WebSphere Business Monitor

版本 6.0.1

数据库服务

注意！

在使用本资料及其支持的产品之前，请阅读第 67 页的『声明和商标』中的常规信息

第 6 版（2006 年 3 月）

该版本适用于 IBM WebSphere Business Monitor V6.0.1（5724-M24）产品及其所有后续发行版和修订版，直至在新版本中另有指明为止。

IBM 欢迎您提出宝贵意见。您可以将意见寄往以下地址:

IBM 中国公司上海分公司，汉化部
中国上海市淮海中路 333 号瑞安广场 10 楼
邮政编码: 200021

请包含与您的意见相关的页码或标题。

当您将信息寄往 IBM 后，即授予 IBM 非专有权，IBM 可以它认为合适的任何方式使用或分发此信息，而无须对您承担任何责任。

目录

管理数据库

管理 WebSphere® Business Monitor 数据库在 WebSphere Business Monitor 的安装和部署阶段尤为重要。

管理数据库包括以下任务:

- 创建数据库
- 设置正确的数据库配置
- 创建静态和动态数据库数据库和索引
- 部署生成的复制脚本
- 部署 Cube Views 元数据定义
- 维护数据库

注: 整个 WebSphere Business Monitor 文档中，讨论了各种数据库的底层模式。有些模式生成器输出深刻反映了 WebSphere Business Monitor 数据库模式的性质。虽然 DBA 可以利用这些信息维护和调优数据存储时，不应将这些信息看作公共 API 的定义。 WebSphere Business Monitor 的未来版本完全可能会更改此底层模式。客户在开发基础结构时，不应假定会在未来发行版中维护这些模式的向后兼容性。为了使用 WebSphere Business Monitor 数据库而编写的定制代码很可能与未来的产品发行版不兼容。

以下信息将帮助您规划和准备管理 WebSphere Business Monitor 数据库。

数据库配置和复制

数据管理在 WebSphere Business Monitor 中扮演重要的角色。

WebSphere Business Monitor 的数据库体系结构支持以下需求:

- 将数据存储的运行时处理与数据存储客户机访问隔离，以保持合适的处理速度
- 能够在客户机访问数据存储上执行更新，并仍对客户机查询作出快速响应
- 优化对历史数据存储的访问，以用于分析和多维报告

WebSphere Business Monitor 数据库中数据的使用模式会有不同，这取决于使用的组件。数据是由两个主要组件使用的: 事件处理器和客户机仪表板。用途的不同导致必须将仪表板的数据库与用于事件处理的数据库分离。可将数据进一步分类到与业务度量模型相关的信息和与处理事件相关的信息。

仪表板显示两种类型的数据，最新数据和历史实例数据。最新实例数比历史实例数少很多。在最新实例执行的查询需要非常快，而且不能受到大量历史实例的影响。这两种类型的数据被分为两个数据库: 运行时和历史数据库。为了增强性能，该体系结构支持所有具有以下内容的功能:

- 用作业务度量模型的定义容器的数据库。它还存储关于其他数据库的信息。
- 用作事务数据库并由事件处理器使用的数据库。
- 用作接近实时分析数据库的数据库，支持分析查询而不影响事务服务器。它是由仪表板使用的。

- 支持对事务历史的多维分析的数据库。仪表板使用它来查看历史数据。

将 WebSphere Business Monitor 数据库分为四个不同的数据库:

- **存储库**: 存储业务度量模型和事件定义。它还存储状态、运行时和历史数据库的模式、名称和主机名。
- **状态**: 存储运行中过程实例的当前状态以及与每个过程实例相关的业务度量值。由 WebSphere Business Monitor 服务器用来进行事件处理。
- **运行时**: 状态和运行时数据库存储的信息基本一致。运行时数据库的区别在于数据的存储方式、最新数据的状况以及数据停留的时间。运行时数据库中的数据至少比在状态数据库中多停留 24 小时。运行时数据库的目的是使用户能够执行近乎实时的分析，而不影响 WebSphere Business Monitor 服务器的事件处理。运行时数据库服务器客户机对最新实例进行查询。它存储业务度量组的运行时信息，以用于有效报告。用于通过仪表板进行查看。
- **历史数据库**: 以星型模式存储已完成实例的信息以及运行中实例的当前状态，以用于多维报告。用于通过仪表板进行查看。

两个数据库存储受监控的事件和“自适应操作管理器”数据。这些数据库是由 WebSphere Business Monitor 内部使用的。没有与过程实例或度量相关的信息存储在其中。

- **发射器**: 存储从引擎发射的事件。发射器数据库表驻留在引擎数据库中。
- **操作目录**: 存储定义为状态的事件以及“自适应操作管理器”必须对其执行的操作。它是在安装期间创建的。

WebSphere Business Monitor 数据库

WebSphere Business Monitor 使用四个数据库来存储事件数据和业务度量模型元数据。这四个数据库是: 存储库、状态、运行时和历史数据库。

存储库数据库

存储库数据库包含描述当前部署的业务度量模型的元数据和有关其他 WebSphere Business Monitor 数据库的信息。存储库数据库包含已部署模型的历史记录。每个 WebSphere Business Monitor 安装只有一个存储库数据库。

存储库数据库是由启动板使用的，启动板用状态、运行时和历史数据库的数据库属性来填充它。这些属性是数据库名称、数据库模式和数据库服务器的主机名。其他 WebSphere Business Monitor 组件使用这些属性在运行时访问状态、运行时和历史数据库。在业务度量模型的导入期间，也会填充存储库数据库。

以下组件会使用存储库数据库:

- **管理控制台**

  通过 WebSphere Business Monitor 管理控制台导入业务度量模型。该导入模型的事件和过程定义存储在存储库数据库中。当导入完成时，业务度量模型视为已部署。导入模型后，过程和事件的定义可供其他 WebSphere Business Monitor 组件检索。

  模式生成器也可使用存储库数据库。模式生成器需要知道用于其数据库工件生成的模式名称。而且，当用户修改了先前部署的业务度量模型并试图为它重新生成模式时，模式生成器会 在生成变更管理工件之前检查存储库数据库中的工件是否存在。
- **仪表板**

仪表板有一组可以显示来自不同透视图数据的视图。其中一些视图使用来自运行时数据库的数据填充，而另一些使用来自历史数据库的数据填充。要使用户能够配置和使用参数表示这些视图，视图就需要检索来自存储库数据库的 WebSphere Business Monitor 元数据。有一些视图需要对照 DB2® Alphablox 立方体来组合查询。这些查询组合需要有关检索自存储库数据库的维、度量和立方体名称的元数据。而仪表板会显示业务过程的过程图，这些过程图保存在存储库数据库中。

- **WebSphere Business Monitor 服务器**

  WebSphere Business Monitor 服务器使用存储库数据库检索过程和事件的定义。

状态数据库

状态数据库存储有关正在运行的实例的信息。这些信息包括度量、业务度量和关键业绩指标（KPI）值。状态数据库得到优化以处理繁重的事务工作负载。每个 WebSphere Business Monitor 安装只有一个状态数据库。

每个过程实例需要状态数据库中的两个表来存储度量、业务度量和 KPI。这些表的结构和过程实例的结构一样是动态的。每个业务度量由两个表之一的某一独立列来表示。根据构建业务度量模型时所选的选项，将状态数据库中的许多或所有信息复制到运行时数据库。

状态数据库由 WebSphere Business Monitor 服务器使用。在运行时，WebSphere Business Monitor 服务器会根据已处理的事件，来插入、检索和更新驻留在状态数据库中的过程实例的信息。

状态数据库存储以下信息:

- 有关作为导入的业务度量模型中的数据的一部分的业务度量组的信息。
- 在 WebSphere Business Monitor 运行时创建的正在运行的过程实例。
- 正在运行的过程的事件条目。事件条目是为更新特定业务度量组而接收的事件数据。

运行时数据库

运行时数据库在结构上与状态数据库类似。它从状态数据库接收有关所有正在运行的过程的当前状态的信息，以及新近完成或失败的过程的最终状态信息。这些信息由 WebSphere Business Monitor 仪表板使用。自适应操作管理器也可使用运行时数据库存储警报通知。每个 WebSphere Business Monitor 安装只有一个运行时数据库。

运行时数据库存储:

- 由自适应操作管理器发送到仪表板的警报通知
- 过程数据
- 度量值

运行时数据库中的信息是从状态数据库复制的。

运行时历史数据库由 WebSphere Business Monitor 仪表板使用。仪表板从运行时数据库检索填充视图所需的正在运行或新近完成的实例数据。仪表板视图使用运行时数据库进行分析，所以针对查询处理和聚集查询处理，对它进行了优化。

历史数据库

历史数据库存储所有已完成和正在运行的过程实例。仪表板通过 DB2 Alphablox，将它用于增强的数据分析。每个 WebSphere Business Monitor 安装只有一个历史数据库。历史数据库中的数据从不会被删除。

历史数据库应只包含两年的历史数据。这是 WebSphere Business Monitor 产品需求之一。如上所述，从不会自动删除历史数据，因此 DBA 负责删除超过两年时限的数据。历史数据库存储关于长时间运行和已完成的实例的信息。这些信息以星型模式存储，而不是状态和运行时数据库的平面事务格式。针对聚集和长时间运行的查询，对历史数据库进行了优化。仪表板视图中的 DB2 Alphablox 使用它来提供高级多维报告。

历史数据库中的信息是从运行时数据库复制的。

在历史数据库中，每个过程实例都有自己的一组集。与状态和运行时数据库不同的是，每组表都是支持多维报告的星型模式。

历史数据库包含根据已部署的业务度量模型创建的动态表。模式生成器生成历史数据模式，用于创建动态表和 Cube Views™ 定义。

历史数据库由 WebSphere Business Monitor 仪表板使用。仪表板从历史数据库检索填充某些视图所需的数据。例如，报告视图主要用于分析从历史数据库抽取的数据。

历史数据库包含以下信息:

- 运行中或已终止的不同版本的过程实例的数据。
- 存储在运行时数据库中的已完成过程实例的数据。在运行时数据库中保留了 24 小时的所有已完成实例。 24 小时是缺省保留时间策略，在数据移动服务配置中可对其进行修改。将数据复制到历史数据库之后，从运行时数据库中删除它以提高性能。

数据库创建和部署

WebSphere Business Monitor 数据库是由启动板在安装期间创建的。创建数据库之前，首先需要规划创建和部署。

如果在安装后不可挽回地删除或破坏了 WebSphere Business Monitor 数据库，数据库管理员可以通过执行保存在 *<monitor_installation_dir>\install\mondb* 中的创建脚本手工重新创建数据库。DBA 还可使用启动板卸载数据库并创建数据库，DBA 应该先从 DB2 手工删除数据库，然后使用启动板重新创建数据库。

数据库工件部署准备

在使用启动板开始创建 WebSphere Business Monitor 数据库之前，需要规划数据库。规划包括分配数据库大小、准备备份策略、配置数据移动服务和设置表空间和缓冲池参数以及确定数据库实例和单个数据库的设置。

在安装期间，启动板创建状态、运行时和历史数据库以及用于管理目的的数据库对象。除了这些对象，模式生成器还创建了一组业务度量模型专用数据库对象（如表）。启动板针对状态、运行时和历史数据库创建一组缺省表空间和缓冲池。这些缺省表空间在表空间配置文件引用，并且旨在使用户能够快速设置和运行，以便进行测试和原型验证。为了避免性能问题和资源约束，需要事前规划表如何在表空间中分布，以及表空间将使用哪些容器和缓冲池。

在安装期间，创建数据库，并只创建静态表定义。在工件生成期间，基于定制文本配置文件将动态表分配给状态、运行时和历史数据库中的表空间。WebSphere Business Monitor 附带有缺省配置文件，位于 <Monitor_install_dir>\install\mondb 目录。该缺省配置文件将所有表都映射到合适大小的一个表空间。为了支持特殊部署，启动板在安装期间创建了对应于缺省配置文件中条目的一组表空间（4KB、8KB、16KB 和 32KB 页面大小）。以下示例显示来自与 WebSphere Business Monitor 一起提供的缺省表空间配置文件的摘要:

```
#

# State database

#

db2.state.Default.TABLE.4K.0=DSDFLTTS4

db2.state.Default.TABLE.8K.0=DSDFLTTS8

db2.state.Default.TABLE.16K.0=DSDFLTTS16

db2.state.Default.TABLE.32K.0=DSDFLTTS32
```

在工件生成期间使用这个简单的配置，所有需要在状态数据库中创建的且满足表空间的页面大小为 4KB 的表，都将被分配到 DSDFLTTS4。页面大小为 8 KB 的表将存储到名为 DSDFLTTS8 的表空间中，而页面大小为 16 KB 的表空间将存储到 DSDFLTTS16，以此类推。由于表空间需求可能发生变化（取决于存储数据的模型的复杂性和数据量），建议不要在测试或生产环境使用缺省表空间配置设置。事先规划和确定适当的存储策略能确保良好的性能。

可使用更多高级配置文件设置，以根据页面大小和存储的数据类型将表映射到表空间。请查看示例配置文件，获取更多信息。

要确定表被分配到哪个表空间，模式生成器要执行以下任务:

- 确定表类型。
- 计算存储至少一行数据所需的最小页面大小。
- 确定该类型的可用表空间；若发现，则定位下一个可用的表空间并使用它。如果未找到表空间，则继续。
- 确定该类型的可用模式生成器；若发现，则定位下一个可用的表空间并使用它。如果未找到任何此类型的可用表空间，则继续。
- 将表分配给数据库缺省表空间（通过在表创建期间不指定表空间子句）。

  注: 如果在数据库中没有为必需的页面大小定义缺省的表空间，则部署会失败。

您可以使用任何文本编辑器编辑表空间配置文件或创建一个新的表空间。使用模式生成器管理控制台中的常规配置选项卡，以使模式生成器使用备用配置文件。

注: 模式生成器实际上不能为此配置文件中的条目创建任何表空间。这需要在部署生成的数据库工件之前手工创建。如果表被分配到不存在的表空间，则工件部署会失败。

数据库工件部署

使用启动板创建了 WebSphere Business Monitor 数据库之后，部署数据库表。在部署阶段，配置模式生成器来生成为完成数据库设置而部署的工件。然后，数据库就准备用数据进行填充。

注: 一次创建数据库。对于每个业务度量模型，更多的数据库表将被添加到数据库。

数据库模式生成:

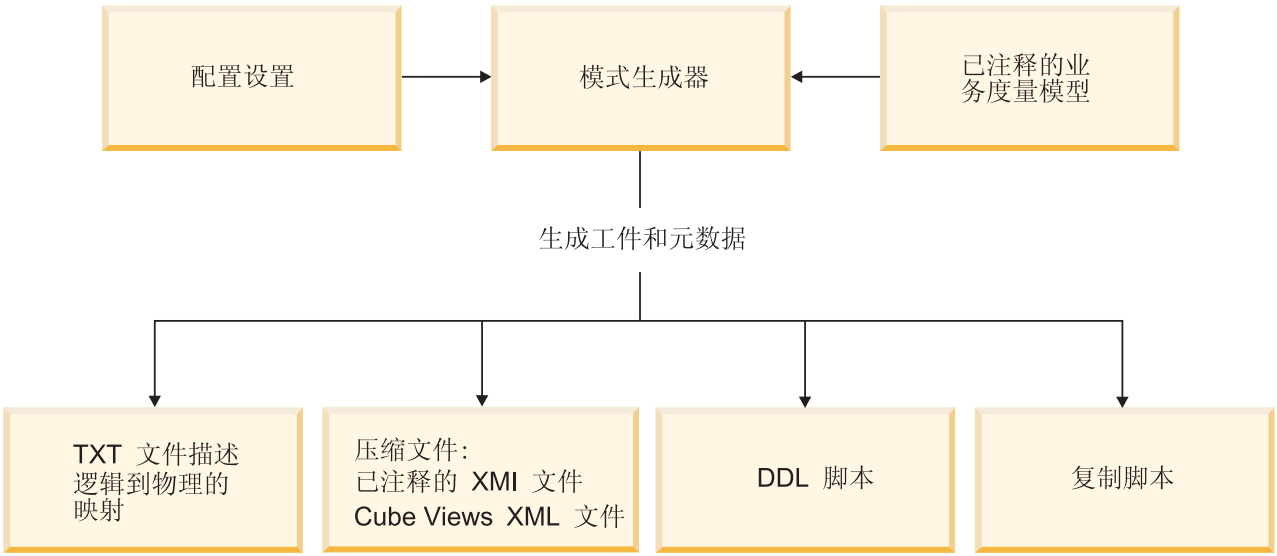
WebSphere Business Monitor 数据库模式基于业务度量模型。这些模式是由模式生成器生成的。

业务度量模型是使用 业务度量编辑器 创建的，它是一个带注释的模型，包含业务模型的元数据。用户可以使用业务度量编辑器来定义可监控的内容: 上下文、关键业绩指标（KPI）、度量和业务状况。业务度量模型通过事件生成监控信息。完成业务度量模型后，将它作为带注释的 XML 文件以 zip 格式导出到 WebSphere Business Monitor，供模式生成器使用。

模式生成器是 WebSphere Business Monitor 管理控制台的一部分。一般情况下，该工具是由数据库管理员（DBA）配置和使用的。它将业务度量模型作为输入；然后生成数据库工件。

以下是生成的工件:

- 用于创建数据库表的数据库定义语言（DDL）脚本。针对生成的每个 DDL 文件（state.ddl、runtime.ddl、datamart.ddl），会创建一个相应的文本文件（StateMapping.txt、runtimeMapping.txt、datamartMapping.txt）。这些文件描述了物理数据库工件（表、列）所表示的度量和过程。
- DB2 Cube Views 定义，描述来星型模式格式的历史数据库。 Cube Views 定义可用于导入到 DB2 OLAP 中心。
- 能够在状态、运行时和历史数据库之间进行复制的复制脚本。对于每个数据库，模式生成器都会创建一个压缩文件，该文件包含在三个数据库之间建立复制所需的所有部署工件。一般而言，这些工件会由 DBA 根据第 38 页的『部署数据移动服务』中的指示信息进行分发和部署。



WebSphere Business Monitor 数据库组件中的数据库表包含两个类型:

- 在安装时一次创建的静态数据库表。这些标在所有业务度量模型之间共享，并不依赖于任何单个业务度量模型。
- 依赖于导入到 WebSphere Business Monitor 管理控制台的业务度量模型的动态数据库表。动态数据库表模式对于每个业务度量模型都是唯一的。对任何与动态表相关的业务度量模型的更改将导致变更管理方案。要获取更多关于变更管理方案的信息，请参阅第 29 页的『变更管理和工件生成』。

Cube Views 定义:

模式生成器创建 DB2 Cube Views XML 文件。数据库管理员（DBA）将此 XML 文件导入到 DB2 OLAP 中心。

模式生成器生成基于业务度量模型的 Cube Views XML 文件。业务度量模型包含有助于描述度量和维的信息。您还可以描述哪些聚集适用于度量。

对于业务度量模型中的每个过程，都会创建一个立方体和一个立方体模型。还会为与过程相关的活动生成立方体和立方体模型。每个立方体模型和立方体都包含一些自动生成的预先定义的度量和维。

每个立方体模型和立方体都包含三个内置度量:

- **耗用的持续时间**: 拥有已定义的聚集函数**“avg”**。
- **工作持续时间**: 拥有已定义的聚集函数**“avg”**。
- **实例计数**: 拥有已定义的聚集函数**“count”**。

自动生成了以下维:

- **CreationTime**: 创建过程实例时的时间
- **StartTime**: 启动过程实例时的时间
- **State**: 包含过程实例所有可能的状态（字符串值），如“已启动”、“正在运行”或“已完成”
- **TerminationTime**: 终止过程实例时的时间

这些基于时间的维使用提供的通用定义维（DIM_TIME）。有三个预定义级别: 年、月、日。

在建模时，可定义自己的业务度量。所创建的业务度量可以是度量或维。WebSphere Business Modeler 文档详细描述了如何使用 WebSphere Business Modeler 创建度量或维。

要获取更多关于 DB2 Cube Views 的信息，请参阅 DB2 文档。

工件生成和部署:

模式生成器根据每个导入的业务度量模型生成数据库和 Cube Views 工件。

作为业务度量模型部署阶段的一个步骤，数据库管理员（DBA）将运行工件。以下任务必须在部署任何业务度量模型之前完成:

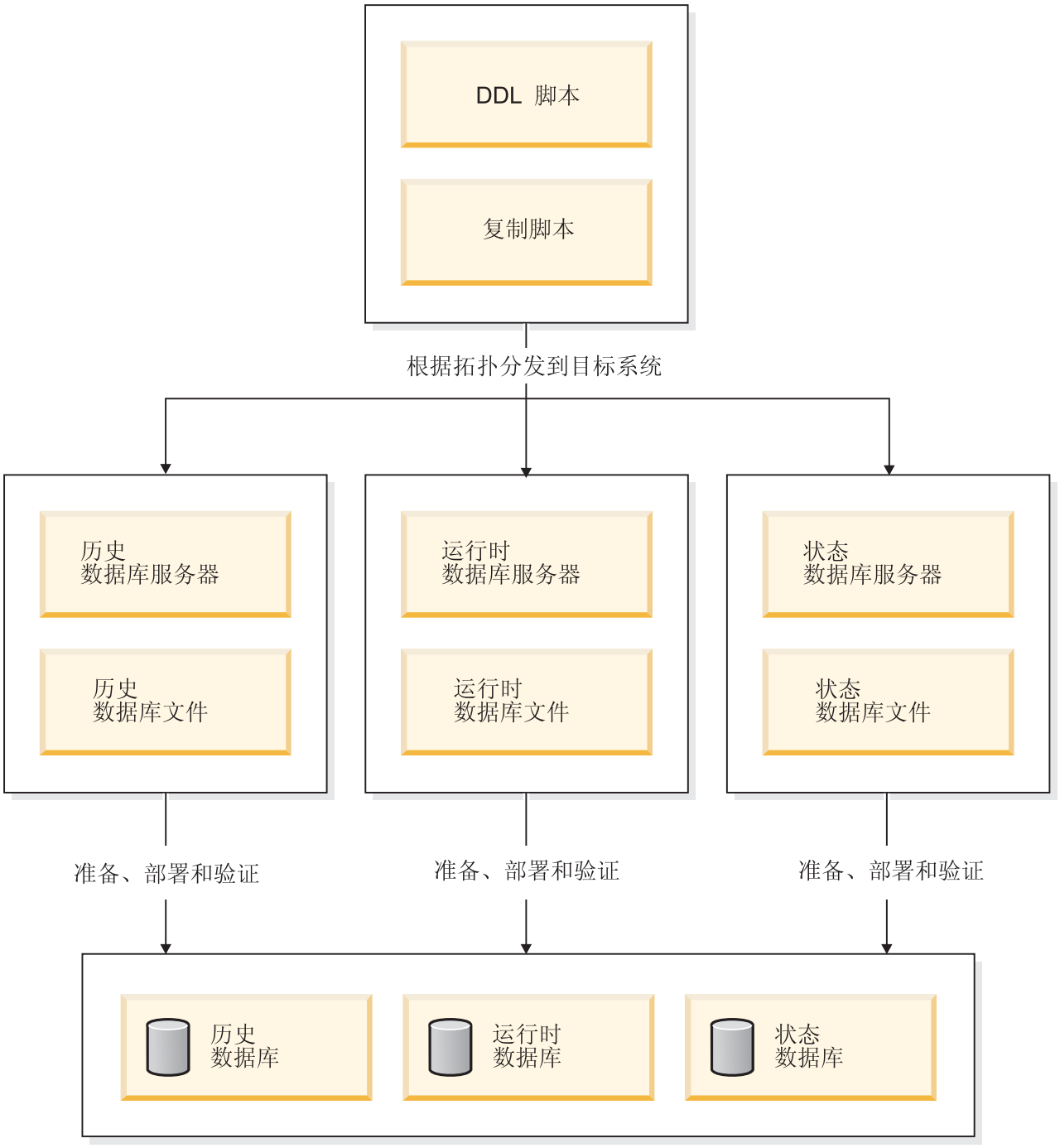
- 参与规划前准备练习。这包括决定拓扑、确定事件率、支持的用户数、数据库实例和数据库参数、缓冲池和表空间以及备份和恢复策略。拥有用于确定工件存储方式和存储位置的策略，也很重要。如果能够根据型号和版本号找到先前生成的一组工件并了解它们是否已部署，将大有帮助。这对于变更管理和支持方案很有用。
- 使用 WebSphere Business Monitor 启动板创建存储库、状态、运行时和历史数据库。创建数据库包括创建一组静态数据库表、设置状态和运行时数据库作为复制源以及创建其他数据库对象，如特定于各种 WebSphere Business Monitor 组件的存储过程和用户定义的函数。
- 运行模式生成器来生成业务度量模型的相关工件。

注: 建议在部署任何生成的工件之前备份所有数据库

生成工件之后，请执行以下步骤来部署它们:

- 部署生成的 DDL 脚本，在状态、运行时和历史数据库中创建业务度量模型动态表。
- 执行复制脚本以启用数据库复制。
- 将 Cube Views 定义文件导入到 DB2 Cube Views。

下图描述了工件部署阶段:



工件定制:

在某些情况下，更改已生成的数据库工件对于提高性能很有帮助。通常可采用两种方法：递进式改进和特殊改进

递进式表空间映射改进

由模式生成器创建的表的数目取决于业务度量模型的复杂程度（以及其他因素）。因此，开始时很难确定如何以最佳方式将这些表分配到表空间。以下简单方法可以帮助

您按照表空间配置文件所定义的那样，逐步改善表到表空间的映射。要获取更多关于数据库规划的信息，请参阅第 4 页的『数据库工件部署准备』。要改善表到表空间的映射，您必须执行以下任务:

1. 使用缺省或任何其他定制的表空间配置文件，运行模式生成器。
2. 将生成的工件抽取到某个临时目录，并确定将在数据库中创建的表的数目和类型。对于每个业务度量模型，会创建以下类型的表:
   - **上下文和活动**: 只存在于状态和运行时数据库中，并通过服务器和仪表板访问。
   - **CD**: 只存在于状态和运行时数据库中。CD 表所含的列通常略多于相关上下文或活动表中包含的列。特定 CD 表的大小主要取决于针对其相应上下文或活动表的事务数、更新与插入事务之比、数据移动服务的相关“应用”组件从 CD 表读取事务并将它们插入到相应 CCD 表的频率，以及相关源生命周期组件修剪 CD 表条目的频率。
   - **CCD**: 只存在于运行时和历史数据库中。CCD 表拥有与对应 CD 表完全相同的结构，其大小也主要取决于上述相同因素。不同之处是，对于 CCD，不是由“应用”组件读取事务，而是由 ETL 组件读取事务，而且由目标生命周期组件修剪条目。
   - **RM 内部表**: 只存在于运行时和历史数据库。这些表使用最大为 4 KB 的页面。
   - **事实和维**: 只存在于历史数据库中。
3. 修改表空间配置文件，以使它包含:
   - 每个表类型的映射
   - 多个表类型到表空间的映射（如果将许多表分配给同一个表空间）

   注:
   - 不要为不在数据库中创建的表类型指定表空间声明，因为将不使用这些表空间。
   - 度量映射到表列。定义的度量越多，表就越大，因此其表空间所需的页面大小也就越大。
4. 在部署生成的工件之前，创建以下表空间（和缓冲池）： 因为没有建立数据库连接，所以模式生成器无法验证配置文件中声明的表空间是否存在。然而，如果表空间不存在，部署将失败。
5. 使用优化的表空间配置文件重新运行模式生成。

   注: 如果创建了新的表空间配置文件，则在模式生成器管理控制台配置中更改配置文件名称。

特殊改进

您可以通过更改以下项来修改生成的工件:

- 任何索引，添加、更改、除去、分配到独立表空间（仅 DMS）以提高数据库性能。

  注: 创建唯一索引要慎重考虑，因为这可能会导致意外的失败。
- 任何表空间分配（分配到不同的表空间，为索引添加表空间分配，如果使用数据库管理的空间（DMS）表空间，则可为大对象添加表空间分配）
- 表的任何注释（不推荐，因为注释说明了每个表和列分别代表什么）

通常情况下，允许的更改不会改变表的基本模式或结构。

更改生成的脚本时，请考虑以下限制:

- 不能更改任何表名。
- 不能更改任何列名。
- 不能更改任何列数据类型。
- 不能删除任何列或任何表。
- 不能向表添加任何列。
- 不能更改表的任何主键。
- 不能更改列的可空性。
- 不能更改任何表的模式分配。
- 不能添加任何新的约束，如唯一性约束或外键约束。

重要: 在下一次为同一业务度量模型执行模式生成时，对生成的数据库工件所做的这些更改将不包括在内。例如，某个用户在部署业务度量模型“Finance Model”的生成工件之前，更改了它们。随后，该用户更改了业务度量模型，并重新生成了所有工件。在这个例子中，由于模式生成器不了解对先前生成的工件的修改，因此用户必须再次修改新生成的工件。

注:

- 工件部署需要使用 Java™ V1.4.2 或更高版本。
- 应在执行复制脚本之前，将 Java bin 目录添加到系统路径。

数据库复制

使用数据库复制技术将业务度量模型相关数据从状态数据库移到运行时数据库，再从运行时数据库移到历史数据库。

要在状态、运行时和历史数据库之间为业务度量模型设置复制，需要执行以下高级别任务:

1. **生成复制设置脚本**。模式生成器会分析要设置复制的业务度量模型，并生成一组设置文件。
2. **分发复制设置脚本**。必须手工将这些设置文件传送到运行状态、运行时和历史数据库的机器上。
3. **执行复制设置脚本**。每个设置文件一旦执行，就会创建必要的数据库对象并配置将数据从一个数据库移到另一个数据库的实用程序。

一旦成功完成，就可启动和运行复制实用程序。以下部分提供了深入的体系结构视图，并说明了使用的基本概念。

数据移动服务

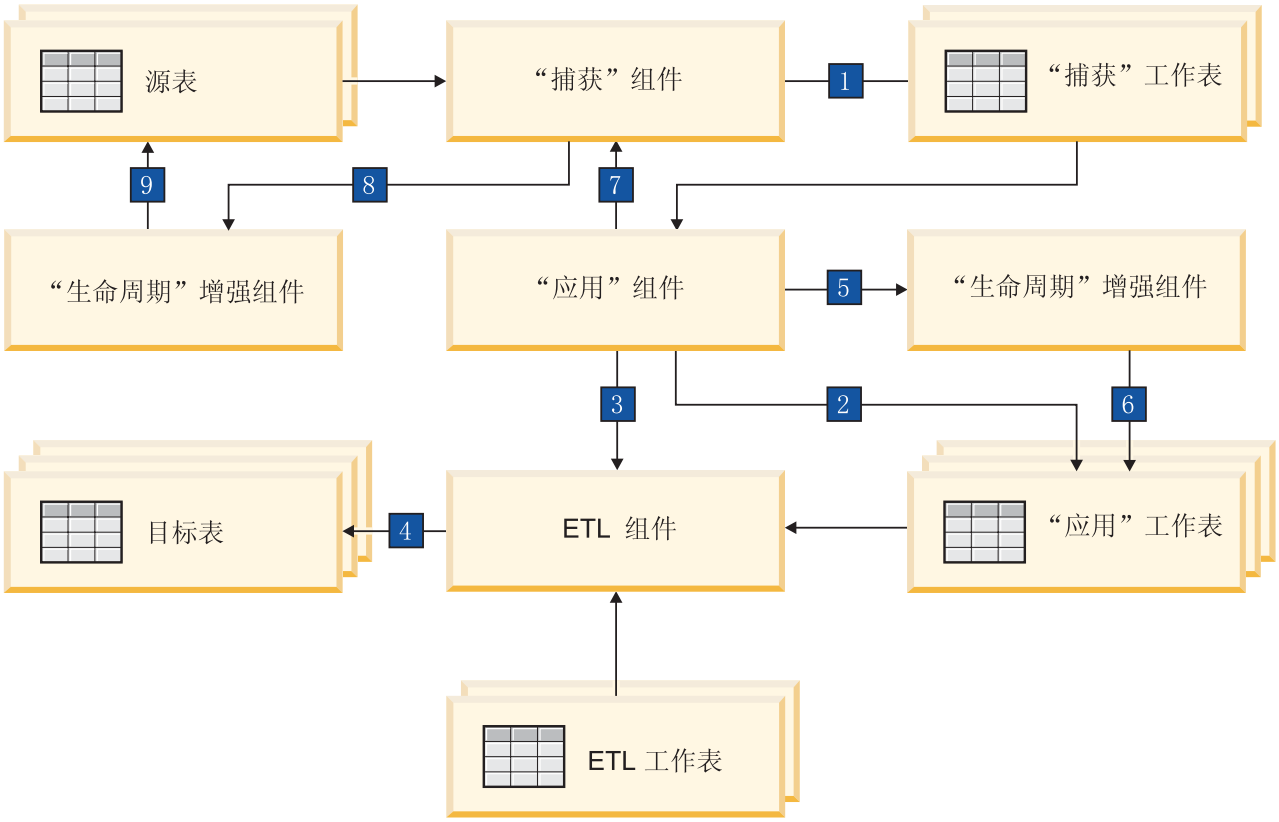
数据移动服务支持应用程序将数据从源数据库移到目标数据库。源数据库和目标数据库可以是同构或异构的，即: 可以安装在单一系统或分布在多个系统上。除移动数据之外，服务还能转换数据，并根据应用程序的需求提供基本的数据生命周期功能。

数据移动服务由 5 种主要组件实施:

1. （源）“捕获”组件
2. （目标）“应用”组件
3. ETL（抽取、变换、装入）组件
4. “源生命周期”组件

5. “目标生命周期”组件

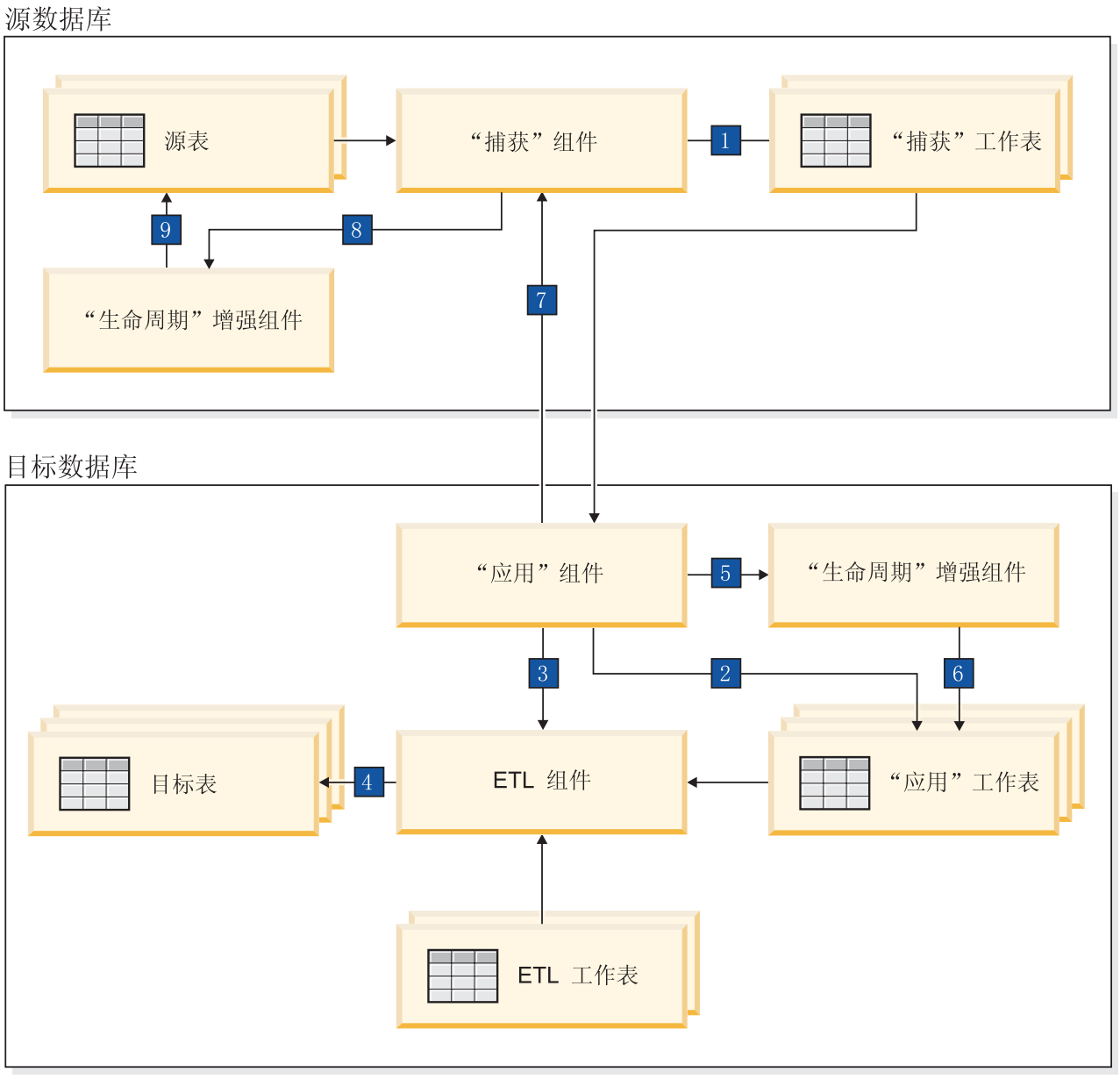
“捕获”和“应用”组件一同工作，将数据从源数据库移到目标数据库。如果源数据库中的数据结构与目标数据库中的数据结构不同，ETL 组件会执行任何需要的数据变换。下图说明了数据移动服务中的过程流:



数据移动服务流执行下列步骤:

1. 存储并经常更新源表中的数据，例如通过监控器服务器。“捕获”组件在工作表中记录了对源表所做的任何数据更改。
2. 在预定义的时间间隔中，通过“应用”组件来识别这些更改，并将其记录在工作表中。
3. 在成功记录更改以后，调用 ETL 组件。
4. ETL 组件使用存储在“应用”工作表中的数据和预定义的规则，来执行任何必要的转换。成功转换的数据将被写入目标表。任何不完全的或错误的数据将保留在另一组工作表中，以便在以后处理。
5. 完成 ETL 处理后，将激活“目标生命周期”组件。
6. 一段时间过后，“应用”工作表中会聚集大量数据。“目标生命周期”组件会删除这些表中已由 ETL 组件成功处理的任何数据。
7. 将数据成功复制到目标数据库后，就不再需要它，并可从“捕获”工作表中删除它。“捕获”组件定期修剪工作表以减少资源不足的情况。
8. 从“捕获”工作表删除数据会触发“源生命周期”组件的调用。
9. 任何已成功处理、被标记为可删除并已通过“源生命周期保留时间策略”的数据，都将从源数据库删除。

“捕获”组件和“源生命周期”组件通常驻留在源系统上；而“应用”、ETL 和“目标生命周期”组件则驻留在目标系统上，如下图所示:



在数据移动服务中，可能会根据源数据库和目标数据库中使用的数据结构，使用多个组件实例。组件实例的数量与业务度量模型中的业务度量组的数目以及源表和目标表的数目直接关联。每个实例都是唯一标识的。以下规则适用于 WebSphere Business Monitor:

- 为每个业务度量模型项目分配一个“捕获”组件实例，由该实例捕获所有属于该业务度量模型项目的源表的更改。
- 为每个业务度量模型项目分配一个“应用”组件实例，由该实例记录需要应用到隶属于此业务度量模型的目标表的更改。
- 为每个目标表分配一个 ETL 组件实例。
- 为每个源表分配一个“源生命周期”组件实例。

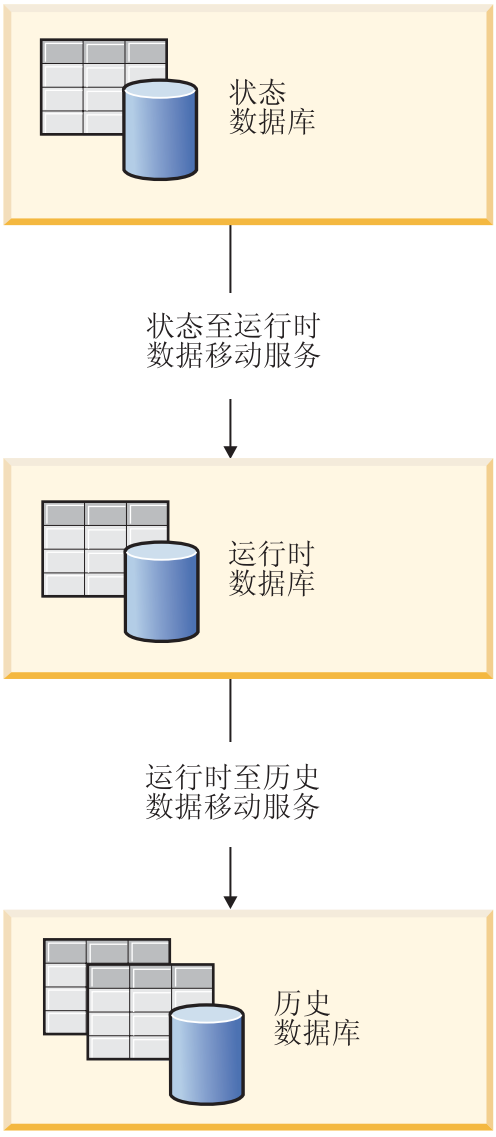
- 为每个“应用”工作表分配一个“目标生命周期”组件实例。

例如，组件实例可以是可执行程序、数据库存储过程或数据库触发器。

WebSphere Business Monitor 中使用了两个数据移动服务实例:

- 状态至运行时数据移动服务
- 运行时至历史的数据移动服务。

状态至运行时数据移动服务处理由监控器服务器存储在状态数据库中的数据，并将这些数据移入可由仪表板访问的运行时数据库。运行时至历史的数据移动服务将数据从运行时数据库移动到历史数据库。下图说明了这一转移:



以下信息描述了这些服务的缺省配置，以及如何配置、启动、停止和监控它们。

状态至运行时数据移动服务:

状态至运行时数据移动服务处理由监控器服务器存储在状态数据库中的数据，并将数据移入运行时数据库，其他 WebSphere Business Monitor 组件和运行时至历史的数据移动服务可以在那里对其进行访问。

以下缺省配置应用于该数据移动服务:

- 将监控器服务器源表（状态数据库）中的更改连续捕获和记录到工作表。
- 由“应用”组件来连续传播已经记录到工作表中的更改，并应用于运行时数据库中的工作表。任何其他 WebSphere Business Monitor 组件都不能访问这些工作表，它们仅供内部使用。
- 每次需要处理新数据时，“应用”组件就会同步调用 ETL 组件。根据调度（初始设置为每 5 小时），ETL 组件会处理存储在“应用”工作表中的数据，或保持不活动直到调度为运行。增加已调度的运行之间的延迟，会延长监控器服务器将数据存储到状态数据库，与在运行时数据库的目标表中“发布”这些数据之间的耗用时间。数据在运行时数据库中后，就可由其他 WebSphere Business Monitor 组件访问。
- “应用”工作表中任何已经由 ETL 组件成功处理的数据都由“目标生命周期”组件根据其计划进行删除。缺省情况下，该组件每 24 小时运行一次。增加调度延迟使工作表变得更大。由于可能会有多个数据服务组件试图更新并同时访问工作表，所以减少延迟也可以引起资源紧缺问题。
- 缺省情况下，已经成功地从“捕获”工作表移到“应用”工作表的数据会自动由“捕获”组件每 5 分钟从“捕获”工作表中删除。
- 每次修剪“捕获”工作表时，都会调用“源生命周期”组件。该组件也是基于进度表的。如果从上次修剪数据起已过去了 5 分钟，则它只将数据从源表（已经由监控器服务器标记为可删除）删除数据。如果将“生命周期”组件修剪时间间隔的值设置为低于“捕获”组件的修剪时间间隔，那么修剪是基于“捕获”组件修剪时间间隔发生的。

  例如: 工作表的“捕获”组件修剪时间间隔设置为 5 分钟，“源生命周期”组件计划设置为 1 分钟。则“捕获”程序必须在 5 分钟后才能开始其修剪周期。由于 5 分钟内没有激活“捕获” 修剪例程，所以不调用“生命周期”组件。5 分钟之后，将数据从工作表删除，调用“源生命周期”组件并将数据从状态数据库中的源表删除。

可更改缺省配置。

运行时至历史的数据移动服务:

运行时至历史的数据移动服务将数据从运行时数据库移到历史数据库，数据将在此处保留至数据库管理员（DBA）将其显式删除。成功移入历史数据库的数据可供其他 WebSphere Business Monitor 组件进行检索和分析。

以下缺省配置应用于该数据移动服务:

- 持续捕获运行时数据库表中的更改和将其记录到工作表中。正受监控的运行时数据库表是由状态至运行时数据移动服务填充的目标表。
- 由“应用”组件来连续传播已经记录到工作表中的更改，并应用于历史数据库中的工作表。任何其他 WebSphere Business Monitor 组件都不能访问这些工作表，它们仅供内部使用。
- 每次需要处理新数据时，“应用”组件就会同步调用 ETL 组件。根据调度（初始设置为每 24 小时），ETL 组件会处理存储在“应用”工作表中的数据，或保持不活动直到调度为运行。增加已调度的运行之间的延迟，会延长状态至运行时数据移动服务将

数据存储到运行时数据库，与在历史数据库的目标中“发布”这些数据之间的耗用时间。数据在历史数据库中后，就可由其他 WebSphere Business Monitor 组件访问。

注: 由于对“应用”组件调用的依赖性和“应用”组件配置，ETL 组件无法每 24 小时（或当前缺省延迟）处理一次新数据。应该将调度解释成“在上一次处理周期完成之后，至少 23 小时 59 分钟不处理新数据”更合适。

- “应用”工作表中任何已经由 ETL 组件成功处理的数据都由“目标生命周期”组件根据其计划进行删除。缺省情况下，该组件每 24 小时运行一次。增加调度延迟使工作表变得更大。由于可能会有多个数据服务组件试图更新并同时访问工作表，所以减少延迟也可以引起资源紧缺问题。
- 已经成功地从“捕获”工作表移到“应用”工作表的数据会自动由“捕获”组件每 5 分钟从“捕获”工作表中删除。
- 每次修剪“捕获”工作表时，都会调用“源生命周期”组件。该组件也是基于进度表的。它只删除运行时数据库（已经由监控器服务器标记为可删除，并至少在运行时数据库中保留 24 小时）中源表的数据。将缺省修剪时间间隔设置为 5 分钟。如果将“生命周期”组件修剪时间间隔的值设置为低于“捕获”组件的修剪时间间隔，那么修剪是基于“捕获”组件修剪时间间隔发生的。

  例如: 工作表的“捕获”组件修剪时间间隔设置为 5 分钟，“源生命周期”组件计划设置为 1 分钟。则“捕获”程序必须在 5 分钟后才能开始其修剪周期。由于 5 分钟内没有激活“捕获” 修剪例程，所以不调用“生命周期”组件。5 分钟之后，将数据从工作表删除，调用“源生命周期”组件并将数据从运行时数据库中的源表删除。

可更改缺省配置。

数据移动服务管理

要启动或停止业务度量模型的数据移动服务，您必须确定相关的主要组件实例以启用或禁用它们。

所有的（源）“捕获”组件实例和所有的（目标）“应用”组件实例都视为主要组件实例。ETL 组件实例和“目标生命周期”组件实例取决于（目标）“应用”组件实例。“源生命周期”组件实例取决于（源）“捕获”组件，无需显式启动或停止它。通常，源数据库中的所有（源）“捕获”组件实例以及所有（目标）“应用”组件实例都必须由用户启动或停止。只有在启动或停止所有这些实例以后，数据移动服务才视为已启动（完全可操作）或已停止。

“捕获”组件实例等同于 DB2 “捕获”程序，“应用”组件等同于 DB2 “应用”程序。根据安装数据库系统的操作系统，可以使用脚本或通过调度工具或服务来手工启动和停止这两个程序。部署期间，会创建随时可用的启动和停止脚本。在 Windows® 系统上，这些脚本都是批处理文件。在 UNIX® 系统上，它们是 shell 脚本。

需要在包含源数据库（对于状态至运行时数据移动服务而言是状态数据库，对于运行时至历史的数据移动服务而言是运行时数据库）的 DB2 所在的系统上启动每个“捕获”程序。“捕获”组件必须拥有对 DB2 日志文件的本地访问权。

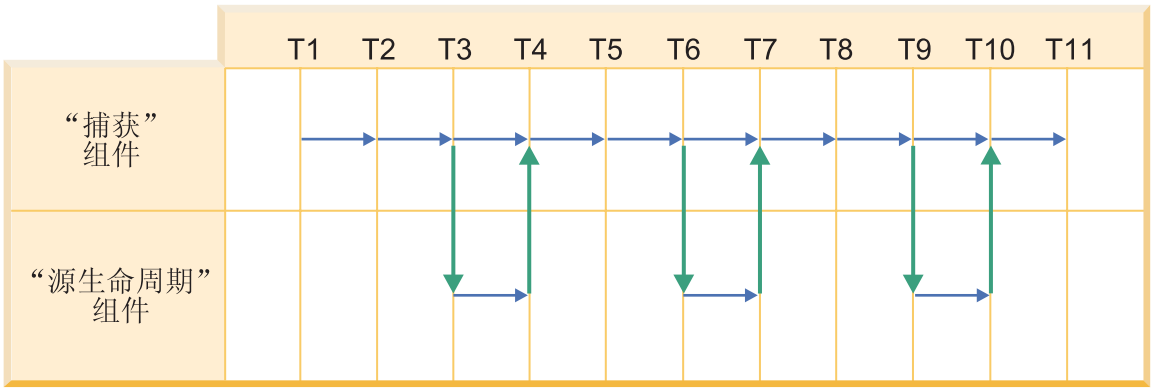
注: 即使“捕获”实例没有运行，也将捕获源数据库中所有的数据更改。然而，在所有实例都可操作之前，不会处理捕获的任何更改并将其应用于目标数据库表。

数据移动服务配置

可以配置每个数据移动服务组件的行为和调度，以满足开发、测试和生产环境的不同需求。更改某一组件的配置可能会直接影响依赖该组件的其他组件的行为。

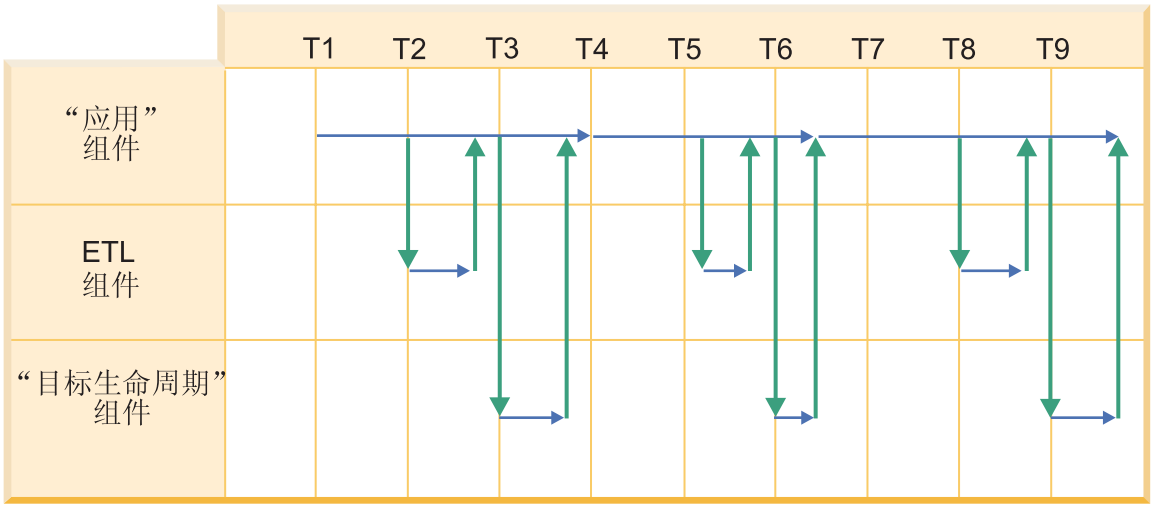
通常，有两种依赖性:

- “捕获”组件定期调用“源生命周期”组件。如果“捕获”组件没有运行，则不会执行源生命周期强制。生命周期组件调用之间的延迟是可配置的。

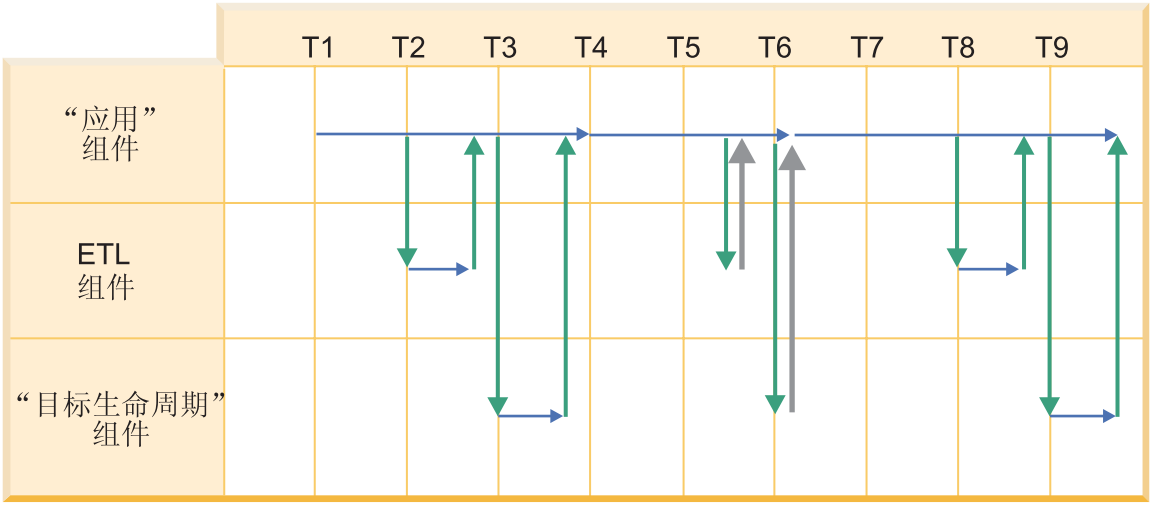


上图中，每 3 个时间单位调用一次“源生命周期”组件，它将执行某项活动，然后将控制返回到“捕获”组件，由“捕获”组件继续处理。

- 在成功将数据从源数据库移到目标数据库之后，由“应用”组件来调用 ETL 组件和“目标生命周期”组件。只有在“应用”组件运行时才调用 ETL 和“目标生命周期”组件。



由于依赖组件的运行调度与被依赖组件不同，所以调用未必会产生执行。每个依赖组件在被调用时会检查其调度，如果执行任务的时间还没到，则将控制返回给调用组件。上例中，如果 ETL 和“目标生命周期”组件的调度限制使对它们的调用每 5 时间单位不能超过一次，则这两个组件可能只执行两次。



如果在 T2（或 T3）调用和执行 ETL 组件（或“目标生命周期”组件）。下一次调用大约在 T6 时发生。由于离上一次执行不到 5 个时间单位，因此控制被立即返回给“应用”组件。如果后续调用大致发生在 T8（或 T9）时，将产生执行，因为离上一次执行已超过 5 个时间单位。每个组件都由一个或多个组件实例来实施。可以单独配置每个实例，以实现更细微的控制。

注：如果进行了更改，除非另外声明，否则将立即生效。

可以通过更改相应的控制表或使用启动脚本中的命令行参数覆盖它们，来修改“捕获”和“应用”组件的缺省配置。可以通过更新其中一个控制表，来配置 ETL 和生命周期强制组件。

执行以下步骤来定制数据移动服务组件，以满足开发、测试和生成环境的需求。

配置（源）“捕获”组件实例

“捕获”组件等同于 DB2“捕获”复制实用程序。缺省情况下，配置该实用程序来不断捕获对源表所做的更改，并将这些更改记录到内部工作表中。通常，不需要更改“捕获”组件实例的缺省配置。

- **识别“捕获”组件实例**。

  使用多个“捕获”组件实例（即 DB2 捕获实用程序）来捕获与业务度量模型相关的数据。要确定已指定服务于某一业务度量模型的“捕获”实用程序，您必须:

  – 识别要更改其“捕获”实用程序配置的数据移动服务。
  – 检查状态数据库（用于状态至运行时数据移动服务）或运行时数据库（用于运行时至历史的数据移动服务）中的 WBIRMADM.RMMETADATA 元数据表，并识别所有“捕获”实用程序名称（SRC_RM_CAP_SVR_NAME 列）。

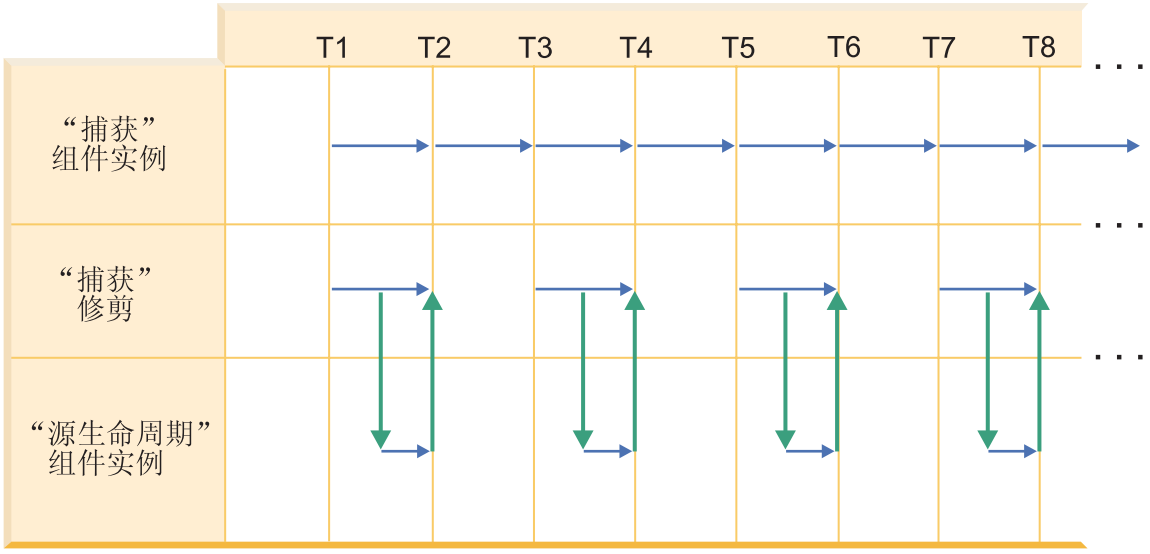
    示例：查询“SELECT OM_NAME, SRC_TAB_NAME, SERVICE_NAME, SRC_RM_CAP_SVR_NAME FROM WBIRMADM.RMMETADATA WHERE SERVICE_NAME=’State to Runtime’”可能会导致以下情况:

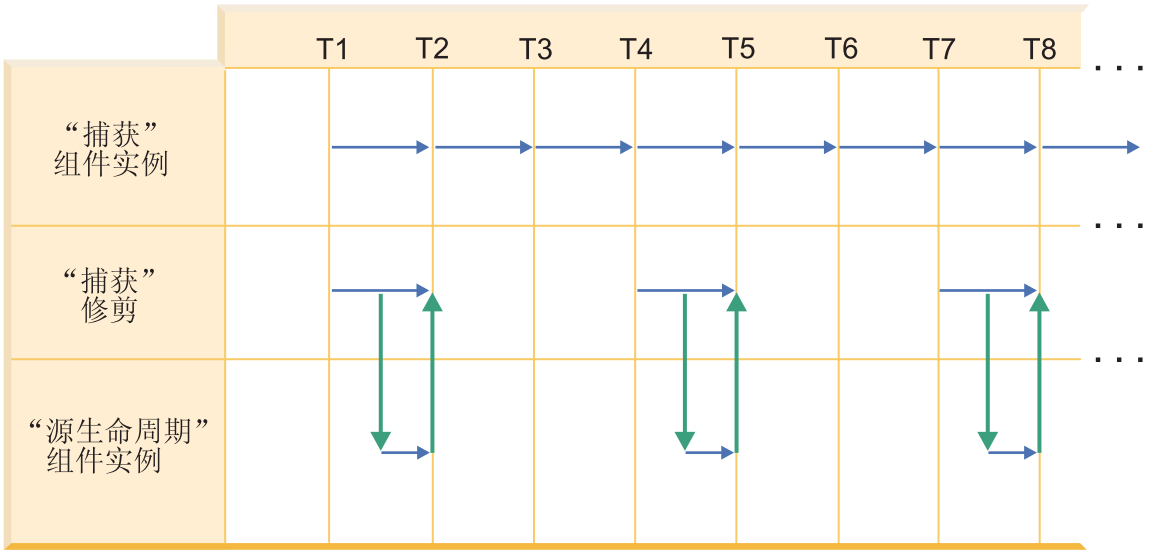
OM_NAME	SRC_TAB_NAME	SERVICE_NAME	SRC_RM_CAP_SVR_NAME
STEW_S	wbi.CTX_TQ4MUFT42JOT5F6R3KSDQDE2UI	State to Runtime	CAPTURE_1
STEW_S	wbi.AI_BVSOYAP1DRWFD5HNQJR5HFQQQE	State to Runtime	CAPTURE_1

上例中，指定用“捕获”实用程序 CAPTURE_1 捕获对状态数据库中业务度量模型 STEW_S 的两个相关源表的所有更改。

- **更改“捕获”工作表的修剪时间间隔。**

  如果启用自动修剪（autoprune 参数等于“y”），“捕获”实用程序会每隔 300 秒（prune_interval 参数的缺省值）自动修剪其工作表。每个修剪活动都将自动产生“源生命周期”组件实例的调用，该实例由数据库触发器实施。更改“捕获”实用程序的修剪时间间隔参数会直接影响“源生命周期”组件修剪源表的频率。以下说明了更改“捕获”的修剪时间间隔对“源生命周期”组件实例的调用有何影响。





将“捕获”实例 prune_interval 参数从 2 个时间单位（如 300 秒）增加到 3 个时间单元（如 450 秒），将会导致:

– “捕获”工作表中可删除的行在工作表中保留更长时间，从而提高潜在的空间需求。工作表将变大，而系统负载和资源短缺风险会降低。
– 源表中根据生命周期保留时间策略可删除的行在源表中保留的时间比预期更长。

通常，如果将“捕获” prune_interval 参数设置为大于生命周期组件的 prune_interval 参数的值，则将优先采用“捕获”参数设置。如果未运行“捕获”实用程序，或禁用了其自动修剪功能，将不会执行源生命周期强制。

配置“源生命周期”组件

在每个源数据库（状态和运行时数据库）中使用多个生命周期组件实例。由触发器实施的每一个实例都强制执行保留时间策略，这些策略是在源数据库上的控制表 WBIRMADM.RMPRUNECTRL 中为该数据移动服务定义的。生命周期保留时间策略是以表为单位指定的。因此，WBIRMADM.RMPRUNECTRL 中的一行对应于一个需要修剪的表。

TABLE_NAME	RETEN...	LAST_PRUNED	PRUNE_IN...	PRUNE_EN...	LOG...	ROW...
wbi.CTR_TQ4MUFT42JOT5F6R3KSDQDE2UI	1440	Oct 11, 2005 4:40:...	5	1	0	0
wbi.AIR_BVSOYAP1DRWFD5HNQJR5HFQQQE	1440	Oct 11, 2005 4:40:...	5	1	0	0

- **识别“源生命周期”组件实例。**

  要确定已指定了哪个触发器来加强给定业务度量模型的保留时间策略，您必须:
  – 识别要修改其 ETL 配置的数据移动服务。
  – 检查状态数据库（用于状态至运行时数据移动服务）或运行时数据库（用于运行时至历史的数据移动服务）中的 WBIRMADM.RMMETADATA 表，并在 SRC_RM_PRUNE_TRG_NAME 列中，查询相关触发器名称。

  示例: 查询“SELECT OM_NAME, SRC_TAB_NAME, SERVICE_NAME, SRC_RM_PRUNE_TRG_NAME FROM WBIRMADM.RMMETADATA WHERE SERVICE_NAME=’State to Runtime’”可能会导致以下情况:

OM_NAME	SRC_TAB_NAME	SERVICE_NAME	SRC_RM_PRUNE_TRG_NAME
STEW_S	wbi.CTX_TQ4MUFT42JOT5F6R3KSDQDE2UI	State to Runtime	WBIRMADM.MCPruneTrig_8
STEW_S	wbi.AI_BVSOYAP1DRWFD5HNQJR5HFQQQE	State to Runtime	WBIRMADM.MCPruneTrig_9

  本例中，两个触发器（WBIRMADM.MCPruneTrig_8 和 WBIRMADM.MCPruneTrig_9）为状态数据库中业务度量模型 STEW_S 源表强制执行生命周期保留时间策略。由于保留时间策略是根据表来定义的，而不是根据生命周期组件实例名称定义的，因此在打算更改生命周期强制行为时，请查询 SRC_TAB_NAME 列。

- **修改“源生命周期”组件实例配置。**
  – 启用和禁用生命周期组件实例:

  修剪会严重影响系统性能。在启用修剪时，它会减少事务服务器（状态）和报告服务器（运行时）需要处理的信息量。根据生命周期组件的参数，它还会在每个调用期间给相同的表添加少量额外负载。禁用修剪时，源表会随着时间的增长而增长，这也会降低性能。

在缺省情况下，根据源表的生命周期保留时间策略自动修剪源表。要临时禁用修剪，可修改相应的 WBIRMADM.RMPRUNECTRL 项：将列 PRUNE_ENABLED 设置为 1 以启用修剪，或设置为任何其他数值（最好是 0）以禁用修剪。

如果使用以下配置，则将修剪源表 wbi.CTX_TQ4MUFT42JOT5F6R3KSDQDE2UI 中的行，但不会修剪表 wbi.AI_BVSOYAP1DRWFD5HNQJR5HFQQQE 中的行。查询："SELECT TABLE_NAME, PRUNE_ENABLED FROM WBIRMADM.RMPRUNECTRL"可能会导致以下情况：

TABLE_NAME	PRUNE_ENABLED
wbi.CTX_TQ4MUFT42JOT5F6R3KSDQDE2UI	1
wbi.AI_BVSOYAP1DRWFD5HNQJR5HFQQQE	0

– 更改保留时间策略：

只能更改位于运行时数据库中的源表的保留时间策略。对于所有位于状态数据库中的表，不管 WBIRMADM.RMPRUNECTRL 中的设置如何，保留期强制为 0。保留期定义为源表中的行在可删除（如果满足两个条件）前必须保留的最小时间。这两个条件中，只有一个是可通过控制表来定制的：以分钟为单位的保留时间。任何已标记为可删除且已在源表中保留了至少 RETENTION_IN_MINUTES 的行都可以删除。

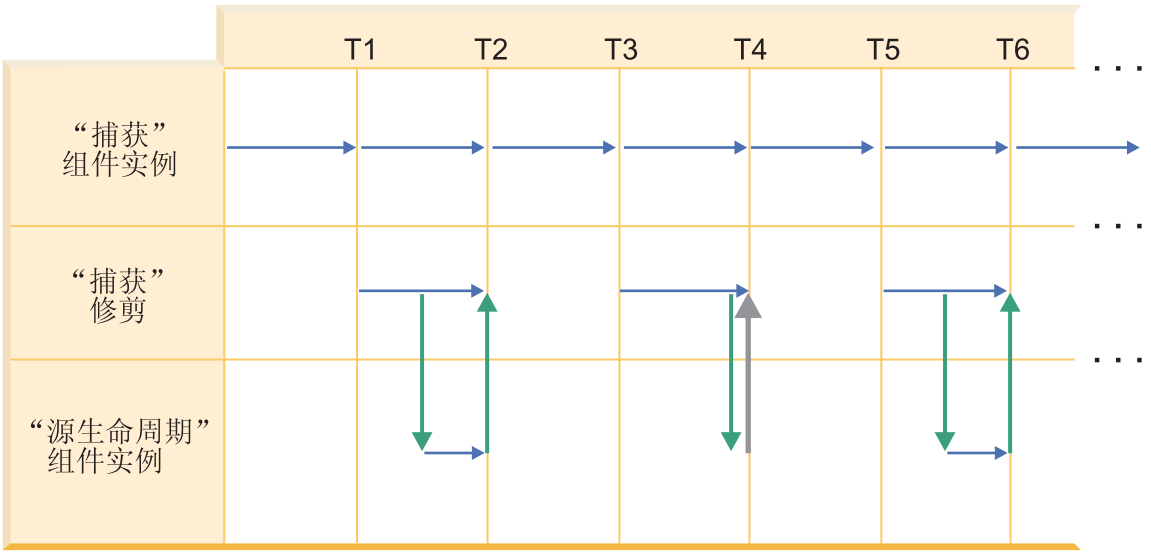
使用运行时数据库中源表的缺省配置，那些已由服务器标记为可删除的行在删除之前，需要保留一天（1440 分钟）。查询："SELECT TABLE_NAME, RETENTION_IN_MINUTES FROM WBIRMADM.RMPRUNECTRL"可能会导致以下情况：

TABLE_NAME	RETENTION_IN_MINUTES
wbi.CTR_TQ4MUFT42JOT5F6R3KSDQDE2UI	1440
wbi.AIR_BVSOYAP1DRWFD5HNQJR5HFQQQE	1440

每次调用"源生命周期"组件时，都会获取对 WBIRMADM.RMPRUNECTRL 控制表条目的更改。

– 源数据修剪调度：

在"捕获"工作表修剪时间间隔和"源生命周期"组件调用之间存在依赖关系。如果"源生命周期"组件实例调用之间没有间隔足够的时间，则调用不会产生执行，如下图所示。



假设将“源生命周期”组件安排为每 4 个时间单位执行一次，而将“捕获”组件配置为每 2 个时间单位修剪一次工作表，则 T4 时刻的调用将不会产生执行。

要更改缺省调度，请定位 WBIRMADM.RMPRUNECTRL 中相应的条目，并更改 PRUNE_INTERVAL 的列值，它表示执行之间的最小延迟（以分钟计）。

TABLE_NAME	LAST_PRUNED	PRUNE_INTERVAL
wbi.CTX_TQ4MUFT42JOT5F6R3KSDQDE2UI	Oct 11, 2005 5:16:44 PM ...	5
wbi.AI_BVSOYAP1DRWFD5HNQJR5HFQQQE		5

增加值将使执行次数减少（但调用次数保持不变）。每次执行都会确定哪些源表行可以删除，并删除它们。定期监控源数据库，以识别和消除由于这些删除而导致的死锁所引起的潜在性能问题。

配置（目标）“应用”组件

“应用”组件的一个实例是 DB2 “应用” 复制实用程序。缺省值情况下，由“捕获”实用程序所捕获的更改被连续应用到目标数据库中的登台表。缺省实用程序配置参数对于大多数环境而言都是足够的，因此不必更改。

- **识别“应用”组件实例。**

  使用多个“应用”组件实例（DB2 “应用”实用程序）将所有数据更改应用到与业务度量模型相关的内部登台表。要确定已指定将哪个“应用”实用程序用于为业务度量模型提供服务:

  – 识别要更改其“应用”实用程序配置的数据移动服务。
  – 检查运行时数据库（用于状态至运行时数据移动服务）或历史数据库（用于运行时至历史的数据移动服务）中的 WBIRMADM.RMMETADATA 元数据表，并识别所有“应用”实用程序名称（TGT_RM_APP_SVR_NAME 列）。查询: “SELECT OM_NAME, SRC_TAB_NAME, SERVICE_NAME, TGT_RM_APP_SVR_NAME FROM WBIRMADM.RMMETADATA WHERE SERVICE_NAME=’State to Runtime’”可能会导致以下情况:

OM_NAME	SRC_TAB_NAME	SERVICE_NAME	TGT_RM_APP_SVR_NAME
STEW_S	wbi.CTX_TQ4MUFT42JOT5F6R3KSDQDE2UI	State to Runtime	APPLY_4
STEW_S	wbi.AI_BVSOYAP1DRWFD5HNQJR5HFQQQE	State to Runtime	APPLY_4

本例中，在状态数据库中捕获的任何对业务度量模型 STEW_S 的数据更改都将通过“应用”实用程序 APPLY_4 应用于运行时数据库中的登台表。

每当“应用”组件处理完“捕获”实用程序记录的所有（已落实）更改时，会调用一个或更多 ETL 组件实例和“目标生命周期”组件实例。

配置 ETL 组件

ETL 组件已在 WebSphere Business Monitor 中作为数据库存储过程来实施。对于任何给定数据移动服务，这些存储过程总是驻留在目标数据库上。因此，所有分配给状态至运行时数据移动服务的 ETL 存储过程都位于运行时数据库中，而分配给运行时至历史的数据移动服务的 ETL 存储过程都驻留在历史数据库中。

- **识别 ETL 组件实例**。

  设置多个 ETL 组件实例，以处理添加到与业务度量模型相关的内部登台表中的数据。要确定已指定用于为给定业务度量模型提供服务的存储过程:

  – 识别要修改其 ETL 配置的数据移动服务。
  – 检查运行时数据库（用于状态至运行时数据移动服务）或历史数据库（用于运行时至历史的数据移动服务）中的 WBIRMADM.RMMETADATA 元数据表，并识别所有 ETL 存储过程名称（TGT_RM_SPETL_NAME 列）。以下查询: “SELECT OM_NAME, SRC_TAB_NAME, TGT_TAB_NAME, SERVICE_NAME, TGT_RM_SPETL_NAME FROM WBIRMADM.RMMETADATA WHERE SERVICE_NAME=’State to Runtime’”可能会导致以下情况:

OM_NAME	SRC_TAB_NAME	TGT_TAB_NAME	SERVICE_NAME	TGT_RM_SPETL_NAME
STEW_S	wbi.CTX_TQ4MUFT42JOT5F6R3KSDQDE2UI	wbi.CTR_TQ4MUFT42JOT5F6R3KSDQDE2UI	State to Runtime	WBIRMADM.WBIRMSP_10
STEW_S	wbi.AI_BVSOYAP1DRWFD5HNQJR5HFQQQE	wbi.AIR_BVSOYAP1DRWFD5HNQJR5HFQQQE	State to Runtime	WBIRMADM.WBIRMSP_14

本例中，由名为 WBIRMADM.WBIRMSP_10 和 WBIRMADM.WBIRMSP_14 的存储过程处理在状态数据库中捕获并应用于运行时数据库中登台表的业务度量模型 STEW_S 的数据更改。成功处理的数据将保存在运行时数据库中的目标表（由 TGT_TAB_NAME 列标识）。

- **修改 ETL 组件实例配置**。

  ETL 组件实例配置存储在 WBIRMADM.RMCONTROL 控制表中。已指定给状态至运行时数据移动服务的实例在运行时数据库中维护其配置；所有其他实例在历史数据库中维护其配置。对配置的任何更改都将在下一次启动时由存储过程获取。通过控制表可配置三种选项:

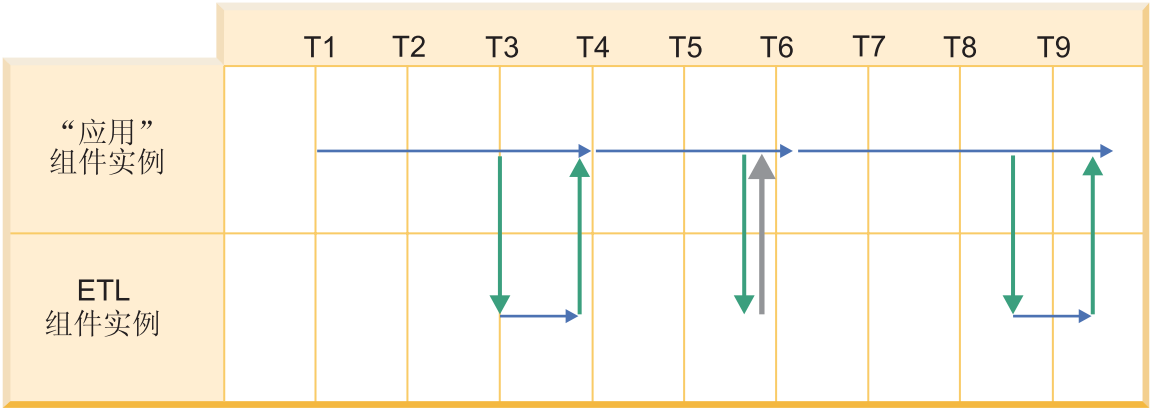
  – 两次 ETL 执行之间的最小耗用时间（ETLSCHEDMETHOD, ETL_0_MINUTES）
  – 日志记录输出的详细程度（LOGLEVEL）
  – 事务持续时间（COMMITINTERVAL）

  该表中的每一行对应于一个 ETL 组件实例，而每个 ETL 组件实例则对应于一个需要填充的目标表。以下示例配置说明了对配置的更改将如何影响实例行为。

TARGETTABLE	TGT_RM_SPETL_NAME	ETLSCHEDMETHOD	ETL_0_MINUTES	LOGLEVEL	COMMITINTERVAL
wbi.CTR_TQ4MUFT42JOT5F6R3KSDQDE2UI ...	WBIRMADM.WBIRMSP_10	1	5	0	1000
wbi.AIR_BVSOYAP1DRWFD5HNQJR5HFQQQE ...	WBIRMADM.WBIRMSP_14	1	5	0	1000

- **更改 ETL 调度。**

  每次“应用”组件实例完成对预订集的处理时，都会调用 ETL 组件实例。一经调用，ETL 实例会检查其调度，然后启动处理或立即将控制返回到“应用”组件实例。它使用存储在控制表 WBIRMADM.RMCONTROL 中的信息，来确定是否需要执行。下图显示了调用和执行之间的区别：根据调度，第一次调用和第三次调用时执行 ETL 组件实例。第二次调用发生在调度以外，并不会导致任何处理活动。



  决定 ETL 组件实例在状态至运行时数据移动服务中和运行时至历史的数据移动服务中运行频率的因素有多种:

  – 可用性：目标表中的数据隔多久可访问。选择较小的时间间隔会使数据较早可用，但也会增加系统负载。
  – 数据量：复制实用程序不间断地（或根据配置）将数据输入登台表，而不管是否由 ETL 组件实例来处理它。使用的数据库资源越多，需要处理的数据就越多。更频繁地处理数据会降低最大资源使用率。
  – 处理时间：ETL 在运行时数据库中处理数据的时间少于在历史数据库中处理数据的时间。请据此规划调度。如果执行的持续时间大于调度延迟，则缩小执行之间的延迟不会提高性能。例如，如果 ETL 组件实例需要 60 秒完成处理，则 30 秒的调度时间间隔实际上变成至少 60 秒的调度时间间隔，因为 ETL 组件实例是连续执行的。

  当前受支持的调度方式有两种:

  – 灵活的调度:

    如果自上一次执行（LASTUPDATED）以来至少经过了 ETL_0_MINUTES，则执行 ETL 实例。例如，假设控制表包含以下信息。

TGT_RM_SPETL_NAME	TARGETTABLE	LASTUPDATED	ETL_0_MINUTES
WBIRMADM.WBIRMSP_10	wbi.CTR_TQ4MUFT42JOT5F6R3KSDQDE2UI ...	Oct 11, 2005 5:20:20 PM ...	60
WBIRMADM.WBIRMSP_14	wbi.AIR_BVSOYAP1DRWFD5HNQJR5HFQQQE ...	Oct 11, 2005 5:20:20 PM ...	60

    存储过程 WBIRMADM.WBIRMSP_10 将在 2005 年 10 月 11 日下午 6 点 20 分 20 秒（2005 年 10 月 11 日下午 5 点 20 分 20 秒 + 60 分钟）以后执行。

如果存储过程在 2005 年 10 月 11 日下午 6 点 20 分 20 秒以后调用，则调度可改变。假设当前时间为下午 7 点而存储过程未在预期的下午 6 点 20 分执行。则将（在大约 40 分钟后）调用和执行该存储过程。在下午 7 点 + 60 分钟 = 下午 8 点之前，它不会再执行。实际调度改变了，因为将 ETL 过程调整在整点过后的 20 分钟每 60 分钟运行一次，而现在是在整点每 60 分钟运行一次。如果需要，您可以通过更改 LASTUPDATED 列中的时间戳记值，来重新设置调度。

如果不需要固定的执行时间段，请使用这一调度机制。要启用该调度形式，对于已指定到同一业务度量组的存储过程，请将 WBIRMADM.RMCONTROL 中的 ETLSCHEDMETHOD 列设置为 0:

TGT_RM_SPETL_NAME	TARGETTABLE	ETLSCHEDMETHOD
WBIRMADM.WBIRMSP_10	wbi.CTR_TQ4MUFT42JOT5F6R3KSDQDE2UI ...	0
WBIRMADM.WBIRMSP_14	wbi.AIR_BVSOYAP1DRWFD5HNQJR5HFQQQE ...	0

– 固定的调度:

它是所有 ETL 组件的缺省调度。如果当前时间超过 NEXTSTARTTIME，则执行 ETL 组件实例。为了避免调度变化，可在每次存储过程执行时，根据当前时间和先前调度的执行时间计算下一次调度的执行时间。以下示例说明了这个调度:

TGT_RM_SPETL_NAME	TARGETTABLE	LASTUPDATED	ETL_0_MINUTES	NEXTSTARTTIME
WBIRMADM.WBIRMSP_10	wbi.CTR_TQ4MUFT42JOT5F6R3KSDQDE2UI ...	Oct 11, 2005 5:20:20 PM ...	60	Oct 11, 2005 6:20:20 PM ...
WBIRMADM.WBIRMSP_14	wbi.AIR_BVSOYAP1DRWFD5HNQJR5HFQQQE ...	Oct 11, 2005 5:20:20 PM ...	60	Oct 11, 2005 6:20:20 PM ...

假设当前时间是下午 7 点，且没有按预期在下午 6 点 20 分执行存储过程。由于过了当天的 NEXTSTARTTIME（下午 6 点 20 分），所以执行存储过程。根据初始调度，将下一次执行安排在下午 7 点 20 分，而不是灵活调度方案中的下午 8 点。如果存储过程必须在预定义的时间段内启动执行，则使用这种调度方式。要启用该调度形式，对于所有已分配给相同业务度量组的存储过程，将 WBIRMADM.RMCONTROL 中的 ETLSCHEDMETHOD 列设置为 1:

TGT_RM_SPETL_NAME	TARGETTABLE	ETLSCHEDMETHOD
WBIRMADM.WBIRMSP_10	wbi.CTR_TQ4MUFT42JOT5F6R3KSDQDE2UI ...	1
WBIRMADM.WBIRMSP_14	wbi.AIR_BVSOYAP1DRWFD5HNQJR5HFQQQE ...	1

由于这些实例之间存在依赖关系，强烈建议您对隶属于同一业务度量模型的 ETL 组件实例使用相同的调度方式。这对于历史数据库中的调度和时间间隔长（几小时或更长时间）的调度尤其重要。将 ETLSCHEDMETHOD 设为 0 或 1 以外的其他值，会禁用 ETL 组件实例。

- **更改日志记录级别**。

存储过程支持两个日志记录级别: 最小（0）和最大（1）。要修改缺省的最小设置，请在 WBIRMADM.CONTROL 中更改需要更改日志记录级别的存储过程（TGT_RM_SPETL_NAME）的 LOGLEVEL 列的值。将所有日志记录输出附加到 WBIRMADM.RMLOG。这两个示例存储过程 WBIRMADM.WBIRMSP_10 和 WBIRMADM.WBIRMSP_14 都会执行最小日志记录:

ENTRYSTMP	NAME	OPERATION	ISTRACEENTRY	ID
Oct 11, 2005 4:40:20 PM ...	WBIRMADM.WBIRMSP_14	SP_START	0	
Oct 11, 2005 4:40:20 PM ...	WBIRMADM.WBIRMSP_14	DEL_TEMP	0	
Oct 11, 2005 4:40:20 PM ...	WBIRMADM.WBIRMSP_14	INS_TEMP	0	
Oct 11, 2005 4:40:20 PM ...	WBIRMADM.WBIRMSP_14	FETCH_TARGET_...	0	
Oct 11, 2005 4:40:20 PM ...	WBIRMADM.WBIRMSP_14	SP_END	0	
Oct 11, 2005 4:40:20 PM ...	WBIRMADM.WBIRMSP_10	SP_START	0	
Oct 11, 2005 4:40:20 PM ...	WBIRMADM.WBIRMSP_10	DEL_TEMP	0	

由于不会自动修剪日志表，因此需要由 DBA 定期监控。除非另有说明，否则应使日志记录输出保持最小。

- **更改事务持续时间**。

  存储过程成功处理的数据将立即写入目标表。然而，根据落实时间间隔设置（WBIRMADM.RMCONTROL 中的 COMMITINTERVAL 列），在处理了指定的行数或没有更多行需要处理之前，对目标表所做的任何更新都不是永久的。增加 COMMITINTERVAL 的值（如，增加到 1500）将使存储过程在落实更改之前处理更多数据。对目标表的锁定将持续更长时间，并可能对其他试图访问同一表的组件产生消极影响。减少持续时间（譬如减少到 500）可以减少那些在目标表中可用前需要处理的行的数目，并可以更早释放锁定。

TARGETTABLE	TGT_RM_SPETL_NAME	ETL_0_MINUTES	LOGLEVEL	COMMITINTERVAL
wbi.CTR_TQ4MUFT42JOT5F6R3KSDQDE2UI ...	WBIRMADM.WBIRMSP_10	5	0	1500
wbi.AIR_BVSOYAP1DRWFD5HNQJR5HFQQQE...	WBIRMADM.WBIRMSP_14	5	0	500

配置“目标生命周期”组件。

只要“应用”组件实例应用新的或已更新的数据，ETL 工作表就会不断增长。将一个由存储过程实施的“目标生命周期”组件实例分配给每个目标（运行时和历史）数据库中的一个工作表。每个实例都会强制执行 WBIRMADM.RMPRUNECTRL 控制表中定义的内部保留时间策略。与在源表中一样，ETL 工作表的生命周期保留时间策略是以表为单位指定的。因此，WBIRMADM.RMPRUNECTRL 中的一行对应于一个需要修剪的表。

- **识别“目标生命周期”组件实例**。

  要确定分配了哪些存储过程来强制执行给定业务度量模型的 ETL 工作表保留时间策略:

  – 识别要修改其 ETL 配置的数据移动服务。
  – 检查运行时数据库（用于状态至运行时数据移动服务）或历史数据库（用于运行时至历史的数据移动服务中的 WBIRMADM.RMMETADATA，在下列表的 TGT_RM_APP_PRUNE_SP_NAME 列中查找相关存储过程。

OM_NAME	SRC_TAB_NAME	TGT_RM_APP_STG_TAB_NAME	TGT_RM_APP_PRUNE_SP_NAME	SERVICE_NAME
STEW_S	wbi.CTX_TQ4MUFT42JOT5F6R3KSDQDE2UI	APP.CCD_6	WBIRMADM.WBIRMSP_P_13	State to Runtime
STEW_S	wbi.AI_BVSOYAP1DRWFD5HNQJR5HFQQQE	APP.CCD_7	WBIRMADM.WBIRMSP_P_17	State to Runtime

本例中，两个存储过程（WBIRMADM.WBIRMSP_P_13 和 WBIRMADM.WBIRMSP_P_17）会强制执行运行时数据库中的业务度量模型 STEW_S ETL 工作表的生命周期保留时间策略。由于保留时间策略是通过表而不是通过生命周期组件实例名称定义的，所以在打算更改生命周期强制行为时，请参照 TGT_RM_APP_STG_TAB_NAME 列。

- **修改目标生命周期组件实例的配置**。

  缺省配置应该适合于大多数部署，但也可以如以下所述进行微调:

  – 启用和禁用“目标生命周期”组件实例。

    在缺省情况下，将根据 ETL 工作表的生命周期保留时间策略，对其进行自动修剪。要临时禁用修剪，可修改相应的 WBIRMADM.RMPRUNECTRL 项: 将列 PRUNE_ENABLED 设置为 1 以启用修剪，或设置为任何其他数值（最好是 0）以禁用修剪。如果运行时数据库中的控制表 WBIRMADM.RMPRUNECTRL 包含以下条目，则将自动修剪两个 ETL 工作表:

TABLE_NAME	PRUNE_ENABLED
APP.CCD_6	1
APP.CCD_7	1

    在禁用任何“目标生命周期”组件实例之前，请确保在相关的表空间容器中有足够的空间可用。每当监控器服务器更新源表中的任意行时，就会在 ETL 工作表中添加一行。因此，源表中的一行可由工作表中的多行来临时表示，这使工作表比源表增长得更快。在下一次调用“目标生命周期”组件实例时，将获取对 WBIRMADM.RMPRUNECTRL 的更改。

  – 更改保留时间策略。

    可以从工作表删除所有已由 ETL 组件实例成功处理的行。运行时和历史数据库的缺省保留期都设为 0 分钟:

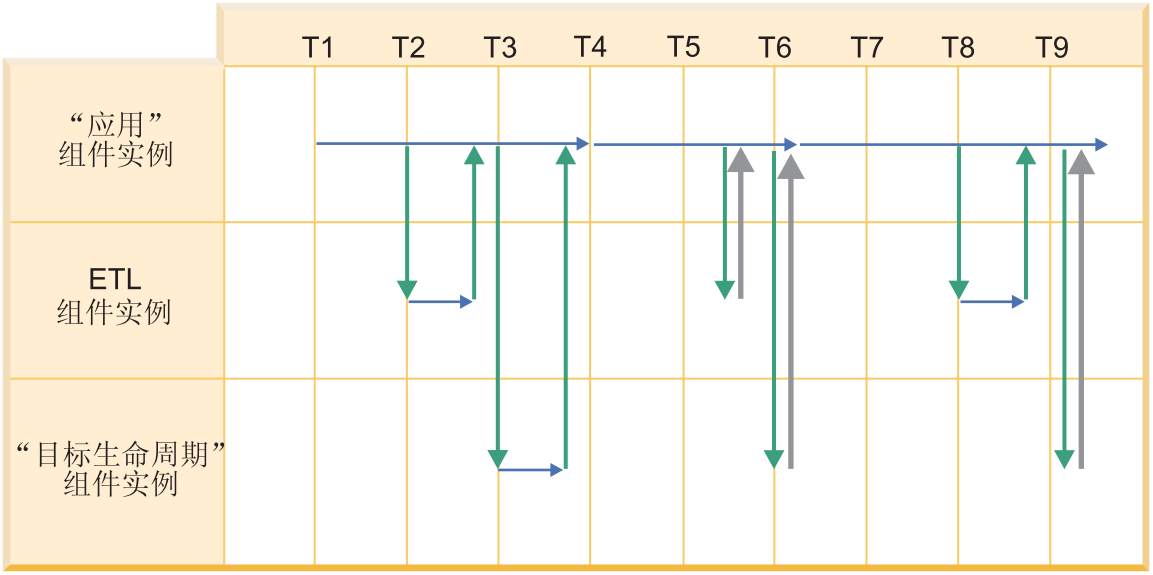
TABLE_NAME	RETENTION_IN_MINUTES
APP.CCD_6	0
APP.CCD_7	0

    在下一次调用“目标生命周期”组件实例时，将删除所有符合条件的行。将保留期设置为 0，并不保证会立即进行删除，这是因为执行生命周期组件的时间由调度确定。

    用户可通过将 WBIRMADM.RMPRUNECTRL 中的 RETENTION_IN_MINUTES 列更改为另一个持续时间（以分钟计），从而将数据保留在表中。

  – 调度 ETL 工作表修剪。

    “目标生命周期”组件调度的原理与 ETL 组件调度非常相似。在“应用”周期和所有相关的 ETL 组件实例完成之后，随后会调用“目标生命周期”组件实例。



如果启用了工作表修剪，且调度为运行，则调用将产生执行。上例中，“目标生命周期”组件实例只在 T3 时执行一次。执行未在 T6 和 T9 时刻发生的原因有几种:

- 在 T4 与 T6 之间，“目标生命周期”组件实例配置被更改，并禁用了修剪。
- 在 T3 与 T9 之间的耗用时间少于为该“目标生命周期”组件实例指定的时间间隔。

注: 本例中，ETL 组件实例和“目标生命周期”组件实例的调度是不同的（假设未禁用修剪）。

通常，对所有相应的实例使用相同的调度，或对生命周期实例使用周期更长的调度。原因是: 在 ETL 组件实例成功处理数据之前，将无法修剪数据。要更改缺省调度，请在 WBIRMADM.RMPRUNECTRL 中找到相应的条目，并更改 PRUNE_INTERVAL 的列值，它代表执行之间的最小延迟（以分钟计）:

TABLE_NAME	LAST_PRUNED	PRUNE_INTERVAL
APP.CCD_6	Oct 11, 2005 4:40:20 PM ...	1440
APP.CCD_7	Oct 11, 2005 4:40:20 PM ...	1440

增加修剪时间间隔会减少执行次数，而调用次数保持不变。每次执行会确定哪些工作表行可删除，并会删除它们。如果使用上述配置，则大概每天（1440 分钟）修剪一次工作表 APP.CCD_6 和 APP.CCD_7。上一次执行修剪的时间是 10 月 11 日下午 4 点 40 分，并将在 10 月 12 日下午 4 点 40 分钟左右再次执行。

数据移动服务配置参数摘要

下表概括每个数据移动服务组件的最常用参数。要获取关于配置参数的更多信息，请参阅 DB2 复制指南。

组件	参数名称	缺省值	有效值	参数位置
捕获	autoprune	Y		
捕获	prune_interval（秒）	300		
源生命周期	PRUNE_ENABLED	1	0 - 禁用 1 - 启用	数据移动服务源 DB: WBIRMADM.RMPRUNECTRL

下表概括每个数据移动服务组件的最常用参数。要获取关于配置参数的更多信息，请参阅 DB2 复制指南。

组件	参数名称	缺省值	有效值	参数位置
源生命周期	RETENTION_IN_MINUTES	0 – 状态至运行时 1440 – 运行时至历史	0 到 BIGINT 的 DB2 限制	数据移动服务源 DB: WBIRMADM.RMPRUNECTRL
源生命周期	PRUNE_INTERVAL（分钟）	5	0 到 BIGINT 的 DB2 限制	数据移动服务源 DB: WBIRMADM.RMPRUNECTRL
ETL	ETLSCHEDMETHOD	1	0 – 灵活的调度 1 – 严格的时间间隔调度 其他 – 禁用 ETL	数据移动服务目标 DB: WBIRMADM.RMCONTROL
ETL	ETL_0_MINUTES	5 – 状态至运行时 1440 – 运行时至历史	0 到 INTEGER DB2 限制	数据移动服务目标 DB: WBIRMADM.RMCONTROL
ETL	LOGLEVEL	0	0 – 用于常规记录 1 – 用于跟踪记录	数据移动服务目标 DB: WBIRMADM.RMCONTROL
ETL	COMMITINTERVAL（记录数）。	1000	0 – 结束之前禁用落实 1 – 落实每个记录。 n 到 BIGINT 的 DB2 限制	数据移动服务目标 DB: WBIRMADM.RMCONTROL
目标生命周期	PRUNE_ENABLED	1	0 – 禁用 1 – 启用	数据移动服务目标 DB: WBIRMADM.RMPRUNECTRL
目标生命周期	RETENTION_IN_MINUTES	0	0 到 BIGINT 的 DB2 限制	数据移动服务目标 DB: WBIRMADM.RMPRUNECTRL
目标生命周期	PRUNE_INTERVAL（分钟）	1440	0 到 BIGINT 的 DB2 限制	数据移动服务目标 DB: WBIRMADM.RMPRUNECTRL

注: IBM® 保留对上述数据库表和列的更改权。因此，在不同的发行版中，可能会更改、删除或添加某些表和列。在信息的不同发行版中所引用的内容或结构的可靠性，由使用者自行确保。IBM 将在此类更改出现时加以记录。

变更管理和工件生成

业务度量模型包含许多事件和过程定义。根据这些定义，模式生成器会生成创建数据库表、Cube Views 定义和复制脚本所需的相应工件。对业务度量模型的更改会导致生成的工件发生更改。

如果发生这样的更改，您需要重新运行模式生成器，生成新的业务度量模型脚本。该活动称为变更管理。

在以下情况下需要变更管理:

- 添加新过程导致添加新表。
- 添加新度量（不是维的一部分）或在新维中创建新度量，以及将新列添加到相应的过程表。

- 字符串类型的度量的长度更改，导致对应列的长度更改。

对业务度量模型的任何更改都需要重复执行以下步骤:

- 将更改的业务度量模型导入到 WebSphere Business Monitor 管理控制台中的模式生成器，以生成相应的工件。
- 运行最新生成的数据库定义语言（DDL）脚本，用这些更改来更新数据库。
- 部署复制脚本以在更改后同步状态、运行时和历史数据库。
- 部署最近生成的 Cube Views 定义。
- 将业务度量模型部署到 WebSphere Business Monitor 管理控制台。

模式生成器会检查业务度量模型的先前版本和新版本。如果新模型尚未部署或不存在于存储库中，则模式生成器会生成与新模型对应的工件。如果存在新模型的先前版本，则模式生成器会检查已部署模型版本和新模型版本之间的差异。如果发现差异，则会根据这些差异重新生成相应的脚本以修改数据库。要获取更多关于由现有模型创建新模型的信息，请参阅 WebSphere Business Modeler 文档。

业务度量模型中的某些更改受到限制，这是由于更改现有业务度量模型的数据库表受限造成的。如果发生以下变更，则应该重新生成整个模型并将其部署为新模型。这会生成和部署一组全新的工件。这些更改是:

- 更改业务度量的 **WebSphere Business Monitor 中的用途**属性，如将度量从**关于正在运行的过程的活动数据**值更改为**维分析中的量化数据**值。
- 更改度量隶属的维组。
- 更改业务度量编辑器中维度量上的 **isPartOfDimensionKey** 复选框的状态。
- 更改度量的数据类型。通过删除当前度量并创建新度量，来处理度量数据类型的变更。
- 删除作为维键一部分的度量。
- 创建新的度量，作为现有维的维键的一部分。
- 将计时器在累计和非累计类型之间转换。
- 更改过程模型或活动。

注: 删除整个过程不需要重新生成模型 ，即使这可能导致删除度量。只会生成更改。

根据业务度量模型中的更改，有三种部署方案:

- 新模型部署
- 新版本模型部署
- 不同的模型部署

添加新的过程

在将新过程添加到业务度量模型时，会将一个新表添加到状态、运行时和历史数据库。

添加了新的过程后，使用模式生成器生成更改已创建数据库（状态、运行时和历史数据库）所需的脚本 和复制设置。

注: 利用此次更改进行规划实践很重要。要获取更多关于数据库规划的信息，请参阅第 4 页的『数据库工件部署准备』。

要在数据库之间进行同步，请用对应于新添加过程的新表更新复制配置。部署这些脚本以在 WebSphere Business Monitor 数据库中添加新表，并在复制设置中做出相应的更改以反映在数据库表中所做的更改。

在添加新过程时，需要执行以下步骤:

- 在部署任何新的或更新的业务度量模型之前备份所有数据库。
- 使用模式生成器生成为添加新表和更改现有数据库而部署的数据库定义语言（DDL）脚本。
- 在 DB2 命令窗口中部署新复制脚本，以启用新过程表的复制。
- 将立方体模型重新导入 DB2 Cube Views 和 ALPHABLOX®，以便立方体模型反映在历史数据库中已创建的任何新立方体。

现有业务度量组存在新的一列

在将新业务度量（度量值）添加到现有业务度量组时，会在状态数据库、运行时数据库和历史数据库中的某些实例表中添加新列。

根据所添加度量值的 **WebSphere Business Monitor 中的用途**属性选择用新列更新的表。下表中显示了受影响的数据库:

WebSphere Business Monitor 数据库中度量值的用途

WebSphere Business Monitor 中的用途	**状态数据库**	**运行时数据库**	**历史数据库**
临时计算	是	否	否
关于正在运行的过程的活动数据	是	是	否
维分析中的定量数据	是	是	是
维分析中的聚集组	是	是	是

在业务度量编辑器中添加了新业务度量（度量值）后，您可以使用模式生成器来生成更改已创建数据库（状态、运行时和历史数据库） 和复制设置所需的脚本。要更改数据库，请在 DB2 命令窗口部署数据库定义语言（DDL）以将新列添加到数据库表。

要在数据库之间同步复制，则通过在 DB2 命令窗口部署复制脚本，以用对应于新添加度量的新列来更新 DB2 复制中心。

在将新度量值添加到过程时，需要执行以下步骤:

- 在部署任何新的或更新的业务度量模型之前，备份所有数据库。
- 停止该过程的监控服务。
- 停止该过程的复制服务。
- 使用模式生成器生成为添加新列和更改现有数据库而部署的 DDL 脚本。
- 部署生成的复制脚本，以反映数据库拓扑中的更改。
- 将立方体模型重新导入 DB2 Cube Views 和 ALPHABLOX，以使立方体反映历史数据库中创建的任何新维。

数据库维护

WebSphere Business Monitor 数据库需要定期维护。一些 DB2 工具可帮助您增强数据库的用途和性能。

建议使用的工具是:

- 配置顾问程序。
- DB2 Web 运行状况中心，它向数据库管理员通知潜在的问题并提供解决问题的建议，以帮助他们完成工作。DBA 可以使用 Web 运行状况中心来远程监控实例，查看警报详细信息并提供建议。
- REORG 和 REORGCHK 命令。REORG 消除溢出行并从已删除的表和索引中重新获取空间。当有大量删除、更新或插入操作时，该工具很有用。REORGCHK 更新了 DB2 优化器工具使用的统计信息。当由于数据库更新而使得数据库表的统计信息不是最新信息时，该工具可以发挥作用。
- RUNSTATS 命令，收集关于数据库对象的统计信息。可在数据检索期间使用这些统计信息，以选择访问数据的路径。因此，DB2 将拥有在选择最有效访问路径时所需的信息。如果管理员没有在某些数据能够在每个数据库中累积后对所有数据库中的所有表运行 DBA RUNSTATS 命令，数据库性能将极差。示例:

  ```
  RUNSTATS ON TABLE tablename WITH DISTRIBUTION AND DETAILED INDEXES ALL
  ```

  注: *tablename* 必须是带有模式名称的标准名称。
  在运行该命令之后。运行 DB2 rebind 命令:

  ```
  db2rbind <database_alias> -l logfile all
  ```
  该执行会影响监控器服务器事件处理性能和复制性能。此外，应该将复制的 RMPRUNECTL 表中的缺省修剪时间间隔集设置为 0 而不是当前的 1440 秒（或 24 小时）。

要获取更多关于 DB2 维护实用程序的信息，请参阅 IBM DB2 文档。

备份数据库并从部署错误中恢复也是数据库维护的一部分。

数据库备份

在运行任何新 WebSphere Business Monitor 数据库定义语言（DDL）脚本之前（无论是部署新的还是更新现有的业务度量模型），您应该备份存储库、状态、运行时和历史数据库。

备份可以确保在发生错误时有一个安全的回滚点。如果您不关心最近的数据收集，可以将数据库回滚到不包含最近部署的业务度量模型的表的先前状态。

要获取更多关于数据库备份和恢复的信息，请参阅 DB2 数据恢复部分。

部署错误之后恢复

如果在部署复制工件时出错。您必须撤消已在单个业务度量模型部署中执行的操作，以撤消更改。

所有部署都是分为几个阶段来完成的，以下是典型方案:

1. DDL 部署
   a. 部署 state.ddl。
   b. 部署 runtime.ddl。

c. 部署 datamart.ddl。

2. 数据移动服务部署

   a. 部署 State_to_Runtime_setup_source。

   b. 部署 State_to_Runtime_setup_target。

   c. 部署 Runtime_to_historical_setup_source。

   d. 部署 Runtime_to_Historical_setup_target。

您必须识别发生错误之处以确定如何采取措施。例如，如果 state.ddl 失败，则只需回滚事务以返回至原始状态。然而，如果 datamart.dll 失败，则回滚 datamart.ddl 只能使系统返回到 runtime.ddl 成功执行后的点。在数据移动服务部署过程中的失败是最难恢复的，但不是不可恢复。第一次部署最易于恢复，新模型的部署次之，而更改模型的最终部署最难恢复。

要从复制脚本部署错误恢复，您需要经历以下阶段：识别、备份、恢复或删除以及重新部署:

识别

- 识别发生的错误并确定是否应联系 IBM支持中心。
- 识别错误发生时正在部署的业务度量模型。
- 识别错误发生时正在创建或更改的模式生成器表。
- 识别错误发生时正在创建或更改的模式生成器工件。
- 识别存储库数据库中业务度量模型的最新有效版本 。
- 如果是变更管理部署，识别为先前模型版本部署的工件的位置。这将给出数据库结构、其描述以及相互之间的关系。这在需要备份数据并稍后恢复的情况下很重要。
- 识别当前工件的位置和部署日志文件。这些对于问题确定很重要，并可以提供给 IBM 支持中心。
- 如果是变更管理部署，识别在任何未经处理的 CCD 表中是否存在数据。您可以使用 *WBIRMADM.RMMETADATA* 表（在运行时和历史数据库中提供）来确定相关 CCD 表（*TGT_RM_APP_STG_TAB_NAME*），该表具有正在部署的业务度量模型项目名称（*OM_NAME*）。标记有 **I** 或 **U** 的任意行可能还未经处理，应该备份。列 *SERVICE_NAME* 包含 CCD 表和目标表的位置，*to* 后的名称即指出了该位置。您应该记录与 *TGT_TAB_NAME* 的关系，以备您决定完全除去所有工件并生成全新的工件。这是因为模式生成器不会生成同名的 CCD 表，而您需要在部署成功之后将此数据恢复到新的 CCD 表。

备份

- 根据识别阶段来确定是否有数据需要备份。因为其他业务度量模型可能在部署期间同一时间运行，所以您可能需要备份与其他业务度量模型相关的数据库对象。
- 您可能需要备份 CCD（一致更改数据）表。模式生成器可能仍在这些表中拥有一些信息，而这些信息并不存在于源或目标数据库表。

  注: 完成的事件可能只存在于历史数据库中。

恢复或删除

- 确定恢复先前的数据库或手工除去工件哪个更容易。

- **恢复**：在没有其他业务度量模型存在时或其他业务度量模型没有活动时，从备份版本恢复可能是个很好的选择。恢复先前的数据库集，对于每个数据库，重新绑定任何需要重新绑定的应用程序，重新注册所有基于 Java 的存储过程和用户定义的函数。

  注:

  – 要获取更多关于数据库备份和恢复的信息，请参阅 DB2 文档中的数据恢复部分。

  – 对于完成的已部署模型，*WBIRMADM.RMMETADATA* 表提供无需除去的模型的信息。然而，在部署时，为了确定某些工件和关系，可能需要检查部署日志来确定哪些可安全除去。

- **除去**:

  – 复制: 历史数据库和运行时数据库

    - 停止所有与业务度量模型相关的“捕获”服务器。（“捕获”服务器运行在运行时和状态数据库上。）
    - 停止所有与业务度量模型相关的“应用”服务器。
    - 除去业务度量模型的所有“ETL”存储过程。
    - 除去所有用于业务度量模型的“ETL”登台表。
    - 除去所有来自业务度量模型的相应目标数据库中 WBIRMADM.RMCONTROL 表的“ETL”控制信息。
    - 除去所有用于业务度量模型的“ETL”修剪存储过程和触发器。
    - 除去所有列举在 *WBIRMADM.RMMETADATA* 表列 *TGT_RM_APP_STG_TAB_NAME* 中的表，这些表具有后缀 *_BKUP* 和 *_M*，而且业务度量模型的 *SERVICE_NAME* 对于历史为 *Runtime_to_Historical*，对于运行时为 *State_to_Runtime*。保留列举在 *TGT_RM_APP_STG_TAB_NAME* 中的表，因为在下一步骤将除去它。
    - 使用 DB2 复制中心除去所有服务于业务度量模型的“应用”预订集成员。
      - 如果“应用”预订集为空，则除去预订集。
      - 如果“应用”服务器没有预订集，则除去“应用”服务器。
    - 从 WBIRMADM.RMMETADATA 表删除所有与业务度量模型相关的元数据项。如果处理历史（运行时）数据库，则您还需要除去来自运行时（状态）数据库中 *WBIRMADM.RMMETADATA* 表的相同条目。在处理历史数据库（运行时数据库）时，您只要除去业务度量模型和服务名称为 *Runtime_to_Historical*（*State_to_Runtime*）的行。

  – 复制: 运行时数据库和状态数据库

    - 停止所有服务于业务度量模型的“捕获”服务器。
    - 除去所有与业务度量模型相关的“捕获”CD 表的相关触发器。
    - 从 WBIRMADM.RMPRUNECTRL 表除去用于业务度量模型的修剪触发器的修剪控制信息。
    - 使用 DB2 复制中心除去与业务度量模型相关的所有表的所有预订。
    - 从 WBIRMADM.RMMETADATA 表删除所有与业务度量模型相关的元数据项。

  – 数据库模式: 一般情况下，会在部署更改的模型之前回滚模式生成期间发生的错误。这不会影响当前复制工件集。

重新部署

当除去所有支持业务度量模型的工件时，您可以用所选的**忽略原部署**再次运行模式生成器。如果生成的模式成功，请不要部署数据库定义语言（DDL）脚本，只需再次部署复制脚本即可。

创建和配置数据库

创建和配置 WebSphere Business Monitor 数据库是安装过程中至关重要的一个阶段。

WebSphere Business Monitor 具有四个数据库:

- 存储库
- 状态
- 运行时
- 历史

在安装 WebSphere Business Monitor 时，只能用启动板创建数据库。如果在安装之后删除了数据库，则可以使用启动板或手工重新创建它们。创建数据库包括创建静态表、表空间和索引，还包括设置适当的数据库配置。状态、运行时和历史数据库包含静态和动态表，而存储库数据库只包含静态表。创建数据库所需的脚本保存在 *<monitor_installation_dir\install\mondb* 中。

要手工创建数据库，请按顺序执行以下步骤:

1. **在 Windows 系统上**:
   a. 转至**开始 -> 程序 -> IBM DB2 -> 命令行工具 -> 命令窗口**。
   b. 要创建存储库数据库，请运行以下脚本: db2CreateRepository.bat <%RepositoryDatabaseName%> <%DB2userID%> <DB2Password> Create_Repository.sql <%Install_Directory%>。
   c. 要创建状态数据库，请运行以下脚本: db2CreateState.bat createStateDB.ddl <%Install_Directory%>。
   d. 要创建运行时数据库，请运行以下脚本: db2CreateRuntime.bat createRuntimeDB.ddl <%Install_Directory%>。
   e. 要创建历史数据库，请运行以下脚本: db2CreateHistorical.bat createDatamartDB.ddl <%Install_Directory%>。

   注: 缺省情况下，*<%Install_Directory%>* 是 ″C:\IBM\Websphere\Monitor″。
2. **在 AIX® 系统上**:
   a. 作为 DB2 实例用户登录。
   b. 要创建存储库数据库，请运行以下脚本: db2CreateRepository.sh <%RepositoryDatabaseName%> <%DB2userID%> <DB2Password> <%PathToDDL%>/Create_Repository.sql <%Install_Directory%>。
   c. 要创建状态数据库，请运行以下脚本: db2CreateState.sh <%PathToDDL%>/createStateDB.ddl <%Install_Directory%> <%DB2UserID%>。
   d. 要创建运行时数据库，请运行以下脚本: db2CreateRuntime.sh <%PathToDDL%>/createRuntimeDB.ddl <%Install_Directory%> <%DB2UserID%>。
   e. 要创建历史数据库，请运行以下脚本: db2CreateHistorical.sh <%PathToDDL%>/createDatamartDB.ddl <%Install_Directory%> <%DB2UserID%>。

   注:

- 如果您已安装在备用目录，则必须根据安装目录来替换相应的路径。
- *<%PathToDDL%>* 是 /opt/IBM/Websphere/Monitor/Install/mondb（缺省情况下）。
- *<%Install_Directory%>* 是 /opt/IBM/Websphere/Monitor（缺省情况下）。
- *<%DB2UserID%>* 是 *db2inst1*（缺省情况下）。

在运行时管理数据库

在运行时管理 WebSphere Business Monitor 数据库包括在 WebSphere Business Monitor 管理控制台中部署模式生成器生成的工件。每当导入一个新的或更改的业务度量模型时，都会重复部署这些工件。

在运行时管理数据库包括完成以下任务。

创建动态数据库表

动态数据库表与特定的业务度量模型对应。创建这些表所需的脚本是从模式生成器生成的。

您必须在 WebSphere Business Monitor 管理控制台中运行模式生成器，以生成创建动态数据库表所需的脚本。

这些脚本为状态、运行时和历史数据库中的每个动态表创建表和索引，并设置配置参数。生成的脚本的位置是在 WebSphere Business Monitor 管理控制台中配置模式生成器期间指定的。

状态数据库:

遵循以下步骤以在状态数据库中创建动态数据库表。脚本存储在用户定义的位置中。该位置是在 WebSphere Business Monitor 管理控制台中设置模式生成器配置时定义的。

在状态数据库中创建动态数据库表所需的数据库定义语言（DDL）脚本存储在根目录上的 **state.ddl** 文件中。要部署脚本，请完成以下步骤:

1. 打开 DB2 **命令窗口**。 在 UNIX上，如果配置了 shell 环境，可以调用 DB2 命令行处理器。
2. 将路径更改为脚本文件的位置。
3. 在您部署新的业务度量模型之前备份状态数据库。
4. 运行命令 ***db2 terminate***。 这确保了任何以前使用不同代码页值的后台进程将不被使用，并且会使用新的后台进程来处理这个请求。
5. 将 ***DB2CODEPAGE*** 环境变量设置为 1208。缺省情况下，DB2 命令行处理器会使用当前代码页解释任何字符数据。但是，生成的 *state.ddl* 包含 UTF-8 字符，除非将 DB2CODEPAGE 环境变量设置成 1208，否则这些字符将会损坏。
   a. 在 UNIX 操作系统上。
      - 使用 *sh*、*ksh*、*bash* 类型 shell，运行命令 ***export DB2CODEPAGE=1208***。
      - 使用 *csh*、*tsch* 类型 shell，运行命令 ***setenv DB2CODEPAGE 1208***。

   b. 在 Windows 操作系统上，运行命令 ***set DB2CODEPAGE=1208***。
6. 通过运行命令 ***db2 connect to <State_DB_Name>*** 来连接到状态数据库。

7. 运行命令 ***db2 +c –stvf state.ddl > state.log***。运行该脚本，并保存日志文件，它记录事务以用于故障诊断。 在决定落实或回滚之前，检查日志文件以查找所有错误。如果需要回滚，则运行命令 ***db2 rollback*** 以撤消操作。如果未发生错误，则运行命令 ***db2 commit***，以落实更改。
8. 运行脚本之后，利用命令 ***db2 disconnect*** **<State_DB_Name>** 从状态数据库断开连接。
9. 运行命令 ***db2 terminate*** 以终止后台进程。

运行时数据库:

遵循以下步骤以在运行时数据库中创建动态数据库表。脚本存储在用户定义的位置中。该位置是在 WebSphere Business Monitor 管理控制台中设置模式生成器配置时定义的。

在运行时数据库中创建正在运行的数据库表所需的数据库定义语言（DDL）脚本存储在根目录上的 **runtime.ddl** 文件中。要部署脚本，请完成以下步骤:

1. 打开 DB2 **命令窗口**。 在 UNIX上，如果配置了 shell 环境，可以调用 DB2 命令行处理器。
2. 将路径更改为脚本文件的位置。
3. 在您部署新的业务度量模型之前备份运行时数据库。
4. 运行命令 ***db2 terminate***。 这确保了任何以前使用不同代码页值的后台进程将不被使用，并且会使用新的后台进程来处理这个请求。
5. 将 ***DB2CODEPAGE*** 环境变量设置为 1208。缺省情况下，DB2 命令行处理器会使用当前代码页解释任何字符数据。但是，生成的 *runtime.ddl* 包含 UTF-8 字符，除非将 DB2CODEPAGE 环境变量设置成 1208，否则这些字符将会损坏。
   a. 在 UNIX 操作系统上。
      - 使用 *sh*、*ksh*、*bash* 类型 shell，运行命令 ***export DB2CODEPAGE=1208***。
      - 使用 *csh*、*tsch* 类型 shell，运行命令 ***setenv DB2CODEPAGE 1208***。

   b. 在 Windows 操作系统上，运行命令 ***set DB2CODEPAGE=1208***。
6. 通过运行命令 ***db2 connect to*** **<Runtime_DB_Name>**，来连接到运行时数据库。
7. 运行命令 ***db2 +c -stvf runtime.ddl > runtime.log***。运行该脚本，并保存日志文件，它记录事务以用于故障诊断。 在落实或回滚之前，请检查日志文件的所有错误。如果需要回滚，则运行命令 ***db2 rollback*** 以撤消操作。如果未发生错误，则运行命令 ***db2 commit***，以落实更改。
8. 运行脚本之后，可以通过运行命令 ***db2 disconnect*** **<Runtime_DB_Name>**，从运行时数据库断开连接。
9. 运行命令 ***db2 terminate*** 以终止后台进程。

历史数据库:

遵循下列步骤以在历史数据库中创建动态数据库表。脚本存储在用户定义的位置中。该位置是在 WebSphere Business Monitor 管理控制台中设置模式生成器配置时定义的。

在历史数据库中创建运行数据库表所需的数据库定义语言（DDL）脚本存储在根目录上的 **datamart.ddl** 文件中。要部署脚本，请完成以下步骤:

1. 打开 DB2 **命令窗口**。 在 UNIX上，如果配置了 shell 环境，可以调用 DB2 命令行处理器。
2. 将路径更改为脚本文件的位置。
3. 在您部署新的业务度量模型之前备份历史数据库。
4. 运行命令 ***db2 terminate***。 这确保了任何以前使用不同代码页值的后台进程将不被使用，并且会使用新的后台进程来处理这个请求。
5. 将 ***DB2CODEPAGE*** 环境变量设置为 1208。缺省情况下，DB2 命令行处理器会使用当前代码页解释任何字符数据。但是，生成的 *datamart.ddl* 包含 UTF-8 字符，除非将 DB2CODEPAGE 环境变量设置成 1208，否则这些字符将会损坏。
   a. 在 UNIX 操作系统上。
      - 使用 *sh*、*ksh*、*bash* 类型 shell，运行命令 ***export DB2CODEPAGE=1208***。
      - 使用 *csh*、*tsch* 类型 shell，运行命令 ***setenv DB2CODEPAGE 1208***。

   b. 在 Windows 操作系统上，运行命令 ***set DB2CODEPAGE=1208***。
6. 通过运行命令 ***db2 connect to*** **<Historical_DB_Name>** 来连接到历史数据库。该脚本运行时不会自动落实更改。
7. 运行命令 ***db2 +c -stvf datamart.ddl > datamart.log***。运行该脚本，并保存日志文件，它记录事务以用于故障诊断。 在落实或回滚之前，请检查日志文件的所有错误。如果需要回滚，请运行命令: ***db2 rollback***，以撤销操作。如果未发生错误，则运行命令 ***db2 commit***，以落实更改。
8. 运行脚本之后，可以通过运行命令 ***db2 disconnect*** **<Historical_DB_Name>**，从历史数据库断开连接。
9. 运行命令 ***db2 terminate*** 以终止后台进程。

注: 当在特定环境中对现有业务度量模型的新版本运行 ***datamart.ddl*** 时，可以看到类似于以下的错误: `SQL0605W Theindex was not created because the index "WBI.I_1133789461307" already exists with the required description。SQLSTATE=01550`。这些错误可以忽略，如果没有出现其他错误，则可以落实事务。

部署数据移动服务

在可以部署状态至运行时和运行时至历史的数据移动服务之前，应创建动态数据库表。在执行动态数据库表创建脚本时出现的任何错误都将导致在数据移动服务部署期间出问题。

在模式生成期间，最多会创建 3 个包含数据移动服务设置文件的压缩（zip 文件或 JAR 文件）归档（名为 DS_State_setup、DS_Runtime_setup 和 DS_Datamart_setup）。在第一次为业务度量模型执行模式生成时，始终会创建这三个归档。在后续的生成中 – 譬如修改业务度量模型之后 – 可以创建 0、1、2 或 3 个新归档。只有当需要更改现有复制环境以满足业务度量模型更改时，才创建归档。数据移动服务部署归档位于模式生成器管理控制台配置中“常规”选项卡下指定的目录中。

数据移动服务部署包括创建和配置源数据库（从这里移动数据）以及目标数据库（将数据移动到这里）中的数据库对象。

- *DS_State_setup* 包含状态至运行时数据移动服务的源数据库设置的部署脚本。
- *DS_Runtime_setup* 包含状态至运行时数据移动服务的目标数据库设置的部署脚本，以及运行时至历史的数据移动服务的源数据库设置的部署脚本。

- DS_Datamart_setup 包含运行时至历史的数据移动服务的目标数据库设置的部署脚本。

1. 部署状态至运行时数据移动服务:
   a. 确定您将在哪台机器上部署状态至运行时数据移动服务的源工件。通常，这是状态数据库所在的机器。
   b. 在该机器上创建工作目录，并将生成的 DS_State_setup 归档复制（如果机器是远程的，则传送）到该目录。由于与操作系统相关的路径长度限制，您必须选择长度小于或等于 100 个字符的路径。
   c. 将归档（Windows 上的 .zip 文件和 UNIX 上的 .JAR 文件）抽取到工作目录。
   d. 在部署期间，将使用各种 DB2 实用程序创建和配置数据库对象。要使用那些工具，必须设置数据库环境。在 Microsoft® Windows 上，通过打开 DB2 命令窗口来执行该操作。在 UNIX上，请确保设置了相应的环境变量。
   e. 浏览至您抽取 DS_State_setup 归档的目录。
   f. 执行 State_to_Runtime_setup_source.bat（在 UNIX 上，扩展名是 .sh），并遵循提示。该脚本会显示状态消息，指出某条命令是成功、生成警报，还是失败。
   g. 检查生成的日志文件 State_to_Runtime_setup_source.log，寻找警报或错误消息。如果显示错误消息，则不要继续。
   h. 备份工作目录。IBM 支持中心可使用其内容进行故障诊断。
   i. 确定您要在哪台机器上部署状态至运行时数据移动服务的目标工件。通常，这是运行时数据库所在的机器。
   j. 在该机器上创建工作目录，并将生成的 DS_Runtime_setup 归档复制（如果该机器是远程的，则传送）到该目录。由于与操作系统相关的路径长度限制，您必须选择长度小于或等于 100 个字符的路径。
   k. 将归档（Windows 上的 .zip 文件和 UNIX 上的 .JAR 文件）抽取到工作目录。
   l. 在部署期间，将使用各种 DB2 实用程序创建和配置数据库对象。要使用那些工具，必须设置数据库环境。在 Microsoft Windows 上，通过打开 DB2 命令窗口来执行该操作。在 UNIX 上，请确保设置了相应的环境变量。
   m. 浏览至 DS_Runtime_setup 归档抽取所至的目录。
   n. 执行 State_to_Runtime_setup_target.bat（在 UNIX 上，扩展名是 .sh），并遵循提示。该脚本会显示状态消息，指出某条命令是成功、生成警报还是失败。
   o. 检查生成的日志文件 State_to_Runtime_setup_source.log，寻找警报或错误消息。如果显示错误消息，则不要继续。
   p. 备份工作目录。IBM 支持中心可使用其内容进行故障诊断。
   q. 如果未报告任何问题，则状态至运行时数据移动服务已设置完毕。
2. 部署运行时至历史的数据移动服务:
   a. 确定您将在哪台机器上部署运行时至历史的数据移动服务的源工件。通常，这是运行时数据库所在的机器。如果您已在同一台机器上部署了状态至运行时数据移动服务的目标工件，则可以继续执行下面的步骤 **e**，因为已抽取了必需的部署文件。
   b. 如果在未运行运行时数据库的机器上执行部署，则在该机器上创建工作目录，并将生成的 DS_Runtime_setup 归档复制（如果该机器是远程的，则传送）到该目录。由于与操作系统相关的路径长度限制，您必须选择长度小于或等于 100 个字符的路径。
   c. 将归档（Windows 上的 .zip 文件和 UNIX 上的 .JAR 文件）抽取到工作目录。

d. 在部署期间，将使用各种 DB2 实用程序创建和配置数据库对象。要使用那些工具，必须设置数据库环境。在 Microsoft Windows 上，通过打开 DB2 命令窗口来执行该操作。在 UNIX 上，请确保设置了相应的环境变量。

e. 浏览至 DS_Runtime_setup 归档抽取所至的目录。

f. 执行 Runtime_to_Historical_setup_source.bat（在 UNIX 上，扩展名是 .sh），并遵循提示。该脚本会显示状态消息，指出某条命令是成功、生成警报还是失败。

g. 检查生成的日志文件 Runtime_to_Historical_setup_source.log，寻找警告或错误消息。如果显示错误消息，则不要继续。

h. 备份工作目录。IBM 支持中心可以使用其内容进行故障诊断。

i. 确定您将在哪台机器上部署运行时至历史的数据移动服务的目标工件。通常，这是历史数据库所在的机器。

j. 在该机器上创建工作目录，并将生成的 DS_Datamart_setup 归档复制（如果该机器是远程的，则传送）到该目录。由于与操作系统相关的路径长度限制，您必须选择长度小于或等于 100 个字符的路径。

k. 将归档（Windows 上的 .zip 文件和 UNIX 上的 .JAR 文件）抽取到工作目录。

l. 在部署期间，将使用各种 DB2 实用程序创建和配置数据库对象。要使用那些工具，必须设置数据库环境。在 Microsoft Windows 上，通过打开 DB2 命令窗口来执行该操作。在 UNIX 上，请确保设置了相应的环境变量。

m. 浏览至 DS_Runtime_setup 归档抽取所至的目录。

n. 执行 Runtime_to_Historical_setup_target.bat（在 UNIX 上，扩展名是 .sh），并遵循提示。该脚本会显示状态消息，指出某条命令是成功、生成警报还是失败。

o. 检查生成的日志文件 State_to_Runtime_setup_source.log，寻找警报或错误消息。如果显示错误消息，则不要继续。

p. 备份工作目录。IBM 支持中心可使用其内容进行故障诊断。

q. 如果未报告任何问题，则已为该业务度量模型设置了运行时至历史的数据移动服务。

配置数据移动服务选项

对于由数据移动服务组件创建和配置的每个“捕获”服务器，有两个参数可影响“捕获”组件的行为。它们是 `lag_limit` 和 `startmode` 参数。

`lag_limit` 和 `startmode` 参数具有缺省值：“7 天”和“WARMSI”。要了解关于这些参数的更多信息，请参阅 DB2 SQL 复制指南和参考。

您在部署好工件之前不能修改这些设置。然而，可在运行任何“捕获”组件服务器之前更改这些设置或为当前正在运行的“捕获”服务器更改参数。

注： 要启用在“捕获”服务器运行时执行的更改，需要停止并重新启动“捕获”服务器。

如果在您部署复制工件时，使用 `lag_limit` 和 `startmode` 参数的缺省设置，并且“捕获”服务器在停止运行超过 7 天后启动，那么“捕获”组件返回一个错误。该错误显示，由于数据太旧，“捕获”服务器无法运行. 您可以多种方式覆盖缺省值。以下描述了三种方式:

1. 修改 <CAPTURESERVERSCHEMA>.IBMSNAP_CAPPARMS 表中的缺省参数。部署好复制工件后，可以对运行时数据库运行以下查询，来确定数据移动服务组件创建的“捕获”服务器数。

   ```
   CONNECT TO RUNTIME DATABASE

   SELECT DISTINCT OM_NAME, SERVICE_NAME, SRC_RM_CAP_SVR_NAME

   FROM WBIRMADM.RMMETADATA

   ORDER BY 1,2,3
   ```

   您会看到如下所示的表:

表 1. RMMETADATA 示例

OM_NAME	SERVICE_NAME	SRC_RM_CAP_SVR_NAME
SubDoctor3	运行时至历史	CAPTURE_18
SubDoctor3	状态至运行时	CAPTURE_1
SubDoctor3	状态至运行时	CAPTURE_115
SubDoctor3	状态至运行时	CAPTURE_156
SubDoctor3	状态至运行时	CAPTURE_194
SubDoctor3	状态至运行时	CAPTURE_212
SubDoctor3	状态至运行时	CAPTURE_250
SubDoctor3	状态至运行时	CAPTURE_41
SubDoctor3	状态至运行时	CAPTURE_59
SubDoctor3	状态至运行时	CAPTURE_97

   OM_NAME 是 WebSphere Business Modeler 项目的名称。 SERVICE_NAME 表示数据移动服务，而 SRC_RM_CAP_SVR_NAME 是用作数据移动服务一部分的“捕获”服务器的标识符（CAPTURE SCHEMA）。上表中，有一个“捕获”服务器用于运行时至历史的数据移动服务，九个用于状态至运行时数据移动服务。

   注: 服务器的数量和名称会随着使用的模型和在工件生成期间指定的策略参数而不同。

   数据库服务组件支持 `lag_limit` 和 `startmode` 参数的所有选项，但应该意识到，如果冷启动（失败后启动“捕获”服务器）次数增加，会有严重的性能问题。如果冷启动频繁，则数据移动服务的 ETL 组件会处理所有现有记录而不仅仅是记录更改。只要识别了所有需要修改的“捕获”服务器，您就可以修改数据库中的缺省参数。修改了每个“捕获”服务器的缺省参数后，可以启动“捕获”服务器。
2. 或修改启动“捕获”服务器的命令行。在生成数据库服务工件期间，会生成可启动和停止“捕获”和“应用”服务器的快捷脚本。捕获启动脚本（StartCapture_#.bat 或 StartCapture_#.sh）位于目录 <data_movement_service_name>\source 中。每个脚本都包含 **asncap** 命令，用于启动“捕获”程序。要了解更多关于这些参数的信息，请参阅 IBM DB2 文档。分别修改启动脚本、保存，然后运行启动脚本以使用新设置运行“捕获”服务器。
3. 或在运行时修改“捕获”服务器。遵循 IBM DB2 文档中的指示信息，该文档描述了如何临时更改正在运行的“捕获”服务器的设置。

结束数据移动服务设置

在缺省情况下，已部署的“捕获”和“应用”组件实例使用启动它们的用户的凭证。尽管这对于某些拓扑来说可能已经足够了，但在两种情况下必须使用备用凭证。

- **第一方案 – 备用用户凭证**: 数据库管理员（DBA）想要以用户 *user1* 身份登录，但希望实用程序使用用户 *user2* 来将数据从源数据库移至目标数据库。
- **第二方案 – 分布式环境**: DBA 准备在 *machine1* 上运行实用程序。在另一台机器 *machine2* 上维护源或目标数据库。

要支持这些方案，您必须创建包含替代用户凭证（而不是当前用户凭证）的密码文件。由于不会在部署期间自动创建密码文件，所以您需要对这两个方案执行以下步骤:

1. 准备一个文件，用来存储连接到源数据库时使用的*用户标识*和*密码*信息。在 DB2 命令行窗口中输入以下命令，并将任何由 <*place_holder_name*> 标记的占位符替换成合适的值。

   `asnpwd INIT encrypt all using <password_file>.` asnpwd 工具创建了一个空文件: <*password_file*>。

   示例调用: `asnpwd INIT encrypt all using password.aut`

2. 保存复制实用程序必须连接的各数据库的数据库访问信息 （用户标识、密码和数据库名称）。在 DB2 命令行窗口中输入以下命令，并将任何由 <*place_holder_name*> 标记的占位符替换成合适的值。

   `asnpwd ADD alias <DB_name> ID <user_ID> PASSWORD <Password> using <password_file>`。

   如果需要，对每个数据库重复该步骤。程序对您输入的信息进行加密，并将其保存在 <*password_file*> 中。

   调用示例:

   - `asnpwd ADD alias STMD7 id MYUSRID password MYPASSWRD using password.aut`
   - `asnpwd ADD alias RTMD7 id MYUSRID2 password MYPASSWRD2 using password.aut`

3. 通过修改生成的可执行文件启动脚本（StartCapture 和 StartApply）来更新实用程序配置文件。您将密码文件参数参数添加到复制实用程序的命令行调用。该实用程序使用存储在指定文件中的加密用户凭证，而非缺省凭证。密码文件必须位于 CAPTURE_PATH（或 APPLY_PATH）参数定义的工作目录中。

   更改示例:

   - “捕获”启动脚本的原文件内容: `db2cmd asncap CAPTURE_SERVER=stmd7 CAPTURE_SCHEMA=CAPTURE_1 CAPTURE_PATH="c:\tmp\state_capture_log"`
   - “捕获”启动脚本的已修改文件内容: `db2cmd asncap CAPTURE_SERVER=stmd7 CAPTURE_SCHEMA=CAPTURE_1 CAPTURE_PATH="c:\tmp\state_capture_log" pwdfile="password.aut"`
   - “应用”启动脚本的原文件内容: `db2cmd asnapply APPLY_QUAL=Apply_1 CONTROL_SERVER=RTMD7 APPLY_PATH="C:\tmp\apply"`
   - “应用”启动脚本的已修改文件内容: `db2cmd asnapply APPLY_QUAL=Apply_1 CONTROL_SERVER=RTMD7 APPLY_PATH="C:\tmp\apply" pwdfile="password.aut"`

4. 将在步骤 1 和步骤 2 中创建的 <*password_file*> 复制到适当的目录中。复制实用程序试图在启动时打开密码文件。如果由 CAPTURE_PATH（或 APPLY_PATH）参数标识的工作目录中不存在文件 <*password_file*>，则会出错。如果未指定任何工作目录参数，则实用程序会尝试在当前工作目录中找出该文件。

要了解有关 DB2

实用程序的信息，请参阅 DB2

SQL 复制指南和参考。

整合启动和停止脚本

要简化启动和停止数据移动服务的过程，您可以整合生成的启动和停止脚本，并通过主脚本调用它们。

由于“捕获”和“应用”组件必须在数据库所在的系统上运行，所以整合选项因所用的拓扑而异。不管如何整合脚本，要预防初始化错误，您需要确保不会同时启动两个组件。

尽管可以单独启动或停止每个“捕获”或“应用”组件实例，但整合所有组件实例启动和停止脚本的内容，以便只用一个脚本来启动或停止单个业务度量模型的数据移动服务，无疑更加方便。您可以通过以下操作来整合脚本:

1. 识别源数据库的“捕获”组件实例启动和停止脚本。
2. 创建“捕获”主启动和停止脚本，它们调用了源数据库的“捕获”组件实例启动和停止脚本。
3. 识别目标数据库的“应用”组件实例启动和停止脚本。
4. 创建“捕获”主要启动和停止脚本，它们调用了目标数据库的“捕获”组件实例启动和停止脚本。

作为整合的结果，只需要执行 4 个启动（或停止）脚本，就可以启动或停止业务度量模型的数据移动服务。

如果不需要单独启动或停止两个数据移动服务，则可以进一步整合。在这种情况下，只需要 3 个启动和停止脚本:

- 一个脚本启动（停止）状态数据库中的所有“捕获”组件实例。
- 一个脚本启动（停止）运行时数据库中的所有“捕获”组件实例和“应用”组件。
- 一个脚本启动（停止）历史数据库中的所有“应用”组件。

如果这三个数据库都位于同一个系统上，则可以将这三个已整合的脚本进一步整合成一个脚本，来启动或停止所有“捕获”和“应用”组件实例。

有种情况下，需要对由不同部署创建的数据移动服务的启动和停止脚本进行整合。在数据移动服务的初始部署期间，会创建所有业务度量组的启动和停止脚本。作为对业务度量模型进行更改的结果，后续部署不包含现有业务度量组的启动和停止脚本。只有新业务度量组的启动和停止脚本是可用的。您需要手工更新先前创建的已整合启动和停止脚本。

以下示例说明了这种情况: 业务度量模型 *FinanceModel* 的初始数据移动服务部署包含三个业务度量组。已为状态数据库创建了三个“捕获”启动和停止脚本。然后，更新模型，并添加新的业务度量。部署期间，只会为新业务度量组创建一个“捕获”启动和停止脚本。需要运行四个“捕获”启动和停止脚本，以启用数据移动服务。

启动和停止数据移动服务

启动和停止给定业务度量模型的数据移动服务是通过启动和停止关联的“捕获”和“应用”组件实例来。在数据移动服务部署期间，启动和停止脚本已经创建，因此，这些脚本可用来启动和停止数据移动服务。

您采用的拓扑决定了应在哪台机器上运行组件实例。通常，“捕获”组件实例必须在状态数据库（用于状态至运行时数据移动服务）所在的机器以及运行时数据库（用于运行时至历史的数据移动服务）所在的机器上运行。“应用”组件应在运行时数据库（用于状态至运行时数据移动服务）所在的机器以及历史数据库（用于运行时至历史的数据移动服务）所在的机器上运行。在该配置中，“应用”组件实例将使用源数据库中的数据，其性能优于在状态数据库（用于状态到至行时数据移动服务）所在的机器以及在运行时数据库（用于运行时至历史的数据移动服务）所在的机器上的性能。

以下信息说明了如何启动状态至运行时数据移动服务以及运行时至历史的数据移动服务。它还描述了如何停止状态至运行时数据移动服务以及运行时至历史的数据移动服务。

注: 状态至运行时数据移动服务和运行时至历史的数据移动服务彼此相互独立。但是，最好在启动运行时至历史服务之前启动状态至运行时服务。在某些实例中，最好在监控器服务器处理了业务度量模型的项且状态至运行时数据移动服务填充了支持该模型的运行时数据库表之后，再启动运行时至历史的数据移动服务。这能使信息进入历史数据库的时间小于等待运行时至历史的数据移动服务时间间隔的时间。

启动状态至运行时数据移动服务:

部署归档 DS_State_setup 和 DS_Runtime_setup 包含了可执行文件脚本，可以使用这些脚本启动用于状态至运行时数据移动服务的“捕获”和“应用”组件实例。如果由于更改业务度量模型而创建了归档，则仅会打包新“捕获”和“应用”组件实例的启动脚本。

注: 您可以整合脚本以启动数据移动服务。要获取更多关于整合脚本的信息，请参阅第 43 页的『整合启动和停止脚本』。

然而，即使未执行整合，仍可使用以下指示信息。要对给定的业务度量模型启动状态至运行时数据移动服务:

1. 识别状态数据库中已经指定给当前业务度量模型的所有“捕获”组件实例。

   如果已经整合了所有“捕获”组件实例启动脚本，则不需要做任何事。继续下一步。如果还未整合脚本（并不希望整合它们），则需要识别所有已为此业务度量模型创建的“捕获”组件实例。在第一次为业务度量模型执行模式生成时，会自动生成“捕获”组件实例启动脚本。随后的模式生成（例如，在更新业务度量模型之后）仅会为新“捕获”组件实例生成启动脚本。要识别所有相关的启动脚本，对于为该业务度量模型执行的每个部署，您需要重复以下步骤。

   a. 浏览至执行该模型部署的目录。
   b. 浏览至 State_to_Runtime\source 子目录，并找出所有的 StartCapture_<number> 脚本。
   c. 对于该业务度量模型的每个部署，重复上述步骤。
2. 启动“捕获”组件实例

已识别的“捕获”组件实例必须在状态数据库所在的机器上启动。如果已整合了启动脚本，则启动已整合的启动脚本。如果未进行整合，则需要执行上一步中确定的每个启动脚本。这些脚本不应同时启动，否则“捕获”实用程序初始化会失败。但是，可以按任意顺序启动这些启动脚本。权限需求，启动“捕获”组件实例的用户标识应拥有:

- 状态数据库上的数据库管理权限（DBADM）。
- 启动脚本中 *CAPTURE_PATH* 参数所引用目录的写访问权。
- 启动脚本中可选参数 *PWDFILE* 所引用文件的读访问权

3. 识别运行时数据库中已经指定给业务度量模型的所有“应用”组件实例。

   如果已经整合了所有“应用”组件实例启动脚本，则不需要做任何事。继续下一步。如果还未整合脚本（并不希望整合它们），则需要识别所有已为此业务度量模型创建的“应用”组件实例。在第一次为业务度量模型执行模式生成时，会自动生成“应用”组件实例启动脚本。随后的模式生成（例如，在更新业务度量模型之后）仅会为新“应用”组件实例生成启动脚本。要识别所有相关启动脚本，对于为该业务度量模型执行的每个部署，您都必须重复下列步骤:

   a. 浏览至执行该模型部署的目录。
   b. 浏览至 State_to_Runtime\target 子目录，并找出所有的 StartApply_<number> 脚本。
   c. 对于该业务度量模型的每个部署，重复上述步骤。

4. 启动“应用”组件实例

   已识别的“应用”组件实例应当在运行时数据库所在的机器上启动。如果已整合了启动脚本，则启动已整合的启动脚本。如果未进行整合，则需要执行上一步中确定的每个启动脚本。这些脚本不应同时启动，否则“应用”实用程序初始化会失败。但是，可以按任意顺序启动这些启动脚本。权限需求，启动“应用”组件实例的用户标识必须拥有:

   - SELECT/INSERT/UPDATE/DELETE 权限，用于状态数据库中的关联“捕获”组件实例控制表。
   - SELECT 权限，用于状态数据库中的关联“捕获”组件实例工作表。
   - SELECT/INSERT/UPDATE/DELETE 权限，用于运行时数据库中的关联复制登台表。
   - SELECT/INSERT/UPDATE/DELETE 权限，用于运行时数据库中的“应用”组件实例控制表。
   - 启动脚本中 *APPLY_PATH* 参数所引用目录的写访问权。
   - 启动脚本中可选参数 *PWDFILE* 所引用文件的读访问权

5. 验证每个“捕获”和“应用”组件实例是否成功启动。

启动运行时至历史的数据移动服务:

部署归档 DS_Runtime_setup 和 DS_Datamart_setup 包含了可执行文件脚本，可以使用这些脚本启动用于运行时至历史的数据移动服务的“捕获”和“应用”组件实例。如果由于更改业务度量模型而创建了归档，则仅会打包新“捕获”和“应用”组件实例的启动脚本。

注: 您可以整合脚本以启动数据移动服务。要获取更多关于整合脚本的信息，请参阅第 43 页的『整合启动和停止脚本』。

然而，即使未执行整合，仍可使用以下指示信息。要对给定的业务度量模型启动运行时至历史的数据移动服务:

1. 识别运行时数据库中已经指定给业务度量模型的所有“捕获”组件实例。

   如果已经整合了所有“捕获”组件实例启动脚本，则不需要做任何事。继续下一步。如果还未整合脚本（并不希望整合它们），则需要识别所有已为此业务度量模型创建的“捕获”组件实例。在第一次为业务度量模型执行模式生成时，会自动生成“捕获”组件实例启动脚本。随后的模式生成（例如，在更新业务度量模型之后）仅会为新“捕获”组件实例生成启动脚本。要识别所有相关的启动脚本，对于为该业务度量模型执行的每个部署，您需要重复以下步骤。

   a. 浏览至执行该模型部署的目录。
   b. 浏览至 Runtime_to_Historical\source 子目录，并找出所有的 StartCapture_<number> 脚本。
   c. 对于该业务度量模型的每个部署，重复上述步骤。

2. 启动“捕获”组件实例。

   已识别的“捕获”组件实例必须在运行时数据库所在的机器上启动。如果已整合了启动脚本，则启动已整合的启动脚本。如果未进行整合，则需要执行上一步中确定的每个启动脚本。这些脚本不应同时启动，否则“捕获”实用程序初始化会失败。但是，可以按任意顺序启动这些启动脚本。权限需求，启动“捕获”组件实例的用户标识应拥有:

   - 运行时数据库上的数据库管理权限（DBADM）。
   - 启动脚本中 *CAPTURE_PATH* 参数所引用目录的写访问权。
   - 启动脚本中可选参数 *PWDFILE* 所引用文件的读访问权

3. 识别历史数据库中已经指定给业务度量模型的所有“应用”组件实例。

   如果已经整合了所有“应用”组件实例启动脚本，则不需要做任何事。继续下一步。如果还未整合脚本（并不希望整合它们），则需要识别所有已为此业务度量模型创建的“应用”组件实例。在第一次为业务度量模型执行模式生成时，会自动生成“应用”组件实例启动脚本。随后的模式生成（例如，在更新业务度量模型之后）仅会为新“应用”组件实例生成启动脚本。要识别所有相关启动脚本，对于为该业务度量模型执行的每个部署，您都必须重复下列步骤:

   a. 浏览至执行该模型部署的目录。
   b. 浏览至 Runtime_to_Historical\target 子目录，并找出所有的 StartApply_<number> 脚本。
   c. 对于该业务度量模型的每个部署，重复上述步骤。

4. 启动“应用”实例。

   已识别的“应用”组件实例应在历史数据库所在的机器上启动。如果已整合了启动脚本，则启动已整合的启动脚本。如果未进行整合，则需要执行上一步中确定的每个启动脚本。这些脚本不应同时启动，否则“应用”实用程序初始化会失败。但是，可以按任意顺序启动这些启动脚本。权限需求，启动“应用”组件实例的用户标识必须拥有:

   - SELECT/INSERT/UPDATE/DELETE 权限，用于运行时数据库中的关联“捕获”组件实例控制表。
   - SELECT 权限，用于运行时数据库中的关联“捕获”组件实例工作表。

- SELECT/INSERT/UPDATE/DELETE 权限，用于历史数据库中的关联复制登台表。
- SELECT/INSERT/UPDATE/DELETE 权限，用于历史数据库中的“应用”组件实例控制表。
- 启动脚本中 *APPLY_PATH* 参数所引用目录的写访问权。
- 启动脚本中可选参数 *PWDFILE* 所引用文件的读访问权

5. 验证每个“捕获”和“应用”组件实例是否成功启动。

停止状态至运行时数据移动服务:

停止状态至运行时的数据移动服务的过程非常类似于启动它的过程。部署归档 DS_State_setup 和 DS_Runtime_setup 包含可执行文件脚本，可以使用这些脚本停止用于运行时至历史的数据移动服务的“捕获”和“应用”组件实例。

如果由于更改业务度量模型而创建了归档，则仅会打包新“捕获”和“应用”组件实例的停止脚本。

注: 建议在停止数据移动服务之前整合脚本。要获取更多关于复制脚本整合的信息，请参阅第 43 页的『整合启动和停止脚本』。

然而，即使未执行整合，仍可使用以下指示信息。

要对给定的业务度量模型停止状态至运行时数据移动服务:

1. 识别状态数据库中已经指定给当前业务度量模型的所有“捕获”组件实例。 如果已经整合了所有“捕获”组件实例停止脚本，则不需要做任何事。继续下一步。如果还未整合脚本（并不希望整合它们），则需要识别所有已为此业务度量模型创建的“捕获”组件实例。在第一次为业务度量模型执行模式生成时，会自动生成“捕获”组件实例停止脚本。随后的模式生成（例如，在更新业务度量模型之后）仅会为新“捕获”组件实例生成停止脚本。要识别所有相关停止脚本，对于为该业务度量模型执行的每个部署，您都需要重复以下步骤。
   a. 浏览至执行该模型部署的目录。
   b. 浏览至 State_to_Runtime\source 子目录，并找出所有的 StopCapture_<number> 脚本。
   c. 对于该业务度量模型的每个部署，重复上述步骤。
2. 停止“捕获”组件实例。 已识别的“捕获”组件实例必须在状态数据库所在的机器上停止。如果已经整合了停止脚本，则启动已整合的停止脚本。如果未进行整合，则需要执行上一步中已识别的每个停止脚本。可以按照任意次序启动停止脚本。

   注: 停止脚本是异步工作的，在有些情况下，在发出停止命令和“捕获”组件停止之间可能会有延迟。这是因为“捕获”组件实例在停止之前要先完成事务。
3. 识别运行时数据库中已经指定给业务度量模型的所有“应用”组件实例。 如果已经整合了所有“应用”组件实例的停止脚本，则不需要做任何事。继续下一步。如果还未整合脚本（并不希望整合它们），则需要识别所有已为此业务度量模型创建的“应用”组件实例。在第一次为业务度量模型执行模式生成时，会自动生成“应用”组件实例的停止脚本。随后的模式生成（例如，在更新业务度量模型之后）仅会为新“应用”组件实例生成停止脚本。要识别所有相关停止脚本，对于为该业务度量模型执行的每个部署，您都必须重复以下步骤:
   a. 浏览至执行该模型部署的目录。

b. 浏览至 State_to_Runtime\target 子目录，并找出所有的 StopApply_<number> 脚本。

c. 对于该业务度量模型的每个部署，重复上述步骤。

4. 停止“应用”组件实例。

已识别的“应用”组件实例应在运行时数据库所在的机器上停止。如果已经整合了停止脚本，则启动已整合的停止脚本。如果未进行整合，则需要执行上一步中已识别的每个停止脚本。可以按照任意次序启动停止脚本。

注: 停止脚本是异步工作的，在有些情况下，在发出停止命令和“应用”组件停止之间可能会有延迟。这是因为“应用”组件实例在停止之前要先完成一个或多个事务。

停止运行时至历史的数据移动服务:

停止运行时至历史的数据移动服务的过程非常类似于启动它的过程。部署归档 DS_Runtime_setup 和 DS_Datamart_setup 包含可执行文件脚本，可以使用这些脚本停止用于运行时至历史的数据移动服务的“捕获”和“应用”组件实例。

如果由于更改业务度量模型而创建归档，则仅会打包新“捕获”和“应用”组件实例的停止脚本。

注: 建议在停止数据移动服务之前整合脚本。要获取更多关于复制脚本整合的信息，请参阅第 43 页的『整合启动和停止脚本』。

然而，即使未执行整合，仍可使用以下指示信息。

要对给定的业务度量模型停止运行时至历史的数据移动服务:

1. 识别运行时数据库中已经指定给业务度量模型的所有“捕获”组件实例。 如果已经整合了所有“捕获”组件实例的停止脚本，则不需要做任何事。继续下一步。如果还未整合脚本（并不希望整合它们），则需要识别所有已为此业务度量模型创建的“捕获”组件实例。在第一次为业务度量模型执行模式生成时，会自动生成“捕获”组件实例停止脚本。随后的模式生成（例如，在更新业务度量模型之后）仅会为新“捕获”组件实例生成停止脚本。要识别所有相关停止脚本，对于为该业务度量模型执行的每个部署，您都需要重复以下步骤。

   a. 浏览至执行该模型部署的目录。

   b. 浏览至 Runtime_to_Historical\source 子目录，并找出所有的 StopCapture_<number> 脚本。

   c. 对于该业务度量模型的每个部署，重复上述步骤。

2. 停止“捕获”组件实例。 已识别的“捕获”组件实例必须在运行时数据库所在的机器上停止。如果已经整合了停止脚本，则启动已整合的停止脚本。如果未进行整合，则需要执行上一步中确定的每个停止脚本。可以按照任意次序启动停止脚本。

   注: 停止脚本是异步工作的，在有些情况下，在发出停止命令和“捕获”组件停止之间可能会有延迟。这是因为“捕获”组件实例在停止之前要先完成事务。

3. 识别历史数据库中已经指定给业务度量模型的所有“应用”组件实例。 如果已经整合了所有“应用”组件实例的停止脚本，则不需要做任何事。继续下一步。如果还未整合脚本（并不希望整合它们），则需要识别所有已为此业务度量模型创建的“应用”组件实例。在第一次为业务度量模型执行模式生成时，会自动生成“应用”组件实例

停止脚本。随后的模式生成（例如，在更新业务度量模型之后）仅会为新“应用”组件实例生成停止脚本。要识别所有相关停止脚本，对于为该业务度量模型执行的每个部署，您都必须重复以下步骤:

a. 浏览至执行该模型部署的目录。

b. 浏览至 Runtime_to_Historical\target 子目录，并找出所有的 StopApply_<number> 脚本。

c. 对于该业务度量模型的每个部署，重复上述步骤。

4. 停止“应用”组件实例。

已识别的“应用”组件实例应在运行时数据库所在的机器上停止。如果已经整合了停止脚本，则启动已整合的停止脚本。如果未进行整合，则需要执行上一步中已识别的每个停止脚本。可以按照任意次序启动停止脚本。

注: 停止脚本是异步工作的，在有些情况下，在发出停止命令和“应用”组件停止之间可能会有延迟。这是因为“捕获”组件实例在停止之前要先完成一个或多个事务。

部署 Cube Views 数据库模式

模式生成器以 XML 文件格式生成 Cube Views 元数据。它表示对应于业务度量模型的 DB2 Cube Views 定义。已在 Windows 和 AIX 平台上部署了 Cube Views 定义。

在 Windows 平台上部署 Cube Views 数据库模式:

Cube Views 元数据存储在模式生成器输出文件夹中。该输出文件夹由用户通过 WebSphere Business Monitor 管理控制台定义。

要部署 Cube Views

元数据文件，请完成下列步骤:

1. 启动 DB2 OLAP 中心。出现 **DB2 数据库连接**对话框。
2. 在 **DB2 数据库连接**对话框中，执行以下操作:
   a. 在**数据库名称**字段中，输入历史数据库的名称。
   b. 在**用户名**字段中，输入具有数据库管理权限的用户的用户标识。
   c. 在**密码**字段中，输入具有数据库管理权限的用户的密码。
   d. 单击**确定**。
   e. 在您第一次连接到数据库时，可能会出现一则消息，通知您需要为 Cube Views 配置数据库。单击该消息上的**是**，开始初始化和配置操作。
3. 在 OLAP 中心窗口中，按以下步骤导入 Cube Views 元数据文件:
   a. 从菜单中选择 **OLAP 中心 → 导入**。启动导入向导。
   b. 选择存储在模式生成器输出文件夹中的 Cube Views XML 文件。文件名是 *model_cv.xml*。
   c. 单击**完成**。启动导入过程。
4. 导入完成之后，在**导入向导**窗口的**导入选项**页面上单击**完成**。

在 AIX 平台上部署 Cube Views 数据库模式。:

Cube Views 元数据存储在模式生成器输出文件夹中。该输出文件夹由用户通过 WebSphere Business Monitor 管理控制台定义。

要部署 Cube Views

元数据文件，请完成下列步骤:

1. 打开 DB2 **命令窗口**编辑器。
2. 通过运行命令 **db2 connect to HISTORICAL_database_name**，以数据库实例用户（例如: db2inst1）身份连接到历史数据库。
3. 将目录更改为 **<DB2_INST_HOME>/sqllib/misc** 目录，然后运行命令: **db2 -tvf db2mdapi.sql**。
4. 运行命令: **db2mdapiclient -d HISTORY -i <GENERATION_DIR>/schemagen/import_model.xml -m <GENERATION_DIR>/schemagen/model_cv.xml> -u <userid> -p <pw> -o <GENERATION_DIR>/schemagen/myoutput.xml**。

   其中

   - -d 是历史数据库名称。
   - -i 是模式生成器生成的 import_model.xml 文件。
   - -u 是用户标识。
   - -p 是密码。
   - -o 是存储 DB2 输出信息的输出操作文件名。
   - -m 是 DB2 的输入元数据命令或指示信息。模式生成器生成了用作多维元数据的 model_cv.xml 文件。
   - <GENERATION_DIR> 指模式生成器存储其生成的工件的输出目录。

示例:

```
su - db2inst1
db2 connect to HISTORY
cd /home/db2inst1/sqllib/misc
db2 -tvf db2mdapi.sql
db2mdapiclient -d HISTORY
-i /opt/IBM/WebSphere/Monitor/generation/schemagen/import_model.xml
-m /opt/IBM/WebSphere/Monitor/generation/schemagen/model_cv.xml
-u db2inst1 -p monPa55w -o /tmp/import_output.xml
```

手工创建 ABX 立方体:

您在安装了 IBM DB2 ALPHABLOX服务器的机器上手工创建了 ABX 立方体。这些立方体可由 WebSphere Business Monitor 仪表板使用。

在部署了 cube-views 定义之后，在您使用仪表板之前，请完成以下步骤:

1. 将 Web 浏览器指向: http://<hostname>:9081/AlphaBlox/home/Admin，然后登录 IBM DB2 ALPHABLOX 管理控制台。
2. 选择**管理**选项卡
3. 单击**立方体**。

4. 要创建立方体，请单击**创建**。
   a. 在**关系数据源**列表中，选择安装时定义的适当的历史数据库。
   b. 选中 **DB2 AlphaBlox 立方体名称**旁边的**已启用**复选框。
   c. 选中**启用 DB2 Cube Views 设置**复选框。等待一会儿，直到字段可见。
5. 对于在**立方体模型**下定义的每个立方体，必须创建一个立方体。
   a. 从**立方体模型**列表中选择立方体模型。
   b. 从**立方体**列表中选择立方体。每个立方体模型只有一个立方体。
   c. 在 **DB2 AlphaBlox 立方体名称**字段中，输入立方体名称。该名称应当就是出现在**立方体**列表中的名称。示例: CISS.NOOP。 不要包括 CISS，它是模式名称。
   d. 选择**使用业务名**选项。
   e. 单击**导入立方体定义**，并在处理时等待。
6. 单击**确定**以保存该立方体。
7. 对每个存在的立方体重复步骤 5（立方体创建）。

手工填充维表

您可能拥有将用作维数据的现有数据 – 例如，用于填充客户维的客户信息数据库。您可以使用历史数据库，利用这些数据来手工填充维表。

在填充表时，有几点需要注意。

- 当在 WebSphere Business Modeler 中创建维时要特别注意，这样才能够用现有数据来填充维。确保您在 WebSphere Business Modeler 中定义的维包含具有适当数据类型的适当度量，从而可以将现有数据存储到模式生成器创建的维表中。
- 在手工插入数据时，请在 SK_<> 列使用负值。这是表的代理键。正代理键值由数据服务在填充这些表时使用；为避免冲突，您必须使用负值。
- 在将数据插入维表时，请确保没有将任何列设置成空值。如果不存在有意义的值来插入给定列，则必须选择一个有意义的缺省值，并使用它。不要将 NULL 插入该表。但是，空字符串（″″）可用于字符串数据类型。
- 在您将新过程实例映射到已插入的维数据时，过程实例可能与现有数据不匹配（例如，与当前未列入维表的新客户关联的过程）。在这种情况下，要在表中为这组数据创建新的一行。现在，该表包含了您输入的数据和其他数据。
- 在输入新数据时，会更新维的非键属性。例如，假设您拥有一个客户维，其键度量是“CustomerName”，而非键度量是“CreditLimit”。最初，该表可能包含来自现有客户数据的行 [’Widgets, Inc’,50000]。如果处理了一个新事件，其中“Widgets, Inc”的 CreditLimit 为 75000，则客户维表行将更新成 [’Widgets, Inc’, 75000]。只在键度量与现有行匹配，而非键度量不匹配的情况下，才会进行该更新。在这些情况下，将更新非键值以反映新数据。

要确定与您手工填充的维对应的维表，以及表中与各种维属性对应的列，请使用 *datamartMapping.txt* 文本文件，该文件位于模式生成器输出目录中（在您运行模式生成器之后）。

历史数据库模式

数据库模式描述了数据库表和它们之间的关系。您可以使用数据库模式来规划数据库的大小。

历史数据库模式中的信息可以帮您了解导入的业务度量模型和数据库表之间的映射。仪表板使用历史数据库进行多维分析并生成报告。

注:

- 存储库、状态和运行时数据库仅用于内部使用，可以对它们进行更改而无需通知。
- IBM 不支持用于直接访问状态、运行时或存储库数据库的客户编写的定制代码。
- 您不能使用历史数据库模式创建自己的仪表板。

历史数据库最初是由日期／时间数据填充的，范围从 1995 年到 2009 年。如果预计记录日期／时间（过程启动／终止时间或其他度量数据）会超出此日期范围，您应当使用以下 SQL 脚本将额外的日期添加到历史数据库中的 DIM_TIME 表:

```
insert into <your WBI schema name>.dim_time( surrogate_key, year, month, day)
with WBITIME (skey, ldate) as
(select surrogate_key+1 as skey,
COALESCE(
DATE(SUBSTR(DIGITS(YEAR),7,4) || '-' ||
SUBSTR(DIGITS(MONTH),4,2) || '-' ||
SUBSTR(DIGITS(DAY), 4,2)) + 1 DAYS,
DATE('YYYY-MM-DD of the first day you'd want to start from,
in case the DIM_TIME table is empty.')
)as ldate
from sysibm.sysdummy1, <your WBI schema name>.dim_time
where
DATE(
SUBSTR(DIGITS(YEAR) ,7,4) || '-' ||
SUBSTR(DIGITS(MONTH),4,2) || '-' ||
SUBSTR(DIGITS(DAY) ,4,2)
) =
(
SELECT
MAX(
DATE(SUBSTR(DIGITS(YEAR),7,4) || '-' ||
SUBSTR(DIGITS(MONTH),4,2) || '-' ||
SUBSTR(DIGITS(DAY), 4,2)))
FROM <your WBI schema name>.DIM_TIME
)
UNION ALL
SELECT parent.skey+1, ldate + 1 DAYS
from WBITIME parent
where YEAR(ldate + 1 days) < where YEAR(ldate + 1 days) <
<YYYY 4 Digit YEAR FOR WHICH YOU DON't WANT DATA to end in>
)
select a.skey, year(a.ldate), month(a.ldate), day(a.ldate)
from WBITIME a
WHERE
```

```
a.ldate >= DATE('YYYY-MM-DD: The start of the range that should be inserted.')
AND a.ldate <= DATE('YYYY-MM-DD: The end of range that
should be inserted.')
```

注: 本脚本更新了四个位置，用以指定希望插入到 DIM_TIME 的数据开始和结束日期。还有三个必须指定 WBI 模式名称（一般为“WBI”）的位置

数据库服务

该参考信息将帮助您使用数据库服务。

历史数据库模式

历史数据库表被分成两种类型。它们是在 WebSphere Business Monitor 安装时创建的静态表，以及针对每个导入的业务度量模型创建的动态表。

下表列出了两种类型的历史数据库表，以及每一列到业务度量模型的映射。

注:

- 可空：表示该列要么接受空值，要么不接受空值
- 描述符：描述列和业务度量模型定义之间的映射。每列不一定都有一个描述符。

静态数据库表

DIM_TIME

时间维表。

列名	列类型	列描述	可空
SURROGATE_KEY	INTEGER	这是主键	N
DAY	SMALLINT	表示日	N
MONTH	SMALLINT	表示月	N
YEAR	INTEGER	表示年	N

动态数据库表

历史数据库使用星型模式结构，中央“事实表”周围围绕着多个维“叶”表。事实表类似于状态和运行时数据库中的上下文表。对于上下文和上下文对应的活动，都存在一个星型。例如，在状态和运行时数据库中，每个上下文可以有一个上下文实例表和一个活动实例表。

上下文事实表

命名规范：FCT_<machine generated name of context>

总是存在的列是:

列名	列类型	列描述	可空
MCI_MCIID	DECIMAL(19,0)	活动实例的唯一标识和表的主键。	N
PARENT_MCIID	DECIMAL(19,0)	父过程实例的唯一标识（如果存在）。	Y
SK_<machine generated name of the dimension>	INTEGER	指向维表的外键。定义了 FK 关系。针对上下文中表示的每个维定义其中的一个列。	Y

上下文事实表

命名规范: FCT_<machine generated name of context>

总是存在的列是:

列名	列类型	列描述	可空
GMT_<machine generated name of metric>	TIMESTAMP	时间戳记用于存储任何时间戳记度量数据类型的 GMT 时间值。(只有将时间度量标记为“维”时，才创建该列。当将“时间”度量标记为维时，只按日、月、年详细程度进行存储，因此该列可以提供查看这些度量的准确时间值的能力。)	Y

当度量标记为“事实”(非维)时，使用下列三个列类型。

当将度量标记为“事实”时，使用的列类型

列名	列类型	列描述	可空
M_<machine generated name>	数据类型根据业务度量模型中定义的数据类型的不同而不同。	用于表示度量或 Keydefinition 值。	Y
C_<machine generated name>	BIGINT	用于表示计数器。	Y
T1_<machine generated name>	BIGINT	用于表示计时器累计时间。(计时器在历史数据库中以单个列表表示；在“状态”数据库中则使用多个列。)	Y

维表

根据上下文定义的维数，为每个上下文定义 0 个或多个维表。. 通常至少有一个“时间”维。

命名规范: DIM_<machine generated name of dimension>

一直定义的列是:

列名	列类型	列描述	可空
SURROGATE_KEY	INTEGER	针对该维行的机器生成的主键。定义了 PK。	N

存在基于定义的列。维表包含与每个定义为该维一部分的度量对应的列。

基于定义的列

列名	列类型	列描述	可空
M_<machine generated name>	数据类型根据业务度量模型中定义的数据类型的不同而不同。	用于表示度量或键定义值。	Y

基于定义的列

列名	列类型	列描述	可空
C_<machine generated name>	BIGINT	用于表示计数器。	Y
T1_<machine generated name>	BIGINT	用于表示计时器累计时间。（计时器在“历史”数据库中以单个列表示，在“状态”数据库中则使用多个列）。	Y

活动的星型模式使用相同的规范，区别在于表分别命名为 AFC_ 和 ADM_。

数据移动服务控制表

本部分描述了“数据移动服务”控制表结构。“状态”、“运行时”和“历史”数据库中的每一个都包含两个控制表，可定制该表以配置本地数据控制移动服务组件的行为。控制表是静态表。

RMCONTROL

包含特定于 ETL 组件实例状态的配置设置。只需要在“运行时”和“历史”数据库中填充和使用该表，因为“状态”数据库中不需要 ETL 组件。表中的每一行都与一个需要填充的目标表相对应。更改给定的行的列值仅会影响指定用于填充目标值的 ETL 组件实例。

列名	列类型	列描述	可空
TARGETTABLE	CHARACTER	该项控制的存储过程填充的目标表的标准表名称。	N
COMMITINTERVAL	NUMERIC	当光标用于将行插入目标表时，存储过程使用的落实时间间隔。	Y
LOGLEVEL	NUMERIC	日志记录级别，它确定将多少信息放入 WBIRMADM.RMLOG 表中。有效值是 0 和 1。0 表示最小日志记录，1 表示最大日志记录。	Y
LASTSEQUENCE	CHARACTER	ETL 存储过程处理来自登台表的最新 SEQUENCE 值。存储过程在运行时更新该列。	N
LASTUPDATED	TIMESTAMP	确定上一次计划调用的时间。该列由存储过程控制，并仅供调度之用。	Y
NEXTSTARTTIME	TIMESTAMP	下一次执行 ETL 调用的时间。	Y
ETLSCHEDMETHOD .	NUMERIC	将要使用的调度方法。只有 0 是有效值。	Y
ETL_0_MINUTES	NUMERIC	调度 ETL 运行期间应耗用的时间（以分钟计）。	Y
TGT_RM_SPETL_NAME	CHARACTER	负责填充 TARGETTABLE 的存储过程的标准名称。	Y

不要更改下列任何列值，否则会发生不可预料的行为:

- TARGETTABLE
- LASTSEQUENCE
- LASTUPDATED

- ETLSCHEDMETHOD
- TGT_RM_SPETL_NAME

当下一次调用 ETL 组件实例时，将落实对以下列的更改:

- COMMITINTERVAL
- NEXTSTARTTIME
- LOGLEVEL
- ETL_0_MINUTES

RMPRUNECTRL

包含特定于“生命周期”组件实例行为的配置设置。在“状态”、“运行时”和“历史”数据库中填充和使用该表。该表中的每一行都与一个需要修剪的（源或工作）表 <TABLE_NAME> 相对应。更改给定行的列值仅会影响指定用于修剪表 <TABLE_NAME> 的“生命周期”组件实例。

列名	列类型	列描述	可空
TABLE_NAME	CHARACTER	将要修剪的表的标准名称。	N
LAST_PRUNED	TIMESTMP	该表中上一次修剪操作的时间。	Y
LOGLEVEL	NUMERIC	日志记录级别，它确定将多少信息放入 WBIRMADM.RMLOG 表中。有效值是 0 和 1。0 表示最小日志记录；1 表示最大日志记录。	N
PRUNE_ENABLED	NUMERIC	确定是否应该发生修剪操作的标志。0 表示否，1 表示是。	N
PRUNE_INTERVAL	NUMERIC	修剪操作期间的最小总时间（单位，分钟）。	N
RETENTION_IN_MINUTES	NUMERIC	在可以修剪符合条件列之后的时间长度（单位，分钟）。	N
ROWS_PRUNED	NUMERIC	上一次修剪操作期间修剪的行数。	N

不要更改下列任何列值，否则会发生不可预料的行为:

- LAST_PRUNED
- ROWS_PRUNED
- TABLE_NAME

当下一次调用 ETL 组件实例时，将落实对以下列的更改:

- LOGLEVEL
- PRUNE_ENABLED
- PRUNE_INTERVAL
- RETENTION_IN_MINUTES

数据移动服务元数据和日志记录表

本部分提供关于 WebSphere Business Monitor 数据库中日志表结构的参考信息。日志表是静态表。

RMMETADATA

有不同的组件实例用于为给定的业务度量模型提供数据移动服务。每个“状态”、“运行时”和“历史”数据库包含一个表，该表针对每个业务度量模型列出了指定的组件实例名称和其他有用的内部信息。在部署阶段每次创建和配置组件实例时，更新该表。不能手工修改其内容。

列名	列类型	列描述
ID	NUMERIC	未使用
OM_NAME	CHARACTER	这些复制工件所服务的相关业务度量模型项目的名称。
OM_ID	NUMERIC	未使用
MC_NAME	CHARACTER	这些复制工件所服务的关联业务度量组的名称。
MC_ID	NUMERIC	未使用
TGT_TAB_NAME	CHARACTER	ETL 存储过程填充的目标表的标准名称。
TGT_RM_APP_SVR_NAME	CHARACTER	负责运行复制应用操作的服务器的名称。
TGT_RM_APP_SS_NAME	CHARACTER	如果存在，则是针对 DB2 SQL 复制的，“应用”服务器管理的组。这是一个预订集。
TGT_RM_APP_STG_TAB_NAME	CHARACTER	复制应用程序作为目标使用的登台表的标准名称。 **注**: 在系统中，还有其他两个扩展名为 *_BKUP* 和 *_M* 的表与该行关联。
TGT_RM_APP_ERR_TAB_NAME	CHARACTER	存储指向登台表（仍需 ETL 进行处理）中行的指针的表的标准名称。
TGT_RM_APP_PRUNE_SP_NAME	CHARACTER	负责修剪目标系统上的应用登台表的存储过程的标准名称。
TGT_RM_APP_TMP_TAB_NAME	CHARACTER	ETL 程序用来确定装入到目标表中的行的临时表的标准名称。
TGT_RM_SPETL_NAME	CHARACTER	负责从登台表中的项填充目标表的 ETL 存储过程的标准名称。
SRC_TAB_NAME	CHARACTER	正在复制到登台表的源表的标准名称。
SRC_RM_CAP_SVR_NAME	CHARACTER	负责运行复制捕获操作的服务器的名称。
SRC_RM_CAP_STG_TAB_NAME	CHARACTER	“捕获”服务器用来存储对源表更改的表的标准名称。

RMMETADATA

有不同的组件实例用于为给定的业务度量模型提供数据移动服务。每个“状态”、“运行时”和“历史”数据库包含一个表，该表针对每个业务度量模型列出了指定的组件实例名称和其他有用的内部信息。在部署阶段每次创建和配置组件实例时，更新该表。不能手工修改其内容。

列名	列类型	列描述
SRC_RM_PRUNE_TRG_NAME	CHARACTER	负责在“捕获”服务器修剪期间，从源表中除去所选行的触发器的标准名称。所选行可以包括表示已完成操作的行。
SERVICE_NAME	CHARACTER	用于标识这些工件所属服务的标签，例如，State_to_Runtime 或 Runtime_to_Historical。

使用下列简化的示例视图:

OM_NAME	SRC_TAB_NAME	SRC_RM_CAP_SV...	SRC_RM_CA...	TGT_RM_AP...	TGT_RM_AP...	TGT_TAB_NAME	SERVICE_NAME
STEW_S	wbi.CTX_TQ4MUF...	CAPTURE_1	CAP.CD_2	APPLY_4	APP.CCD_6	wbi.CTR_TQ4MUF...	State to Runtime
STEW_S	wbi.AI_BVSOYAP...	CAPTURE_1	CAP.CD_3	APPLY_4	APP.CCD_7	wbi.AIR_BVSOYA...	State to Runtime

可以容易地确定“捕获”组件实例 CAPTURE_1 正在监控“状态”数据库中源 WBI.CTXTQ4MUF。任何对源表的更改都记录在工作表 CAP.CD_2 中，供“应用”组件实例 APPLY_4 应用到工作表 APP.CCD_6。该表供 ETL 组件实例使用，来填充“运行时”数据库中的目标表 WBI.CTXTQ4MUF。

RMLOG

“运行时”和“历史”数据库都各包含一个可用于获取统计信息、进度、调试或错误信息的日志表。所有 ETL 组件和“目标生命周期”组件将消息写入该表，但是不能从中读取消息。可以将日志记录级别设置为最小值，来禁止某些消息。

列名	列类型	列描述
ENTRYSTMP	TIMESTAMP(10)	该日志表中特定项的时间戳记。
ID	NUMERIC	将同一实例中的多个行关联在一起的标识。该标识来自 SEQUENCE WBIRMADM.RMSPTRIGID。
ROWS_INSERTED	NUMERIC	该实例中插入的行数的指示符。
ROWS_UPDATED	NUMERIC	该实例中更新的行数的指示符。
ROWS_DELETED	NUMERIC	该实例中删除的行数的指示符。
ROWS_INERROR	NUMERIC	该实例中，标记为导致可恢复错误的行数的指示符。
NAME	CHARACTER	使该表中添加项的存储过程、触发器或过程的标准名称。
OPERATION	CHARACTER	标识输入该项时执行的操作的标签。
RESULT	CHARACTER	可以在其中找到所发生操作的更多信息的列。

RMLOG

“运行时”和“历史”数据库都各包含一个可用于获取统计信息、进度、调试或错误信息的日志表。所有 ETL 组件和“目标生命周期”组件将消息写入该表，但是不能从中读取消息。可以将日志记录级别设置为最小值，来禁止某些消息。

列名	列类型	列描述
ISTRACEENTRY	NUMERIC	标识该项是否需要将 LOGLEVEL（位于 WBIRMADM.RMCONTROL 中）设置成 1 的列。 0: 该日志项不是跟踪项。 1: 该日志项是跟踪项（可以取消 – 请参阅 WBI.RMCONTROL 表）。

该表中的每一行都与 <ENTRYSTMP> 中的组件实例 <NAME> 发布的消息保持一致。具有相同 <ID> 和 <NAME> 的行表示在同一 <NAME> 调用期间生成的消息。下列示例包含 ETL 组件实例 WBIRMADM.WBIRMSP_10 和 WBIRMADM.WBIRMSP_14，以及“目标生命周期”组件实例 WBIRMADM.WBIRMSP_P13 和 WBIRMADM.WBIRMSP_P_17 生成的日志项。WBIRMADM.WBIRMSP_10 (4:40:20 PM) 和 WBIRMADM.WBIRMSP_14 (4:40:27 PM) 各发出五条消息，WBIRMADM.WBIRMSP_P_13 (4:40:20 PM) 和 WBIRMADM.WBIRMSP_P_17 (4:40:20 PM) 各发出一条消息。

ENTRYSTMP	ID	NAME	OPERATION	ROWS_INSERTED
Oct 11, 2005 4:40:20 PM 3...	1	WBIRMADM.WBIRMSP_10	SP_START	0
Oct 11, 2005 4:40:20 PM 3...	1	WBIRMADM.WBIRMSP_10	DEL_TEMP	0
Oct 11, 2005 4:40:20 PM 3...	1	WBIRMADM.WBIRMSP_10	INS_TEMP	0
Oct 11, 2005 4:40:20 PM 3...	1	WBIRMADM.WBIRMSP_10	FETCH_TARGET_...	0
Oct 11, 2005 4:40:20 PM 3...	1	WBIRMADM.WBIRMSP_10	SP_END	0
Oct 11, 2005 4:40:20 PM 3...	2	WBIRMADM.WBIRMSP_P_13	PRUNESTAGING	0
Oct 11, 2005 4:40:20 PM 3...	3	WBIRMADM.WBIRMSP_P_17	PRUNESTAGING	0
Oct 11, 2005 4:40:27 PM 1...	4	WBIRMADM.WBIRMSP_14	SP_START	0
Oct 11, 2005 4:40:27 PM 1...	4	WBIRMADM.WBIRMSP_14	DEL_TEMP	0
Oct 11, 2005 4:40:27 PM 1...	4	WBIRMADM.WBIRMSP_14	INS_TEMP	0
Oct 11, 2005 4:40:27 PM 1...	4	WBIRMADM.WBIRMSP_14	FETCH_TARGET_...	0
Oct 11, 2005 4:40:27 PM 1...	4	WBIRMADM.WBIRMSP_14	SP_END	0

不能自动修剪该表。DBA 应该定期监控和修剪它。使用 WBIRMADM.RMMETADATA 中的信息确定组件实例 <NAME> 所服务的业务度量模型。请注意，WBIRMADM.RMCONTROL 表中的 LOGLEVEL 和 ETL_0_MINUTES 列的值以及 WBIRMADM.RMPRUNECTRL 表中 LOGLEVEL 和 PRUNE_INTERVAL 的值会影响该表的增长速度。当 LOGLEVEL 设置成 1，并且 ETL_0_MINUTES 减少而 PRUNE_INTERVAL 减少时，会生成更多的项。

数据库服务故障诊断

在生成、部署或运行 WebSphere Business Monitor 的数据库服务期间，可能会发生与数据库服务相关的错误。下列是如何针对相关问题对数据库进行故障诊断的信息。

部署问题

在 WebSphere Business Monitor 数据库工件的不同部署方案中，可能发生错误。以下是针对每个错误所建议的解决方案。

在部署生成的数据库工件期间，可能会发生由以下原因导致的问题:

- 不正确的配置
- 用户特权不够
- 环境设置问题

表 2. 部署错误

问题	解决方案
分配给表的表空间不存在。	• 请确保在表空间属性文件中具有定义的表空间，并且具有所描述的特征。 • 利用相应的属性特征创建缺少的表空间，然后重新运行 DDL；或者更新表空间属性文件，以与定义的表空间匹配，然后重新生成模式。
分配给表的表空间太小，无法容纳表。	• 请确保在表空间属性文件中具有定义的表空间，并且具有所描述的特征。 • 修订并重新运行，或者手工编辑 DDL，来纠正表空间分配。

表 2. 部署错误 (续)

问题	解决方案
表已存在于数据库中。	假设先前未运行该 DDL，则造成该问题的原因可能有两种。 • 一种原因是，在选择了**忽略先前部署**选项的情况下，生成 DDL 脚本。 “模式生成器”生成新的表创建语句，而不是更改现有表。只有在您试图从头开始创建数据库表时，才应该使用该选项；也就是说，当您删除了现有数据库表时。如果您知道数据库表已存在，并且想要保留它们，则请在不选择**忽略先前部署**选项的情况下，重新运行“模式生成器”，并重新运行生成的 DDL 脚本。 • 另一种原因是，在某时刻，该业务度量模型的一个版本已经被删除，但未选定*删除并保留以进行报告*选项。如果删除了业务度量模型的一个版本，并且未选择*保留以进行报告*选项，那么“模式生成器”不能继续管理对于该业务度量模式的支持数据库表的更改。此时有两个选项。 – 分支出来，根据当前业务度量模型创建新的，并将其部署为带有新表集的新业务度量模型。您可以手工将数据从现有表集中迁移到新表。 – 使用提供的映射文件作为向导，手工删除支持业务度量模型的现有数据库表。一旦删除该表，则选择**忽略先前部署**选项，重新运行“模式生成器”。生成的 DDL 脚本将创建新的表集，这些表将支持最新版本的业务度量模型。 **注**: 除非在删除旧表前手工备份旧表中的数据，然后将数据迁移到新创建的表中，否则将不会向您报告运行旧版本业务度量模型的任何过程的历史记录。 建议您，在除去模型时，不要使用*删除*选项来代替*删除并保留报告*选项，除非您不打算在将来某一时刻部署新版本的业务度量模型。
表空间太小。（虽然初始分配给该表的表空间对于列大小而言足够大，但后来添加到表中的度量使其超出了当前表空间的页面大小）。	您将需要备份该表，删除它，然后重新创建表，为其分配更大的表空间。然后，应该将备份数据装入新表。一旦在更大表空间中重新创建了该表，您将能够运行添加必需新列的 DDL 脚本。

由于各种原因，您可能想要再次从数据库表开始。例如，您可能拥有不再需要的大量度量，而因为它们存在于先前版本的业务度量模型中，而仍存在于数据库中。最简单的方法在 WebSphere Business Modeler 中重命名项目。业务度量模型将作为新方式处理，在数据库中将创建新的唯一表。

注: 在这种情况下，先前过程实例中的历史信息将不再可用。

如果想要查看历史信息，请将数据从原始数据库表复制到新创建的表。列名称将不匹配，但是数据类型匹配。您可以使用映射文件（利用 DDL 脚本生成）或数据库列注释，来标识哪些列应该与哪些度量相对应，哪些表应该与哪些过程对应。

注: 如果在最新版本的业务度量模型中不再存在这些度量，那么原表中的某些列在新表中没有对应的列。

运行时问题

当您重新启动已停机多天的“捕获”服务器时，会收到“捕获”服务器从 IBM DB2 复制生成的错误消息。消息会出现在 Windows 系统的“捕获”窗口中、系统上的日志文件中、IBMSNAP_CAPTRACE 表中、或通过复制监控传输的电子邮件发送出来。

错误消息

ASN0121E CAPTURE "CAPTURE_141" : "WorkerThread". 由于现有数据太旧，导致“捕获”程序热启动失败。“捕获”程序终止。

要解决该错误，请参阅准备数据库工件部署

停止运行时数据库

当由于某些原因停止或关闭 WebSphere Business Monitor 运行时数据库时，您首先应该停止自适应操作管理器应用程序。

您可以从 WebSphere

Process Server 管理控制台停止自适应操作管理器应用程序。WebSphere Business Monitor 运行时数据库应该在启动自适应操作管理器之前启动。

声明和商标

声明

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:

Lab Director
IBM RTP Laboratory
3039 Cornwallis Road
P.O. BOX 12195
Raleigh, NC 27709-2195
U.S.A

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

本资料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下测得的。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。

这些信息可能包含日常业务操作中使用的数据和报告示例。为了尽可能完整地描述它们，示例可能包含个人、企业、品牌和产品的名称。所有这些名称都是虚构的，如果与实际商业企业所用的名称和地址有任何相似，纯属巧合。

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。

版权许可

本信息可能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。

编程接口信息

提供编程接口信息是为了帮助您使用本程序来创建应用软件。

通用的编程接口使您可以编写获得本程序工具的服务的应用软件。

但是，本信息也可能包含诊断、修订和调优信息。提供诊断、修订和调优信息是为了帮助您调试您的应用软件。

警告：由于这些诊断、修订和调优信息会随时更改，请不要将它们用作编程接口。

商标和服务标记

以下术语是 International Business Machines Corporation 在美国和／或其他国家或地区的商标或注册商标:

IBM
IBM（徽标）
WebSphere
DB2
Tivoli
MQSeries
AIX
z/OS

Excel、Microsoft、Windows、Windows NT 和 Windows 徽标是 Microsoft Corporation 在美国和 / 或其他国家或地区的商标。

Intel、MMX 和 Pentium 是 Intel Corporation 或其子公司在美国和 / 或其他国家或地区的商标或注册商标。

UNIX 是 The Open Group 在美国和 / 或其他国家或地区的注册商标。

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

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。

ALPHABLOX 是 Alphablox Corporation 在美国和 / 或其他国家或地区的注册商标。

Adobe 是 Adobe Systems Incorporated 在美国和 / 或其他国家或地区的商标。

其他公司、产品和服务名称可能是其他公司的商标或服务标记。