

目录

关于空间数据库.....	3
空间数据库的用途.....	3
FME 支持的空间数据库	5
常见问题.....	5
创建一个数据库写模块.....	7
数据检验.....	11
创建一个数据库读模块.....	12
Writer Level 参数.....	14
写模块模式（Writer Mode）	15
事务处理参数（Transaction Parameters）	15
Strict 参数	16
Feature Type Level 参数	16
更新键.....	17
几何属性列.....	18
删掉/清空表	18
创建索引.....	19
导入目标要素类定义.....	21
创建时导入要素类.....	22
在编辑过程中使用导入要素类选项	23
用转换器格式化数据属性	25
数据更新.....	26
为什么要更新？	26
示例方案.....	26
数据库和 Table Level 的更新.....	27
重新加载所有内容.....	28

Feature Level 更新	32
使用格式化的属性值来标注要素	32
Transformer 更新	35
SQLExecutor	36
ArcSDEQuerier	36
OracleQuerier	37
变化检测	38
变化检测	39
数据分发	45
数据分发的益处	45
使用 FME Desktop 进行数据分发	45
简单的导出	46
针对其他用户的工作空间	47
数据转换的改善（Transformation Possibilities）	50
使用 FME Server 进行数据分发	51
Workspace Considerations 工作空间的思考	51
发布一个工作空间	52
阶段回顾	54
从这个模块中你应该学会些什么	54

空间数据库



尽管空间数据库的用途十分明显，但是仍有必要提及他们，以便对FME在其中的定位有明确的认识。

空间数据库的用途

虽然现在有上千种空间数据格式（FME就支持二百多种格式），但是总的来说可以分为两大类：数据转换格式和数据储存格式。

而空间数据库几乎是专门用来存储数据而不是转换数据的，因此，它们的主要功能是向数据库导入和从数据库导出数据。这些操作包括把数据转换为正确的格式和结构。

这样的操作通常应用在如下三个领域，并且整个培训模块也是依照这几个领域组织的：

- **数据导入**
- **数据更新**
- **数据分发**

迁移到空间数据库（数据导入）

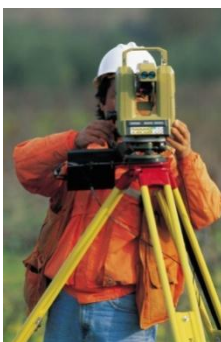
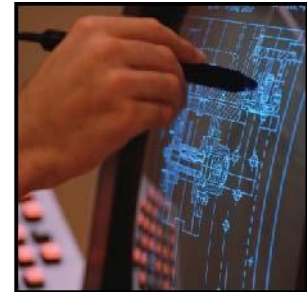
当向空间数据库迁移数据时，这些原始文件可能存储在基于文件的数据集中。因此最初面临的挑战是：

- 从基于文件数据存储格式中迁移数据
- 格式解译
- 结构映射

日常操作（数据更新）

原始数据迁移入新的空间数据库后，许多组织仍然继续着使用原来的应用程序，，这些应用程序不提供对数据库的直接访问或者有着特定功能。

例如，一家公共事业管理公司要求员工使用自定义的 CAD 软件设计了设施网络，并把部分信息存储到他们的空间数据库中。另外，公司的外业员工带着移动设备采集数据或标记出这些编辑数据。在这种情形下，处理日常更新数据的挑战在于：



- 使用从基于文件转换格式中获得的信息对数据定期的更新
- 格式解译
- 变更检测
- 将复杂数据转换为简单的空间数据

为外部应用获取数据（数据分发）



一旦数据库的数据加载和维护好之后，就要在一些应用中使用数据了。许多用户会在数据库里直接操作数据，而不直接访问数据库的另一些用户，首先需要把数据导出来。因此空间数据库中的数据需要分发给员工或客户，这些员工或客户需要数据的不同视图或结构。对这种情形，存在的挑战是：

- 按照用户的具体要求写入数据
 - 数据解译
 - 数据下载
- 提供了一种直接访问空间数据库的方法
 - 数据流

FME支持的空间数据库



现在有多种格式的空间数据库，每种空间数据库都有它自己的数据规则和规范。Safe Software 公司的目的是用一种尽可能快捷的方式支持尽可能多的数据格式和数据结构。

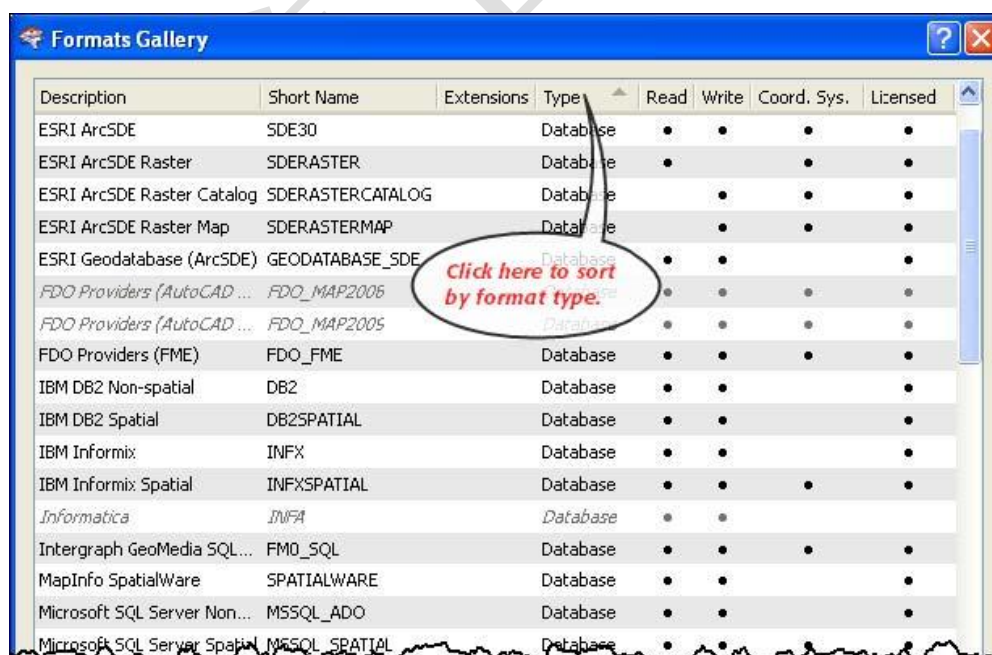
在进一步深入了解数据导入，数据更新和数据导出之前，知道 FME 都支持哪些空间数据库非常重要。

常见问题

关于 FME 和它对空间数据库支持的一些常见问题是：

FME 都支持哪些空间数据库

使用 Tools>Browse Formats 菜单浏览 Formats Gallery 窗口.....在这个窗口中你可以使用 Type 字段分类所有的格式。你会看到目前 FME 支持三十多种不同的数据库格式。



当然，这些数据库中并非所有的数据库都是“空间”数据库；有些是非空间数据库，有

些是为了在空间数据库中存储非空间数据的非空间读写器。不过，FME支持大多数主流空间数据库（不管是读操作还是写操作）例如：ESRI ArcSDE、GeoMedia Warehouses、Oracle Spatial、IBM DB2、SQL Server Spatial、PostGIS、IBM Informix、MySQL、Smallworld。

FME2010版本的目标是实现Netezza数据库的支持

不同的 FME Editions 支持的空间数据库类型

FME 有许多不同等级的许可证，称为“Editions”。



一般的，Edition 相当于你希望读写的数据库的格式。

每个版本授权你访问同等级的数据库格式。例如，ESRI Edition 允许你访问 ESRI 和

Intergraph 数据库；Oracle Edition 允许你访问 Oracle、SQL Server 和 DB2 数据库。

不过这只是一般规则，一些格式对于读写操作有不同的许可证要求，或对于空间和非空间数据库有不同的许可证要求。

在 Safe 的网站（www.safe.com/formats）上的格式页面中有每种格式和它的版本支持的详细信息。

例如，这里（下图）我们可以看到 Oracle Non-Spatial 的读出操作和写入操作在 Professional 和它以上的版本支持任何许可等级。然而对于 Oracle Spatial，Professional、ESRI or Intergraph 版本允许读出操作，但是只有 FME Oracle Edition 或与它同级的版本才支持写入操作。

Format	FME Professional Edition	FME ESRI Edition	FME Intergraph Edition	FME Microsoft SQL Server Edition	FME Oracle Edition	FME Smallworld Edition
Oracle Non-spatial <i>** requires installation of application software **</i>	R / W	R / W	R / W	R / W	R / W	R / W
Oracle Spatial Object <i>** requires installation of application software **</i>	R	R	R	R / W	R / W	R / W

虽然这看起来有些令人困惑，但是这种根据用户的需求选择他们需要的功能等级是 Safe 公司减少用户成本的一种方法。这也是我们没有在网站上公布价格而是建议您与销售代表商讨价格的原因：确保你获得满足你要求的合适版本。

使用 FME 操作数据库还有其他要求么？

从一个数据库中进行读写操作常常需要某些形式的数据库客户端。例如 Oracle 读和写需要安装 Oracle Client 或 Oracle Instant Client，Smallworld 读和写需要 Smallworld-FME 接口。像这样的客户端多数是免费的，但是在 FME 中开始使用一个特定的格式前，应该知道它需要的任何额外要求。



数据导入

很明显，使用数据库需要在数据库中确实存有数据。

逻辑上讲，首先你要做的就是将数据导入到数据库；你可能想从另一个系统上迁移大量的数据，或者通过添加数据来逐步建立你的数据库内容，直到数据库可用。不论哪种方式，你都需要数据库格式写模块来达到这个目标。

创建一个数据库写模块

这里（下图），一个用户正要转换 City Parks 数据集到 Oracle 空间数据库中。因为在



导入到数据库前要设置一系列的参数，所以需要点击 Settings 按钮以便在尽可能早的阶段完成工作空间的设置。

小提示：如果你选择的数据库写模块没有出现在写模块列表上或者它是灰色显示，你需要做如下的检查：

- 是否需要数据库客户端？如果需要安装，客户端的安装是否正确？

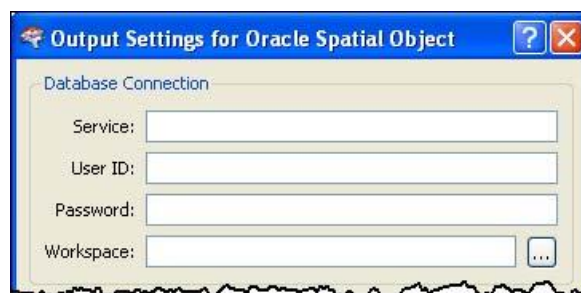
FME 不显示有效的格式，除非也安装了所需的所有应用软件。

- 你是否有正确的 FME 许可？

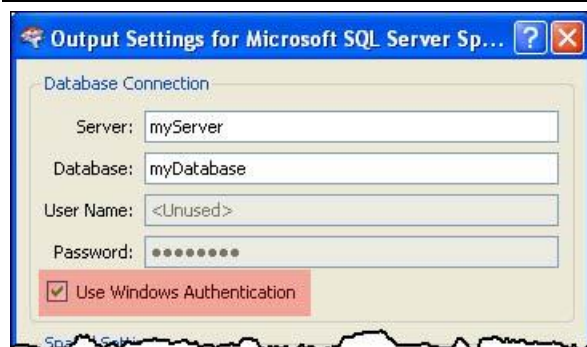
使用 Help > About FME 菜单，找到你的当前 FME 授权版本的详细信息。

连接参数

对于 Oracle 的这些连接设置(如右图所示)，是所有数据库参数中最重要的：没有这些参数，你根本不能写（或读）任何要素。



有些写模块，比如 SQL Server Spatial 写模块（下图所示），允许通过 Windows Authentication 模式连接，也就是 Windows 系统担保你的连接资格。



连接 Gotchas!

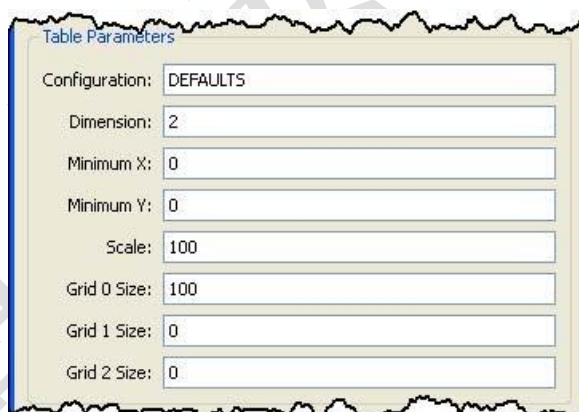
可能有很多原因导致用户不能连接到数据库，这种情况下，请查阅FME数据库读写器手册，其中针对每种数据库格式，都有一个章节负责讲解有关如何检查和修

复问题链接。

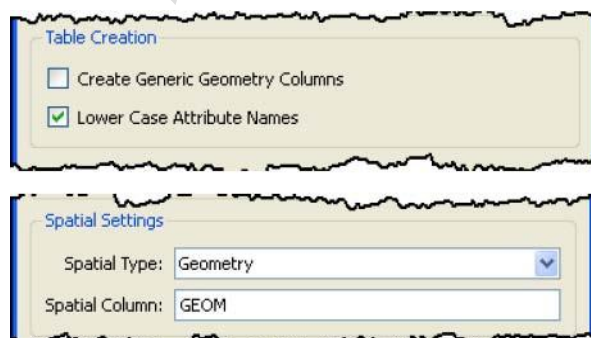
空间参数

除连接设置以外，通常每个数据库都有一套空间参数需要定义。

如右图：ArcSDE 数据库写模块需要一些其它参数。参数如 Scale 和 Grid Size 是非常重要的，所以清楚的了解你的数据并能正确设置这些参数是十分重要的。



在这里填写上所有的空间范围字段是一个好习惯。如果你不这样做，空间数据库仍能被创建，但是之后便不能再更改这个范围了。这些数据是仍可以加载，但是其他的数据库功能如索引可能不能像预料的一样工作。所以要花些时间设置好这些参数。



左图：PostGIS（上面的）and SQL Server Spatial（下面的）都是非常简单的一套空间参数。

例1— 基本的数据库写入

这个简单的练习是把一套 SDF 格式的数据迁移到 Oracle Spatial Object 格式中。

不要忘了，FME 的最佳实践经验告诉我们在转换数据前（也就是说现在）应当检查数据。

1) 创建工作空间

创建一个工作空间，用以将一个 Interopolis 市的数据集写入到 Oracle 数据库，参数设置如下：

置如下：

Source Format Autodesk MapGuide Enterprise SDF

Source Dataset

C:\FMEData\Data\DemoData\InteropolisDatabase.sdf

Destination Format Oracle Spatial Object

Destination Settings

Service: XE

User ID: training

Password: training

Minimum X: 3080000 Minimum Y: 10060000

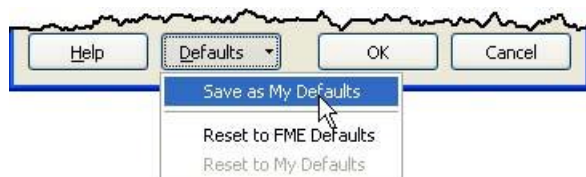
Maximum X: 3210000 Maximum Y: 10140000

Index Creation Yes

在点击 Ok 之前，其他的目标数据库设置可以保留默认值

2) 保存参数为默认值

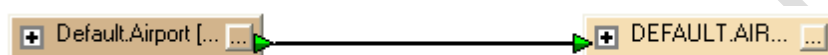
在退出数据库设置对话框前，点击对话框底部的“ Defaults” 按钮，选择 “Save as My Defaults”。



现在不论用户何时创建这种格式的一个新的工作空间，数据库连接参数将会自动完成上述设置。

3) 选择所有要素类型

弹出提示框后，选择读取所有源数据要素类型。工作空间的每一张表都有源数据和目标数据要素类型，如下图所示。



但是，目标数据库的用户名取自源数据表，我们想在一个不同的用户下写入数据。所以打开目标要素类型参数（对于每张目标数据表），更改数据库用户为“training”（如下图所示）：

小提示：“Apply to...” 按钮

虽然不是 FME 最直观的功能，

但是在这个操作中却能节省你

大量的时间。



4) 运行工作空间

运行工作空间一次，也只运行一次。

数据检验

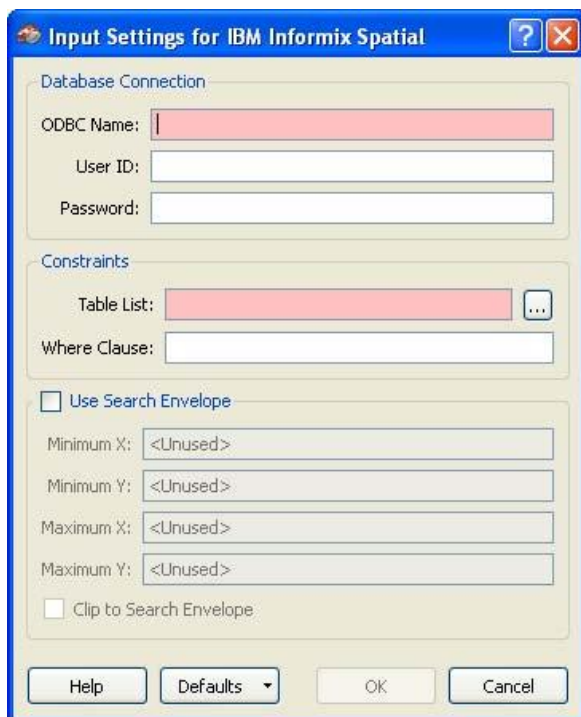


**无论你有什么信任 FME，通过检验结果数据来验证转换是否正
确都是一个好主意。**

既然数据已导入了系统中，现在我们需要检验这些数据。也就是使用 FME Universal

Viewer 中的数据库读模块。

创建一个数据库读模块



在使用一个写模块之前,在 FME 的初始化对话框中检查数据库设置是必要的。

这里(左图)一位用户正在检查 Informix (Spatial) Database 的参数。

注意数据库连接参数设置是相同的,只是有一个选项来选择读入哪个数据表,有一个选项用来设置读入过程中的 where 从句。

还有一套 Search Envelope 参数。用

户可设置 Search Envelope (空间范围) 参数,数据库读模块一般都有这个功能,设定空间范围参数意味着,数据入库时,系统会将数据按包围区域指定的范围进行剪切。请记住,所有的读写参数可以被发布,以使用户在运行时能得到及时提示。

数据库读模块的最佳实践

因为对于这个任务,数据库的工具被优化了,所以最佳实践指示我们应该尽可能多的把处理交给数据库完成。另外,我们也不希望读取过多额外的数据。记住下面几点:

什么时候使用 Where 从句

当你打算用指定的属性值过滤数据时,请使用 Where 从句。

一个 Where Clause 比起读取整个数据集,然后使用 FME 的 Tester 转换器移除多余的数

据更加高效。

什么时候使用 Search Envelope

当你打算用空间范围过滤数据时，使用 Search Envelope。

Search Envelope比起读取整个数据集然后使用FME的Clipper转换器在空间范围上限制数据流，更加有效。

例2— 基本的数据库读取操作

这个简单的例子要求你检验在例1中写入的数据

1) 打开 FME Universal Viewer

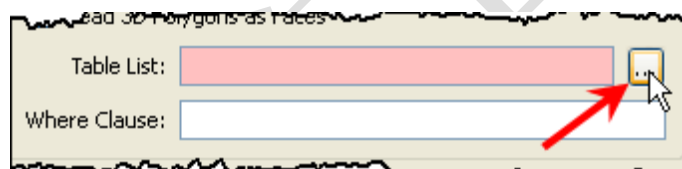
启动 FME Universal Viewer，选择 “open dataset” 工具。

2) 设置连接参数

设置格式为Oracle Spatial Object格式，点击 “Settings” 来填写数据库连接参数。

3) 读取所有数据

现在开始读取数据库中的所有数据。



点击Table List右边的[...]按钮。

这会产生一个可以读取的表的列表。选择所有表。点击OK并检查

数据。

小提示：不再有 “Deselect All” 按钮，因为现在 “Select All” 工具是一个 “三态的” 切换按钮。

4) 只读取一个 Park 要素

现在使用Where Clause参数只读取一个Park要素。

重新打开 “Open Dataset” 对话框，点击Table List右边的[...]按钮。这会产生一个可

以读取的表的列表。选则表TRAINING.CITYPARKS ,在Where Clause字段后输入 :NAME='Bartholomew'

注意：至少对Oracle而言，字段名不区分大小写，属性值应使用单引号(')而不是双引号 (")。这个信息直接传给数据库，而不是由FME解析，所以，你使用的语法应该符合所使用的数据库。

点击OK按钮，FME已经从数据库只读取到了这个指定的Park要素。

5) 使用 Search Envelope

下面试一下使用Search Envelope。

重新打开Open Dataset settings对话框并在数据表的列表中选择所有的表。

清空Where Clause，勾选上 “Use Search Envelope” 复选框。

设置参数如下：

Minimum X: 3127500 Maximum X: 3130750

Minimum Y: 10083000 Maximum Y: 10085000

点击OK按钮并检查数据。

重复这一过程，这次选中 “Clip to Search Envelope” 选项，看看结果有什么区别？

提示：如果有错误消息提示，可能是由于你忽略了设置 “Create Index” 参数。如果没有索引，则不能使用搜索范围。

使用Non-Spatial格式，尝试使用Oracle Non-spatial格式，重复第一步到第三步，只获取属性数据。

Writer Level参数



现在我们已经检查过数据并且写入过程运行顺利，下面进一步看看写模块级别的一些参数。

到现在你应该知道，写模块级别参数会影响整个数据集。所以对数据库而言，写模块参数会影响所有已写入数据库的数据。

写模块模式 (Writer Mode)

最重要的写模块级别参数就是 Writer Mode。大部分 FME 的数据库格式的写模块 (不是所有的) 都有这个参数。

这个参数的值可以是下面三个中的一个：INSERT, UPDATE 或者 DELETE。UPDATE 和 DELETE 将在后面的单元中讲述。现在我们只讨论 INSERT 选项。

就像你所猜想一样，在 INSERT 模式下的写模块只是简单的将所有要素添加到表里。如果新添加的要素在数据库中已经存在，例如新要素与已库中已有的要素有相同的 ID 号，那么没有更新操作被执行。要么数据库里会存入这个重复的新纪录；要么如果 ID 字段被定义为唯一索引，会产生一个警告并放弃这个记录。

事务处理参数 (Transaction Parameters)

对于数据库写模块，另外一个常用的参数是 Transaction Parameters。事务处理 (Transaction) 就是分批地将数据写入到数据库中，这样一方面可以提高性能，另一方面可以避免因单个错误要素而导致的整个处理过程的异常终止。事务处理参数 (Transaction Parameters) 指定是否启用事务处理方式及每批写入多少个要素。

另外一个相关的参数是 “Transaction to Start Writing at”。假设由于第5个事务里的一个错误要素导致该事务处理失败，用户可以修复这个数据中的错误，重新运行这个事务并且指定在第五个事务开始写操作。已经知道最初的四个事务是正确的，就不需要再将它们写

入到数据库。这样的话，在转换时就不需要再花时间来重写数据。

Strict 参数

有些数据库写模块有所谓的 “Strict” 参数。

当在一个事件中从数据库中返回一个次要错误时，这些参数告诉 FME 读入器该如何处理这个事件。

例如，如果一个几何或属性太大了不能存入数据库表中分配的空间里，FME 可能会清空数据来适应。不过这种数据的补救对于用户不是总有用的。对于用户来说，Strict 参数是一种方式，这种方式可以指定是严格遵照规则写入数据（这种情形下，某个问题会导致事务处理的失败），还在在一些次要的情况下可以忽略这个规则（这种情形下，某个问题只是作为警告记录在 FME 的日志文件里）。

Oracle 和 ArcSDE 就是两种严格定义了处理参数的写入方式的格式。

Feature Type Level 参数



在 FME 的参数等级中，要素类级别参数比写模块级别参数低一级。

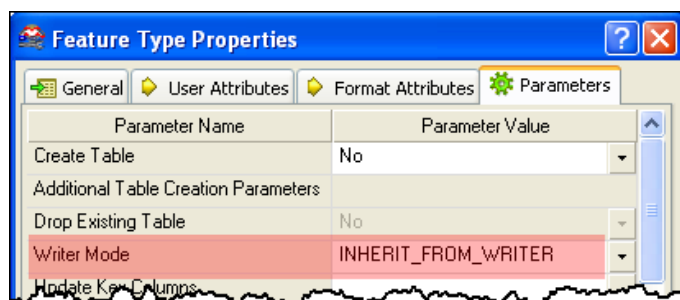
要素类级别参数是一些能够影响各自要素类的参数。就数据库而言，每个要素类代表一个表，要素类级别参数只影响写入指定表的要素。

写模块模式 (Writer Mode)

与数据库写模块一样，每个数据库要素类都有一个写模块模式参数。默认情况下，这个参数通常继承 Writer 参数的值，那就是说，这个写模块参数负责设置所有要素类写模块参

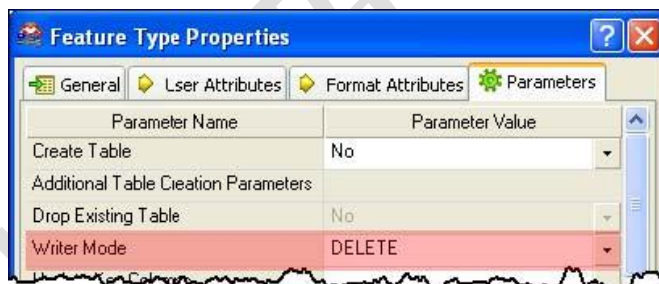
数。

不过在某些情况下，比如在一个表上执行 INSERTS 操作，而在另一张表上执行 UPDATES 或 DELETES 操作也是重要的。在这种情形下，设置的要素类写模块模式将取代写模块级别的设置。



左图：Feature Type默认情况下是从Writer-level参数中继承了它的写模块模式。

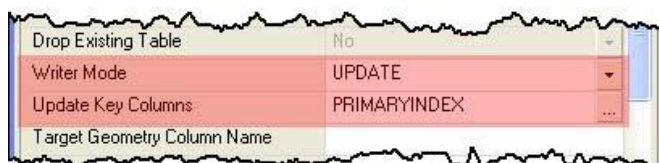
右图：用户将这个 Feature Type 的 Writer Mode 更改为 DELETE。输入这个 Feature Type 的要素类会导致数据库中一个记录的删除。



其它的 Feature Type 仍然是继承了 Writer-level 参数的写模块模式，或者是它们自己独立的设置。

更新键

当一张表的写模块模式是 UPDATE 时，识别哪些刚写入的要素将更新哪些已经存在的记录是必要的。



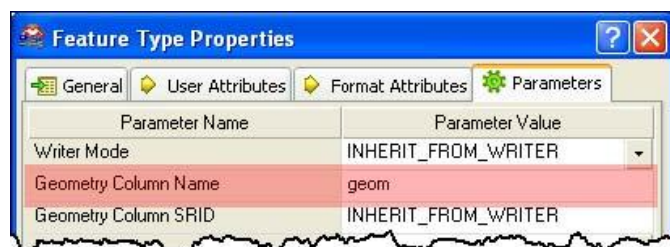
这个 Update Key Column 参数（左图），是用户定义连接要素和记录的属性的方法。这个连接甚至可以由多个属性来定义。

几何属性列

空间数据库通常将几何属性存储于一个特殊的列类型中。以 Oracle 为例，这个列是 SDO_GEOMETRY 类型列。

一般的，这个几何列都有一个默认的名字如 GEOM。不过有些情况下，用户可能希望使用其他名字，比如，可以把一个额外的几何列写入到在现有的表中。

在 FME 的数据库的要素类中，通常有一个称为 Geometry Column Name 的参数，这个参数可以用来更改或重命名几何列。



左图：在这个 PostGIS 的要素类属性中，几何列的名称默认的值“geom”。

删掉/清空表

当对一个已有的数据库表进行写入操作时，你或许想截取部分表，或者完全删掉这张表。

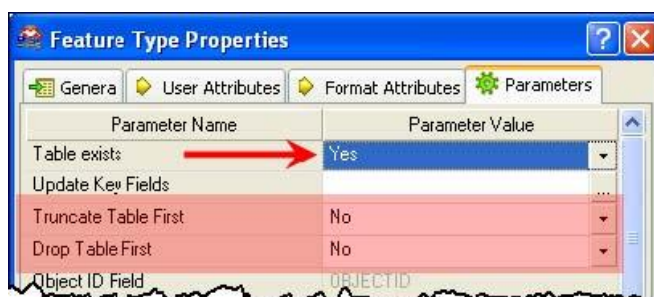
大部分数据库写模块在 Feature Type level 上有一个参数可以实现这个操作。

如果用到这两个选项，要能区分它们：

Truncate: 清空一个表是清空表的内容。当你想要替换所有表中的数据但要保留这个表的结构时是非常有用的。

Drop: 删掉一个表是将表从数据库中删除。当你希望替换表中所有数据及表的结构时，显得特别有用。

右图：在写入数据库前，对于 Geodatabase (SDE)的要素类属性

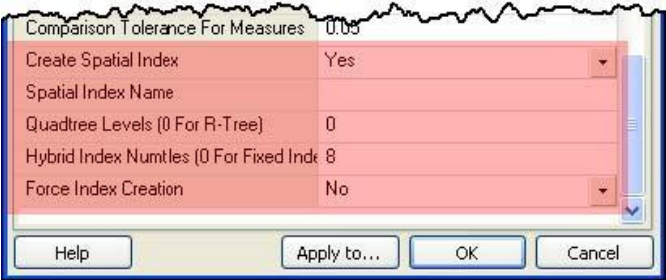


中，有 Truncate Table First 和 Drop Table First 两个选项。注意只有在 Table Exists 参数设为 Yes 时，这两个选项才可用，否则显示为灰色。

创建索引

用户是否要为表创建索引应该由每个表的具体情况而定，所以，这也是一个要素类级别的参数。

右图：并不是所有的数据库格式都有索引参数，但是Oracle要素类有很多索引参数。



例3— 重新导入到已有的数据表

在例1中，我们从SDF数据集中迁移了一套数据到Oracle空间数据库中，但是，检查数据时出现了一些问题，需要修复。

问题1：作为迁移的一部分，对于每个城市公园的记录，添加面积属性是很有用的。作为一个ETL工具，我们能够轻松地运用FME解决这个问题。

问题2：BusRoutes源数据有了更新，但是，我们SDF数据在更新前已经全部迁移到了数据库中。所以我们需要加载更新数据到现有的表中，覆写之前上传的所有数据。

Q+A

问题：上面两个问题都涉及到取代已有的数据表，但是我们是应该选择Drop还是Truncate呢？将你的答案在正确的表格中标出来。

<i>Problem 1</i>	<i>Truncate?</i>		<i>Drop?</i>	
<i>Problem 2</i>	<i>Truncate?</i>		<i>Drop?</i>	

在这个例子中我们只解决第一个问题。

1) 打开一个工作空间添加

转换器

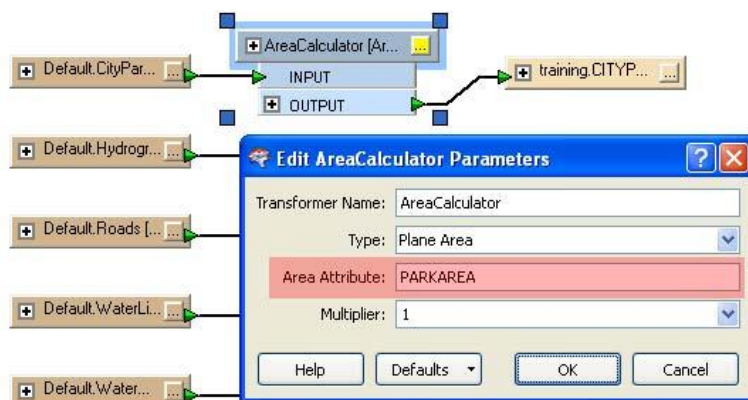
打开例 1 中的工作空间。

“cityparks” 源数据

和目标要素类之间放置一

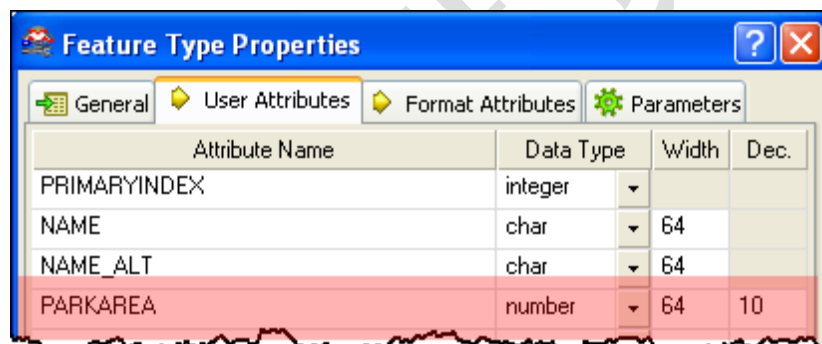
个 AreaCalculator 转换器来计算每个公园的面积。

写入一个新的属性：PARKAREA。



2) 调整目标结构

打开目标要素类CITYPARKS，在表结构中添加一个新的字段PARKAREA，这个列数据类型为"number"。



3) 调整要素类参数

检查和调整这个要素类的参数。

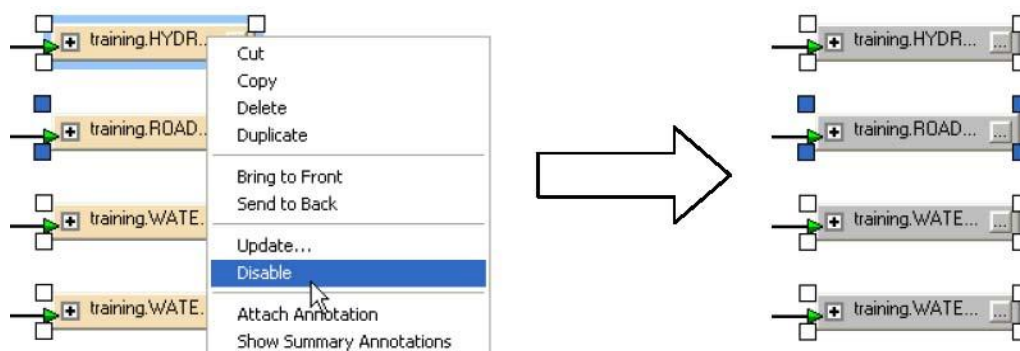
根据你前面Q+A部分的回答，决定从下面三个值中选择一个：

- Create Table • Drop Existing Table • Truncate Existing Table

4) 禁用多余的要素类

因为 我们只想更新CITYPARKS表，所以选中另外的所有表，单击右键，选择“Disable”。

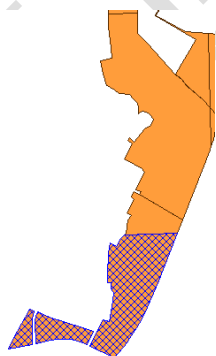
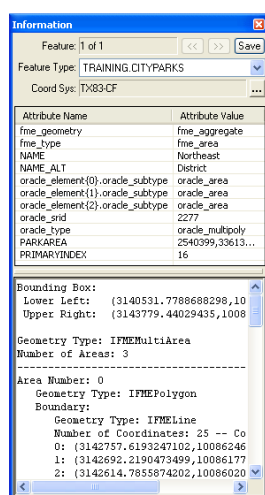
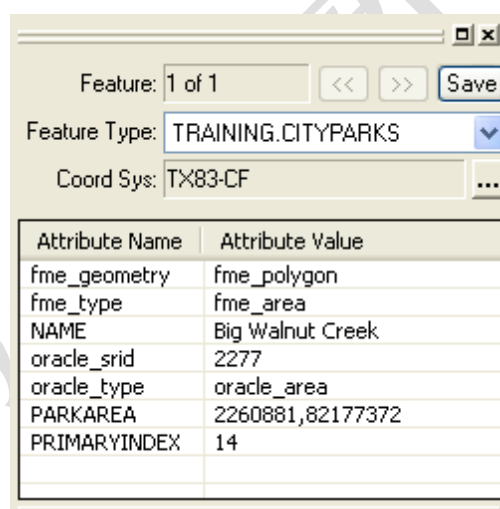
这样可以避免数据写入到这些表中。



5) 运行工作空间

运行这个工作空间，检查输出数据，确保它是正确的。

如右图所示：从 Oracle 数据库中读回数
据，会看到 PARKAREA 属性。



如左图所示：顺便说一句，如果你注意到这个聚集在一起的 park 类,想知道在 Oracle 中它属于什么类型，可以告诉你它属于 Multi-polygon 要素。

导入目标要素类定义



这个 *Import Feature Type Definitions* 工具对数据库（数

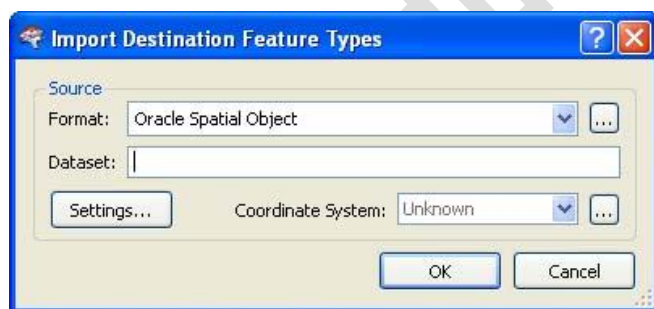
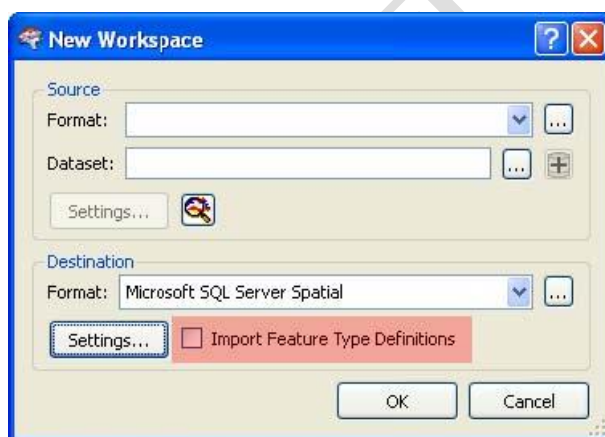
数据库中通常有预定义的表结构) 特别有用。

有时候, 你想创建一个工作空间并使用已有的数据库结构, 而不是FME根据源数据要素产生的数据库结构。这种情况下, 你就要用到Import Feature Type Definitions工具。

创建时导入要素类

你可能没注意到, 在 New Workspace 对话框有个内置的选项允许用户选择已有的数据库表结构作为目标要素类。

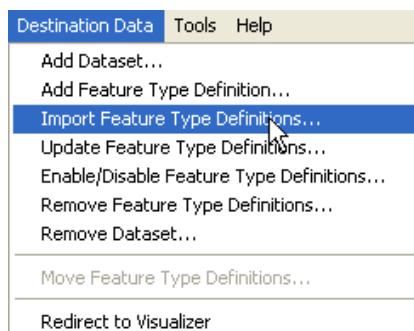
这个复选框通常是灰色不可用状态, 只有当目标写模块被设置为数据库格式时它才可用。



选中这个复选框, 在工作空间创建前会出现 Import Feature Types 对话框。在选定了格式和数据集后, 出现要素类的列表供选择。

记住你想要导入的要素类不一定必须与你想要写入的数据库相同, 或者根本就不来自于任何数据库。然而, 该功能的主要用途是添加一个已有数据库表的结构。

因为你可以手工的拒绝添加要素类, 而在编辑过程中使用导入要素类选项, 所以添加数据集工作空间没有相同的选项。



在编辑过程中使用导入要素类选项

即使工作空间已创建好，你仍可使用菜单栏中的Destination Data > Import Feature Type Definitions菜单，从已有的数据库表中添加表结构定义。

例4— 向现有的表中重新导入数据：第二部分

现在我们解决之前提出的第二个问题。

问题2：BusRoutes源数据有了更新，但是，我们SDF数据在更新前已经全部迁移到了数据库中。所以我们需要加载更新数据到现有的表中，覆写之前上传的所有数据。

1) 创建一个工作空间

使用New Workspace对话框创建一个新的工作空间，设置如下：

Source Format MapInfo MIF/MID

Source Dataset

C:\FMEData\Data\Transit\BusRouteUpdates\BusRouteUpdates.mif

Destination Format Oracle Spatial Object

Destination Settings

Service: XE

User ID: training

Password: training

Index Creation: Yes

Import Feature Type Definitions: Yes

2) 导入要素类的定义

当出现提示时，选择表TRAINING.BUSROUTES作为导入要素类定义的来源。现在工作空间看上去应该是像这样的：



3) 调整要素类参数

检查和调整这个要素类的参数。

根据你前面Q+A部分的回答，决定从下面三个值中选择一个：

- Create Table • Drop Existing Table • Truncate Existing Table

4) 运行工作空间

连接源数据和目标要素类，然后运行工作空间。

检查并确认结果是否正确。尤其要检查一下在BUSROUTES表中是否有重复的要素。

也要查看下Workbench日志窗口确认FME是否按照你希望的方式更新了数据库。

空间数据库中的时间和日期属性



将时间和日期属性导入、导出数据库一向都是比较棘手的。

时间和日期属性对FME来说是一个比较复杂的问题，因为每种不同的数据库格式对数据可能都有它自己独特的结构。

下面是从FME文档中摘录的一些内容，可能会对你有所帮助：

Microsoft SQL Server

DateTime 字段代表从 1753 年 1 月 1 日到 XXXX 年 12 月 31 日的日期和时间。

例如，值 20061231235959 表示 2006 年 12 月 31 日晚上 11 点 59 分 59 秒。当写入到数据库时，写模块要求数据的格式为 YYYYMMDDHHMMSS（年月日时分秒）。

SmallDateTime 代表从 1900 年 1 月 1 日到 2079 年 6 月 6 日的日期和时间。例

如, 20060101101000 值代表 2006 年 1 月 1 日上午 10 点 10 分 00 秒。当写入到数据库时, 写模块要求数据格式是 YYYYMMDDHHMMSS。

IBM Informix

Informix 的读模块对每个 DATE 字段返回两个属性值。

第一个属性是数据库列的名称, 格式为 YYYYMMDD。第二个属性带有.full 的后缀, 格式为 YYYYMMDDHHMMSS。例如, 如果日期字段叫做 UPDATE_DATE, 其属性值为 UPDATE_DATE 和 UPDATE_DATE.full。

当输出一个 DATE 或 DATETIME 列时, Informix 写模块查找上述两个属性值。或者可以指定两个属性中的任意一个。如果两个属性都被指定了, 则属性 <name>.full 优先。

IBM DB2

依照 IBM Informix reader/writer 支持的格式, 除此之外还支持 DATE, TIME 和 TIMESTAMP 类型。

ESRI ArcSDE

依照 IBM Informix reader/writer 支持的格式。

Oracle

Oracle 数据库要求日期数值采用 YYYYMMDDHHMMSS 格式, 即使在数据库表里的时间日期值显示为其它格式, 如: 01-JAN-08 12:00:00。

用转换器格式化数据属性

为了将日期写入到数据库的 DATE 或 DATETIME 字段, 你可以使用转换器 TimeStamper 或 DateFormatter 来获取日期正确的格式。格式字符串 ^Y^m^d^H^M^S 将返回

YYYYMMDDHHMMSS的格式；格式字符串^Y^m^d行返回YYYYMMDD的格式。

数据更新



更新数据库的重要的事情是什么？

为什么要更新？

信息被存入数据库后，它不可能保持不变。会发生更改。

有时，这些更新是直接在数据库里修改数据。有时是使用来自另外的应用程序的数据更新数据库。

有时更新涉及重新加载整套数据。大多数情况下只是某些要素需要更新；有时候源数据会指明更改哪些要素，而有时候又需要通过一系列变化检测来判定哪些需要更新。

示例方案

查看文件 `C:\FMEData\Data\Addresses\AddressUpdates\AddressUpdates.txt` 的内容。这个实例是，Interopolis 市打算对数据库中的地址记录进行年度更新。到目前为止，他们已经将数据库导出到 CSV 格式的文件中，该文件已经由研究人员进行了验证和更新。

在这一节，我们将探讨将这些更新应用到主要地址数据库里的不同方法。这些技术同样适用于任何数据库格式

数据结构

源数据还没有插入到数据库，不过可以浏览一下这个数据集，浏览设置如下：

Source Format Intergraph GeoMedia Access Warehouse

Dataset :

C:\FMEData\Data\Addresses\roadAllowancesAndAddressPoints.mdb

Table Name : ADDRESS_POINTS

属性PRIMARYINDEX是每个地址的唯一ID号。

CSV数据大体是相同的，但是它有一个附加的字段UPDATE_TYPE，来记录了我们编辑的类型，如下：

- I** Insert（插入），作为一个新地址在数据库中插入这条记录，
- D** Delete（删除），作为一个旧地址从数据库中删除这个记录
- U** Update（更新），作为一个更改了的地址更新记录
- N** Unchanged（没有更改），这个记录没有发生变化，不需执行更新操作

例5— 创建地址数据库表

创建一个地址数据库表，在这个表上我们将使用FME来加载GeoMedia Access Warehouse中的 ADDRESS_POINTS表到Oracle数据库里来执行更新操作。

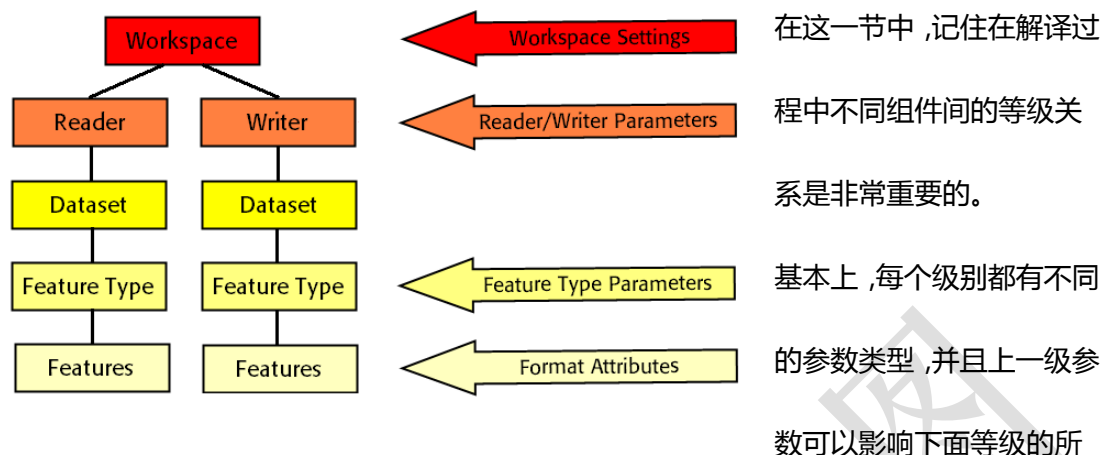
Q+A :

考虑下，当你试图删除或清空一个不存在的数据库表时，会发生什么？如果好奇，你可以试一下，找出答案。

数据库和 Table Level 的更新



如果你有一大堆更新需要写入，最简单的方法是要么重写，要么更新整个数据表。



有参数。写模块模式是一个在多种级别上都可用的参数。

重新加载所有内容

大家都知道,在把Writer Mode设为INSERT,把Drop Existing Table设为Yes的条件下,重新读入整个源数据集,您可以有效地做一个高层次的更新,该更新将替换表中的所有内容。

重新加载所有数据是一种简单的方法,但是很粗糙——它不允许你保留作为普遍要求的数据历史变更记录。

例 6

1) 创建一个工作空间

创建一个工作空间,用来解译CSV文件并将更新导入到Oracle,工作空间设置如下:

Source Format Comma Separated Value (CSV)

Dataset

C:\FMEData\Data\Addresses\AddressUpdates\AddressUpdates.txt

Source Settings Has Field Names = Yes

Lines to Skip = 1

Destination Format Oracle Spatial Object

使用New Workspace对话框里的 “Import Feature Type Definitions” 选项来添加ADDRESS_POINT表作为目标要素类。

完成后最初的工作空间看起来如下图：



2) 添加转换器 2DPointReplacer

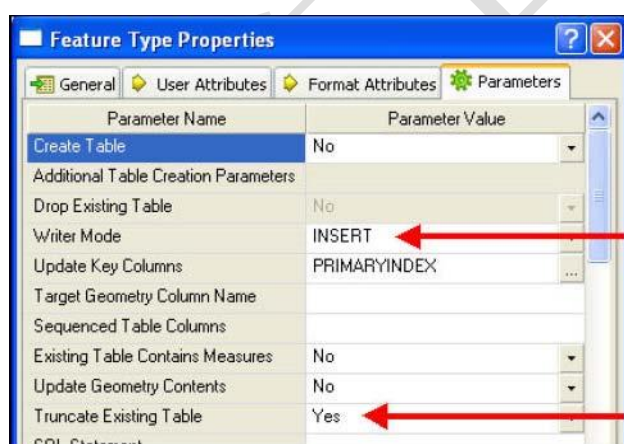
添加一个2DPointReplacer转换器将CSV格式的记录转变为真正的空间要素，X、Y的属性值设置为X_COORD和Y_COORD。

3) 清空表记录

打开目标的Feature Type Properties对话框，点击Parameters标签。

确认 *Create Table* 设置为No。更改 *Truncate Existing Table* 的设置为Yes。

这个设置在工作空间运行时，将会清空已存在的表。



4) 把 Writer Mode 设置为

Insert

在同一个对话框中，设置Writer

Mode为Insert。

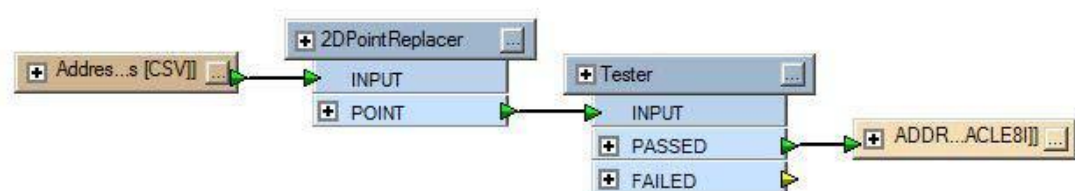
如左图所示：Feature Type

Properties的设置。

5) 添加转换器 Tester

显然，我们不想把已经标志为删除的记录插入到数据库表中，所以添加一个Tester转换器过滤掉属性值UPDATE_TYPE = D的要素。剩余的要素导入到输出要素类中。

如下图： 这时你的工作空间看起来如下图所示。因为我们保留了从Tester的“PASSED”端口输出的要素，所以必须把测试条件设置为UPDATE_TYPE!=D。



6) 运行工作空间

好了，现在来运行工作空间，点击run按钮，在FME Viewer中查看数据库的内容。

7) 选择多个源数据文件

作为一项高级任务，尝试处理多个CSV源数据。不要忘记在Source Feature Type对话框中选中Merge Filter复选框。

你可以尝试加载所有文件，在我电脑上只用了53秒就完成了。

小提示 经验告诉我，将要素数目减至200个，对于每个事务参数的读取是有益的。

一次运行1000个要素，就会让Oracle Express难以承受，并产生“end-of-file”的错误信息。

只在写入时更新 (Updates Only Writing)

注意在上面提到的例子中，我们如何把写模块模式设置成了INSERT。其它的模式

还有UPDATE, DELETE 和INHERIT_FROM_WRITER。

所以我们不用替换整个表，我们有更多的选择来更改表：例如在UPDATE模式下，我们只更新被标识为已更改的要素。

例 7

1) 打开工作空间

打开例6的工作空间继续下面的练习。

不要编辑源数据参数。我们想确认下这个例子运行的数据确实是加载到例6中的数据（记住-这张表已经被清空了）。

2) 设置更新键列

打开Feature Type Properties对话框，点击Parameters标签。

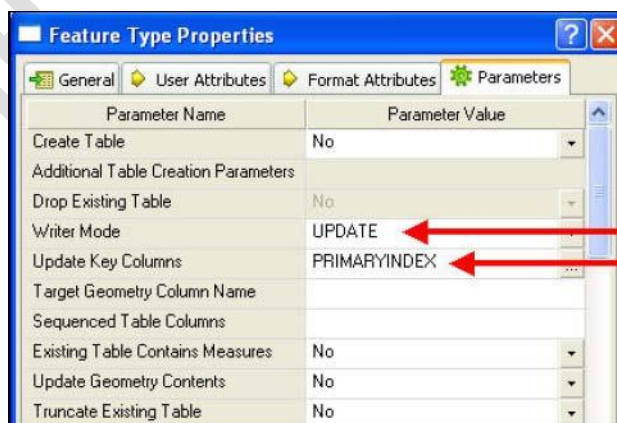
在Update Key Columns选项上，选择属性PRIMARYINDEX。这意味着数据表中的已有记录会被任何一个带有相同PRIMARYINDEX值的新写入的要素所取代。

3) 设置模式为更新 (Set mode to update)

在同一个对话框中，将Writer Mode设置更改为UPDATE。

如右图：Feature Type

properties设置如右图所示。如果



CREATE TABLE被设置为YES就把它改回NO！

同样确认Truncate Existing Table设置的是NO，Update Geometry Contents设置为YES也是重要的。

4) 编辑 Tester 参数

这次我们只需要保留被更新的要素，所以将Tester转换器的参数的检测条件设置为

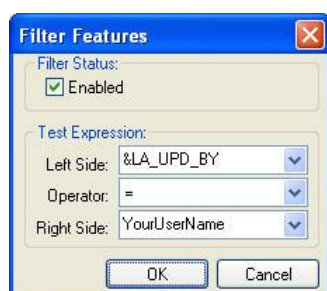
UPDATE_TYPE = U。

5) 添加标识符属性

判读是否发生了更改是很困难的，所以在Tester后面插入一个TimeStamper转换

器。用TimeStamper设置属性LA_UPD_DAT。添加一个AttributeCreator (或者

AttributeSetter)来设置属性LA_UPD_BY的值。



6) 运行工作空间

运行工作空间，查看一下数据是否进行了更新。

讨论要点

这种技术存在的问题是：更新、插入和删除等所有的操作

必须要在不同的进程中运行。因为每个Feature Type（数据库表）只能在一个工作空

间里表示一次，不能同时运行所有的操作。

Feature Level更新



表级别更新比较简单，但是不够灵活。为了有更多的控制，你

需要使用要素级别的更新。

使用格式化的属性值来标注要素

数据库和表级别的更新是比较粗糙和费时的，因为，你可能写入了比实际需要的更多的

数据，使用了比理想状态下更多的进程。

使用要素级别控制更新一个较低的级别，在这更新中提供了更多的精确度。每个要素必

须以某种方式标记出来，在FME中是用格式属性（*format attribute*）。

fme_db_operation

*fme_db_operation*是一个格式属性，用以识别数据库写模块如何处理正写入的要素。就像写模块和要素类参数一样，它的值可能是DELETE, INSERT或UPDATE。

fme_where

fme_where 是一个格式属性，它指定强制更新的匹配条件。例如，有个 Where 条件 *MyField* = @Value(*MyIndex*)是说当数据库中的字段 *MyField* 与属性 *MyIndex* 匹配时，则对该要素进行更新。

例 8

1) 打开一个工作空间

打开例 7 的工作空间，继续下面的练习。

2) 设置更新键列

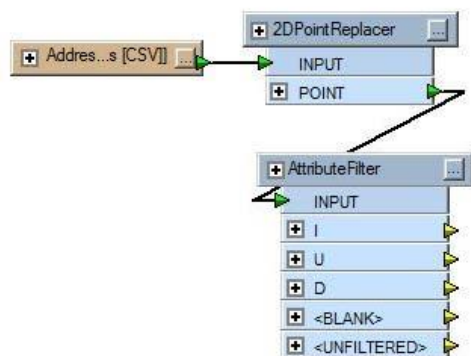
你可以使用 *fme_where* 格式属性确定要更新的记录。不过，使用 *fme_db_operation* 来设置操作也是可接受的，但是要使用参数 Feature Type Update Key Column 来决定这个查找属性。

跟例 7 中讲的一样，打开目标 Feature Type Properties 对话框，点击 Parameters 标签。确认 Update Key Columns 被设置为 PRIMARYINDEX。

3) 过滤更新值

这时我们想过滤所有要素类的更新，因此，

用一个链接连接三个 Tester 来分别检测更新类别 I,U,D。



或者可以使用一个 AttributeFilter (如右图)。

小提示 在这个例子中 ,我们一定要更改要素类上的写模块模式吗 ? 答案是"可能" !

当为 UPDATE 或 DELETE 模式时 , 格式属性将会覆写要素类的写模块模式。如果是 INSERT 模式 , 格式属性不会覆写这个写模块模式。因为不同的写模块有不同的行为 , 做个测试运行下是值得的。

1) 添加一个 AttributeCreator 转换器

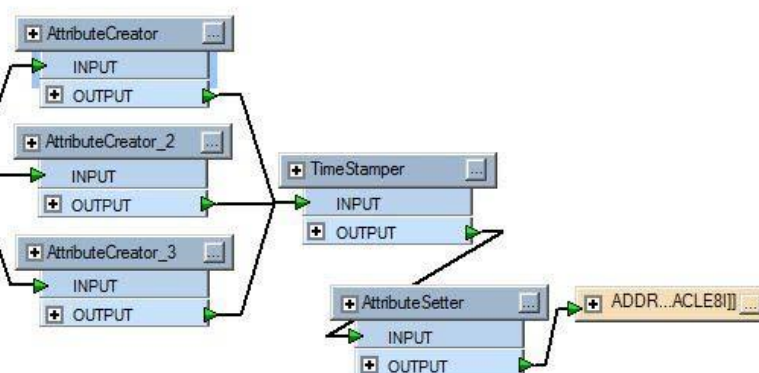
添加三个 AttributeCreator 转换器——每一个针对一类 PASSED 端口的要素。

使用 AttributeCreators 来创建一个名为 fme_db_operation 的属性。

每个转换器根据它要处理的要素集把这个属性的值设置为 UPDATE, DELETE 或 INSERT。

这三个 AttributeCreators 都连接到 TimeStamper 上。

如下图 : 工作空间的最后部分看起来像这样。

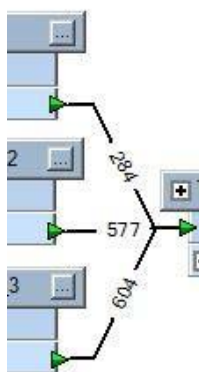


2) 运行工作空间

运行这个工作空间

连线上的要素计数显示了有多少要素被更新、插入或删除。

在 FME Viewer 中 , 你可以使用你的用户名属性过滤要素 , 并将其与在 CSV



文件中的变化列表进行匹配。

作为一个额外的任务,你可以试着根据最后更新日期(LA_UPD_DAT)来过滤要素,查看所有刚刚执行的更新。

问题:在目标要素类别中,能把 fme_db_operation 设置成常量吗?如果可以,这样对我们的工作有没有好处呢?

记事本:如果由于一个错误或类似原因,你需要重新运行工作空间,那么你可能要恢复原始表。这只需要重新运行例 6 就可以了。要确保对于每个工作空间都使用的相同的源数据。

3) 高级任务

使用 ValueMapper 转换器代替 Tester/AttributeFilter-AttributeCreator 的组合,每个要素还能与它们各自的 fme_db_operation 匹配吗?试一试,找出答案。

N (uNchanged)更新类型会有什么操作类型呢?

试一下用 fme_where 属性代替要素类的属性参数。

Transformer更新



除了用写模块更新数据库,有许多转换器也可以用来发送更新。

FME 中有许多设计好的用于数据库的 Workbench 转换器。这些转换器属于函数集的数据库子类。这类函数一般用于数据库的查询,但是也可用于数据库的更新和插入操作。

在这些转换器中，对于每个输入要素都会发送一个 SQL 命令或其它数据库命令。如果输出要素是由一个数据库的查询得来的，则输出要素可能是全新的。

这些转换器有时比使用写模块来执行更新操作效果更好，因为你或许希望只对一小部分数据进行更新，或者你希望使用一个指定的Where从句，这些在写模块UPDATE模式下都不能实现。

然而，除非你有特别的原因需要使用这些转换器，否则你还是应该使用写模块。

SQLExecutor

SQLExecutor 是向数据库中发送 SQL 命令的主要转换器。这个转换器常用 Select 命令，但也可以发送 Insert，Update 和 Delete 命令。

这个转换器对于格式是中立的，也就是说，你可以向这个转换器发送任何数据库格式的命令。不过，它不会在日志文件中返回太多的信息，所以你要非常确定发送的命令的正确性。

使用 FME 的函数@Value()访问属性，如下所示：

```
delete from ADDRESS_POINTS  
  
where PRIMARYINDEX = @Value(PRIMARYINDEX)
```

这种&符号表示的方法（例如：&PRIMARYINDEX）也可以运行，但是不推荐这种方法。

ArcSDEQuerier

ArcSDEQuerier 是一个发送查询命令到 ArcSDE 数据库的转换器。

最初，这个转换器是这种 ArcSDE 数据库更新的主要工具，所以带有一个模式设置 (QUERY, UPDATE 或 DELETE)。不过随着 SDE 写模块的改进，更新数据时使用写模块比使

用这个转换器更好些。

OracleQuerier

OracleQuerier 执行 Oracle 数据库表的空间查询。

这个转换器没有模式设定功能，也不打算包含这个功能。

这个转换器中有个设置来运行一个 SQL 语句（你可以在导航面板上找到它，而不是在转换器设置向导里），它也可以发送非查询命令，不过这对于最优的方法也不是必要的。

例 9

1) 重新设置数据库

运行例 6 的工作空间，重新把数据加载到数据库。

打开例 8 的工作空间，继续下面的练习。使用带 Tester 或 AttributeFilter 转换器的工作空间，而不是用了高级的 ValueMapper 转换器的工作空间。

2) 处理删除操作

定位和放置一个 SQLExecutor 转换器，与 DELETED 端口连接。

这个转换器的重要设置如下：

Type: Oracle 11g/10g/9/8i

Service: xe

User ID: training

Password: training

SQL Query: delete from ADDRESS_POINTS

where PRIMARYINDEX = @Value(PRIMARYINDEX)

3) 处理编辑 (Handle Edits)

放置另一个 SQLExecutor 转换器，与 UPDATED 端口连接。

像前面一样设置连接参数，不过这次的 SQL Query 语句不同，如下：

```
update ADDRESS_POINTS set ADDRESS=' @Value(ADDRESS)'
```

```
where PRIMARYINDEX = @Value(PRIMARYINDEX)
```

4) 处理插入 (Handle Insertions)

SQLExecutor 转换器可以处理插入操作，但是提供一个完整的 SQL 查询来插入一条新纪录对于我们的方案来说并不是一个简单的方法。使用写模块，我们能更简单的处理插入数据操作。

连接这些要素到 INSERT 端口，写入到输出要素类。

检查要素类设置。确认写模块模式设置为 INSERT，是否清空或丢弃现有表设置为 NO。

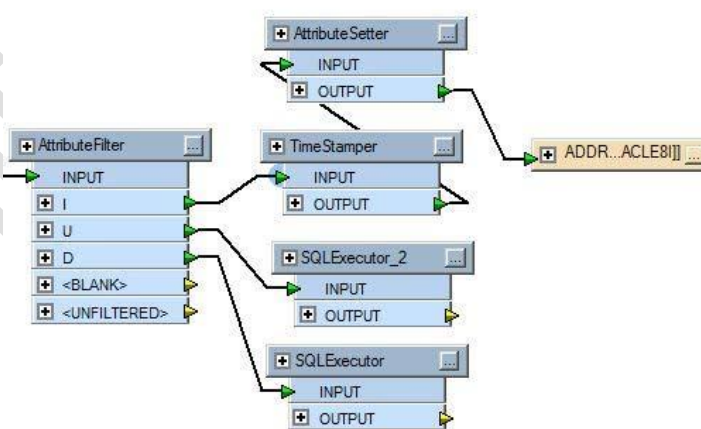
5) 运行工作空间

运行这个工作空间。

如右图：此时工作空间的后半部分应看起来是这个样子。

注意 SQLExcuter 转换器没有输出连接。一旦这些要素触发了已

经设置的 SQL Query 语句，我们就不再需要这些要素了。



变化检测



通过与现有数据库的对照，变化检测功能让我们决定需要更新

哪些要素。

尽管我们使用的源数据已经标识成了要求的操作类型,但并不是每位用户都清楚他们的源数据与初始数据相比是否已经发生了改变。在这种情况下,我们需要执行某种形式的变化侦测。

变化检测

为了检测现有数据和新数据之间的变化,运行变更检测显然需要读取两套数据。这使变化检测处理比起更新一套已定义的记录来多花费两倍的时间,因为在数据读取和变化检测过程中的都需要额外的操作。

对于变化检测的最大障碍就是数据的删除。通常的方法是假设新数据中没有的要素,一定是被删除的数据。不过这强迫用户在一个操作上处理整个的新数据集,因为如果不这样做,那些尚待处理的数据会被认为是已被删除,并会被从数据库中移除。

ChangeDetector Transformer

ChangeDetector 的功能正如它的名字暗示的一样:输入两套数据(原始数据和修改后的数据)并检测两者之间的变化。

ChangeDetector 转换器的主要缺点是不能检测特定要素的编辑,而是查找这两套大数据的添加和删除情况。换句话说,以我们的 address database 为例,它不能判别正在被编辑的特定记录,而会将其标示为一个新纪录(INSERTION),原有的记录将会被标示为 DELETION!

对于这个转换器,检测编辑的更多信息请登陆

<http://www.fmepedia.com/index.php/UpdateDetector>。

CRC Calculator Transformer

一个 CRC (Cyclic Redundancy Check, 循环冗余码校验) 是一个检验和,通常

是用来判定数据传输过程中是否被损坏。不过我们可以用这个功能检验要素是否在数据库和一套被编辑的数据之间做了更改。

基于一组属性值和要素几何属性，新要素和旧数据都被分配了一个 CRC 值。在要素(如 ID 号)间对照 CRC 值，看看是否不同。如果不同，我们则假设这个要素被编辑过了。

如果有些要素的 ID 号不匹配，要么假设他们是新添加到数据库的（如果他们来自于新数据），要么假设它们是从数据库里删除掉的（如果他们来自于老数据）。

CRCCalculator 的优势在于处理过程更加快捷，耗费更少的系统资源（因为它使用更少的基于组的操作），并且 CRC 值能够存储在数据库中，不再需要计算未发生变化的要素值。

例 10

该例将使用 CRC 值进行变化检测。它将忽略在源 CSV 数据中的 UPDATE_TYPE 字段，假设这类信息并不存在。

1) 重新设置数据库

首先，我们需要重新加载数据库，添加最初的 CRC 值。打开例 5 的工作空间。添加 CRCCalculator 转换器，计算一个新的属性 CRCVALUE。你应根据几何和属性 ADDRESS + STREET_NAM + STREET_TYP 来计算这个 CRCVALUE 值。

在目标要素类型上添加一个新的属性（Char 16），用来匹配这个新的属性。运行工作空间。

2) 打开工作空间

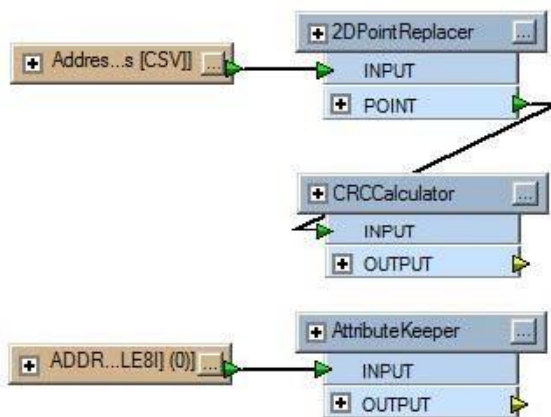
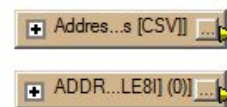
使用例 9 中的工作空间，继续下面的练习。

删除所有的 SQLExecutor 和 AttributeFilter 转换器。使用 Destination Data >

Update Feature Type 菜单选项，在工作空间中的目标要素类中添加这个新的属性 CRCVALUE。

3) 添加一个新的读模块 (Add new Reader)

通过添加一个新的 Oracle 读模块 (如右图所示)，添加



ADDRESS_POINTS 表作为一个源数据。我们需要重新读取源数据以便能将它与随后的更新做比较。

4) 添加 CRCCalculator

添加一个 CRCCalculator 转换器，为将要引入的 CSV 要素计算一个新属性

NEWCRCVALUE。使用和前面相同的参数计算这个属性值。

5) 添加 AttributeKeeper

给将要添加的 Oracle 要素增加一个 AttributeKeeper 转换器。我们只想保留属性 PRIMARYINDEX (唯一的 ID) 和 CRCVALUE (预定义的 CRC)。其它的属性对我们来说都不重要。

6) 添加 FeatureMerger

添加 FeatureMerger 转换器，这个转换器是整个操作的关键。它将使用 PRIMARYINDEX 作为关键字来匹配原始数据和修改后的数据。输出结果可能有以下几种情形：

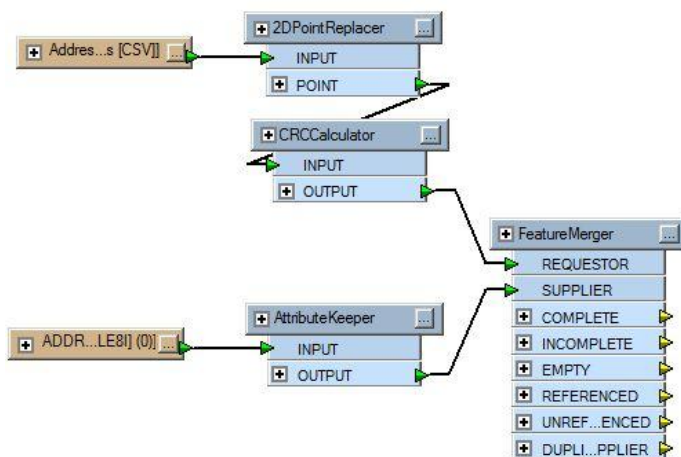
- 如果关键字匹配上了，则说明要素确实存在过并且仍然存储在数据库中，现在需要进行 CRC 值的测试，看看是否对要素进行了编辑：

- 如果 CRC 值匹配，则这个要素未发生变化。

○ 如果 CRC 值不匹配，则要素被编辑过了。

- 如果修改过的要素没有匹配，那么它一定是新插入的要素。
- 如果原有要素没有匹配值，那么它一定是被删除并且标识为 Deletion 了。

这时，你的**工作空间**应当看起来像这样（如下图）



7) *Incomplete* 端口的处理 (*Incomplete Processing*)

没有匹配的修改过的要素会从 INCOMPLETE 端口输出。因为他们一定是新要素，用一个 AttributeCreator 来设置 fme_db_operation 的值为 INSERT。

8) *UNREFERENCED* 端口的处理

没有匹配值的原有的要素会从 UNREFERENCED 端口输出。因为他们一定是被删除了的要素，用一个 AttributeCreator 来设置 fme_db_operation 的值为 DELETED。

9) *Complete* 端口的处理

能匹配的被修改过的要素一定是一个更新或者没有更改。

为判定到底是哪种情况，我们添加一个 Tester 转换器。

添加一个 Tester 来测试 CRCVALUE=NEWCRCVALUE 是否成立。

如果要素通过测试（也就是说它们的值是匹配的），该要素一定没发生变化。它可

以被忽略。如果没能通过测试（也就是它们的值不同），该要素一定是被编辑过的。对于这类要素，使用 AttributeCreator 转换器来设置 fme_db_operation 的值为 UPDATE。

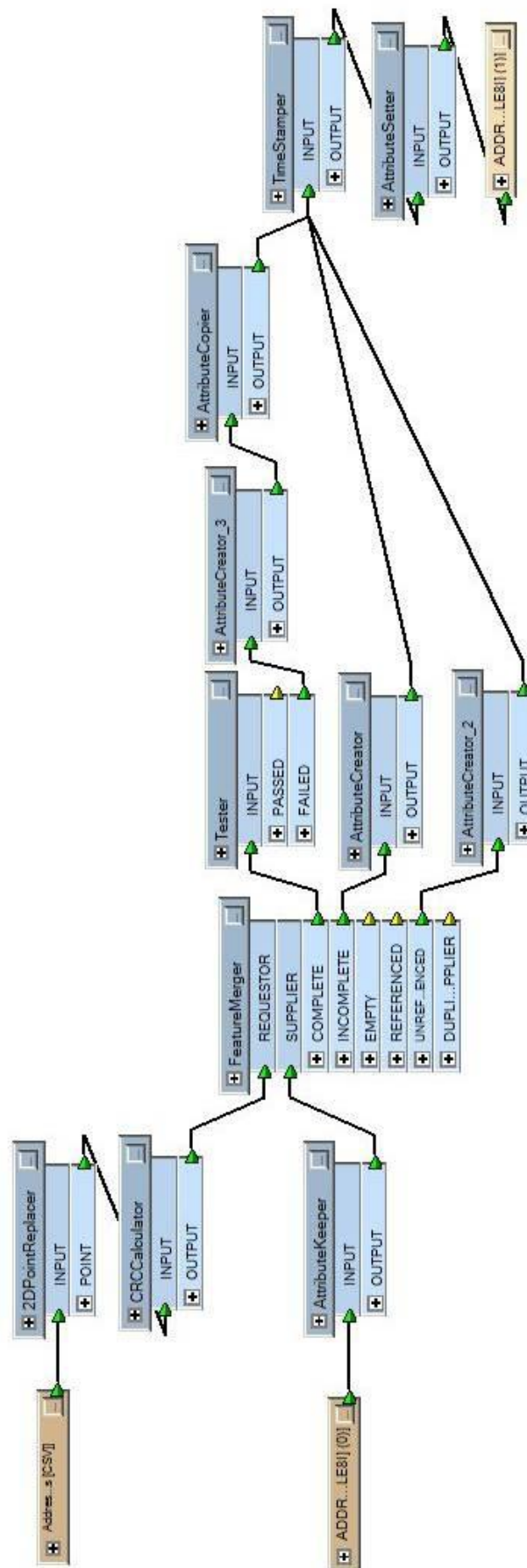
10) 更新 CRC 的值

对于更新的要素，我们需要将新的 CRC 值复制到正确的属性里，以便能添加到数据库中。放置一个 AttributeCopier 转换器，复制 NEWCRCVALUE 值到 CRCVALUE。

11) 运行工作空间

运行这个工作空间。

例 10 的工作空间如下所示



数据分发



许多用户会在空间数据库上直接操作数据；其他不能直接访问数据库的用户，需要首先将数据导出来。

为了能够分发数据，一个普遍的要求就是能够从数据库中导出数据。导出工作可以是简单地把数据以用户选择的格式传送给那些不能直接访问数据库的用户（比如承包商），也可以是创建一个数据门户入口，使用户能够下载数据或以流传输的方式获取数据。



数据分发的好处

通过创建一个预定义的、灵活的且可以向终端用户提供不同选项的数据分发处理，你能获得如下好处：

- 时间和资源：当用户能够自助服务时，处理数据传送请求就只需要较少的资源。
- 授权给用户：用户有选择权来选择下载他们需要的数据、所需的格式和下载的时间。

小提示：你可以在Safe Software的网站下上查看一个数据分发的典型案例，网址为：

http://www.safe.com/technology/documents/casestudies/CaseStudy_IHS.pdf

pdf

把数据导出为另一种格式，通常使用FME Desktop。

数据下载门户和数据流传输的方式则需要安装FME Server。

使用FME Desktop进行数据分发



使用FME Desktop进行数据分发可以像只包含一个Reader和一个Writer的工作空间那么简单, 或者也可以包括其他数据集、函数或脚本。

使用FME Desktop进行数据分发通常意味着, 要么你运行工作空间来导出数据, 要么你作为FME的管理员创建工作空间让其他人来使用。

简单的导出

对于简单的导出, 你只需要参照数据检查 (Data Inspection) 这一部分创建一个数据库读模块。你所要做的只是创建工作空间, 从数据库表中读出数据并写入到指定的格式文件。

例11

这个例子是一系列检验数据分发技术的第一步。这个例子假设将一些数据库表转换为KML格式的数据库。

1) 创建工作空间

创建工作空间, 读取一个 Oracle数据库, 输出到另一种格式的数据库, 设置如下:

Source Format Oracle Spatial Object

Source Settings

Service: XE

User ID: training

Password: training

Table List AIRPORT, BUSROUTES, ROADS

Destination Format OpenGIS KML Encoding Standard

2) 运行工作空间

运行这个工作空间，将数据写入到文件

C:\FMEData\Output\TrainingAdvanced\export.kml。



3) 检查数据

使用Google Earth检查数据。

这些数据看起来应该像这样（如左图）。

针对其他用户的工作空间

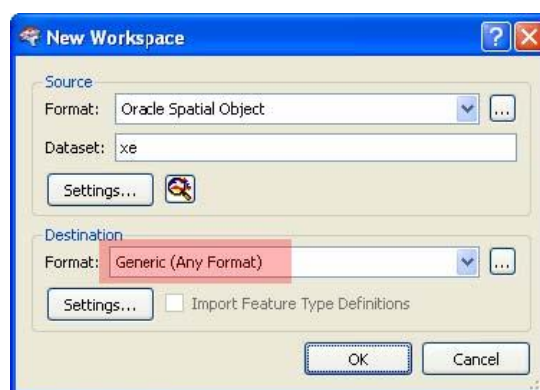
当你为其他用户创建了一个工作空间后，可能用户并不是专家，你想添加不同的选项使他们不用编辑工作空间也能调整数据的输出。

Generic Writer

Generic Writer对于自定义输出格式是一个非常好的工具。

这个写模块的格式直到运行时才会定义。发布了这个参数就能设置你希望的格式，这样用户就能按照提示选择他们想要的输出格式。

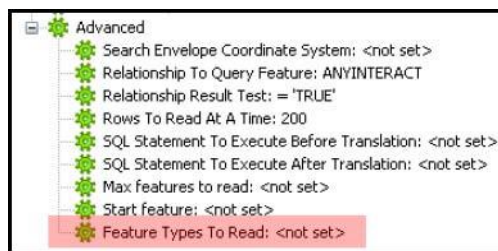
如右图所示：创建的带有Generic格式写模块的工作空间。



Feature Types to Read

Feature Types to Read是在大多数读入器上都能找到的参数。它允许用户在源数

据中选择读取哪些要素类（在这个例子中读取的是数据表）。



如左图所示：从Navigator窗口中的Reader的Advanced参数部分，你就能找到这个Feature Types to Read参数。

例12

在这个例子中我们继续在例11的工作空间上操作。

1) 设置Feature Types to Read

在Workbench中的Navigator窗口上找到Feature Types to Read参数。在这个参数上点击右键，选择Publish Parameter选项。在弹出的对话框中，把Prompt的值从“Feature Types to Read”改为“Select Feature Classes to Read”，让非FME用户能清楚的知道到底要求他们做什么。

2) 移除Published Parameter

清空提示框，删除与Published Parameter相关的Oracle服务设置。

3) 运行工作空间

用Prompt and Run选项运行工作空间，或者保存这个工作空间，在FME Universal Translator中运行。你会被提示读取哪个数据表。尝试不同的组合，查看输出的数据有什么不同。

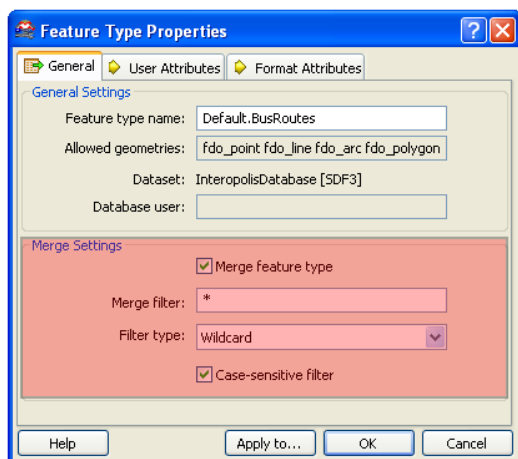
Dynamic Feature Types to Read

你也许已经注意到，出现在Feature Types to Read列表中的表（如右图所示）都是来自于工作空间本身的。

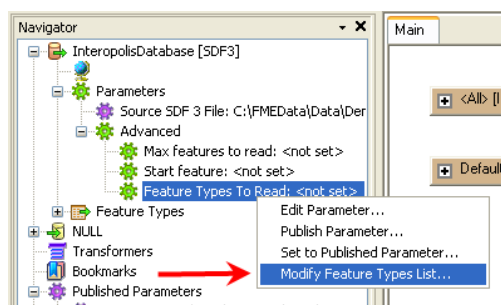


在数据库中还有很多表，比如CITYPARKS表，但是我们没有被提示去选择他们。

FME2010弥补了这个缺陷。它允许你使用Feature Types to Read的一个新的Dynamic形式来选择表（即使这些表没在工作空间定义）。

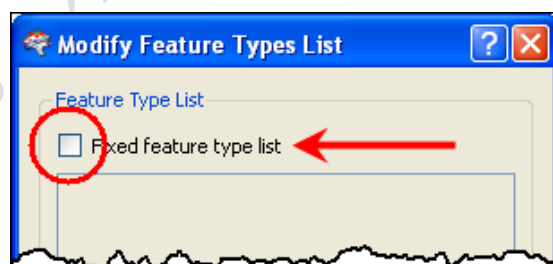


如左图所示：为了允许未定义的元素类能进入工作空间，需要选中Merge Feature Type选项。为了从数据库的SDF版本中读取数据，我们在表BusRoutes（目前唯一在工作空间中定义的数据表）上设置一个全包含过滤器。

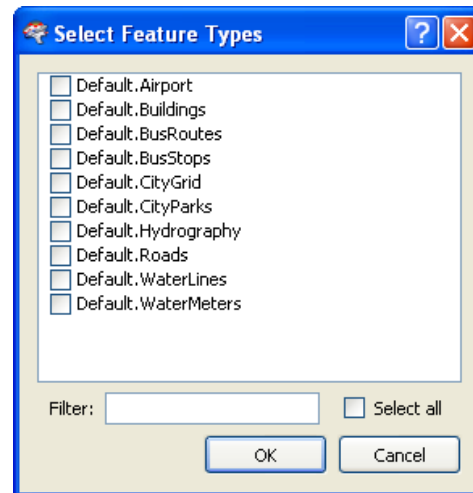


如右图所示：可以看到 Feature Types to Read 参数中有个新的选项：Modify Feature Types List.

如下图所示：在修改对话框中，确定没有选中“Fixed features type list”。



如下图所示：现在Feature Types to Read选项提示用户可选择数据库中的所有表。



如果你已安装了 FME2010，为什么不在 Oracle 练习数据库中尝试下使用 Dynamic Feature Types to Read 呢？

数据转换的改善 (Transformation Possibilities)

和其它FME工作空间一样，一个读取数据库的工作空间能够从其他源数据中合并数据，进行数据转换。

FME的最佳实践告诉我们尽早地在工作空间中删除多余的属性是非常有用的。

AttributeKeeper和AttributeRemover就是实现这一操作的非常有用的工具。通过减少属性数量到最少，你就能改进你操作的工作空间的Workbench的性能。（如果你想知道原因的话，这是因为没必要追踪这么多的属性映射）。

重新符号化 (Resymbolisation)

变换数据的一个主要原因就是为了赋予要素一套合适的符号。空间数据库往往只储存数据，而忽略符号，除非这些符号被一个特别的应用需要。例如，ArcSDE会储存要素颜色，而Oracle或PostGIS数据库就不会。

例 13

本例中我们将继续操作我们的数据分发工作空间。

1) 打开工作空间

打开例12的工作空间。

2) 添加 *KMLStyler* transformer

在工作空间中，为每个要素流添加一个KMLStyler转换器。使用这些转换器用不同颜色（例如，红色代表公路，白色代表机场）和不同的线宽来表示不同的要素。

3) 运行这个工作空间

运行工作空间，在Google Earth中检查要素新的外观。

使用FME Server进行数据分发



使用FME Server进行数据分发包括向服务器发布预先定义好的工作空间，这个工作空间可以被任何人使用，即使他们没有安装或并不了解FME。

使用FME Server进行数据分发，通常意味着你创建一个工作空间，可以让其他人使用。

使用这个服务器架构的工作空间输出的数据甚至能以流传输的方式到另一个应用程序。



Workspace Considerations工作空间的思考

当向FMEServer发布工作空间时，有几点你必须要考虑：

源数据的位置

首先，服务器本身要能够访问源数据。事实上，使用工作空间访问源数据库是非常简单的，仅仅需要建立客户端和正确的连接。

目标数据的位置

第二，为了能在服务器上运行工作空间，它必须要能访问输出的位置。它不能是如 C:\FMEData folder 这种我们使用的本地目录，因为服务器不能直接访问它。它必须是一个 UNC (Universal Naming Convention，通用命名标准) 路径，如 [\\mycomputer\myoutput](#)。

流传输数据或者使用数据下载服务时，能够访问输出位置并不是必要的。在这种情况下，FME Server 会提供一个临时的数据存放路径。

发布一个工作空间

发布工作空间到 FME Server 很简单，只要在 Workbench 中打开工作空间，使用菜单选项 File > Publish to Server 就可以了。

你需要来发布工作空间的服务器名称和服务器接收连接的端口名（通常是 7071）。一旦进行了连接，你就可以选择将工作空间放到数据储存器，也就是一个工作空间、脚本和数据的目录下。

最后，你需要指定将工作空间注册到哪些 Web 服务下，例如，数据下载服务能提供下载工作空间输出结果的 ZIP 压缩文件，KML Network Link 提供一个到能定期更新的 KML 数据的链接。

例 14

现在让我们把这个数据分发工作空间发布到 FME Server 上。

1) 打开和发布一个工作空间

打开例13中的工作空间，选择File > Publish to Server菜单选项，发布工作空间到FME Server。在Safe Software公司的培训室里，连接详细信息。

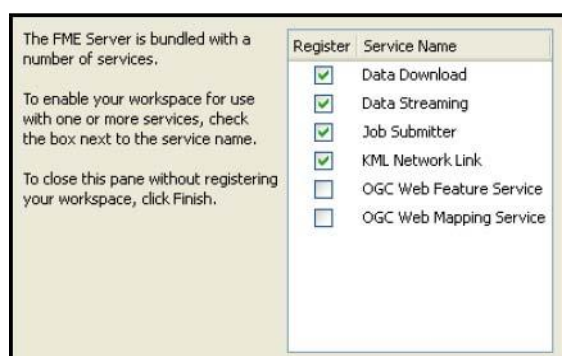
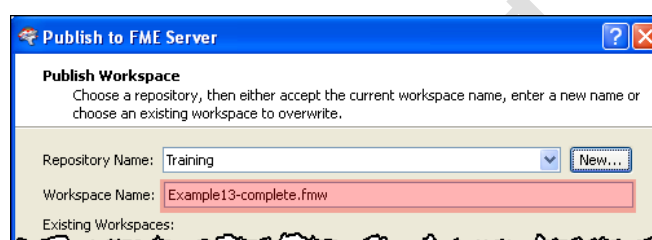
FME Server的设置为：

Hostname: trm17

Port Number: 7071

发布工作空间到一个名为Training的存储器。一定要对工作空间重命名，避免覆盖其他人的工作。

如右图所示：在把它发布到服务器的时候，你可以重命名这个工作空间



在下面这些服务上注册工作空间：

- 下Job submitter
- oData Download
- aData Streaming
- aKML Network Link

3) 打开服务器界面

在网页浏览器中打开这个服务界面，访问的地址为：<http://trm17/fmeserver>

在Data Downloads服务下找到发布的工作空间。

运行它，以证实能产生正确结果。

记事本：在操作之前，需要确认辅导人员已将数据加载到数据库里了。不然系统虽然会报告运行成功，却没有数据输出。

4) 运行一个 *Job Submitter* 服务

JobSubmitter服务仅仅运行工作空间，并将数据输出到定义的位置。

它作为一个高级的操作，只在该服务下运行这个工作空间。你需要哪些安装来确保

数据会被写回到你自己的电脑呢？

阶段回顾



这个模块是为了帮助你使用FME来处理空间数据。

从这个模块中你应该学会些什么

下面是需要从本模块学习到的几个要点：

理论

- FME支持一系列的数据库格式，包括空间数据和非空间数据
- 像FME其他部分一样，空间数据库使用层次结构(R/W parameters, Feature Type parameters, Format Attributes)来控制转换。
- 让数据库自己执行的处理，比如剪切源数据，比起读入整个表在FME中处理高效的多的。

FME 技巧

- 使用FME写入数据到空间数据库
- 使用FME检查空间数据库中的数据
- 进行Table-Level和Feature-Level更新
- 使用与数据库相关的转换器执行空间数据库的处理
- 使用Import Feature Type Definitions菜单选项写入现有的数据库的定义
- 在空间数据库中进行变化检测 (Change Detection)

- 使用FME Desktop或FME Server来分发空间数据库数据

Q + A

问题：这两个问题都涉及到取代已有的数据表，但是我们是应该选择 Drop 还是 Truncate 呢？将你的答案在正确的表格中标出来。

<i>Problem 1</i>	<i>Truncate?</i>		<i>Drop?</i>	√
<i>Problem 2</i>	<i>Truncate?</i>	√	<i>Drop?</i>	

答案：在问题 1 中，你需要删除数据表，因为数据表要更新表结构来储存新的属性。

在问题 2 中，我们只需要取代数据（不是表结构），因此只需要清空。

Q + A

问题：考虑下，当你试图删除或清空一个不存在的数据库表时，会发生什么？如果好奇，你可以试一下，找出答案。

答案：如果你尝试删掉一个不存在的表，你会收到一个警告信息：

- **Failed to drop table 'ADDRESS_POINTS'. Table may not exist**
- **（删除表'ADDRESS_POINTS'失败。表可能不存在）**
- 如果你尝试清空一个不存在的表，你会收到一个警告信息：
- **Failed to truncate table 'ADDRESS_POINTS'. Table may not exist**
- **（清空表'ADDRESS_POINTS'失败。表可能不存在）**

.....但是因为只有一种方法可以访问 Oracle 清空命令，就是设置 Create Table = NO，

所以当 FME 试图写入表时，会返回错误信息如下：

Oracle table 'ADDRESS_POINTS' does not exist and no table creation parameters were specified. Either use a table that exists or specify the create table parameter as 'Yes'

(Oracle数据库表'ADDRESS_POINTS'不存在，不能指定表的创建参数。要么使用一个存在的表，要么指定Create Table的参数值为 'yes')