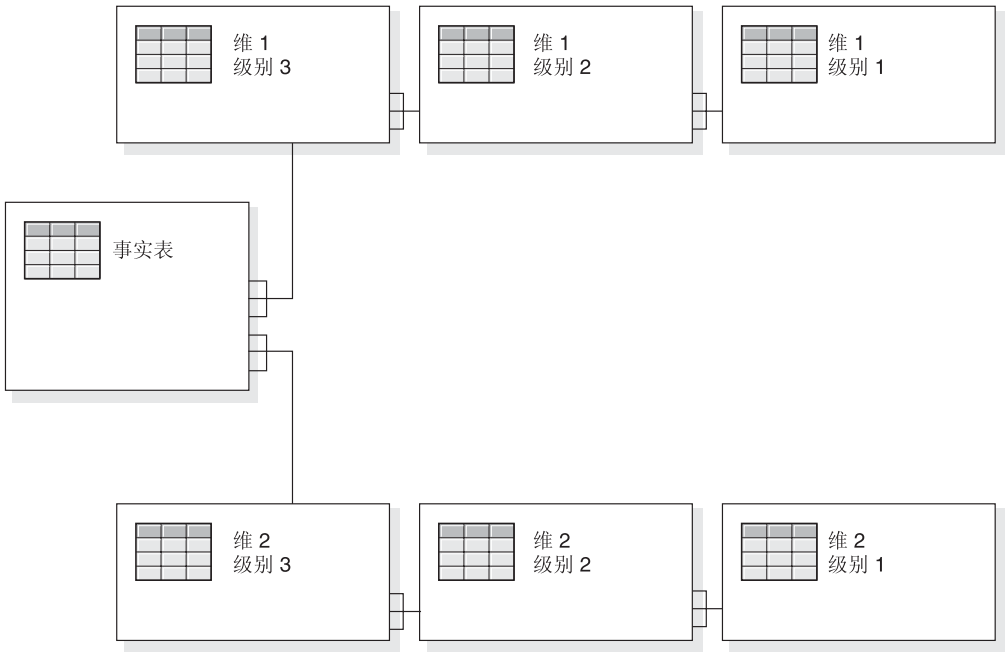
星型模式



雪花模式

下图显示了具有两个维的雪花模式，其中，每个维都具有 3 个层次。雪花模式可以具有任意数目的维，并且每个维可以具有任意数目的层次。

雪花模式



要了解有关维的不同层次如何形成层次结构的详细信息，请参阅『层次结构』。

层次结构

层次结构是一组相互之间具有多对一关系的层次，并且这一组层次共同构成维。在关系数据库中，层次结构的不同层次既可以存储在单个表中（如在星型模式中），也可以存储在单独的表中（如在雪花模式中）。

在 DB2 Alphablox Cube Server 中，支持以下层次结构类型:

- 平衡
- 未对齐
- 未平衡

多对一关系

多对一关系是指一个实体（通常是一个列或一组列）包含的值引用另一个具有唯一值的实体（一个列或一组列）。在关系数据库中，这些多对一关系通常是由外键／主键关系强制实施的，并且，这些关系通常是事实表与维表之间以及层次结构中的层次之间的关系。此关系通常用来描述分类或分组。例如，在具有表 *Region*、*State* 和 *City* 的地理模式中，在给定的地区有许多州，但没有任何州同时位于两个地区。同样，对于城市，一座城市仅位于一个州（同名但位于多个州的城市的处理方式必须略有不同）。关键在于每座城市都刚好位于一个州，但一个州可以有许多城市，因而符合“多对一”这个术语。

层次结构的不同元素（即层次）在子代层次与父代层次之间必须具有多对一关系，而无论该层次结构在物理上是以星型模式还是以雪花模式表示的；即，数据必须遵守这些关系。强制实施多对一关系所需的干净数据是维模式的一项重要特征。此外，这些关系使得有可能根据关系数据创建 DB2 Alphablox 立方体。

在定义 DB2® Alphablox 立方体时，用于定义层次结构的多对一关系变为维中的层次。您通过管理用户界面来输入此信息。要了解关于设置元数据以定义 DB2 Alphablox 立方体的详细信息，请参阅第 13 页的第 3 章，『创建和修改立方体』。

平衡及未平衡层次结构

当维具有递归层次结构时，则不需要创建任何层次。但需要指定任何必需的成员信息。

平衡层次结构

在平衡层次结构（**平衡／标准**）中，层次结构的分支都将下移至同一层次中，并且每个成员的父代都在紧邻该成员的上一层次中。平衡层次结构的一个常见示例是表示时间的层次结构，其中每个层次（year、quarter 和 month）的深度是一致的。DB2 Alphablox Cube Server 支持平衡层次结构。

未平衡层次结构

未平衡层次结构包括这样一些层次，它们有一个一致的父子关系但有一些在逻辑上不一致的层次。层次结构分支也可以有不一致的深度。机构图表是未平衡层次结构的一个示例，它显示机构中职员之间的报告关系。机构结构中的层次是不平衡的，层次结构中的某些分支比其他分支有更多的层次。

对于标准的未平衡层次结构（**未平衡／标准**），DB2 Alphablox Cube Server 将忽略跳过的层次，视它们不存在。标准部署层次结构使用层次结构中层次定义的关系，其中层次结构中的每个层次都被用作部署中的一项。支持将标准部署层次结构用于未平衡层次结构。使用层次结构中的层次时，每个层次都至少需要维表中的一列，而缺少的层次则包含 null 值。在 DB2 Alphablox 立方体中不支持未平衡的，且使用层次结构各层次之间固有的父子关系的递归部署层次结构。

DB2 Alphablox Cube Server 中也支持未平衡层次结构的递归变体（**未平衡／递归**）。当选择此层次结构类型时，还必须指定数据成员是隐藏还是可视的。缺省情况下，数据成员是隐藏的。使用递归层次结构时，任何成员都可以将数据放置在事实表中，而不只是放置在叶成员中。例如，如果事实表中 California 的销售值为 100，San Jose 的销售值为 15，而 Oakland 的销售值为 20，并且城市积累至州，则 California 的销售值将为 135。如果“数据成员”是可视的，则 California 成员将有一个值为 100 的子成员，名称类似于“California 数据”。如果“数据成员”是隐藏的，则在成员层次结构中，California 将只有两个子成员（San Jose 和 Oakland），但仍然有已积累的销售值 135（即仍包含隐藏值 100）。

在未平衡层次结构中，层次结构的较低层次中可以出现 null 值。父成员的子代始终在父代所在层次的下一层次中。在此层次结构中，层次不对其成员提供有意义的上下文 - 在本示例中，Washington DC 与 CA 在同一层次。机构图表可能是未平衡层次结构的一个较好的示例。

第 1 层	**第 2 层**	**第 3 层**
USA	CA	San Francisco
USA	CA	Los Angeles
USA	Washington DC	<NULL>
Vatican City	Vatican City	<NULL>

有关 null 值和层次键的更多信息，请参阅第 18 页的『创建层次』。

未对齐层次结构

在未对齐层次结构中，维的至少一个成员的父成员未处于紧邻成员的上一层次中。与未平衡层次结构类似，这些层次结构的分支也可下移至不同的层次中。

DB2 Alphablox Cube Server 支持使用未对齐层次结构。将忽略未对齐层次结构中跳过的层次，视它们不存在。对于未对齐层次结构，只支持标准部署层次结构。使用层次结构中的层次时，每个层次都至少需要维表中的一列，而缺少的层次将包含 null 值。

在未对齐层次结构中，任何层次列中都可以出现 null 值。将跳过成员名之间的空列值，所以父代可以有一个在父层次下有多个层次的子成员。以下示例中显示的 USA 的子代是 CA 和 Washington DC。在未对齐层次结构中，这些层次对其成员提供了有意义的上下文。因此，虽然 Washington DC 是 USA 的子代，但它包含在具有 San Francisco 和 Los Angeles 的 City 层次中。

Country	State	City
USA	CA	San Francisco
USA	CA	Los Angeles
USA	<NULL>	Washington DC
Vatican City	<NULL>	Vatican City

有关 null 值和层次键的更多信息，请参阅第 18 页的『创建层次』。

将关系模式映射为立方体

对于负责设计和构建 DB2 Alphablox 立方体的管理员来说，了解（至少在较高层次了解）关系数据库与 DB2 Alphablox 立方体之间的映射十分重要。了解此映射有助于确保在设计或创建 DB2 Alphablox 立方体时不会出错。由于立方体是由对底层关系数据库执行的查询填充的，因此，可以通过将立方体的查询结果与关系数据库的查询结果作比较来对立方体执行质量保证测试。

维、层次和属性

可以在 DB2 Alphablox 立方体中定义任意数目的维，并且，对于每个维，可以定义任意数目的层次。在典型的雪花模式中，每个层次都被规划化为单独的表，并且最详细的层次被事实表中的外键引用。DB2 Alphablox Cube Server 依靠这些不同的表之间的关系来创建立方体中的维。在定义 DB2 Alphablox 立方体时，必须提供有关模式的详细信息来作为 DB2 Alphablox 立方体定义的一部分。

对于每个维，可以指定维层次结构中的层次。每个维至少需要一个层次。层次用来指示层次结构内的位置。例如，在 Time 维中，您可以有 Year、Quarter、Month 和 Week 这几个层次。

“所有”层次是位于层次结构的顶层、具有单个成员的层次。“所有”层次中的这个成员也称为“所有”层次成员，当维中的层次对象以这个成员作为模型时，该成员表示它下面所有成员的聚集。对于大多数维，具有“所有”层次是有意义的，但对于 Scenario 维以及某些情况下的 time 维，您将不会想要显示“所有”层次。

对于维中的每个层次，必须定义一个层次键。层次键由一个或多个层次键表达式组成。将层次键表达式组合在一起，就可以唯一地标识层次中的每个成员。例如，城市层次的层次键可以由三个层次键表达式组成（<Country Name, State Name, City Name>），它也可以由单个表达式组成（例如，<city_id>）。如果未定义成员排序表达式，则层次键表达式的顺序可能会影响层次中的成员排序。注意，最好的习惯指示层次键表达式不应该是完全可空的（对于层次的每个成员，至少有一个层次键表达式应该不为空）。

注: 在 DB2 Alphablox 8.4 之前迁移 DB2 Alphablox Cube Server 中现有的立方体定义期间，将自动生成并添加层次键。生成的结构顶部级别的级别键将由一个与级别表达式匹配的级别键表达式构成。顶层之下的任何层次将有一些层次键表达式，这些表达式与该层次本身的层次表达式以及该层次的祖代的所有层次表达式对应。

量度

DB2 Alphablox 立方体的量度根据关系数据库中的事实表进行计算。当一个查询需要量度时，DB2 Alphablox Cube Server 将计算该查询中指定的每个成员的直接同代的值。例如，DB2 Alphablox Cube Server 将某一年份的销售量度计算成该年份中 12 个月份的销售量度总计。

注意，在定义量度的 SQL 表达式中，所有列名都被包含那些列的表限定，以防出现指代不清的问题（当不同的表包含同名的列时，就会发生这种问题）。因此，对用于量度的 SQL 表达式有几项限制:

1. 表达式中的第一个标记必须是量度表中的列。由于以下表达式以左括号开头，所以它是无效的:

   ```
   (store_sales - unit_sales) / store_cost
   ```

2. 表达式其余部分中的所有列都必须刚好存在于一个表中。
3. 表达式中的列一定不能是量度表中的任何外键列。

第 3 章 创建和修改立方体

管理员使用**管理**选项卡的**立方体**部分来定义 DB2 Alphablox 立方体。本主题描述创建 DB2 Alphablox 立方体时需要执行的步骤。

用于创建立方体的任务的核对表

本节提供定义 DB2 Alphablox 立方体时所需执行的任务的核对表以及对每个任务的简要描述。本章后面的内容提供了详细的任务指示信息。

任务		描述
1	了解底层数据库的模式	要定义 DB2 Alphablox 立方体，您必须了解用来构建立方体的关系数据库的模式。使用数据库工具来浏览数据库并确保您具有访问该数据库中的表名、列名、主键名和外键名的权限。要了解有关模式的信息，请参阅第 7 页的第 2 章，『维模式设计』。
2	确定立方体所需要的维、量度和层次	除了了解模式以外，您还必须了解 DB2 Alphablox 立方体的底层关系数据库中的数据。您必须了解要在立方体中定义的量度、那些量度在数据库中的存储位置以及每个维的层次结构的不同层次之间的关系。
3	第 14 页的『创建关系数据源』	为用来创建 DB2 Alphablox 立方体的底层关系数据库创建 DB2 Alphablox 数据源定义。
4	第 15 页的『定义立方体』	使用**立方体**管理用户界面来定义 DB2 Alphablox 立方体的属性。
5	第 15 页的『定义量度』	指定要在 DB2 Alphablox 立方体中量度的事实以及每个量度的关系事实表到 DB2 Alphablox 立方体映射。
6	第 16 页的『定义维』	指定 DB2 Alphablox 立方体中的每个维以及每个维的每个层次。定义关系表与 DB2 Alphablox 立方体维和层次之间的映射。

任务		描述
7	第 21 页的『创建 Alphablox Cube Server Adapter 数据源定义』	要查询 DB2 Alphablox 立方体，请定义通过 **Alphablox Cube Adapter** 多维驱动程序创建的数据源。
8	第 22 页的『指定和管理立方体资源』	输入 DB2 Alphablox 立方体连接限制、更新频率和其他管理参数。
9	第 24 页的『复审立方体』	确保输入的用于定义 DB2 Alphablox 立方体的信息没有任何错误。

创建关系数据源

DB2 Alphablox 立方体要求将底层关系数据源定义为 DB2 Alphablox 数据源。每个 DB2 Alphablox 立方体都必须引用关系数据源。该数据源必须引用具有维模式设计的关系数据库。要获取有关 DB2 Alphablox 立方体关系模式要求的描述，请参阅第 6 页的『模式要求』。有关维模式的讨论，请参阅第 7 页的第 2 章，『维模式设计』。

如果已经为关系数据库定义了数据源，则请跳至下一个主题（第 15 页的『定义立方体』）。有关数据源的更多信息，请参阅*《管理员指南》*。

要将关系数据库指定为 DB2 Alphablox 数据源，请执行下列步骤:

1. 作为 *admin* 用户或作为隶属于 administrators 组的用户登录到 DB2 Alphablox 主页。
2. 单击**管理**选项卡然后单击**数据源**链接。
3. 在“数据源”窗口中，单击位于数据源列表下面的**创建**按钮。
4. 从**适配器**菜单中，选择其中一个可用的关系驱动程序选项（例如，IBM DB2 JDBC 4 类驱动程序）。
5. 在**数据源名**字段中输入新数据源的名称。
6. 为**客户机主机名**、**端口号**、**SID** 和**数据库**字段添加适当的信息。当选择特定驱动程序选项时，将显示相关字段。
7. 输入**缺省用户名**和**缺省密码**。用户名和密码必须对该关系数据库有效。当 DB2 Alphablox 立方体访问关系数据库时，总是使用缺省用户名和密码。指定的数据库用户需要对该数据库具有读访问权。

   注: 当使用此数据源来填充 DB2 Alphablox 立方体时，将忽略**使用 DB2 Alphablox 用户名和密码**列表的值。使用访问控制表（ACL）来控制用户对 DB2 Alphablox 立方体的访问权。有关 ACL 的信息，请参阅*《管理员指南》*。
8. 当使用此数据源来填充 DB2 Alphablox 立方体时，忽略**最大行数**和**最大列数**的值。您仍然可以输入这些值，并且其他应用程序使用该数据源时将使用这些值，但是 DB2 Alphablox 立方体将忽略这些值。
9. 除非您想将 JDBC 日志记录信息写至 DB2 Alphablox 日志文件，否则，请将**启用 JDBC 跟踪**菜单设置为**否**。仅当您遇到问题并且需要调试问题原因时，才应该启用 JDBC 跟踪。
10. 单击**保存**按钮以保存该数据源。

定义立方体

要定义 DB2 Alphablox 立方体的一般属性:

1. 作为 *admin* 用户或作为隶属于 administrators 组的用户登录到 DB2 Alphablox 管理页面。
2. 单击**管理**选项卡，然后单击**立方体**链接。
3. 单击**创建**按钮。“立方体管理”对话框将显示在新的浏览器窗口中。
4. 在 **DB2 Alphablox 立方体名**字段中，为 DB2 Alphablox 立方体输入一个唯一名称。DB2 Alphablox 立方体名允许的字符是 A-Z、a-z、0-9、下划线（_）和空格。
5. 现在或稍后您作好启动此立方体的准备时选择 **DB2 Alphablox 立方体名**字段右边的**启用**复选框。当选择了**启用**时，无论服务器何时重新启动，此立方体都将自动启动。如果您正在处理立方体定义并且认为它不会正常运行，或者还不想允许别人访问它，则使此复选框处于未选取状态，以便以后再启用此立方体。
6. 在**关系数据源**菜单中，选择先前在第 14 页的『创建关系数据源』 中定义的关系数据源。如果列表为空，则尚未任何定义 DB2 Alphablox 关系数据源。
7. （可选）如果您需要限制对立方体的访问，则在**安全角色**字段中输入预定义的角色（在应用程序服务器或 DB2 Alphablox 中定义的)并选择“启用”复选框。
8. （可选）如果您正在使用 IBM DB2 UDB 作为数据源，并且您的数据源提供了 DB2 Cube Views 立方体，则**启用 DB2 Cube Views 设置**选项可用。如果您选择此选项，则 DB2 Cube Views 中的可用立方体定义可以用来指定 DB2 Alphablox 立方体。要使用此选项:
   a. 通过使用**立方体模型**菜单，选择立方体模型。
   b. 通过使用**立方体**菜单，选择立方体。
   c. 选择**使用业务名**或**使用对象名**单选按钮以指定在 DB2 Alphablox 立方体中定义对象时要使用的名称类型。

      使用业务名选项显示用户更容易理解的对象备用描述性名称，而**使用对象名**选项则显示已分配给物理对象的标注。
   d. 单击**导入立方体定义**按钮以导入立方体定义并预先填充 DB2 Alphablox 立方体中的量度和维。根据所导入的立方体定义的不同，DB2 Alphablox Cube Server 将尝试指定与 DB2 Cube Views 中的立方体最匹配的 DB2 Alphablox 立方体。单击**显示导入日志**按钮以查看日志，该日志包含与导入操作相关的信息和调试消息。
   e. 此时，您可以编辑导入的立方体量度和维（如下所述）以定制立方体，也可以选取**在启动、重建和编辑时导入立方体定义**选项。选择此选项时，每当启动、重建或者打开 DB2 Alphablox 立方体以进行编辑时，DB2 Alphablox 立方体都装入最新的 DB2 Cube Views 立方体定义。
9. 单击**确定**按钮以保存 DB2 Alphablox 立方体。

定义量度

必须对所有 DB2 Alphablox 立方体定义一个或多个量度。要获取量度的描述，请参阅第 12 页的『量度』。要定义 DB2 Alphablox 立方体中的量度:

1. 作为 *admin* 用户或作为隶属于 administrators 组的用户登录到 DB2 Alphablox 主页。
2. 单击**管理**选项卡，然后单击**立方体**链接。
3. 从立方体列表中选择 DB2 Alphablox 立方体，然后单击**编辑**按钮。所选立方体的 DB2 Alphablox **立方体管理**对话框将显示在新的 Web 浏览器窗口中。
4. 在立方体导航树中，单击**量度**节点。量度面板显示。
5. 在**量度事实表**字段中，输入事实表的标准名称，此名称是在底层关系数据库中定义的（例如 `CVSAMPLE.SALESFACT`）。此外，也可以从菜单中选择正确的模式、目录和表组合以便自动地插入事实表名。
6. 在指定了事实表之后，可以通过单击**创建新量度**按钮来创建新量度。将显示一组新选项。
7. 在**名称**字段中，通过输入新量度的名称替换“New Measure”。量度名允许的字符是 A-Z、a-z、0-9、下划线（_）和空格。

   注: 您输入的名称将在发送到 DB2 Alphablox 应用程序的结果集中出现，因此，请输入易于阅读并能描述其内容的名称。例如，如果量度计算的是一间商店的销售总额，则可以将该量度命名为 *Store Sales*。
8. 在**表达式**字段中，输入一个有效的表达式。可以使用“表达式构建器”工具来帮助您为列和函数输入正确的语法。虽然为常用函数（AVG、COUNT、MAX、MIN 和 SUM）提供了快捷按键，但是，您可以手工输入所需的任何有效函数。在生成要发送到底层数据库的 SQL 时，将使用这些函数来计算新量度。以下表达式示例定义了 COGS 量度:

   ```
   SUM(@col(CVSAMPLE.SALESFACT.COGS))
   ```
9. 单击**应用**按钮以将该量度添加到列表中。
10. 根据需要重复这些步骤以定义所需的任何其他量度。要删除量度，请单击导航树中的量度标注，然后单击树下方的**删除所选**按钮。
11. 当完成了修改 DB2 Alphablox 立方体定义时，请单击**确定**按钮，也可以继续定义维和层次。

定义维

必须输入信息以便为 DB2 Alphablox 立方体定义维、层次、连接、属性和其他信息。以下任务说明如何创建和修改维、事实表连接、维连接以及层次。

有关维和层次的描述，请参阅第 11 页的『维、层次和属性』。

创建维

要创建或编辑维:

1. 作为 *admin* 用户或作为隶属于 administrators 组的用户登录到 DB2 Alphablox 主页。
2. 单击**管理**选项卡，然后单击**立方体**链接。
3. 从立方体列表中选择 DB2 Alphablox 立方体，然后单击**编辑**按钮。所选立方体的 DB2 Alphablox **立方体管理**对话框将显示在新的 Web 浏览器窗口中。

4. 在左边的 DB2 Alphablox 立方体树中，单击**维**标注。在右边的面板中，将显示“创建维”按钮。要编辑现有的维，请单击维名，将显示现有的维定义。
5. 单击**创建新的维**按钮以创建新的维，或者从**维**列表中选择一个维以编辑现有的维。
6. 在**名称**文本框中，输入该维的名称。维名所允许的字符是 A-Z、a-z、0-9、下划线（_）和空格。此处指定的名称将出现在 DB2 Alphablox 立方体中。对于列示产品成员的维，您可以输入要作为维名显示的产品。
7. （可选）在**描述**字段中，输入维的简短描述。此描述仅仅是一个注释字段；它不会对维定义产生任何影响。
8. 指定**维类型**。缺省情况下，将选择**常规**。如果新的维与显示时间值相关，则选择**时间**。
9. （可选）在**缺省成员**字段中输入缺省成员。缺省情况下，将显示您输入的值。例如，在 Time 维中，通常是将当前年份作为缺省成员显示。

   注: 如果未指定缺省成员，且维不具有“所有”层次，则第一个可视成员将为缺省成员。
10. 从**层次结构类型**菜单中，选择所代表的层次结构的适当类型。选项包括**平衡 / 标准**、**未对齐 / 标准**、**未平衡 / 递归**或**未平衡 / 标准**。当选择了**未平衡 / 递归**时，“数据成员”选项允许您选择**显示**或**隐藏**（缺省值）。
11. 对于**“所有”层次**设置，请通过选择**具有“所有”层次**选项来表示您是否想要您的维显示“所有”层次。缺省情况下，已选择了此选项。您也可以指定**“所有”层次成员名**。例如，如果您具有 Product 维，则可以选择将所有层次成员名的缺省值设置为 `All Products`。如果您未在“所有”层次成员名中输入值，则缺省成员名将显示为“All Products”。

    注: 当**“所有”层次成员名**保留为空白时，将根据服务器语言环境转换“所有”。根据您想要在用户界面上显示的内容，可以通过指定**“所有”层次成员名**来覆盖此选项。
12. 单击**确定**按钮以保存该维。

在创建了新的维之后，就可以开始定义任何必需的事实表连接和维连接了。

创建事实表连接

对于星型模式或雪花模式，您需要在事实表和与事实表有直接关系的每个表之间定义事实表连接。如果您具有雪花模式，则必须为直接连接至事实表的每个表创建一个事实表连接。在雪花模式中，您还需要为与事实表没有直接关系的其他相关表创建维连接。

要创建或编辑维中的事实表连接，请执行下列步骤:

1. 在所选维之下单击**事实表连接**节点。
2. 要创建事实表连接，请单击**创建新连接**按钮。将显示连接指定面板。如果已存在事实表连接，则展开“事实表连接”文件夹并单击该连接。
3. 在**表达式**文本框中，输入指定了事实表连接的表达式。也可以使用表达式构建器来帮助您输入用来定义连接的表达式。示例:

   ```
   @col(qcc_fact.Week_Ending) = @col(qcc_time.Week_Ending)
   ```

4. 单击**应用**按钮以便在不关闭对话框的情况下应用并保存这些设置。单击**确定**按钮以保存层次定义并关闭“立方体管理”对话框。

创建维连接

维连接是维中与事实表没有直接关系的相关表之间的连接。维连接只能与雪花模式配合使用。

要在选择的维中创建或编辑维连接:

1. 在“立方体管理”对话框中，单击维的“连接”文件夹下方的**维连接**节点。
2. 要创建新的维连接，请单击显示的**创建新连接**按钮。维连接对话框显示。要编辑现有的维连接，请展开“维连接”文件夹并选择要编辑的连接。
3. 在**表达式**字段中，输入维连接表达式。也可以使用表达式构建器来帮助您输入用来定义连接的表达式。示例:

   ```
   @col(QCC_PRODUCTS.FAMILYID) = @col(QCC_PRODUCTFAMILIES.FAMILYID)
   ```

4. 单击**应用**按钮以便在不关闭“立方体管理”对话框的情况下应用并保存这些设置。单击**确定**按钮以保存更改并关闭“立方体管理”对话框。

创建层次

对于每个维，可以指定维层次结构中的层次。每个维至少需要一个层次。层次用来指示层次结构内的位置。例如，在 Time 维中，您可以有 Year、Quarter、Month 和 Week 这几个层次。

“所有”层次是位于层次结构的顶层、具有单个成员的层次。“所有”层次中的这个成员也称为“所有”层次成员，当维中的层次对象以这个成员作为模型时，该成员表示它下面所有成员的聚集。对于大多数维，具有“所有”层次是有意义的，但对于 Scenario 维以及某些情况下的 time 维，您将不会想要显示“所有”层次。

对于维中的每个层次，必须定义一个层次键。层次键由一个或多个层次键表达式组成。将层次键表达式组合在一起，就可以唯一地标识层次中的每个成员。例如，城市层次的层次键可以由三个层次键表达式组成（<Country Name, State Name, City Name>），它也可以由单个表达式组成（例如，<city_id>）。如果未定义成员排序表达式，则层次键表达式的顺序可能会影响层次中的成员排序。注意，最好的习惯指示层次键表达式不应该是完全可空的（对于层次的每个成员，至少有一个层次键表达式应该不为空）。

如果具有未平衡或未对齐的层次结构并且使用标准部署，则维表中可能有 null 值。假定有一个地理表，它包含以下各列:

Country	Region	State	City	City_ID
USA	West	California	San Jose	1
USA	East	<null>	Washington, D.C.	2
Canada	Quebec	<null>	Quebec	3
Canada	Quebec	<null>	Montreal	4
Canada	British Columbia	<null>	Vancouver	5

对类似这样的维建立模型时，需要创建四个层次并为每个层次指定一个唯一层次键。您最初的反应可能是创建下列四个层次:

- Country: 层次键 = Country; 层次表达式 = Country
- Region: 层次键 = Country, Region; 层次表达式 = Region
- State: 层次键 = State; 层次表达式 = State
- City: 层次键 = City_ID; 层次表达式 = City

在上面显示的数据中，有两个国家或地区，四个区域以及五个城市。但是，有多少个州呢？由于存在 null 值，乍看似乎只有一个州（California）。然而，DB2 Alphablox Cube Server 将为 null 值创建内部"哑元"州。例如，如果钻取到 Quebec 中，您将看到 Quebec 和 Montreal。对于最终用户来说，只有一个州，即 California。因为这种内部表示，所以 State 层次中有四个成员（California、<null>、<null> 和 <null>）。这将导致无法启动立方体，因为在 State 层次中有三个具有相同键的成员（三个 <null> 值）。

为了解决这种情况，需要将 State 层次的层次键更改为下列内容:

- State: 层次键 = Country, Region, State; 层次表达式 = State

那么四个 State 成员（一个常规成员以及三个哑元成员）的层次键将为:

- <USA, West, California>
- <USA, East, null>
- <Canada, Quebec, null>
- <Canada, British Columbia, null>

由于这些层次键是唯一的，所以立方体将正常启动。

需要注意的是，即使在运行时期间查询立方体时 <null> 成员不可视，求值为 <null> 的层次表达式值仍需要唯一键。

要在维中创建或编辑层次:

1. 要创建新层次，请单击该维下方的**层次**节点并单击**创建新层次**按钮。要编辑现有的层次，请打开"层次"文件夹并选择要编辑的层次。
2. 对于新层次，在**名称**字段中输入一个名称。出现的缺省名称是"New Level"。DB2 Alphablox 立方体名允许的字符是 A-Z、a-z、0-9、下划线（_）和空格。
3. 在**类型**菜单中，选择层次类型。缺省情况下，类型为 REGULAR。可选类型包括: REGULAR、TIME、UNKNOWN、TIME_YEARS、TIME_HALF_YEARS、TIME_QUARTERS、TIME_MONTHS、TIME_WEEKS、TIME_DAYS、TIME_HOURS、TIME_MINUTES、TIME_SECONDS 或 TIME_UNDEFINED。如果您正在使用 Time 维，则选择与时间相关的适当层次类型（例如，TIME_QUARTERS）。如果没有与时间相关的适当层次类型，则使用 TIME_UNDEFINED 层次类型。TIME 层次类型只能与常规维中的层次配合使用。与时间相关的 MDX 功能能否正常起作用取决于是否正确使用了与时间相关的层次类型。如果您未在维内使用 TIME_* 层次，则可以将 REGULAR、TIME 和 UNKNOWN 层次类型混合在一起使用。
4. 在**表达式**字段中，输入指定了层次的表达式。也可以使用表达式构建器来帮助您输入用来定义层次的表达式。表达式可能会造成您需要考虑的相关性能损失。在以下示例中，使用日期函数的 SQL 表达式用来创建 Year、Month 和 Week Ending 这三个新层次:

```
YEAR(week_ending_date)
MONTHNAME(week_ending_date)
week_ending_date
```

5. 在应用或保存这些更改之前，必须创建至少一个或多个唯一地标识层次的层次键（请参阅上面的描述）。
   a. 在导航树中的新层次（在应用或保存更改之前，显示为“New Level”）下面，单击**层次键**文件夹，然后单击**创建新的层次键**按钮。
   b. 手工输入层次键表达式或使用表达式构建器创建一个层次键表达式。
   c. 单击**保存**按钮以保存新的层次键。新的层次键将显示在导航树中的“层次键”文件夹之下。如果需要，则继续创建其他层次键。
6. 单击**应用**按钮以便在不关闭“立方体管理”对话框的情况下应用并保存这些设置，或单击**确定**按钮以保存您的更改并关闭“立方体管理”对话框。

设置层次顺序

如果您不想使用层次的缺省排序，则可以修改将显示的层次顺序。

要设置层次顺序:

1. 单击要修改的维下的**层次**节点。将显示**设置层次顺序**选项。
2. 要将某个层次在层次顺序列表中上移或下移，可在列表中选择该层次，然后单击**上移**或**下移**按钮。
3. 修改完层次顺序后，单击“保存”按钮。

创建和编辑属性

属性是一些可以对层次定义的属性，它们可用来提供关于该层次的成员的其他信息。例如，名为 Product 的成员可能有大小、颜色、成本或其他相关产品信息的属性。DB2 Alphablox Cube Server 支持的 MDX Properties() 函数可用来显示 DB2 Alphablox 应用程序中的属性。

要创建或编辑属性:

1. 要创建新属性，请单击显示在维下方的**属性**节点，然后单击**创建新属性**按钮。属性定义对话框显示。要编辑现有的属性，请单击要编辑的属性节点。
2. 在**表达式**文本框中，输入指定了属性的表达式。也可以使用表达式构建器来帮助您输入用来定义属性的表达式。示例:

   ```
   @col(FAMILY.FAMILYID)
   ```

3. 单击**应用**按钮以便在不关闭“立方体管理”对话框的情况下应用并保存这些设置。单击**确定**按钮以保存更改并关闭“立方体管理”对话框。

在层次内设置成员排序

缺省的成员排序按成员名进行。如果您不想使用缺省排序，则可以使用“成员排序”来修改一个层次中显示的成员的排序。如何您不指定不同的排序表达式，则层次将按成员名排序。

例如，返回的月份在缺省情况下按字母顺序排序，而您想要更改成员排序，以便月份按年月日顺序显示。

如果在 time 维内定义了如下月份:

```
MONTHNAME(week_ending_date)
```

并且它们按字母顺序显示，则您可以创建如下成员排序表达式:

```
MONTH(@COL(week_ending_date)
```

它会导致月份预期地按年月日顺序进行排序。

要修改层次中的成员排序:

1. 在要修改的层次下单击**成员排序**标注。
2. 单击**创建新的成员排序**按钮。
3. 使用**表达式构建器**来为您要使用的成员排序创建一个表达式，或者手工输入一个表达式。
4. 单击**应用**按钮以便在不关闭“立方体管理”窗口的情况下应用并保存这些设置，或单击**确定**按钮以保存您的更改并关闭窗口。

创建持久计算的成员

1. 作为 *admin* 用户或作为隶属于 administrators 组的用户登录到 DB2 Alphablox 主页。
2. 单击**管理**选项卡。
3. 单击**立方体**链接。
4. 从立方体列表中选择 DB2 Alphablox 立方体，然后单击**编辑**按钮。 所选立方体的“DB2 Alphablox 立方体管理”对话框将显示在新的浏览器窗口中。
5. 在所选立方体的导航树中，找到并单击**计算的成员**标注。
6. 单击**创建新的计算的成员**按钮。
7. 在**成员名**字段中，输入有效的成员名，它包括维名和成员名（*DimensionName*].[*MemberName*] ）。 例如，要添加新的计算量度以表示库存储备，则计算的成员名可以为“[Measures].[CalculatedCost]”。
8. 在“MDX 表达式”字段中，输入一个用于指定新成员值的有效 MDX 表达式。 例如，对于新的 [Measures].[CalculatedCost] 计算的成员，有效的 MDX 表达式可能类似于以下内容:

   ```
   [Measures].[Sales Amount]-[Measures].[Profit Amount]
   ```

9. 选择一个整数作为**求解顺序**值。 求解顺序值必须是整数值（正数、零或负数），应按照该顺序来对指定的计算成员求值。 求解顺序值是按彼此间的相对顺序进行求值的，负数值先于零或正数值进行计算。

创建 Alphablox Cube Server Adapter 数据源定义

在可以查询 DB2 Alphablox 立方体之前，必须定义使用 **Alphablox Cube Server Adapter** 的 DB2 Alphablox 数据源。可以从多个应用程序中使用单个数据源来访问多个 DB2 Alphablox 立方体。被访问的立方体是由 Alphablox 应用程序使用的 MDX 查询的 FROM 子句确定的。要创建 DB2 Alphablox Cube Server Adapter 数据源，请执行下列步骤。

1. 作为 *admin* 用户或作为隶属于 administrators 组的用户登录到 DB2 Alphablox 主页。
2. 单击**管理**选项卡然后单击**数据源**链接。

3. 单击**创建**按钮。
4. 从**适配器**菜单中，选择名为 **Alphablox Cube Server Adapter** 的适配器。
5. 在**数据源名**文本框中输入名称。
6. （可选）在**描述**文本框中输入描述。
7. 在**最大行数**和**最大列数**文本框中指定数值。这些值限制了对通过此数据源输入的查询返回的行数或列数。缺省值是 1000。
8. 单击**保存**按钮以保存该数据源。

指定和管理立方体资源

对于每个 DB2 Alphablox 立方体，可以定义一个时间表来根据底层数据库刷新该立方体的数据。也可以为每个立方体设置若干个调整参数。

定义刷新时间表

当用于 DB2 Alphablox 立方体的关系数据库中的底层数据更改后，DB2 Alphablox 立方体中高速缓存的任何数据都可能会变为旧数据。当数据变旧时，您应该重建该立方体以确保从该 DB2 Alphablox 立方体得到的应答对于底层数据库来说是正确的。您可以通过停止并重新启动 DB2 Alphablox 立方体或使用 `REBUILD CUBE <cube_name>` 控制台命令来手工重建该立方体，这将重建维并清空内存高速缓存。另外，如果尚未更改维，但是已将新的或更改过的数据添加到数据库中，则可以手工地使用 `EMPTYCACHE <cube_name>` 控制台命令来仅将内存高速缓存清空。

如果底层数据库以定期的并且可预测的时间间隔更新，则安排对引用该数据库的 DB2 Alphablox 立方体进行定期更新可能会有意义。例如，如果该数据库每晚 9:00 PM 被更新，则您可能想在每天早上 3:00 AM 重建 DB2 Alphablox 立方体。

要将 DB2 Alphablox 立方体配置为定期地重建它自己，请执行下列步骤:

1. 作为 *admin* 用户或作为隶属于 administrators 组的用户登录到 DB2 Alphablox 主页。
2. 单击**管理**选项卡。
3. 单击**立方体**链接。
4. 从立方体列表中选择 DB2 Alphablox 立方体，然后单击**编辑**按钮。所选立方体的 DB2 Alphablox **立方体管理**对话框将显示在新的 Web 页面窗口中。
5. 在左边的立方体导航树中，单击**时间表**标注。安排时间表面板显示。
6. 选取**刷新间隔**框以启用已安排的 DB2 Alphablox 立方体重建。
7. 通过单击期望的按钮并修改相应的时间段来设置刷新时间间隔。例如，要将 DB2 Alphablox 立方体设置为在每天 3:00 AM 进行重建，请选择第二个按钮并输入时间 3:00 AM。
8. 单击**保存**按钮以更新 DB2 Alphablox 立方体定义。

设置调整参数

对于每个 DB2 Alphablox 立方体，可以设置若干个用于资源管理的调整参数。要完成此任务，请执行下列步骤:

1. 作为 *admin* 用户或作为隶属于 administrators 组的用户登录到 DB2 Alphablox 主页。
2. 单击**管理**选项卡。

3. 单击**立方体**链接。
4. 从立方体列表中选择 DB2 Alphablox 立方体，然后单击**编辑**按钮。所选立方体的 DB2 Alphablox **立方体管理**对话框将显示在新的 Web 页面窗口中。
5. 在左边的立方体导航树中，单击**调整**节点以打开调整面板。
6. 根据系统和用户要求设置调整参数选项。下表描述了可用选项。要了解有关这些调整参数和其他调整参数的详细信息，请参阅第 29 页的『调整立方体』。

调整参数	描述
最大并发连接数	对此 DB2 Alphablox 立方体建立的并发连接的最大数目。仅当所有连接同时执行查询时才会达到此限制。达到此限制后，新连接必须等待空闲的连接。
数据源连接池	通过在“数据源连接池”组中选择**启用连接池**复选框，可以启动 DB2 Alphablox 立方体的连接池。当启用了连接池时，还应该指定可以连接至底层关系数据库的最大持续连接数。**最大持续连接数**的缺省值是 10。达到指定的连接数后，新连接必须等待空闲的数据库连接。当使用此限制时，一旦打开每个连接，这些连接就将保持处于打开状态（连接数最高可达指定的最大持续连接数）以供其他 SQL 查询使用。当未选择“启用连接池”时，每个查询使用单独的连接，然后关闭该连接。 **注**: 尽管可以在 DB2 Alphablox Cube Server 中使用此设置来获取连接池，但此设置主要用于连接池不可用的 Apache Tomcat 3.2.4 上的 DB2 Alphablox 安装。对于 WebSphere 和 WebLogic 服务器上的 DB2 Alphablox 安装，您应使用应用程序服务器的连接池功能。要在 WebSphere 或 WebLogic 服务器上使用连接池，当定义 DB2 Alphablox 关系数据源定义时，需要使用 Application Server Adapter 选项。有关用受支持的数据源配置连接池的详细信息，请参阅您的应用程序服务器文档。
数据高速缓存	缺省情况下，对与选择的 DB2 Alphablox 立方体相关的数据高速缓存选择了**不受限制**选项。要限制所选立方体数据高速缓存的大小，您可以取消选择**不受限制**选项，然后指定要存储在 DB2 Alphablox Cube Server 的内存数据高速缓存中的最大数据高速缓存大小（行数）。达到此限制后，将把高速缓存中最近最少使用的查询结果从高速缓存中清除，以便腾出空间来供新行使用。 （可选）可以在**高速缓存种子查询（可选）**字段中指定 MDX 查询。缺省情况下，未指定种子查询。要启用可选查询，您还必须选择**启用**选项。 当启动或重建立方体时，将执行您在此文本框中输入的查询。此查询将使用一组初始结果来填充 DB2 Alphablox Cube Server 内存高速缓存。这些结果使用从底层数据库中检索到的数据来作为高速缓存的*种子*。如果后续 DB2 Alphablox 立方体查询只需要已存在于 B2 Alphablox Cube Server 高速缓存中的数据，那么该查询将直接从高速缓存中获取结果，这就避免了对底层数据库执行其他的 SQL 查询，从而缩短了响应时间。MDX 查询的 FROM 子句中引用的立方体名必须是先前在 **DB2 Alphablox 立方体名**文本框中定义的名称。

调整参数	描述
成员高速缓存	缺省情况下，已选择**静态（所有成员将装入内存）**选项，所有成员都将装入成员高速缓存中。（可选）您可以选择**动态（成员将在需要时装入内存）**选项，成员将在需要时装入成员高速缓存中。如果选择了此动态成员高速缓存选项，则缺省最大大小（成员数）设置为 100,000。您可以选择另一个非零值，或选择**不受限制**选项，以在需要时将所有成员装入数据高速缓存内存中。

7. 单击**保存**按钮以更新 DB2 Alphablox 立方体定义。

复审立方体

通常，在创建 DB2 Alphablox 立方体之后，花费几分钟的时间来确保正确定义了量度、维和层次是很值得的。如果您找到任何错误，可以很容易地更正它们。要对 DB2 Alphablox 立方体执行检查，请执行下列步骤:

1. 作为 *admin* 用户或作为隶属于 administrators 组的用户登录到 DB2 Alphablox 主页。
2. 单击**管理**选项卡。
3. 单击**立方体**链接。
4. 从立方体列表中选择 DB2 Alphablox 立方体，然后单击**编辑**按钮。显示了“编辑 DB2 Alphablox 立方体常规”选项卡的页面出现。
5. 请验证**关系数据源**文本框中指定的数据源是否引用了期望的关系数据库。您可能需要检查**数据源**管理页面上的数据源设置。
6. 在尝试启动 DB2 Alphablox 立方体之前，请验证是否从 **Alphablox 立方体名**文本框旁边选择了**启用**。如果未启用该立方体，则您在尝试启动该立方体时将看到一条错误消息。
7. 验证是否已正确地创建了量度:
   a. 单击**量度**节点以验证**量度事实表**文本框中指定的表是否是关系模式中正确的表、名称的拼写是否正确以及该名称是否是标准名称。
   b. 对于每个已定义的量度，检查是否已在**表达式**文本框中指定了期望的聚集。
8. 验证是否已正确地定义了所有所需的维以及名称是否正确。对于每个维，检查下列各项:
   a. 验证是否已添加了任何必需的事实表连接和维连接，并验证表达式是否正确。
   b. 验证是否正确地指定了层次，并且层次的出现顺序是否正确。第一层应该是最概略的层次，随后的每个层次都应该是层次结构中相邻的下一个层次。例如，如果 *Time* 维的层次结构是 *Year*、*Month* 和 *Day*，则 *Year* 应该位于第一层，接着是 *Month*，最后是 *Day*。
   c. 验证已定义的任何属性是否正确（包括期望的名称和表达式）。
9. 单击**时间表**选项卡并验证是否所有设置都符合您的要求。
10. 单击**调整**选项卡并验证是否所有设置都符合您的要求。

在检查 DB2 Alphablox 立方体完成之后，就可以启动该立方体了。要了解有关启动 DB2 Alphablox 立方体的详细信息，请参阅第 25 页的『启动、停止和重建立方体』。

第 4 章 维护立方体

DB2 Alphablox Cube Server 提供了对 DB2 Alphablox 立方体执行管理任务的功能。这些任务是通过 DB2 Alphablox 管理用户界面或通过控制台执行的。

启动、停止和重建立方体

您需要对 DB2 Alphablox 立方体执行的最常见管理任务是启动、停止和重建立方体。

启动 DB2 Alphablox 立方体

必须启动 DB2 Alphablox 立方体才能使其可用于查询。可以从 DB2 Alphablox 主页中启动立方体，也可以从控制台窗口的命令行启动它。启动立方体时，DB2 Alphablox Cube Server 将对底层关系数据库运行查询。这些查询的结果将用来把维成员装入到立方体的内存高速缓存中。可以将高速缓存种子 MDX 查询指定为 DB2 Alphablox 立方体定义的组成部分，该查询用来预先计算一些要存储在立方体高速缓存中的结果。如果指定了 MDX 查询，则在启动时，DB2 Alphablox Cube Server 将对 DB2 Alphablox 立方体运行 MDX 查询以使用该 MDX 查询返回的量度值来填充高速缓存。

从 DB2 Alphablox 管理页面启动立方体

要从 DB2 Alphablox 管理页面启动 DB2 Alphablox 立方体:

1. 作为隶属于 administrators 组的用户登录到 DB2 Alphablox 管理页面。
2. 单击**管理**选项卡。将打开“常规”页。
3. 在“运行时管理”部分下面，单击**立方体**链接。
4. 从 **DB2 Alphablox 立方体**列表中选择要启动的 DB2 Alphablox 立方体。
5. 要查看 DB2 Alphablox 立方体的当前状态，请单击**详细信息**按钮。
6. 单击**启动**按钮。当 DB2 Alphablox 立方体完成启动操作后，状态字段将显示**正在运行**。

从控制台窗口启动立方体

要从控制台窗口中启动 DB2 Alphablox 立方体，请执行下列操作。

1. 如果 DB2 Alphablox 尚未运行，则启动它。有关启动 DB2 Alphablox 的详细信息，请参阅*《管理员指南》*。
2. 在控制台窗口中，输入以下命令:

   ```
   start cube cubeName
   ```

   其中 *cubeName* 是要启动的 DB2 Alphablox 立方体的名称。当在 Web 浏览器中使用 DB2 Alphablox 管理页面时，也可以通过选择**管理 > 常规 > 启动控制台会话**来打开控制台窗口。

当立方体未启动时进行故障诊断

如果 DB2 Alphablox 立方体无法启动，则会显示一条错误消息，该消息可以帮助您确定问题的原因。在对问题进行故障诊断时，下列记录工具可以提供更多信息:

- 检查 DB2 Alphablox 日志文件。
- 在控制台上，通过在控制台窗口中输入以下命令来将消息层次提高到 DEBUG:

  ```
  report debug
  ```
- 在 DB2 Alphablox 关系数据源中启用 JDBC 跟踪。

要了解关于启用任何这些日志记录选项的信息，请参阅《*管理员指南*》。

下表显示了一些可能会导致启动操作失败的常见情况并列示了建议的问题更正操作。在确定问题后，请更正问题并再次尝试启动 DB2 Alphablox 立方体。

错误	描述
“确保启用了立方体”	检查是否已启用了该立方体。在命令行上，输入以下命令来了解是否已启用了该立方体: `show cube cubeName` 要启用 DB2 Alphablox 立方体，在**立方体**用户界面的**常规**选项卡中，选择 **DB2 Alphablox 立方体名**字段旁边的**启用**。
连接至底层数据库时发生错误。	各种问题都可能会导致连接错误。您通常需要检查下列常见情况: • 检查关系数据源是否具有正确的连接信息。 • 检查关系数据源是否具有有效的非空用户名和密码。 • 确保数据库可供连接。
底层数据库返回语法错误。	关系数据库返回语法错误时，通常是立方体的定义有错误。例如，如果语法错误指示找不到某个列，则请检查维定义以确保指定的列名和表名与数据库中的列名和表名完全相同。

停止 DB2 Alphablox 立方体

停止 DB2 Alphablox 立方体的操作将使该立方体不可用于查询，并且将从该立方体的内存高速缓存中除去所有条目并从该立方体的轮廓中除去所有维成员。

从 DB2 Alphablox 管理页面停止立方体

要通过 DB2 Alphablox 主页来停止 DB2 Alphablox 立方体:

1. 作为 *admin* 用户或作为隶属于 administrators 组的用户登录到 DB2 Alphablox 管理页面。
2. 单击**管理**选项卡。**常规**页面显示。
3. 在**运行时管理**部分下面，单击 **DB2 Alphablox 立方体**链接。
4. 从 **DB2 Alphablox 立方体**列表中选择要停止的 Alphablox 立方体。
5. 要查看 DB2 Alphablox 立方体的当前状态，请单击**详细信息**按钮。
6. 单击**停止**按钮。当 DB2 Alphablox 立方体完成关闭操作后，状态字段将显示**已停止**。

从控制台窗口停止立方体

要从控制台窗口中停止 DB2 Alphablox 立方体，请输入以下命令:

```
stop cube cubeName
```

其中 *cubeName* 是要停止的 DB2 Alphablox 立方体的名称。当在 Web 浏览器中使用 DB2 Alphablox 管理页面时，也可以通过选择**管理 > 常规 > 启动控制台会话**来打开控制台窗口。

注: 在任何正在执行的查询完成之前，DB2 Alphablox 立方体不会停止。

重建 DB2 Alphablox 立方体

当底层数据库中的数据（包括维数据）更改时，您应该重建或重新启动 DB2 Alphablox 立方体。更改立方体定义后，必须重建或重新启动立方体（或者如果配置了刷新时间间隔，则等待下一个刷新时间间隔）以使更改能够对查询产生作用。

在重建操作期间，立方体不可用于查询；新查询将等待并在重建操作完成后执行。重建操作将等待到任何正在运行的查询完成后才启动操作。维的大小以及根据底层数据库填充维的查询的性能确定了操作的持续时间。

要重建 DB2 Alphablox 立方体，请在控制台窗口中输入以下命令:

```
rebuild cube cubeName
```

其中 *cubeName* 是要重建的 DB2 Alphablox 立方体的名称。当在 Web 浏览器中使用 DB2 Alphablox 管理页面时，也可以通过单击**管理 > 常规 > 启动控制台会话**来打开控制台窗口。

如果维数据未更改但事实数据已更改（例如，已将上个季度的销售数据添加到数据库中），则只能清空内存高速缓存的内容。要清空高速缓存中的所有条目但保持维成员不变，请从控制台窗口中输入以下命令:

```
emptycache cube cubeName
```

管理策略

在定义并启动 DB2 Alphablox 立方体之后，只有在发生下列其中一种情况时才需要执行维护任务:

- 底层数据库中的数据已更改。
- 立方体定义已更改。

由于 DB2 Alphablox 立方体位于内存中，所以不需要管理磁盘空间。有一些内存注意事项，但是那些注意事项通常并不要求您执行日常管理任务。要了解关于内存问题的信息，请参阅 第 32 页的『DB2 Alphablox 立方体内存注意事项』。

您必须了解底层关系数据库的操作环境。底层数据库的管理方式对 DB2 Alphablox 立方体有着重要的影响。

了解数据库环境

每当 DB2 Alphablox 立方体的底层数据库中的数据发生更改时，DB2 Alphablox 立方体中的数据可能不会随底层的关系数据更改进行同步更改（或不是最新的）。DB2 Alphablox 立方体通过查询底层数据库来获取数据。当查询向 DB2 Alphablox 立方体请求获取数据时，DB2 Alphablox Cube Server 将检查结果是否位于它的内存数据高速缓存中。如果结果位于内存数据高速缓存中，则它们立即可供应用程序使用，从而使响应时间较短。虽然结果最初是从底层数据库中检索到的，但是那些结果是在过去的某

个时间检索到的。如果底层数据源的数据未更改，则没有问题。如果底层数据库中的数据在高速缓存条目生成之后并在查询请求获取结果之前发生更改，则结果将不匹配。

此外，如果在数据库中插入、更新或删除了 DB2 Alphablox 立方体中的某些成员，则 DB2 Alphablox 立方体提供的结果就不会反映维的真实状态。对 DB2 Alphablox 立方体执行的新查询所得到的结果可能仍与底层数据库中的结果相匹配，但也可能不匹配。这完全取决于在数据库中更改的值、Alphablox 立方体的内存高速缓存中存储的内容以及查询所请求的数据。

由于当底层数据库中的数据更改时没有可靠的方法来了解 DB2 Alphablox 立方体是否仍有效并且是否是最新的，所以最安全的做法是重建立方体。因此，了解底层数据库何时更改以及如何更改十分关键。

例如，如果您知道数据库永远不更改，则永远不需要重建 DB2 Alphablox 立方体。如果数据库仅将新数据添加到未在立方体中定义的数据库部件，则可能不需要进行重建。

如果每晚都使用对所有部件进行的潜在更改来更新数据库，则可能需要在每晚完成数据库更新后重建 DB2 Alphablox 立方体。您对数据库的操作环境了解得越多，您预测的 DB2 Alphablox 立方体数据变旧时间就越准确。

安排定期更新

根据有计划的定期时间表更新数据仓库和数据集市数据库是非常常见的情况。根据该时间表，可以安排对 DB2 Alphablox 立方体进行定期更新。可以使用 `REBUILD CUBE` 或 `EMPTYCACHE CUBE` 控制台命令来手工执行更新。此外，可以为每个 DB2 Alphablox 立方体设置自动重建时间表。要了解有关设置自动时间表的详细信息，请参阅第 22 页的『定义刷新时间表』。

安排对 DB2 Alphablox 立方体进行更新并没有最佳的方法。了解关系数据库中发生的情况非常重要。了解用户团体的喜好和需求也同样重要。根据立方体及其底层数据库大小的不同，重建 DB2 Alphablox 立方体可能要花费一些时间。通常，安排的最佳重建时间是在很少用户或没有用户使用系统的深夜。并且，特别是当重建操作需要花费很长时间的情况下，请确保用户了解立方体在那些时间内将不可用。

控制台命令

可以从 DB2 Alphablox 控制台窗口中执行大部分的立方体管理任务。要访问控制台，请单击**管理**选项卡，然后打开**常规**页面并单击**启动控制台会话**链接，或者使用在您启动 DB2 Alphablox 时打开的 DB2 Alphablox 控制台窗口。下表列示了立方体命令以及每个命令的功能描述。

命令语法

描述

delete cube *cubeName*

删除立方体及其整个定义。

disable cube *cubeName*

将立方体设置为“禁用”状态。已禁用的立方体在被启用之前无法启动，因此不会在 DB2 Alphablox 启动时自动启动。在禁用立方体之前，应该先将其停止。

emptycache cube *cubeName*

从立方体的内存高速缓存中除去所有条目。在清空高速缓存之后，对 DB2 Alphablox 立方体执行的查询就必须从底层数据库中检索结果。如果底层数据库已更改，请使用此命令以确保从 DB2 Alphablox 立方体中检索到的结果与存储在数据库中的数据相同。注意，EMPTYCACHE 命令不会重建立方体的维轮廓。要重建维轮廓，请使用 REBUILD 命令或者停止然后启动立方体。

enable cube *cubeName*

将立方体设置为“启用”状态。必须先启用立方体才能启动它。已启用的立方体将在 DB2 Alphablox 启动时自动启动。

rebuild cube *cubeName*

首先从内存高速缓存中除去所有维的成员名以及所有量度；然后查询底层数据库以重新填充所有维的维成员名。如果在立方体定义中指定了初始 MDX 高速缓存种子查询，则将执行该查询以填充高速缓存。

show cube*cubeName*

显示立方体的当前状态。立方体状态可以是:

- 已禁用
- 已停止
- 正在启动
- 正在运行

要显示所有已定义的 DB2 Alphablox 立方体的状态，请输入以下命令:

```
show cube cubeName
```

start cube *cubeName*

启动立方体并使其可用于查询。当立方体启动时，它将查询底层数据库以填充维成员，并且将运行 MDX 高速缓存种子查询（如果在立方体定义中指定了 MDX 高速缓存种子查询的话）。

stop cube *cubeName*

停止正在运行的立方体。立方体停止后，它变为不可用于查询，并且将从内存高速缓存中除去维成员和量度。

修改立方体

您随时可以更改 DB2 Alphablox 立方体定义的任何部分。对已停止的立方体所作的更改将立即得到应用。对正在运行的立方体所作的更改将立即被保存到立方体定义中，但在通过控制台或通过已安排的刷新来重建或重新启动该立方体之前，不会将那些更改应用于正在运行的立方体。

使用**立方体**管理页面来修改 DB2 Alphablox 立方体，这与创建它时采用的方法完全相同。您可以更新立方体定义的任何部分并进行保存。要了解有关如何在用户界面的每个部分中输入定义的详细信息，请参阅第 13 页的第 3 章，『创建和修改立方体』。

调整立方体

可以使用许多管理控件来调整和配置 DB2 Alphablox 立方体。由于 DB2 Alphablox 立方体在内存中运行，并且，它可能会发展到使用大量的内存，所以，您应该了解一些内存注意事项。

调整控件

使用本节描述的控件来控制 DB2 Alphablox 立方体的资源。

连接和高速缓存大小限制

通过打开“立方体管理”对话框并单击立方体导航树中的**调整**标注，可以在**立方体**页面上为每个已定义的 DB2 Alphablox 立方体指定连接限制和高速缓存大小限制。

最大并发连接数:

当有许多用户同时查询 DB2 Alphablox 立方体时，运行 DB2 Alphablox 的计算机上的机器资源消耗速度可能比只有几个用户时的消耗速度快。但是，请记住，查询必须在完全*相同的时间*执行，因此存在资源争用情况。即使有许多用户同时进行连接，发生这种情况的频率也不会很高。如果系统上存在此问题，则可以限制每个 DB2 Alphablox 立方体所允许的连接数。

所使用的资源量完全依赖于所发出的查询的类型。许多查询使用非常少量的机器资源，但某些长时间运行的查询可能会消耗大量的资源。

要调整 DB2 Alphablox 立方体的最大并发连接数:

1. 作为 *admin* 用户或作为隶属于 administrators 组的用户登录到 DB2 Alphablox 主页。
2. 单击**管理**选项卡。
3. 单击**立方体**链接。
4. 从立方体列表中选择 DB2 Alphablox 立方体，然后单击**编辑**按钮。所选立方体的 DB2 Alphablox **立方体管理**对话框将显示在新的 Web 页面窗口中。
5. 单击**调整**选项卡。
6. 选取要设置的任何限制的框并输入相应的数值。
7. 单击**保存**按钮以将限制保存到 DB2 Alphablox 立方体定义中。

数据源连接池:

通过在 DB2 Alphablox 立方体定义“调整”部分的“数据源连接池”组中选择**启用连接池**复选框，可以启用 DB2 Alphablox 立方体的连接池。当启用了连接池时，还必须指定可以连接至底层关系数据库的最大持续连接数。**最大持续连接数**的缺省值是 10。达到指定的连接数后，新连接必须等待空闲的数据库连接。当使用此限制时，一旦打开每个连接，这些连接就将保持处于打开状态（连接数最高可达指定的最大持续连接数）以供其他 SQL 查询使用。未选取**启用连接池**时，则 DB2 Alphablox Cube Server 发送至数据库的每个查询都将打开新连接并且在结果返回后关闭该连接。即，无论任何其他连接具有何种状态，都将打开新连接。既不共享连接，也不会留下空闲的连接。

注: 尽管可以在 DB2 Alphablox Cube Server 中使用此设置来获取连接池，但此设置主要用于连接池不可用的 Apache Tomcat 3.2.4 上的 DB2 Alphablox 安装。对于 WebSphere 和 WebLogic 服务器上的 DB2 Alphablox 安装，您应使用应用程序服务器的连接池功能。要在 WebSphere 或 WebLogic 服务器上使用连接池，当定义 DB2 Alphablox 关系数据源定义时，需要使用 Application Server Adapter 选项。有关用受支持的数据源配置连接池的详细信息，请参阅您的应用程序服务器文档。

对数据库打开的每个连接都有相关联的成本，然而，此成本较低。在许多情况下，响应时间的差别并不显著，但在某些情况下却很显著。底层数据库也可能会限制它所接受的连接数，因此 DBA 可能不想您使用太多的连接。

数据和成员高速缓存:

通过对特定的 DB2 Alphablox 立方体使用“调整”面板中的可用选项，可以提高该立方体的性能。可以修改两种可用的高速缓存（数据高速缓存和成员高速缓存）以潜在地提高立方体的性能。存储在高速缓存中的数据越多，对 DB2 Alphablox 立方体执行的查询需要从底层数据库中检索结果的频率就越低，从而缩短了查询响应时间。但是，如果高速缓存增大得太大，则将耗尽机器上的内存，从而有可能使所有用户的性能下降。为了确定系统的最优高速缓存大小，您需要进行试验并考虑内存资源、用户负载和查询负载。根据用户负载和查询负载的不同，进行折衷以确定最佳的数据高速缓存大小和成员高速缓存大小。

数据高速缓存存储从关系数据库中访存的立方体单元。存储的数据一旦装入数据高速缓存中，则是由并发查询和后续查询共享使用的。数据高速缓存的大小是可配置的。要了解有关数据高速缓存选项的详细信息，请参阅第 22 页的『设置调整参数』。

成员高速缓存存储维元数据（成员），并可进行调整以高速缓存全部或部分成员。对于成员高速缓存，存在两种可用的方式：静态高速缓存（缺省值）和动态高速缓存。使用**静态高速缓存**时，将在立方体启动期间从关系数据源读取维成员并将它们预装入内存中。

选择**动态高速缓存**时，将从关系数据源读取维成员，并将它们存储在文件系统上用户指定位置处的压缩数据文件中。使用的磁盘空间大小与立方体中的成员数成正比。用于成员高速缓存的内存大小与成员数和每个成员的大小成正比。在 32 位系统上，每个成员的固定成本大约为 168 字节；在 64 位系统上，该成本大约为 290 字节。可变成本取决于成员键、平均成员名长度、成员属性的编号和类型、成员具有的子成员数以及其他成本。

对于立方体中的每个维，在成员高速缓存目录中创建了两个文件：包含维数据的文件（名称为 [*cubeName*].[*dimensionName*]）以及相关联的索引文件（名称为 [*cubeName*].[*dimensionName*].idx）。维文件包含关于成员名、成员属性、成员键和其他成员数据的信息。对于每个立方体，也有一个立方体索引文件（名称为 [*cubeName*].idx）。

启动或刷新立方体时，会生成该立方体的成员高速缓存文件并覆盖所有先前文件。只要立方体在运行，就会根据需要从维文件中将维成员自动读取到内存中。如果成员高速缓存增大得比用户指定的参数要大，将根据 DB2 Alphablox Cube Server 的高速缓存策略管理空间。

可以使用 DB2 Alphablox 管理页面的**管理**选项卡上的 **DB2 Alphablox 立方体管理器**链接来指定此目录的位置，此位置适用于所有立方体。**成员高速缓存目录路径**值指定每个 DB2 Alphablox 立方体的维数据的磁盘存储器目录路径。缺省情况下，该目录路径为 *alphablox_dir*\analytics\repository\temp（例如，c:\alphablox\analytics\repository\temp）。可以根据性能或备份需求指定不同的目录路径位置。为了获得最佳 I/O 性能，成员高速缓存目录应该在本地文件系统而不是网络安装的文件系统上。要了解有关成员高速缓存选项的详细信息，请参阅第 22 页的『设置调整参数』。

要了解关于内存的更多信息，请参阅 『DB2 Alphablox 立方体内存注意事项』。

要调整 DB2 Alphablox 立方体的数据高速缓存和成员高速缓存:

1. 作为 *admin* 用户或作为隶属于 administrators 组的用户登录到 DB2 Alphablox 主页。
2. 单击**管理**选项卡。
3. 单击**立方体**链接。
4. 从立方体列表中选择 DB2 Alphablox 立方体，然后单击**编辑**按钮。所选立方体的 DB2 Alphablox **立方体管理**对话框将显示在新的 Web 页面窗口中。
5. 单击**调整**选项卡。
6. 选取要设置的任何限制的框并输入相应的数值。
7. 单击**保存**按钮以将限制保存到 DB2 Alphablox 立方体定义中。

最大立方体数

如果已定义了许多 DB2 Alphablox 立方体，并且如果每个立方体都开始使用大量的内存和机器资源，则整个系统的性能都将受到影响。为了帮助控制这种情况，可以限制允许在 DB2 Alphablox 中运行的 DB2 Alphablox 立方体数目。此限制控制着可以同时运行的 DB2 Alphablox 立方体数目；它并不限制可以定义的数目。

要对并发运行的 DB2 Alphablox 立方体数目设置限制:

1. 作为隶属于 administrators 组的用户登录到 DB2 Alphablox 主页。
2. 单击**管理**选项卡。将显示“常规”页。
3. 在“一般属性”部分下面，单击 **DB2 Alphablox 立方体管理器**链接。
4. 选中**最大立方体数**复选框并输入要设置的限制数目。
5. 单击**保存**按钮以保存更改。

最大行数和列数

通过限制 DB2 Alphablox 立方体数据源中的最大行数和列数，可以限制应用程序不得发出返回大量数据的查询。请在**数据源**管理页面上的 DB2 Alphablox 立方体数据源中设置这些限制。该数据源就是用来对 DB2 Alphablox 立方体发出 MDX 查询的数据源。

DB2 Alphablox 立方体内存注意事项

DB2 Alphablox Cube Server 是作为 DB2 Alphablox 运行所在的 Java™ 进程的一部分运行的。因此，当 Cube Server 使用更多的内存时，Java 进程就会使用更多的内存。DB2 Alphablox Java 进程的内存限制是在安装时设置的。如果您发现 DB2 Alphablox 由于 DB2 Alphablox 立方体使用大量的内存而耗尽了内存，则可以执行几项可能的操作:

- 限制每个立方体的内存高速缓存大小。要了解详细信息，请参阅第 30 页的『连接和高速缓存大小限制』。
- 限制系统中的 Alphablox 立方体数目。要了解详细信息，请参阅『最大立方体数』。
- 更改 DB2 Alphablox 运行所在的 Java 进程的内存堆最大大小。要了解详细信息，请参阅第 33 页的『更改最大内存堆大小』。
- 增大运行 DB2 Alphablox 的计算机的内存容量。要了解详细信息，请参阅第 33 页的『在系统中添加更多的内存』。

更改最大内存堆大小

DB2 Alphablox Cube Server 作为 DB2 Alphablox 的 Java 进程的一部分运行。如果您在 DB2 Alphablox 中遇到内存不足的错误，则可能需要提高 Java 进程的最大内存堆大小。请将最大内存堆大小设置为足够大以便满足内存要求，但又设置得足够小以便不会导致操作系统在进程大小接近最大值时过度地交换到磁盘。并且，为机器上预料不到的内存使用留下一些空间。例如，如果机器有 1024 兆字节的内存，并且机器上的其他资源使用了大约 300 兆字节的内存，则考虑将最大内存堆大小设置为 600 兆字节的值。

可能需要进行一些试验才能找到系统的理想最大值。如果没有任何问题、性能较佳并且在 DB2 Alphablox 立方体中没有内存不足错误，则表示环境的限制设置得不错。

在系统中添加更多的内存

一种经常被忽略的内存问题解决方案是在运行 DB2 Alphablox 的系统中添加更多的内存。请与您的硬件供应商一起进行检查以确定可以在计算机上安装的内存容量。当系统上的内存使用量增大到接近已安装的物理内存的限制时，系统将把内存交换到磁盘以便为新的内存请求腾出空间，从而导致内存管理效率更低。

内存升级通常是成本相对较低的提高服务器容量的方法。并且，它通常能够帮助解决或消除内存使用问题。如果系统上有空间可以添加更多的内存，您应该考虑这样做。

第 5 章 使用 MDX 来查询 DB2 Alphablox 立方体

DB2 Alphablox 应用程序使用多维表达式（MDX）语言来查询 DB2 Alphablox 立方体。MDX 是 OLE DB for OLAP 规范的查询语言组件，它由 Microsoft 创建并维护。DB2 Alphablox 立方体支持 MDX 语法和函数的一个子集。本节描述在查询 DB2 Alphablox 立方体时支持的 MDX 语法并提供了示例查询。

受支持的 MDX 语法

MDX 是由包括 Microsoft Analysis Services 在内的数种多维数据库使用的多维查询语言。DB2 Alphablox Cube Server 使用 MDX 语法的一个子集来作为 DB2 Alphablox 立方体的查询语言。对于访问 DB2 Alphablox 立方体的 DB2 Alphablox 应用程序来说，MDX 查询被用作 DataBlox 查询参数（或相关方法）的值。

基本语法

针对 DB2 Alphablox 立方体执行的 MDX 查询的基本语法如下所示:

```
SELECT {axisSpecification} ON COLUMNS,
       {axisSpecification} ON ROWS
FROM cubeName
WHERE (slicerItems)
```

其中:

axisSpecification
: 是一个或多个元组的集合。可以以列表形式输入元组，也可以通过 CrossJoin 函数来“生成”元组。

cubeName
: 是已定义的 Alphablox 立方体的名称。

slicerItems
: 是一个元组（通常是用逗号分隔的成员列表），将对这个元组过滤查询结果集。如果有多个切片成员，则每个成员必须来自不同的维，并且不能在查询中指定的任何轴中引用该维。

用法说明

以下几点包括有关 DB2 Alphablox 的上下文中 MDX 用法的重要信息。

- 维只能在查询中的一个轴上出现。将维放在多个轴上的查询将由于出错而失败。
- 虽然查询通常指定两个轴，但它可以指定零个或多个轴。还可以将 COLUMNS 轴指定为 AXIS(0) 以及将 ROWS 轴指定为 AXIS(1)。后续的每个轴将被指定为 AXIS(*n*)，其中 *n* 是下一个连续整数。注意，显示查询结果数据的 DB2 Alphablox 应用程序（GridBlox、ChartBlox 或 PresentBlox）只能接受最多指定了两个轴的查询。作为 XML 数据集显示的查询可以接受任意个轴。
- 在 DB2 Alphablox 中，MDX 中的关键字是不区分大小写的，但是 MDX 查询中的成员名在被方括号 [ ] 括住时是区分大小写的。当成员名未被方括号 [ ] 括住时，它们在被发送到服务器之前将被转换为大写。除非数据库中的所有成员名都是大写的，否则应该使用方括号语法。

- 如果在需要成员的 MDX 函数中只包含维名称，则 DB2 Alphablox Cube Server 将返回一些使用 [dimensionName].currentMember 作为值的结果。

指定成员集

成员集由来自同一个维的一个或多个成员组成。虽然没有规定必须将成员名括在方括号中，但良好的做法是始终用一对方括号 [ ] 将成员名括起来。当成员名包含空格时，必须将该成员名括在方括号中。成员名是区分大小写的；因此，下列成员指定不是等效的:

```
[Time].[Fiscal Year]
[Time].[fiscal year]
```

限定成员名

可以使用维名以及它在层次结构中的父代来限定成员名，这与对象语法类似，如下所示:

```
[Dimension].[Level].[Member]
```

也可以使用维名以及成员的一个或多个祖代来限定成员名，如下所示:

```
[Dimension].[Member].[Member]
```

注: 您总是应该对成员名进行限定，最低的要求是使其具有唯一性。

花括号

花括号表示集合，必须将放在 MDX 查询中的轴上的集合括在花括号 { } 中。例如，如果要指定包含产品 Golden Oats 和 Sugar Grains 的集合，语法如下所示:

```
{[Product].[Golden Oats], [Product].[Sugar Grains]}
```

FROM:TO 语法

通过使用冒号（:）来分隔成员，可以指定从层次中的一个点扩展到另一个点（包括这两个点）的成员集。例如，如果维名为 *Alphabet* 并且包含成员 A-Z，则以下内容将求值为集合 {D, E, F, G, H}:

```
{[Alphabet].[D]:[Alphabet].[H]}
```

计算的成员

计算的成员允许在不将新成员添加至底层关系数据源的情况下创建派生的成员。

计算的成员（也称为派生的成员）是维中通过使用数学或逻辑运算从其他成员的值派生的成员。写入它们中的任何值将在执行下一个计算时被覆盖。

计算的成员允许将新成员添加到数据源，而不要求重建立方体或将新值添加至数据源。在派生的值很少被访问的情况下，计算的成员也很有用。

非持久计算的成员

通过使用 MEMBER 子句，可以将非持久计算的成员定义为 MDX 查询的一部分。非持久计算的成员仅在该查询的生存期内可用，并在分析应用程序中可能有有限的用处。

持久计算的成员

在定义立方体时，持久计算的成员被定义为立方体的一部分。持久计算的成员的优点是它们可用于任何查询。在 DB2 Alphablox Cube Server 中，持久计算的成员属于指定的维，其行为类似于一般成员，它有一个父代并在维层次结构中。要创建持久计算的成员，必须指定计算的成员的全名，包括计算的成员在层次结构中的维和位置，并且必须指定表示计算的成员的表达式。也可以有选择地指定求解顺序。

受支持的 MDX 函数

MDX 函数用来简化和拓宽 MDX 查询的可能作用域。下表列示了在对 DB2 Alphablox 立方体执行的查询中支持的 MDX 函数和运算符的子集。

要了解有关下面列示的 MDX 函数的语法和用法的信息，请参阅下列信息资源:

- Microsoft MDX Function Reference (http://msdn.microsoft.com/library/en-us/olapdmad/agmdxfunctintro_6n5f.asp)
- Spofford, George. 2001. *MDX Solutions*. New York: John Wiley & Sons.

运算符
Is、And、Or、Not、XOR、>、>=、<、<=、= 和 <>

MDX 函数	语法
支持的运算符	
Aggregate	Aggregate(*Set*[,*NumericExpression*])
AllMembers	*Dimension*.AllMembers
Ancestor	Ancestor(*Member*,*Level*) Ancestor(*Member*,*NumericExpression*)
Ancestors	Ancestors(*Member*,*Level*) Ancestors(*Member*,*NumericExpression*)
Ascendants	Ascendants(*Member*)
Avg	Avg(*Set*[,*Count*])
BottomCount	BottomCount(*Set*,*Member*[,*NumericExpression*])
BottomPercent	BottomPercent(*Set*,*Percentage*[,*NumericExpression*]) **注**: *NumericExpression* 在此处是可选的，但在 MSAS 中是必需的。
BottomSum	BottomSum(*Set*,*Value*[,*NumericExpression*]) **注**: *NumericExpression* 在此处是可选的，但在 MSAS 中是必需的。
Children	*Member*.Children
ClosingPeriod	ClosingPeriod(*Level*,*Member*)

MDX 函数	语法
CoaleseEmpty	CoalesceEmpty(*NumericExpression* [,*NumericExpression*]... | *StringExpression*[,*StringExpression*]...)
Count	Count(*Set*[, ExcludeEmpty | IncludeEmpty]) **注**: 此处只支持 Count(*Set*[, ExcludeEmpty | IncludeEmpty])。此处不支持 .Count 语法。
Cousin	Cousin(*Member1*,*Member2*)
CrossJoin	Crossjoin(*Level*,*Member*)
CurrentMember	*Dimension*.CurrentMember
DataMember	*Member*.DataMember
DefaultMember	{*DimensionExpression* | *HierarchyExpression*}.DefaultMember
Descendants	Descendants(*Member*,[*Level*[,*DescFlags*]]) **注**: 此处仅支持 Descendants(*Member*,[*Level*[,*DescFlags*]])。此处不支持带有 *Set* 选项的 Descendants()。
Distinct	Distinct(*Set*)
DrilldownLevel	DrilldownLevel(*Set*[,{*Level*|,*Index*}])
DrilldownMember	DrilldownMember(*Set1*,*Set2*[,RECURSIVE])
DrillupMember	DrillupMember(*Set1*,*Set2*)
Except	Except(*Set1*,*Set2*[,ALL])
Filter	Filter(*SetExpression*, { Logical_Expression | [CAPTION | KEY | NAME] =String_Expression })
FirstChild	*Member*.FirstChild
FirstSibling	*Member*.FirstSibling
Generate	Generate(*Set1*,*Set2*[,ALL]) **注**: 支持 Generate(*Set1*,*Set2*[,ALL])。此处不支持 Generate(*Set*,<String Expression>[,*Delimiter*])。
Head	Head(*Set*[,*NumericExpression*])
Hierarchize	Hierarchize(*Set*[,POST])
Hierarchy	*Member*.Hierarchy *Level*.Hierarchy
IIf	IIf(*LogicalExpression*, {*Expression1*, *Expression2*})

MDX 函数	语法
Intersect	Intersect(*Set1*,*Set2*[,ALL])
IsEmpty	IsEmpty(*MDXExpression*)
Item	*Set*.Item(*Index*) **注**: 不支持 *Set*.Item(*StringExpression*[,*StringExpression*])。
Lag	*Member*.Lag(*NumericExpression*)
LastChild	*Member*.LastChild
LastPeriods	LastPeriods(*Index*,*Member*)
LastSibling	*Member*.LastSibling
Lead	*Member*.Lead(*NumericExpression*)
Level	*Member*.Level
Max	Max(*Set*[,*NumericExpression*])
Median	Median(*Set*[,*NumericExpression*])
Members	*Dimension*.Members *Hierarchy*.Members *Level*.Members **注**: 不支持 Members(*StringExpression*)。
Min	Min(*Set*[,*NumericExpression*])
MTD	MTD([*MemberExpression*])
Name	*Dimension*.Name *Level*.Name *Member*.Name *Hierarchy*.Name
NameToSet	NameToSet(*MemberName*)
NextMember	*Member*.NextMember
NonEmptyCrossjoin	NonEmptyCrossjoin(*SetExpression*[,*SetExpression ...*] [,*CrossjoinSetCount*])
OpeningPeriod	OpeningPeriod(*Level*,*Member*)
Order	Order(*Set*,*NumericExpression*[,ASC|DESC|BASC|BDESC])
Ordinal	*Level*.Ordinal

MDX 函数	语法
ParallelPeriod	ParallelPeriod(*Level*,*NumericExpression*,*Member*)
Parent	*Member*.Parent
PeriodsToDate	PeriodsToDate(*Level*,*Member*)
PrevMember	*Member*.PreviousMember
Properties	*Member*.Properties(*StringExpression*) **注**: 在此处，Properties() 函数只支持用户定义的成员属性。可以使用此函数来访问层次中成员的成员属性，一般在计算的成员定义中引用此函数。
QTD	QTD([*MemberExpression*])
Rank	Rank(*Tuple*, *Set*[, *CalcExpression*])
Stdev	Stdev(*SetExpression*[,*NumericExpression*])
Stdevp	Stdevp(*SetExpression*[,*NumericExpression*])
Stddev	Stddev(*SetExpression*[,*NumericExpression*])
Stddevp	Stddevp(*SetExpression*[,*NumericExpression*])
Subset	Subset(*Set*,*Start*[,*Count*])
Sum	Sum(*Set*,*NumericExpression*)
Tail	Tail(*Set*[,*Count*])
TopCount	TopCount(*Set*,*Count*[,*NumericExpression*])
TopPercent	TopPercent(*Set*,*Percentage*[,*NumericExpression*]) **注**: *NumericExpression* 在此处是可选的，但在 MSAS 中是必需的。
TopSum	TopSum(*Set*,*Value*[,*NumericExpression*]) **注**: *NumericExpression* 在此处是可选的，但在 MSAS 中是必需的。
Union	Union(*Set1*,*Set2*[,ALL]) Union({*Set1*,*Set2*})
UniqueName	*Dimension*.UniqueName *Level*.UniqueName *Member*.UniqueName *Hierarchy*.UniqueName
Value	*SetExpression*.*Value*
Var	Var(*NumericExpression*[,*NumericExpression*])
Variance	Variance(*SetExpression*[,*NumericExpression*])

MDX 函数	语法
VarianceP	VarianceP(*SetExpression*[,*NumericExpression*])
VarP	VarP(*SetExpression*[,*NumericExpression*])
WTD	WTD([*MemberExpression*])
YTD	YTD([*MemberExpression*])

MDX 查询示例

本节提供一些对名为 *DB2AlphabloxCube* 的 DB2 Alphablox 立方体执行的 MDX 查询示例。假定示例中的 DB2 Alphablox 立方体具有下列维、层次和量度。

Time	Products	量度
Year {1998, 1999, 2000, 2001}	Imported {Yes, No}	{Sales, Cost, Profit}
Quarter {Q1, Q2, Q3, Q4}	Product Name {A-Z}	
Month {1-12}		

示例 1

以下查询从列轴上的 *Product Name* 层次中选择若干个成员（*A*、*B*、*C*、*D* 和 *Z*），对行轴的 *Time* 维使用 Children 函数来生成一组年份，并在 WHERE 子句中按 *Sales* 量度对查询进行切片。

```
SELECT {[Products].[Product Name].[A]:[D],
     [Products].[Product Name].[Z]} ON COLUMNS,
       {[Time].Children} ON ROWS
FROM [DB2AlphabloxCube]
WHERE ([Sales])
```

Time	A	B	C	D	Z
2001	12.5	14.25	34.95	2,503.22	
2002					
2003					
2004					179.7

示例 2

以下查询使用 CrossJoin 函数来在列轴上显示产品成员 E 和 F 以及 1999 年的 4 个季度。行轴显示 DB2 Alphablox 立方体中的三个量度。

```
SELECT CrossJoin({[Products].[Product Name].[E],
     [Products].[Product Name].[F]}, [Time].[1999].Children)
     ON COLUMNS,
  {[Sales], [Cost], [Profit]} ON ROWS
FROM [DB2AlphabloxCube]
```

	E				F			
量度	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
Sales	17,700	16,80044	44	18,100	1,413.87	1413.87	5,510	1,413.87

Cost	12,300	12,300	50	13,200	599.97	599.97	4,400	599.97
Profit	5,400	4,500	—6	4,900	813.9	813.9	1,110	813.9

声明

本信息是为在美国提供的产品和服务编写的。

IBM 可能在其他国家或地区不提供本文档中讨论的产品、服务或功能特性。有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并非意在明示或暗示只能使用 IBM 的产品、程序或服务。只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务，都可以代替 IBM 产品、程序或服务。但是，评估和验证任何非 IBM 产品、程序或服务，则由用户自行负责。

IBM 公司可能已拥有或正在申请与本文档内容有关的各项专利。提供本文档并未授予用户使用这些专利的任何许可。您可以用书面方式将许可查询寄往:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

有关双字节（DBCS）信息的许可查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往:

IBM World Trade Asia Corporation, Licensing, 2-31 Roppongi 3-chome, Minato-ku, Tokyo 106-0032, Japan

本条款不适用英国或任何这样的条款与当地法律不一致的国家或地区: International Business Machines Corporation “按现状” 提供本出版物，不附有任何种类的（无论是明示的还是暗含的）保证，包括但不限于暗含的有关非侵权、适销和适用于某种特定用途的保证。某些国家或地区在某些交易中不允许免除明示或暗含的保证。因此本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。此处的信息将定期更改；这些更改将编入本资料的新版本中。IBM 可以随时对本资料中描述的产品和／或程序进行改进和／或更改，而不另行通知。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式充当对那些 Web 站点的保证。那些 Web 站点中的资料不是 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的: （i）允许在独立创建的程序和其他程序（包括本程序）之间进行信息交换，以及（ii）允许对已经交换的信息进行相互使用，请与下列地址联系:

IBM Corporation, J46A/G4, 555 Bailey Avenue, San Jose, CA 95141-1003 U.S.A.

只要遵守适当的条件和条款，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本文档中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际软件许可协议或任何同等协议中的条款提供。

此处包含的任何性能数据都是在受控环境中测得的。因此，在其他操作环境中获得的数据可能会有明显的不同。有些测量可能是在开发级的系统上进行的，因此不保证与一般可用系统上进行的测量结果相同。此外，有些测量是通过推算而估计的，实际结果可能会有差异。本文档的用户应当验证其特定环境的适用数据。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其他可公开获得的资料中获取。IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其他关于非 IBM 产品的声明。有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

所有关于 IBM 未来方向或意向的声明都可随时更改或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

本信息包含在日常业务操作中使用的数据和报告的示例。为了尽可能完整地说明这些示例，示例中可能会包括个人、公司、品牌和产品的名称。所有这些名称都是虚构的，与实际商业企业所用的名称和地址的任何雷同纯属巧合。

本信息包括源语言形式的样本应用程序，这些样本说明不同操作平台上的编程方法。如果是为按照在编写样本程序的操作平台上的应用程序编程接口（API）进行应用程序的开发、使用、经销或分发为目的，您可以任何形式对这些样本程序进行复制、修改、分发，而无须向 IBM 付费。这些示例并未在所有条件下作全面测试。因此，IBM 不能担保或暗示这些程序的可靠性、可维护性或功能。用户如果是为了按照 IBM 应用程序编程接口开发、使用、经销或分发应用程序，则可以任何形式复制、修改和分发这些样本程序，而无须向 IBM 付费。

商标

下列各项是 International Business Machines Corporation 在美国和／或其他国家或地区的商标或注册商标:

DB2	DB2 OLAP Server	DB2 Universal Database
IBM	WebSphere®	

Alphablox 和 Blox 是 Alphablox Corporation 在美国和／或其他国家或地区的商标或注册商标。

Java 和所有基于 Java 的商标是 Sun Microsystems, Inc. 在美国和／或其他国家或地区的商标。

Linux® 是 Linus Torvalds 在美国和／或其他国家或地区的商标。

其他公司、产品或服务名称可能是其他公司的商标或服务标记。

索引

D

E

M

R

S

程序号: 5724-L14

中国印刷

S151-0145-03

Spine information:

IBM DB2 Alphablox

DB2 Alphablox Cube Server 管理员指南

版本 8.4