IBM® DB2® Universal Database

XML Extender 管理與程式設計

版本 7

IBM® DB2® Universal Database

XML Extender 管理與程式設計

版本 7

使用本資訊及其支援的產品之前，請先閱讀第245頁的『注意事項』的一般資訊。

目錄

表

關於本書

本節說明下列資訊：

- 『適合本書的讀者』
- 『如何使用本書』
- 第x頁的『強調的慣例』
- 第xi頁的『如何閱讀語法圖』
- 第xiii頁的『相關資訊』

適合本書的讀者

本書是專門爲下列人員提供的：

- 在 DB2 應用程式中使用 XML 資料及熟悉 XML 概念的人。 本文件的讀者應對 XML 及 DB2 有一般的認識。 欲進一步瞭解 XML 及相關主題，請參照下列網站：

  http://www.w3.org/XML

  欲進一步瞭解 DB2，請參閱下列網站：

  http://www.ibm.com/software/data/db2/library
- 熟悉 DB2 管理概念、工具及技術的 DB2 資料庫管理員。
- 熟悉 SQL 及一或多種用於 DB2 應用程式的程式設計語言之 DB2 應用程式設計師。

如何取得本書的現行版本

您可以到 XML Extender 網站取得本書的最新版本：

http://www.ibm.com/software/data/db2/extenders/xmlext/library.html

如何使用本書

本書的結構如下所示：

第 1 章：簡介

本章提供 XML Extender 的概觀，並說明如何使用於商業應用方面。 它包含協助您準備及執行的入門實務範例。

第 2 章：管理

本章說明如何準備及維護 XML 資料的 DB2 資料庫。如果您需要管理含有 XML 資料的 DB2 資料庫，請熟讀本章。

第 3 章：程式設計

本章說明如何管理您的 XML 資料。如果您需要在 DB2 應用程式中存取及操作 XML 資料，請熟讀本章。

第 4 章：參照

本章說明如何使用 XML Extender 管理指令、使用者定義的類型、使用者定義函數，以及儲存程序。 本章亦列示 XML Extender 發出的訊息與訊息碼。 如果您熟悉 XML Extender 概念及作業，但需要有關使用者定義的類型 (UDT)、使用者定義的函數 (UDF)、指令、訊息、中間資料表、控制表或程式碼的資訊，請熟讀本章。

第 5 章：附錄

本附錄說明文件存取定義的 DTD、範例的樣本與入門實務範例，以及其它 IBM XML 產品。

將本書併入 DB2 UDB 版本 7 資訊中心

您可以使用下列步驟將 XML Extender 文件併入「DB2 資訊中心」中：

- 在 XML 產品中，從您所使用之語言下的 DOC次目錄內複製本書的 HTML 檔案，將它們安裝到 DB2 UDB 文件目錄中您所使用之語言的次目錄下：

  就 Windows 而言，「資訊中心」目錄爲 `sqllib\doc\html\db2sx`

  就 UNIX 而言，「資訊中心」目錄爲 *install directory*`/doc/html/db2sx`
- 重新啓動「資訊中心」，則本書將內含於**書籍**欄標中。

強調的慣例

本書使用下列慣例：

粗體字	粗體字表示： • 指令 • 欄位名稱 • 功能表名稱 • 按鈕
斜體字	斜體字表示： • 用某值取代的變數參數 • 強調的字

	• 第一次使用名詞解釋
UPPERCASE	大寫字母表示： • 資料類型 • 直欄名稱 • 表格名稱
範例	範例文字表示： • 系統訊息 • 鍵入的值 • 撰寫程式碼範例 • 目錄名稱 • 檔名 • 路徑名稱

如何閱讀語法圖

本書的所有指令語法及 SQL 陳述式，皆使用語法圖來描述。

閱讀語法圖的方式如下：

- 由左而右、由上而下，順著路線讀取語法圖。

  ▶▶── 符號表示陳述式的開始。

  ──▶ 符號表示陳述式語法將續接下一行。

  ▶── 符號表示陳述式接自上一行。

  ──▶◀ 符號表示陳述式終止。

  非完整陳述式的語法單元圖解，是以 ▶── 符號開頭，而以 ──▶ 符號結尾。
- 必要的項目會出現在水平線 (主要路徑) 上。

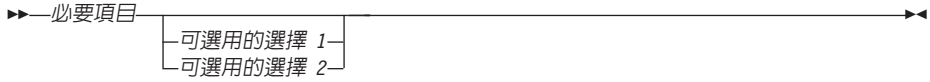
- 可選用的項目會出現在主要路徑的下方。

  如果有某個可選用的項目出現在主要路徑上方，則該項目對陳述式的執行並無任何影響，且僅用於閱讀上。

- 如果您可以從兩個或兩個以上的項目中選擇，它們會以垂直堆疊的方式呈現。

  如果您*必須*選擇其中一個項目，則該堆疊的一個項目會出現在主要路徑上。

  如果可以選擇其中一個項目，則整個堆疊會出現在主要路徑下方。



  如果其中一個項目是預設值，則它會出現在主路徑的上方，而其他的選項會顯示在下方。

- 若在主線上有一個指回左邊的箭頭，表示該項目可以重複。

  如果重複箭頭包含標點符號，則您必需使用所指定的標點符號來區隔重複的項目。

  在堆疊上方的重複箭頭表示您可以重複堆疊中的項目。

- 關鍵字會以大寫來表示 (例如，FROM)。在 XML Extender 中，並不限定關鍵字是大寫或小寫。但是非關鍵字的詞彙將會以小寫字母方式呈現 (例如，*column-name*)。它們是代表使用者提供的名稱或值。
- 如果有顯示標點符號、括弧、數學運算子或其他這樣的符號，則您必須將它們當作語法的一部份來輸入。

相關資訊

下列文件可能對您使用 XML Extender 及相關產品上有用：

文件	序號	說明
Call Level Interface Guide and Reference	SC09-2950	本書說明如何使用 CLI 來存取 DB2 伺服器，以撰寫應用程式。
DB2 Application Development Guide	SC09-2949	本書說明應用程式開發處理程序，以及如何編碼、編譯及執行使用內含的 SQL 及 API 來存取資料庫的應用程式。
DB2 Extender page	N/A	本頁包含 DB2 Extender 的相關資訊，以及一些與擴充元有關的技術。 DB2 Extender 頁面的網址爲：http://www.software.ibm.com/data/db2/extenders
Universal Database Parts 1 及 2 的 DB2 SQL 參考手冊	• 第 1 章：SC09-2974 • 第 2 章：SC09-2975	這些書說明 SQL 語法、語意及語言規則。也包括有關版次對版次不相容、產品限制及型錄檢視的資訊。
• *DB2 Universal Database Administration Guide: Implementation* • *DB2 Universal Database Administration Guide: Performance* • *DB2 Universal Database Administration Guide: Planning*	• 施行：SC09-2944 • 效能：SC09-2945 • 規劃：SC09-2946	這些書說明如何設計、施行及維護 DB2 資料庫。
DB2 Universal Database Image, Audio 和 Video Extenders 管理與程式設計	SC26-9929	本書說明如何管理影像、音效及視訊資料的 DB2 資料庫。本書也說明如何使用擴充元提供的應用程式設計介面，來存取及操作這些資料類型。

文件	序號	說明
DB2 Universal Database Text Extender Administration and Programming	SC26-9930	本書說明如何管理文字資料的 DB2 資料庫。本書也說明如何使用擴充元提供的應用程式設計介面，來存取及操作這些資料類型。

第1篇 簡介

本章提供 XML Extender 的概觀，並說明如何使用於商業應用方面。

第1章 XML Extender 簡介

IBM® DB2® Extenders™ 系列提供資料和 meta 資料管理解決方案來處理傳統和非傳統的資料。 The XML Extender helps you integrate the power of IBM's DB2 Universal Database (DB2 UDB)™ with the flexibility of XML.

DB2 的 XML Extender 有能力儲存和存取 XML 文件，或從現存的關聯式資料中產生 XML 文件，以及將 XML 文件分解成 (分解、儲存未標示的元素或屬性內容) 關聯式資料。 XML Extender 提供新的資料類型、函數及儲存程序，讓您在 DB2 中管理 XML 資料。

XML Extender 應用於下列作業系統：

- Windows NT
- AIX
- Sun Solaris
- LINUX

XML 文件

電腦產業中有許多應用程式，各有其優缺點。現在，使用者有機會選擇最適合工作需求的應用程式。不過，因爲使用者傾向於在不同應用程式之間共用資料，所以面臨的問題，仍然是如何將資料抄寫、轉換、匯出或儲存成不同格式，然後匯入另一個應用程式。這在商業應用程式來說是一個重要的問題，因爲這些轉換程序容易遺失一些資料，或至少需要使用者執行繁瑣的程序來確保資料的一致性。徒然浪費時間與金錢。

今日，解決此問題的方法之一，就是讓應用程式開發者撰寫 *Open Database Connectivity (ODBC)* 應用程式，將資料儲存到 DBMS。如此一來，資料就能夠以另一個應用程式所需要的格式來操作及呈現。不過，由於應用程式容易過時，所以必須撰寫資料庫應用程式，將資料轉換成應用程式所需的格式。將資料轉換成 HTML 的應用程式提供簡報解決方案，但實際呈現的資料無法做爲其他用途。如果有其他方法可以將資料和簡報分隔開來，該方法就可以作爲應用程式之間的實際交換方式。

XML 爲解決這個問題而問世。XML 是 *eXtensible Markup Language* 的字首語。該語言本身是一種 meta 語言，可讓您根據企業的需求，建立自己的語言，所以具延伸性。 XML 不僅可用來擷取特定應用程式的資料，還可擷取資料結構。 XML

不是唯一的交換格式。 不過，XML 已逐漸成爲資料交換的標準。 只要遵循這個標準，應用程式就可共用資料，不必使用專利格式來轉換資料。

XML 應用程式

因爲 XML 現在已成爲資料交換的標準，許多可利用 XML 優點的應用程式正在開發中。

假設您使用一個特殊的專案管理應用程式，且想要將部份的資料與行事曆應用程式一起共用。透過 XML，就可輕易達成目標。在今日這個環環相扣的世界裡，應用程式供應商必須在應用程式中提供 XML 交換功能，才能與同業競爭。所以，在本範例中，您的專案管理應用程式可將作業匯出成爲 XML，然後原封不動地匯入您的行事曆應用程式中 (如果資訊符合接受的 DTD)。

為何是 XML 和 DB2？

雖然 XML 提供資料交換的標準格式，解決許多問題，但仍有其它問題尚待克服。建置企業資料應用程式時，您需要回答如下的問題：

- 我要多久抄寫一次資料？
- 應用程式之間需要共用何種資訊？
- 我如何才能快速搜尋所需的資訊？
- 我如何使用一個特定動作，例如新增的新登錄，在全部應用程式之間觸發自動資料交換程序？

這些問題只能經由 DBMS 來解決。將 XML 資訊及 meta 資訊直接納入資料庫中，您可更直接 (更迅速) 取得其它應用程式在特定用途上所需的 XML 結果。這是 XML Extender 能夠輔助您的地方。有了 XML Extender，您可在許多 XML 應用程式中，充份利用 DB2 的功能。

在 DB2 資料庫中使用結構化 XML 文件的內容，您可結合結構化 XML 資訊與傳統的關聯式資料。根據應用需求，您可選擇在 DB2 中以非傳統的使用者定義資料類型來儲存整個 XML 文件，或在關聯式表格中將 XML 內容對映成傳統的資料。對於非傳統的 XML 資料類型，XML Extender 除了具備 DB2 UDB Text Extender 所提供的結構式文字搜尋之外，另外新增強大的功能來搜尋 XML 元素或屬性值的繁複資料類型。

有了 XML Extender，您的應用程式可以：

- 將整個 XML 文件儲存成應用程式表格中的*直欄資料*，或在外部儲存成本端檔案，同時取出所要的 XML 元素或屬性值，然後放入*輔助表格*內來搜尋。使用 XML 直欄方法，您可以：

- 對已取出並放到輔助表格或已編排索引的 SQL *一般資料類型*的 XML 元素或屬性，執行快速搜尋
- 更新 *XML 元素*的內容或 *XML 屬性*的值
- 使用 SQL 查詢來動態取出 XML 元素或屬性
- 在插入或更新期間驗證 XML 文件
- 使用 Text Extender 來執行結構式文字搜尋

• 透過 XML 集合儲存和存取方法，使用一或多個關聯式表格來撰寫或分解 XML 文件的內容

將 XML 整合到 DB2

XML Extender 提供下列特性，協助您使用 DB2 來管理和運用 XML 資料：

• 管理工具，協助您在關聯式表格中整合管理 XML 資料
• 儲存和使用方法，處理您的 XML 資料。
• 一個 DTD 儲存庫，讓您儲存驗證 XML 資料所用的 DTD：DTD_REF
• 一個對映方法，稱爲「文件存取定義」檔案 (Document Access Definition, DAD)，讓您將 XML 文件對映到關聯式資料

管理工具

XML Extender 管理工具協助您啓用 XML 的資料庫和表格直欄，將 XML 資料對映到 DB2 關聯式結構。XML Extender 提供數個管理工具，您可以自行開發管理作業的應用程式或僅使用精靈來執行管理作業。您可使用下列工具來完成 XML Extender 的管理作業：

• XML Extender 管理精靈提供圖形式使用者介面來進行管理作業。
• **dxxadm** 指令提供指令行選項來進行管理作業。
• XML Extender 管理儲存程序提供應用程式開發選項來進行管理作業。

儲存和存取方法

XML Extender 提供兩個儲存和存取方法，將 XML 文件整合到 DB2：XML 直欄和 XML 集合。這些方法的用途極爲不同，但可使用於相同的應用程式中。

XML 直欄

這個方法協助您在 DB2 中儲存完整 XML 文件。XML 直欄非常適合用來保存文件。文件插入已啓用 XML 的直欄後，可以更新、擷取及搜尋。元素和屬性資料可對映到 DB2 表格 (輔助表格)，再經過索引編排，達成更快的結構式搜尋。

XML 集合

這個方法協助您將 XML 文件結構對映到 DB2 表格，所以您可使用現存的 DB2 資料來製作 XML 文件，或將 XML 文件分解 (儲存未標示的元素或屬性內容) 成 DB2 資料。這個方法非常適用於資料交換應用程式，尤其當 XML 文件的內容經常更新時，更爲適合。

DTD 儲存庫

XML Extender 提供一個 XML *文件類型定義 (DTD) 儲存庫*，這是 XML 元素和屬性的一組宣告。當一個資料庫 *啓用* XML 的支援時，就會建立一個 *DTD 參照表 (DTD_REF)*。 本表格的每一橫列代表一個具有其它 meta 資料資訊的 DTD。使用者可存取此表格來插入自己的 DTD。 DTD_REF 表格中的 *DTD* 是用來驗證 XML 文件。

文件存取定義 (DAD)

您在*文件存取定義 (DAD)* 中指定如何處理結構式 XML 文件。 DAD 本身就是 XML 格式的文件。 使用 XML 直欄或 XML 集合時， DAD 可結合 XML 文件結構和 DB2 資料庫。 DAD 的結構在定義 XML 直欄和 XML 集合時有所不同。

DAD 檔案是透過 XML_USAGE 表格來管理，此表格是在啓用資料庫的 XML 支援時建立的。

XML 直欄：結構式文件儲存和擷取

因爲 XML 包含建立一組文件所需的全部資訊，所以您可能有時必須儲存及維護目前的文件結構。

例如，若您是一個在 Web 上提供文章的新聞出版公司，您可能要保存出版過的文章。在此實務範例中，XML Extender 可讓您將完整或部份的 XML 文章儲存於 DB2 表格的某直欄中。這種 XML 文件儲存類型稱爲 *XML 直欄*，如第7頁的圖1所示。

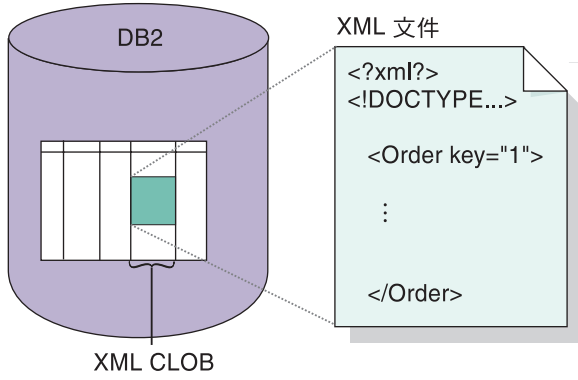


圖 1. 在 DB2 表格直欄中儲存結構化 XML 文件

XML Extender 提供下列使用者定義類型 (UDT) 來使用於 XML 直欄：

- XMLVarchar
- XMLCLOB
- XMLFILE

所有 XML Extender 的 UDT 皆以 db2xml 爲字首，這是 DB2 XML Extender UDT 的*綱目名稱*。 這些資料類型在應用程式表格中用來識別 XML 文件的儲存類型。 XML Extender 支援舊型純文字檔；您不必在 DB2 中儲存 XML 文件。您亦可將 XML 文件儲存成*本端檔案系統*的檔案，由本端檔案名稱指定。

DB2 XML Extender 提供功能強大的*使用者定義函數 (UDF)*，在 XML 直欄中儲存及擷取 XML 文件，以及取出 XML 元素或屬性值。 UDF 是針對資料庫管理系統 (DBMS)所定義的函數，可在 SQL 查詢中參考到。 XML Extender 提供下列 UDF 類型：

- 儲存：將完整 XML 文件以 XML 資料類型儲存在啓用 XML 的直欄中
- 取出：以基本資料類型取出 XML 文件，或指定值的元素和屬性
- 更新：更新整個 XML 文件或指定的元素和屬性值

取出函數可讓您在一般 SQL 資料類型上執行強大的搜尋功能。此外，您可使用 DB2 UDB Text Extender 和 XML Extender，對 XML文件中的文字執行結構式和*全文搜尋*。 這種強大的搜尋能力可用來增進出版大量閱讀文字的網站可用性，如報紙文章或*電子資料交換 (EDI)* 應用程式 (經常含有可搜尋的元素或屬性)。

所有 XML Extender 的 UDF 皆以 db2xml 爲字首，這是 DB2 XML Extender UDF 的綱目名稱。 UDF 適用於 XML UDT，主要使用於 XML 直欄中。

位置路徑

*位置路徑*是一連串 *XML 標示*，用來識別 XML 元素或屬性。XML Extender 使用位置路徑從識別 XML 文件的結構，指出元素或屬性的環境定義。 A single slash (/) path indicates that the context is the whole document. 位置路徑使用於下列狀況：

- 取出 UDF 時，識別所要取出的元素和屬性
- To specify the mapping file between an XML element or attribute and a DB2 column when defining the indexing scheme in the DAD for XML columns
- 使用 Text Extender 來執行結構式文字搜尋時，識別 XML 元素或屬性

圖2顯示位置路徑的範例，以及與 XML 文件結構的關係。

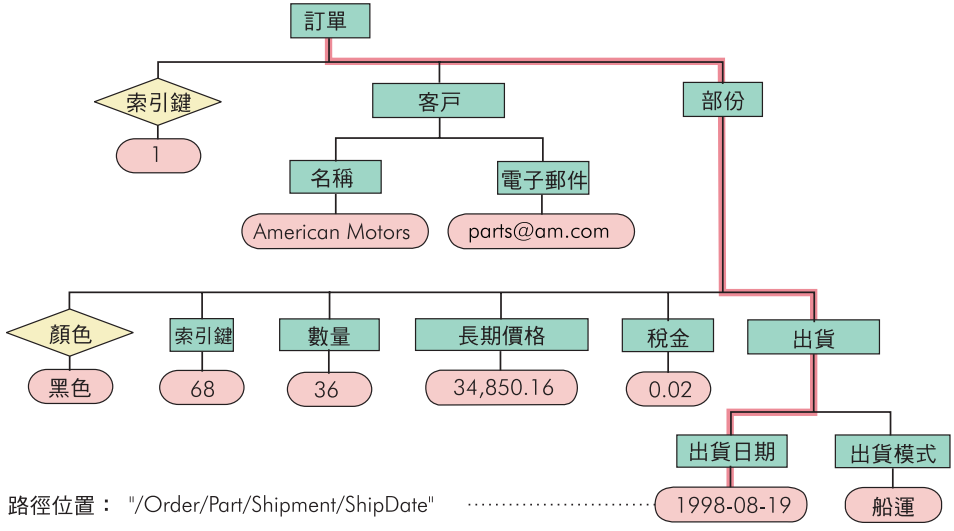


圖 2. 將文件儲存爲 DB2 表格直欄中的結構化 XML 文件

爲了指定位置路徑，XML Extender 使用一組 *XML 樣式表語言轉換 (XSLT)* 和 *XML 路徑語言 (XPath)*。本書使用 XPath 規格所定義的*位置路徑*。位置路徑是一連串 XML 標示，用來識別 XML 元素或屬性。本書亦使用*絕對位置路徑*的 XSLT 或 XPath 縮寫語法，這是在 XPath 規格中指定的。絕對位置路徑是物件的完整路徑名稱。

XSLT 是一種語言，可將 XML 文件轉換成其它 XML 文件。主要是用來做爲 *XML Stylesheet Language (XSL)* (XML 的樣式表語言) 的一部份。除了 XSLT 之外，XSL 尚包括 XML 指令字典來指定格式。 XSL 使用 XSLT 指定 XML 文件的樣式，說明如何將文件轉換成另一個使用格式化指令字典的 XML文件。

XPath 是一種語言，用來尋找 XML 文件的部份，主要是設計供 XSLT 使用。Every location path can be expressed using the syntax defined for XPath.

關於 XSLT 及 XPath 的詳細資訊，請參閱下列網頁：

- 關於 XSLT，請參閱：http://www.w3.org/TR/WD-xslt
- 關於 XPath，請參閱：http://www.w3.org/TR/xpath

相關語法和限制，請參閱 第48頁的『位置路徑』。

XML 直欄術語

本節說明本書採用的 XML 概念及術語。

文件存取定義 (DAD)
以 XML 直欄而言，指 XML 文件結構到 DB2 輔助表格的對映，爲了進行結構式查詢，輔助表格經過索引編排。

DXX_INSTALL
XML Extender 安裝目錄。

輔助表格
XML Extender 建立的附加表格，主要是在 XML 直欄中搜尋元素或屬性時改進效能。

XML 直欄
儲存和存取 XML 文件的一種方法，係啓用一個 DB2 直欄來處理 XML 資料類型，然後將完整 XML 文件儲存在已啓用的直欄中。亦指透過管理工具來啓用 XML 功能的一個直欄。

XML 表格
一種應用程式表格，其中含有一或多個直欄，各直欄已透過管理工具來啓用 XML 的功能。

XML UDF
XML Extender 提供的 DB2 使用者定義的函數。

XML UDT
XML Extender 提供的 DB2 使用者定義類型。

XML 集合：整合資料管理

傳統的 SQL 資料是從送進的 XML 文件中*分解*出來，或用來*製作*送出的 XML 文件。 如果其他應用程式要共用您的資料，您要能夠製作和分解進入及送出的 XML 文件以及管理資料，充份利用 DB2 的關聯式功能。這種 XML 文件儲存類型稱爲 *XML 集合*。

圖3 顯示 XML 集合的範例。

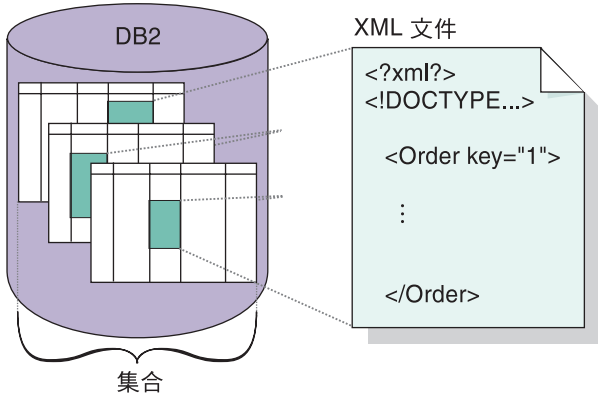


圖 3. 將文件儲存成 DB2 表格中的未標示資料

XML 集合定義於 DAD 檔案中，該檔案指定元素和屬性如何對映到一或多個關聯式表格。若要定義集合名稱，您可啓用集合名稱，然後搭配儲存程序來製作或分解 XML 文件。

當您在 DAD 檔案中定義集合時，您可使用兩種對映方法， SQL 對映或 RDB_node 對映。SQL 對映使用 SQL SELECT 陳述式，定義集合的 DB2 表格和條件。 RDB_node 對映使用 XPath 爲主的 RDB_node，定義表格、直欄及條件。

儲存程序可用來製作或分解 XML 文件。儲存程序使用字首 db2xml，此爲 XML Extender 的*綱目名稱*。對 XML 集合使用下列儲存程序：

- 組合：
  - dxxGenXML()：針對 XML 使用 DAD 檔案來製作 XML 文件
  - dxxRetrieveXML()：使用已啓用的 XML 集合來製作 XML 文件
- 分解：
  - dxxShredXML()：針對 XML 集合使用 DAD 檔案來分解 XML 文件
  - dxxInsertXML()：使用已啓用的 XML 集合來分解 XML 文件

XML 集合術語

下列術語是 XML Extender 專用的，在本書中亦經常使用。

組合 使用 DAD 檔案所定義的現存關聯式資料來建立 XML 文件。

分解 將 XML 文件儲存成 DAD 檔案所定義的未標示關聯式資料。

文件存取定義 (DAD)

以 XML 集合而言，指 XML 文件結構到 DB2 資料結構的對映，主要用於製作或分解 XML 文件。

DXX_INSTALL

XML Extender 安裝目錄。

XML 集合

使用一組關聯式表格來儲存及存取 XML 資料的一種方法。 未標示的資料可製作成 XML 文件，或從 XML 文件中分解出來。亦參照用來製作或分解 XML 文件的表格集。

XML 儲存程序

製作或分解 XML 文件所用的儲存程序。

第2章 XML Extender 啓動

本章說明如何開始使用 XML Extender 存取及修改應用程式的 XML 資料。透過提供的教學課程，您可使用提供的範例資料設定資料庫，將 SQL 資料對映到 XML 文件，在資料庫中儲存 XML 文件，以及從 XML 文件搜尋及取出資料。

在管理課程中，在 DB2 指令視窗內使用管理指令。您可使用 XML Extender 管理精靈來完成這些作業，本書也有說明該精靈。 在 XML 資料管理課程中，您會使用 XML Extender 提供的 UDF 及儲存程序。 本書其它章節的大部份範例是根據本章使用的範例資料。

要求：若要完成本章的課程，您必須使用 DB2 UDB 版本 6.1 或更新版本。如果您使用的是 6.1 版本，則必須安裝 Fixpack 2。此外，該課程的步驟假設您使用的是 Windows NT®。

課程如下：

- 在 DB2 表格直欄中儲存完整的 XML 文件
  - 計劃頻繁地搜尋 XML UDT，以儲存文件及 XML 元素和屬性。
  - 設定資料庫及表格
  - 啓用 XML 的資料庫
  - 將 DTD 插入 DTD 儲存庫表
  - 準備 XML 直欄的 DAD
  - 將直欄新增至 XML 類型的現存表格
  - 啓用 XML 的新直欄
  - 在輔助表格中建立索引
  - 在 XML 直欄中儲存 XML 文件
  - 使用 XML Extender UDF 搜尋 XML 直欄
- 從現存的資料中建立 XML 文件
  - 規劃 XML 文件的資料結構
  - 設定資料庫及表格
  - 啓用 XML 的資料庫
  - 準備 XML 集合的文件存取定義 (DAD) 檔
  - 使用現存資料製作 XML 文件
  - 從資料庫中擷取 XML 文件

- 清除資料庫

課程實務範例

在課程中，您是 ACME Auto Direct 的員工，該公司的業務是將客貨車分送給汽車代理商。您會被指派 兩項作業。第一，您將設定系統以保存訂購記錄於 SALES_DB 資料庫中，供銷售部門查詢。第二項作業爲在現存的採購訂購資料庫、SALES_DB 中取出資訊，並從中取出將儲存在 XML 文件內的關鍵資訊。

課程：在 XML 直欄中儲存 XML 文件

實務範例

您被指派一個保存服務部門銷售資料的作業。該資料儲存在使用相同 DTD 的 XML 文件中。以後服務部門在處理客戶要求及抱怨時就會使用這些 XML 文件。

已提供服務部門一個 XML 文件的建議結構，並指定他們認爲會經常被查詢的元素資料。他們希望 XML 文件儲存在 SALES_DB 資料庫中的 SALES_TAB 表格內，而且能夠快速找到文件。 SALES_TAB 表格將包含兩個具有每一個銷售的直欄，及包含 XML 文件的第三個直欄。該直欄被稱爲 ORDER。

您要決定用來儲存 XML 文件的 XML 資料類型，以及要經常查詢的 XML 元素和屬性。接下來您會設定 XML 的 SALES_DB 資料庫，建立 SALES_TAB 表格，以及啓用 ORDER 直欄，使您可以在 DB2 儲存完整文件。您也會爲 XML 文件插入一個 DTD 以進行驗證，然後將文件儲存成 XMLVARCHAR 資料類型。啓用此直欄之後，您會定義輔助表格以便在文件存取定義 (DAD) 檔索引文件的結構搜尋，該檔案是一個指定輔助表格結構的 XML 文件。若要查看 DAD 檔、DTD 和 XML 文件範例，請參閱 第231頁的『附錄B. 範例』。

SALES_TAB 說明於 表1。

表 1. SALES_TAB 表格

直欄名稱	資料類型
INVOICE_NUM	CHAR(6) NOT NULL PRIMARY KEY
SALES_PERSON	VARCHAR(20)
ORDER	XMLVARCHAR

規劃

開始使用 XML Extender 儲存文件之前，您需要瞭解 XML 文件結構才能夠決定如何搜尋文件。規劃如何搜尋文件時，您需要決定：

- 要儲存 XML 文件的 XML 使用者定義類型
- 服務部門會經常搜尋的 XML 元素和屬性，以便將它們編排索引來增進效能。

下列各節會說明如何做這些決定。

XML 文件結構

本課程的 XML 文件結構採用特定訂單的資訊，該訂單的結構是以訂單索引作爲頂層，然後以客戶、零件和出貨資訊作爲下一層。 XML 文件說明於 第16頁的圖4。

本課程也提供範例 DTD，協助您瞭解和驗證 XML 文件結構。您可在 第231頁的『附錄B. 範例』 找到 DTD 檔。 它符合 第16頁的圖4 中的結構。

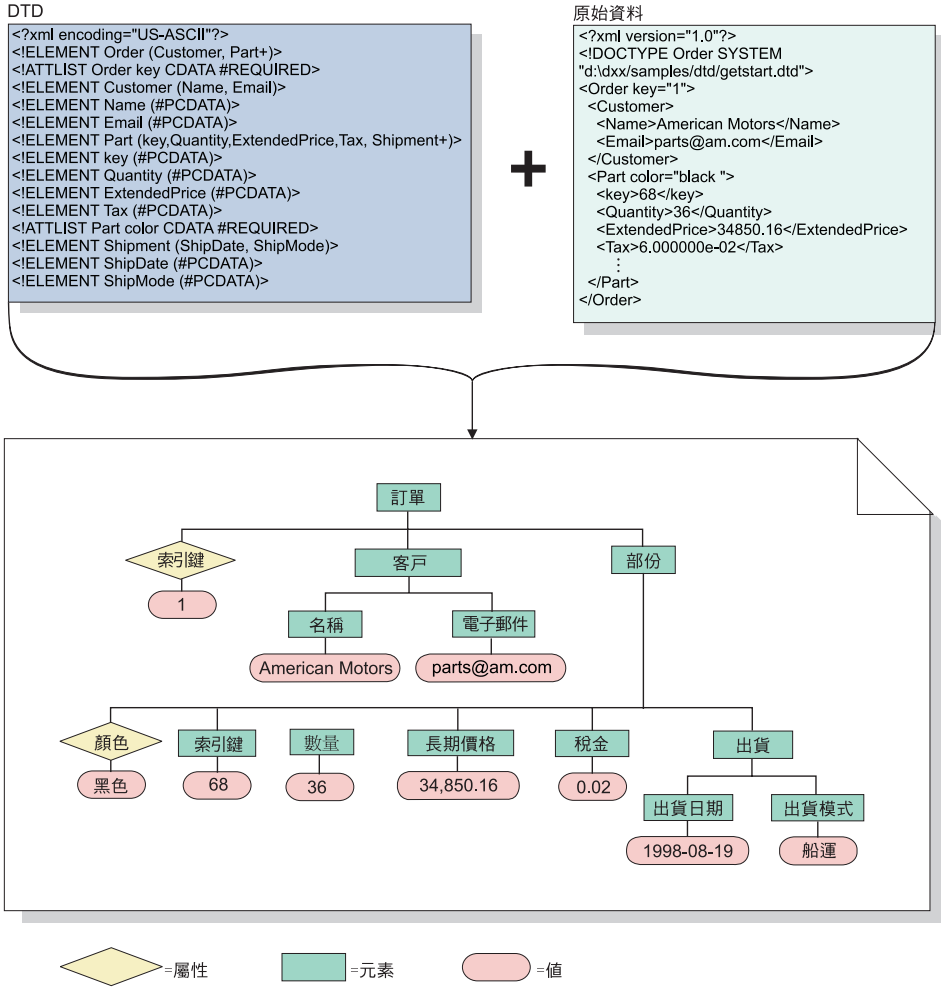


圖 4. DTD 及 XML 文件的階層式結構

決定 XML 直欄的 XML 資料類型

XML Extender 提供一些 XML 使用者資料類型，您用這些資料類型定義要存放 XML 文件的直欄。這些資料類型如下：

- XMLVarchar：適合儲存在 DB2 的小型文件
- XMLCLOB：適合儲存在 DB2 的大型文件
- XMLFILE：適合儲存在 DB2 以外的文件

在本課程中，您會在 DB2 儲存一個小型文件並使用 XMLVarchar 資料類型。

決定要搜尋的元素和屬性

瞭解 XML 文件結構和應用程式的需求之後，您可決定要搜尋哪些元素和屬性，最常搜尋或取出的元素和屬性，或最昂貴的查詢。服務部門表示會經常查詢訂單索引、客戶名稱、價格和訂單的出貨日期，因此需要這些搜尋能夠有快速的效能。本資訊內含於 XML 文件結構的元素和屬性。 表2 說明每一個元素和屬性的位置路徑。

表 2. 要搜尋的元素和屬性

資料	位置路徑
訂單索引	/Order/@key
客戶	/Order/Customer/Name
價格	/Order/Part/ExtendedPrice
出貨日期	/Order/Part/Shipment/ShipDate

將 XML 文件對映到輔助表格

在本教學指導中，您會爲 XML 直欄建立 DAD 檔，並利用它將 XML 儲存在 DB2 中。它也將 XML 元素和屬性內容對映到用於編排索引的 DB2 輔助表格，此動作增進搜尋效能。在上一節，您已了解要搜尋哪些元素和屬性。在本節中，您會進一步學習將這些元素值和屬性值對映到可以編排索引的 DB2 表格。

確認要搜尋的元素和屬性之後，您要決定它們在輔助表格中如何組織、有多少個表格以及哪些直欄是哪一個表格。通常您會在同一個表格放置類似的資訊來組織輔助表格。您也根據在文件中是否可以重複元素的位置路徑來決定此結構。例如在我們的文件中，可以多次重複零件元素，因此可以多次出現價格和日期元素。可多次出現的元素必須在它們自己的表格中。

此外，您也需要決定元素或屬性值應使用什麼 DB2 基本類型。通常根據資料格式就能輕易決定那一種基本類型。若資料是文字，請選取 VARCHAR，若資料是整數，請選取 INTEGER，若資料是日期而且您要執行範圍搜尋，請選取 DATE。

在本教學指導中，元素和屬性會對映到下列輔助表格：

ORDER_SIDE_TAB

直欄名稱	資料類型	位置路徑	多次出現嗎？
ORDER_KEY	INTEGER	/Order/@key	否
CUSTOMER	VARCHAR(16)	/Order/Customer/Name	否

PART_SIDE_TAB

直欄名稱	資料類型	位置路徑	多次出現嗎？
PRICE	DECIMAL(10,2)	/Order/Part/ExtendedPrice	是

SHIP_SIDE_TAB

直欄名稱	資料類型	位置路徑	多次出現嗎？
DATE	DATE	/Order/Part/Shipment/ShipDate	是

就本教學指導而言，我們提供一組 script 讓您用來設定環境。 這些 script 位於 *DXX_INSTALL*\samples\cmd 目錄 (其中 *DXX_INSTALL* 是安裝 XML Extender 的磁碟機和目錄，例如 c:\dxx\samples\cmd)，它們如下：

getstart_db.cmd
建立資料庫並移入四個表格。

getstart_prep.cmd
連結資料庫與 XML Extender 儲存程序和 DB2 CLI。

getstart_insertDTD.cmd
在 XML 直欄插入用來驗證 XML 文件的 DTD。

getstart_createTabCol.cmd
建立一個應用程式表格，該表格會有已啓用 XML 的直欄。

getstart_alterTabCol.cmd
新增要爲 XML 啓用的直欄來變更應用程式表格。

getstart_enableCol.cmd
啓用 XML 直欄。

getstart_createIndex.cmd
在輔助表格爲 XML 直欄建立索引。

getstart_insertXML.cmd
將 XML 文件插入 XML 直欄。

getstart_queryCol.cmd
在應用程式表格執行 select 陳述式並傳回 XML 文件。

getstart_clean.cmd
清除教學指導環境。

設置

在本節中，您準備資料庫以便與 XML Extender 一起使用。您將執行下列動作：

1. 建立資料庫。
2. 啓用資料庫。

建立資料庫

在本節中，您使用指令設定資料庫。本指令建立範例資料庫、連接該資料庫、建立用來保存資料的表格，然後插入資料。

建立資料庫：

1. 變更到 *DXX_INSTALL*\samples\cmd 目錄，其中 *DXX_INSTALL* 是安裝 XML Extender 的磁碟機及目錄。 這些課程採用 c:\dxx 目錄。若您的磁碟機和目錄不同，請變更這些值。
2. 從 Windows NT「開始」功能表開啓「DB2 指令視窗」，或從 Windows NT 指令提示輸入下列指令：

   ```
   DB2CMD
   ```

3. 從「DB2 指令視窗」，執行下列指令：

   ```
   getstart_db.cmd
   ```

啓用資料庫

若要在資料庫儲存 XML 資訊，您必須爲 XML Extender 啓用資料庫。爲 XML 啓用資料庫時，XML Extender 會執行下列動作：

- 建立全部使用者定義類型 (UDT) 及使用者定義函數 (UDF)。
- 建立控制表格並在其中移入 XML Extender 所需的必要 meta 資料。
- 建立 db2xml 綱目並指定必需的專用權。

啓用 XML 的資料庫：

從「DB2 指令視窗」，執行下列 script 啓用 SALES_DB 資料庫：

```
getstart_prep.cmd
```

這個 script 把資料庫連結到 XML Extender 儲存程序及 DB2 CLI。 它也執行啓用資料庫的 **dxxadm** 指令選項：

```
dxxadm enable_db SALES_DB
```

建立 XML 直欄

XML Extender 提供一種在資料庫儲存和存取 XML 文件的方法，稱爲 XML 直欄。 使用 XML 直欄方法時，您可以使用 XML 檔案類型儲存文件，在輔助表格

中編製直欄索引，然後查詢或搜尋 XML 文件。對於不常更新文件的歸檔應用程式來說，本儲存方法特別有用。 就本教學指導的目的而言，您會在 XML 直欄儲存提供的 XML 文件。

在本課程中，您會在 SALES_TAB 表格儲存文件。若要儲存文件，您會執行下列動作：

1. 將 XML 文件的 DTD 插入 DTD 參照表 DTD_REF。
2. 準備 DAD 檔，該檔案指定 XML 文件位置和輔助表格以執行結構搜尋。
3. 在 SALES_TAB 表格中，使用 XML 使用者定義類型 XMLVARCHAR 新增一個直欄。
4. 啓用 XML 的直欄。
5. 編排輔助表格索引以執行結構搜尋。
6. 使用使用者定義的函數儲存文件，XML Extender 提供該函數。

在 DTD 儲存庫儲存 DTD

您可以使用 DTD 驗證 XML 直欄中的 XML 資料。 XML Extender 在 XML 啓用的資料庫建立名稱是 DTD_REF 的表格。此表格稱爲 DTD 參照而且可儲存 DTD。決定要驗證 XML 文件之後，您必須在本儲存庫儲存 DTD。本教學指導的 DTD 是 `c:\dxx\samples\dtd\getstart.dtd`。

插入 DTD：

從「DB2 指令視窗」，輸入下列 SQL INSERT 指令 (全部在同一行)：

```
DB2 CONNECT TO SALES_DB
DB2 INSERT into db2xml.dtd_ref values('c:\dxx\samples\dtd\getstart.dtd',
    db2xml.XMLClobFromFile('c:\dxx\samples\dtd\getstart.dtd'), 0, 'user1',
    'user1', 'user1')
```

您也可執行下列指令檔插入 DTD：

```
getstart_insertDTD.cmd
```

準備 DAD 檔

XML 直欄的 DAD 檔案有簡單的結構。您指定儲存模式爲 XML 直欄，而且定義要編製索引的表格及直欄。

在下列步驟中，DAD 中的元素稱爲*標示*， XML 文件結構的元素稱爲*元素*。 DAD 檔的範例類似您會建立的檔案是在 `c:\dxx\samples\dad\getstart_xcolumn.dad`。它與在下列步驟中建立的檔案有一些小差異。若在此課程中使用它，請注意檔案路徑可能與您的環境不同，<validation> 值是設爲 NO，而不是設爲 YES。

準備 DAD 檔：

1. 開啓文字編輯器並命名檔案 getstart_xcolumn.dad

   請注意：用於 DAD 檔的標示有區分大小寫。

2. 建立 DAD 標頭，具有 XML 和 Doctype 宣告。

   ```
   <?xml version="1.0"?>
   <!DOCTYPE DAD SYSTEM "c:\dxx\dtd\dad.dtd">
   ```

   DAD 檔是 XML 文件，需要 XML 宣告。

3. 插入開啓和關閉 <DAD></DAD> 標示。其它標示位於這些標示裡面。

4. 插入開啓和關閉 <DTDID></DTDID> 標示指定 DTD ID 識別字，該識別字使 DAD 與 XML 文件 DTD 相關，然後在從屬站指定 DAD 位置。

   ```
   <dtdid>c:\dxx\samples\dtd\getstart.dtd</dtdid>
   ```

   確認在 第20頁的『在 DTD 儲存庫儲存 DTD』 中的 DTD 參照表插入 DTD 時，本字串符合作爲第一個參數值的值。 例如，若在不同機器磁碟機工作，那麼用於 DTDID 的路徑可能與上述字串不同。

5. 指定開啓和關閉 <validation></validation> 標示，表示 XML Extender 要使用您插入到 DTD 儲存庫表格的 DTD 驗證 XML 文件結構。

   ```
   <validation>YES</validation>
   ```

   <validation> 值必須是大寫。

6. 插入開啓和關閉 <Xcolumn></Xcolumn> 標示定義儲存方法作爲 XML 直欄。此方法定義 XML 資料要儲存在 XML 直欄中。

   ```
   <Xcolumn>
     </Xcolumn>
   ```

7. 爲要建立的每一個輔助表格插入開啓和關閉 <table></table> 標示。

   ```
   <Xcolumn>
        <table name="order_side_tab">
        </table>
        <table name="part_side_tab">
        </table>
        <table name="ship_side_tab">
        </table>
     </Xcolumn>
   ```

8. 爲要併入輔助表格的直欄插入開啓和關閉 <column></column> 標示。每個 <column> 標示有四種屬性：

   - **名稱**：直欄名稱
   - **類型**：直欄的資料類型
   - **路徑**：XML 文件中的對應元素的位置路徑，使用 XPath 語法。關於位置路徑語法，請參閱 第48頁的『位置路徑』。
   - **多次出現**：指示在 XML 文件結構是否可以多次出現元素的位置路徑

```
<Xcolumn>
     <table name="order_side_tab">
  <column name="order_key"
          type="integer"
            path="/Order/@key"
          multi_occurrence="NO"/>
  <column name="customer"
          type="varchar(50)"
          path="/Order/Customer/Name"
          multi_occurrence="NO"/>
     </table>
     <table name="part_side_tab">
         <column name="price"
          type="decimal(10,2)"
          path="/Order/Part/ExtendedPrice"
          multi_occurrence="YES"/>
     </table>
     <table name="ship_side_tab">
  <column name="date"
          type="DATE"
          path="/Order/Part/Shipment/ShipDate"
          multi_occurrence="YES"/>
     </table>
 </Xcolumn>
```

9. 確定最後一個 </table> 標示後面有一個結束 </Xcolumn>。
10. 確定 </Xcolumn> 標示後面有一個結束 </DAD>。
11. 將檔案儲存成 getstart_xcolumn.dad。

您可比較剛才建立的檔案與範例檔 c:\dxx\samples\dad\getstart_xcolumn.dad。這個檔案是 DAD 檔的工作複本，啓用 XML 直欄和建立輔助表格需要這個檔案。範例檔包含一些路徑陳述式，可能需要將這些陳述式變更爲符合您的環境，才能順利執行它們。

建立 SALES_TAB 表格

在本節中您會建立 SALES_TAB 表格。該表格一開始有兩個直欄，直欄內有訂單的銷售資訊。

建立表格：

從「DB2 指令視窗」，輸入下列 CREATE TABLE 陳述式：

```
DB2 CONNECT TO SALES_DB

DB2 CREATE TABLE SALES_TAB(INVOICE_NUM CHAR(6) NOT NULL PRIMARY KEY,
        SALES_PERSON VARCHAR(20))
```

另外，您可以執行下列指令檔建立表格：

```
getstart_createTabCol.cmd
```

新增 XML 類型的直欄

現在，將一個新直欄新增至 SALES_TAB 表格。本直欄會含有先前建立的完整 XML 文件，而且必須是 XML UDT。 XML Extender 提供多種資料類型，說明於第145頁的『第8章 XML Extender 使用者定義類型』。在本教學指導中，您會將文件儲存成 XMLVARCHAR。

新增 XML 類型的直欄：

從「DB2 指令視窗」，輸入下列 SQL 陳述式：

```
DB2 ALTER TABLE SALES_TAB ADD ORDER DB2XML.XMLVARCHAR
```

另外，您可執行下列指令檔變更表格：

```
getstart_alterTabCol.cmd
```

啓用 XML 直欄

建立 XML 類型的直欄之後，您爲 XML Extender 啓用該直欄。啓用此直欄後，XML Extender 會讀取 DAD 檔並建立輔助表格。啓用此直欄之前，您必須執行下列動作：

- 決定是否要建立 XML 直欄的預設概略表 (該表格包含 XML 文件) 和輔助表格直欄。查詢 XML 文件時可以指定預設概略表。在本課程中，您會使用 -v 參數指定概略表。
- 決定是否要指定主要鍵作爲 *ROOT ID*、應用程式表格中的主要鍵的直欄名稱以及唯一識別字，該識別字使全部輔助表格與應用程式表格相關。若沒有指定主要鍵，XML Extender 就會將 DXXROOT_ID 直欄新增至應用程式表格和輔助表格。 ROOT_ID 直欄連結應用程式表格和輔助表格，可讓 XML Extender 自動更新輔助表格 (若已更新 XML 文件)。在本課程中，您會在指令 (INVOICE_NUM) 中使用 -r 參數指定主要鍵名稱。然後 XML Extender 會使用指定的直欄作爲 ROOT_ID 並將該直欄新增至輔助表格。
- 決定是否要指定表格空間或使用預設表格空間。在本課程中，您會使用預設表格空間。

啓用 XML 的直欄：

從「DB2 指令視窗」，輸入下列指令：

```
dxxadm enable_column SALES_DB SALES_TAB ORDER GETSTART_XCOLUMN.DAD
        -v SALES_ORDER_VIEW -r INVOICE_NUM
```

另外，您可執行下列指令檔啓用 XML 的直欄：

```
getstart_enableCol.cmd
```

XML Extender 會建立使用 INVOICE_NUM 直欄的輔助表格和建立預設概略表。

重要事項：不可修改輔助表格。您只能使用 XML Extender 提供的 UDF 更新 XML 文件。在 XML 直欄更新 XML 文件時，XML Extender 會自動更新輔助表格。

檢視直欄及輔助表格

啓用 XML 直欄之後，請建立 XML 直欄概略表及輔助表格。 處理 XML 直欄時您可使用本概略表。

檢視 XML 直欄及輔助表格直欄：

從「DB2 指令視窗」，輸入下列 SQL SELECT 陳述式：

```
DB2 SELECT * FROM SALES_ORDER_VIEW
```

概略表顯示輔助表格中的直欄，一如 `getstart_xcolumn.dad` 檔所指定。

在輔助表格建立索引

在輔助表格建立索引可讓您執行快速結構搜尋 XML 文件。在本步驟中，您會在輔助表格 (啓用 XML 直欄 ORDER 時建立該表格) 中的鍵值直欄建立索引。服務部門已指定員工可能最常查詢的直欄。 表3 說明您要編排索引的直欄：

表 3. 要編排索引的輔助表格直欄

直欄	輔助表格
ORDER_KEY	ORDER_SIDE_TAB
CUSTOMER	ORDER_SIDE_TAB
PRICE	PART_SIDE_TAB
DATE	SHIP_SIDE_TAB

將輔助表格編排索引：

從「DB2 指令視窗」輸入下列 SQL 指令：

```
DB2 CREATE INDEX KEY_IDX
      ON ORDER_SIDE_TAB(ORDER_KEY)

DB2 CREATE INDEX CUSTOMER_IDX
      ON ORDER_SIDE_TAB(CUSTOMER)

DB2 CREATE INDEX PRICE_IDX
      ON PART_SIDE_TAB(PRICE)

DB2 CREATE INDEX DATE_IDX
      ON SHIP_SIDE_TAB(DATE)
```

另外，您可執行下列指令檔建立索引：

```
getstart_createIndex.cmd
```

儲存 XML 文件

現在您已啓用一個含有 XML 文件的直欄以及編排輔助表格索引，因此可以使用 XML Extender 提供的函數儲存文件。將資料儲存到 XML 直欄時，您會使用預設強制轉型函數或 XML Extender UDF。 因爲您會將基本類型 VARCHAR 的物件儲存到 XML UDT XMLVARCHAR 直欄，所以您會使用預設強制轉型函數。 儲存預設強制轉型函數及 XML Extender 提供之 UDF 的其它資訊，請參閱第104頁的『儲存資料』。

儲存 XML 文件：

重要事項：開啓 XML 文件c:\dxx\samples\xml\getstart.xml。請確定 DOCTYPE 中的路徑與指定在 DAD 中的 DTD ID，及將 DTD 插入 DTD 儲存庫時的路徑相符。您可以藉由查詢 db2xml.DTD_REF 表格及檢查 DAD 檔中的 DTDID 元素，來驗證它們是否相符。 如果您使用不同於預設值的磁碟機及目錄結構，那麼您可能需要變更 DOCTYPE 宣告中的路徑。

從「DB2 指令視窗」，輸入下列 SQL INSERT 指令：

```
DB2 INSERT INTO SALES_TAB (INVOICE_NUM, SALES_PERSON, ORDER) VALUES('123456',
   'Sriram Srinivasan', db2xml.XMLVarcharFromFile('c:\dxx\samples\cmd\getstart.xml'))
```

儲存 XML 文件時，XML Extender 會自動更新輔助表格。

另外，您可執行下列指令檔儲存文件：

```
getstart_insertXML.cmd
```

若要確認已更新表格，請從「DB2 指令視窗」爲表格執行下列 SELECT 陳述式：

```
DB2 SELECT * FROM SALES_TAB

DB2 SELECT * FROM PART_SIDE_TAB

DB2 SELECT * FROM ORDER_SIDE_TAB

DB2 SELECT * FROM SHIP_SIDE_TAB
```

搜尋 XML 文件

您可透過對輔助表格直接查詢來搜尋 XML 文件。在本步驟中，您會搜尋價格超過 2500.00 的訂單。

查詢輔助表格：

從「DB2 指令視窗」輸入下列 SELECT 陳述式：

```
DB2 "SELECT DISTINCT SALES_PERSON FROM SALES_TAB S, PART_SIDE_TAB P
   WHERE PRICE > 2500.00 AND
   S.INVOICE_NUM=P.INVOICE_NUM"
```

結果集應該顯示賣出價格超過 2500.00 的商品的售貨員姓名。

另外，您可執行下列指令檔搜尋文件：

```
getstart_queryCol.cmd
```

您已完成將 XML 文件儲存在 DB2 表格的入門教學指導。本書中許多範例都是根據這些課程而定。

課程：製作 XML 文件

教學指導實務範例

您的工作是收集現存訂單資料庫 SALES_DB 的資訊，然後選出要儲存在 XML 文件的主要資訊。 以後服務部門在處理客戶要求及抱怨時就會使用這些 XML 文件。服務部門已要求包含特定資料並且爲 XML文件提供了一個建議結構。

使用現存資料時，您會使用這些表格中的資料製作 XML 文件 `getstart.xml`。

您也會規劃及建立 DAD 檔，將相關表格的直欄對映到提供訂單記錄的 XML 文件結構。因爲是使用多個表格製作本文件，所以您會建立 XML 集合，該集合使這些表格與 XML 結構及 DTD 相關。您使用這個 DTD 定義 XML 文件結構。您也可在應用程式使用它驗證已製作的 XML 文件。

下列表格說明 XML 文件的現存資料庫資料。 以*斜體字*呈現的直欄名稱是服務部門在 XML 文件結構中要求的直欄。

ORDER_TAB

直欄名稱	資料類型
ORDER_KEY	INTEGER
CUSTOMER	VARCHAR(16)
CUSTOMER_NAME	VARCHAR(16)
CUSTOMER_EMAIL	VARCHAR(16)

PART_TAB

直欄名稱	資料類型
PART_KEY	INTEGER
COLOR	CHAR(6)
QUANTITY	INTEGER
PRICE	DECIMAL(10,2)

直欄名稱	資料類型
TAX	REAL
ORDER_KEY	INTEGER

SHIP_TAB

直欄名稱	資料類型
DATE	DATE
MODE	CHAR(6)
COMMENT	VARCHAR(128)
PART_KEY	INTEGER

規劃

開始使用 XML Extender 製作文件之前，您必須決定 XML 文件結構以及文件如何對應資料庫資料結構。 本節會提供一個概觀，概述服務部門所要求的 XML 文件結構、您用來定義 XML 文件結構的 DTD 、以及這個文件如何對映那些含有用來移入文件的資料的直欄。

決定文件結構

XML 文件結構從多個表格取得某特定訂單的資訊，並建立該訂單的 XML 文件。這些表格含有此訂單的相關資訊，而且可以在它們的鍵值直欄合併該相關資訊。 服務部門想要一個文件，其結構是以訂單號碼作爲頂層，然後是客戶、零件和出貨資訊。他們希望文件結構具有直覺性及彈性，其元素說明資料而不是文件的結構。 (例如，客戶姓名應該是在一個叫作 『客戶，』 的元素中，而不是在一個段落。) 根據他們的要求，DTD 和 XML 文件的階層式結構應類似 第28頁的圖5 說明的其中一種階層式結構。

設計文件結構之後，您應建立 DTD 說明 XML 文件結構。本教學指導提供 XML 文件及 DTD。 您可在 第231頁的『附錄B. 範例』 找到 DTD 檔。 您會發現它符合 第28頁的圖5 中的結構。

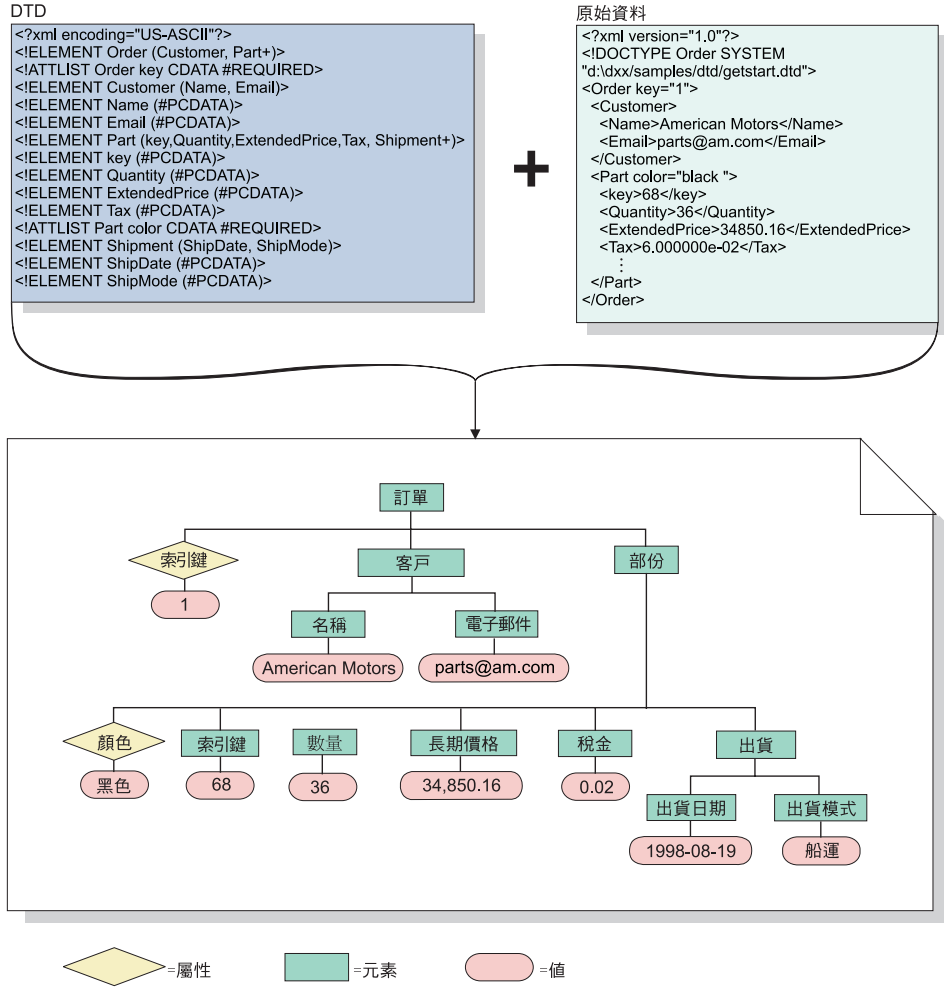


圖 5. DTD 及 XML 文件的階層式結構

對映 XML 文件及資料庫關係

設計結構及建立 DTD 之後，您必須說明文件結構與 DB2 表格 (您要用來移入元素及屬性的) 之間的關聯性。您可將階層式結構對映到關聯式表格中的特定直欄，如第29頁的圖6所示。

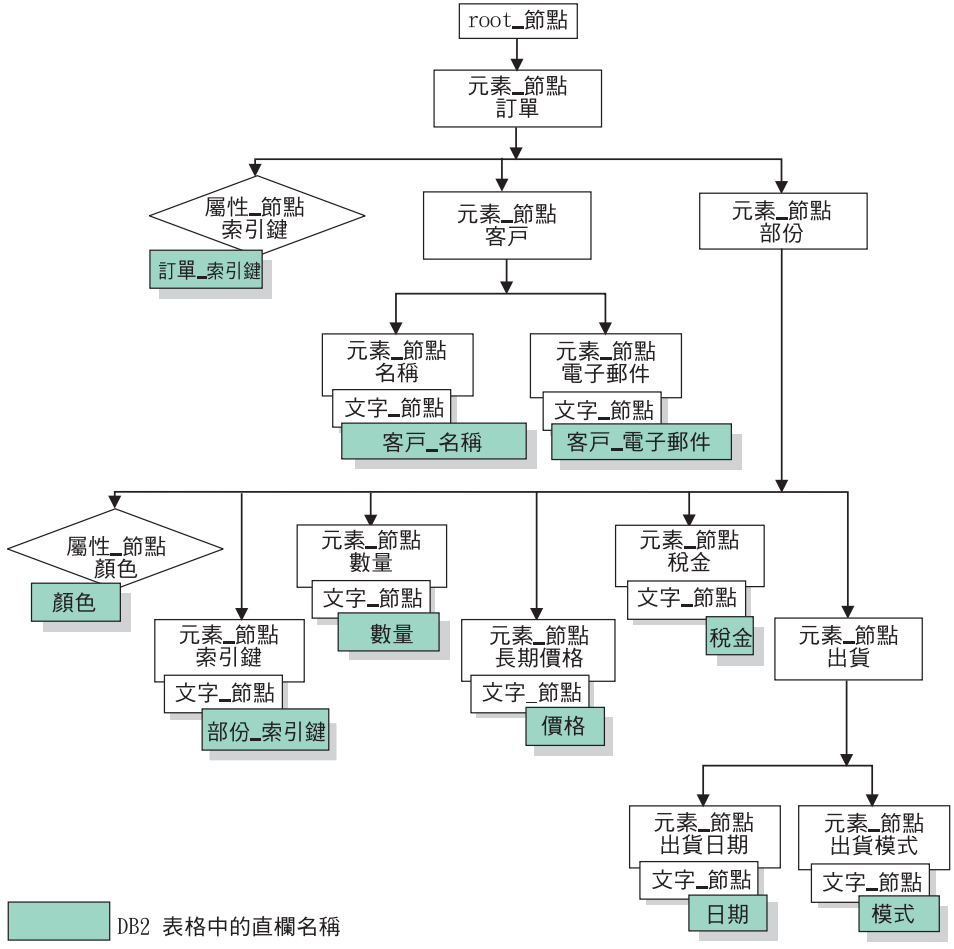


圖 6. XML 文件對映關聯式表格直欄

使用本關係說明建立一些 DAD 檔，定義關聯式資料與 XML 文件結構之間的關係。

若要建立 XML 集合 DAD 檔，您必須瞭解 XML 文件如何對應資料庫結構 (說明於圖6)，才能說明 XML 文件結構從哪些表格及直欄衍生元素及屬性的資料。 您會使用本資訊建立 XML 集合的 DAD檔。

就本教學指導而言，我們提供一組 script 讓您用來設定環境。 這些 script 位於 *DXX_INSTALL*\samples\cmd 目錄 (其中 *DXX_INSTALL* 是安裝 XML Extender 的磁碟機和目錄，例如 c:\dxx\samples\cmd)，它們如下：

getstart_db.cmd

建立資料庫並移入四個表格。

getstart_prep.cmd
連結資料庫與 XML Extender 儲存程序和 DB2 CLI。

getstart_stp.cmd
執行儲存程序來製作 XML 集合。

getstart_exportXML.cmd
從資料庫匯出 XML 文件以便在應用程式中使用。

getstart_clean.cmd
清除教學指導環境。

設置

建立資料庫

在本節中，您使用指令設定資料庫。本指令建立範例資料庫、連接該資料庫、建立用來保存資料的表格，然後插入資料。

重要事項：若完成 XML 直欄課程而且沒有清除環境，您就可以略過本步驟。查看是否有 SALES_DB 資料庫。

建立資料庫：

1. 變更到 *DXX_INSTALL*\samples\cmd 目錄，其中 *DXX_INSTALL* 是安裝 XML Extender 的磁碟機及目錄。 這些課程採用 c:\dxx 目錄。若您的磁碟機和目錄不同，請變更這些值。
2. 從 Windows NT「開始」功能表開啓「DB2 指令視窗」，或從 Windows NT 指令提示輸入下列指令：

   ```
   DB2CMD
   ```

3. 從「DB2 指令視窗」，執行下列指令：

   ```
   getstart_db.cmd
   ```

啓用資料庫

若要在資料庫儲存 XML 資訊，您必須爲 XML Extender 啓用資料庫。爲 XML 啓用資料庫時，XML Extender 會執行下列動作：

- 建立全部使用者定義類型 (UDT) 及使用者定義函數 (UDF)。
- 建立控制表格並在其中移入 XML Extender 所需的必要 meta 資料。
- 建立 db2xml 綱目並指定必需的專用權。

重要事項：若完成 XML 直欄課程而且沒有清除環境，您就可以略過本步驟。

啓用 XML 的資料庫：

從「DB2 指令視窗」，執行下列 script 啓用 SALES_DB 資料庫：

```
getstart_prep.cmd
```

這個 script 把資料庫連結到 XML Extender 儲存程序及 DB2 CLI。 它也執行啓用資料庫的 **dxxadm** 指令選項：

```
dxxadm enable_db SALES_DB
```

建立 XML 集合：準備 DAD 檔

因爲許多個表格都含有此資料，所以您會建立 XML 集合，該集合使表格與 XML 文件相關。若要建立 XML 集合，請準備 DAD 檔來定義此集合。

在第27頁的『規劃』中決定具有資料的關聯式資料庫有哪些直欄，以及來自表格的資料如何在 XML 文件中構成。 在本節中，您會在 DAD 檔建立對映方法，該方法指定表格與 XML 文件結構之間的關係。

在下列步驟中，DAD 中的元素稱爲*標示*， XML 文件結構的元素稱爲*元素*。 類似您會建立的 DAD 檔的範例是在 `c:\dxx\samples\dad\getstart_xcollection.dad`。它與下列步驟中建立的檔案有一些小差異。 若在此課程使用它，請注意檔案路徑可能與您的環境不同。

建立 DAD 檔以製作 XML 文件：

1. 從 `c:\dxx\samples\cmd` 目錄，開啓文字編輯器並建立名稱是 `getstart_xcollection.dad` 的檔案。
2. 使用下列文字建立 DAD 標頭：

   ```
   <?xml version="1.0"?>
   <!DOCTYPE DAD SYSTEM "c:\dxx\dtd\dad.dtd">
   ```

   XML Extender 假設在 `c:\dxx` 安裝此產品。若這個值不正確，請將這個值變成在本步驟安裝本產品時以及在執行下列步驟時指定的磁碟機和目錄。
3. 插入 <DAD></DAD> 標示。其它標示位於這些標示裡面。
4. 指定 <validation> </validation> 標示，指出 XML Extender 是否使用您插入 DTD 儲存庫表格的 DTD 驗證 XML 文件結構。

   ```
   <validation>NO</validation>
   ```
5. 使用 <Xcollection> </Xcollection> 標示將存取及儲存方法定義成 XML集合。存取及儲存方法定義 XML 資料儲存在 DB2 表格集合中。

   ```
   <Xcollection>
     </Xcollection>
   ```

6. 指定一個 SQL 陳述式來指定用於 XML 集合的表格及直欄。本方法稱爲 SQL 對映，而且是將關聯式資料對映到 XML 文件結構的兩種方法的其中一種。(對映方法的細節，請參閱第54頁的『對映方法的類型』。) 輸入下列陳述式：

```
 <SQL_stmt>
SELECT o.order_key, customer_name, customer_email, p.part_key, color, quantity,
price, tax, ship_id, date, mode from order_tab o, part_tab p,
                table(select substr(char(timestamp(generate_unique())),16)
  as ship_id, date, mode, part_key from ship_tab) s
      WHERE o.order_key = 1 and
            p.price > 20000 and
            p.order_key = o.order_key and
            s.part_key = p.part_key
      ORDER BY order_key, part_key, ship_id
</SQL_stmt>
```

使用 SQL 對映時，這個 SQL 陳述式使用下列準則。關於文件結構的資訊，請參閱 第29頁的圖6。

- 按照由上而下順序指定直欄 (根據 XML 文件結構階層)。例如，訂單和客戶元素的直欄是第一層，零件元素是第二層，出貨是第三層。
- 一個實體的直欄集合在一起，每一個群組有一個物件 ID 直欄：ORDER_KEY、PART_KEY 和 SHIP_ID。
- 物件 ID 直欄是每一個群組中的第一個直欄。例如，O.ORDER_KEY 在關於鍵值屬性的直欄前面， p.PART_KEY 在「零件」元素的直欄前面。
- SHIP_TAB 表格沒有單一鍵值條件式直欄，因此會使用 generate_unique DB2 內建函數產生 SHIP_ID 直欄。
- 然後會在 ORDER BY 陳述式按照由上而下順序列示物件 ID 直欄。不可以根據綱目和表格名稱定義 ORDER BY 中的直欄而且應符合 SELECT 子句中的直欄名稱。

寫入 SQL 陳述式時，請參閱 第56頁的『對映方法需求』 以瞭解需求。

7. 新增要用於已製作的 XML 文件的下列前言資訊。

```
<prolog>?xml version="1.0"?</prolog>
```

全部 DAD 檔需要此完全相同的文字。

8. 新增 <doctype></doctype> 標示以用於正在製作的 XML 文件。 <doctype> 標示包含儲存在從屬站的 DTD 的路徑。

```
<doctype>!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
```

9. 使用 <root_node></root_node> 標示定義 XML 文件的 root 元素。在 root_node 中，您可指定組成 XML 文件的元素和屬性。
10. 使用下列三種節點將 XML 文件結構對映到 DB2 關聯式表格結構：

element_node

指定 XML 文件中的元素。Element_nodes 可以有子項 element_node。

attribute_node
指定 XML 文件中某元素的屬性。

text_node
在底層 element_nodes 的關聯式表格中，指定元素及直欄資料的文字內容。

這些節點的其它資訊，請參閱第51頁的『DAD 檔』。 第29頁的圖6 顯示 XML 文件的階層式結構及 DB2 表格直欄，並指出使用那幾種節點。 灰暗框指出取出資料製作 XML 文件的 DB2 表格直欄名稱。

下列步驟可讓您新增每一種節點，一次新增一種。

a. 定義 XML 文件中的每一個元素的 <element_node> 標示。

```
      <root_node>
<element_node name="Order">
           <element_node name="Customer">
    <element_node name="Name">
    </element_node>
    <element_node name="Email">
    </element_node>
    </element_node>
  <element_node name="Part">
    <element_node name="key">
    </element_node>
    <element_node name="Quantity">
    </element_node>
     <element_node name="ExtendedPrice">
    </element_node>
     <element_node name="Tax">
    </element_node>
     <element_node name="Shipment" multi_occurrence="YES">
              <element_node name="ShipDate">
    </element_node>
        <element_node name="ShipMode">
    </element_node>
    </element_node> <!-- end Shipment -->
  </element_node> <!-- end Part -->
</element_node> <!-- end Order -->
</root_node>
```

請注意：<Shipment> 子項元素有 multi_occurence=″YES″ 屬性。本屬性用於沒有屬性的元素，文件中一直重複這些元素。 <Part> 元素不使用多次出現的屬性，因爲它有一個顏色屬性，該屬性使它成爲唯一的元素。

b. 定義 XML 文件中的每一個屬性的 <attribute_node> 標示。在這些屬性的 element_node 中它們以巢狀呈現。以粗體字高亮度顯示新增的 attribute_nodes：

```
      <root_node>
<element_node name="Order">
  <attribute_node name="key">
  </attribute_node>
          <element_node name="Customer">
    <element_node name="Name">
    </element_node>
     <element_node names"Email">
    </element_node>
    </element_node>
  <element_node name="Part">
    <attribute_node name="color">
    </attribute_node>
    <element_node name="key">
    </element_node>
    <element_node name="Quantity">
    </element_node>

...

  </element_node> <!-- end Part -->
</element_node> <!-- end Order -->
</root_node>
```

c. 就每一個底層 element_node 而言，定義 <text_node> 標示，表示 XML 元素含有當製作文件時要從 DB2 取出的字元資料。

```
      <root_node>
<element_node name="Order">
  <attribute_node name="key">
  </attribute_node>
          <element_node name="Customer">
    <element_node name="Name">
        <text_node>
      </text_node>
    </element_node>
    <element_node name="Email">
        <text_node>
      </text_node>
    </element_node>
    </element_node>
  <element_node name="Part">
    <attribute_node name="color">
    </attribute_node>
    <element_node name="key">
       <text_node>
       </text_node>
    </element_node>
    <element_node name="Quantity">
       <text_node>
      </text_node>
    </element_node>
     <element_node name="ExtendedPrice">
       <text_node>
      </text_node>
```

```
    </element_node>
    <element_node name="Tax">
        <text_node>
      </text_node>
    </element_node>
    <element_node name="Shipment" multi-occurence="YES">
              <element_node name="ShipDate">
        <text_node>
        </text_node>
    </element_node>
        <element_node name="ShipMode">
        <text_node>
        </text_node>
    </element_node>
    </element_node> <!-- end Shipment -->
  </element_node> <!-- end Part -->
</element_node> <!-- end Order -->
</root_node>
```

d. 就每一個底層 element_node 而言，請定義 <column> 標示。這些標示指定當製作 XML 文件時從哪一個直欄取出資料，而且通常是在 <attribute_node> 或 <text_node> 標示內。請記得！在此處定義的直欄必須在 <SQL_stmt> SELECT 子句中。

```
      <root_node>
<element_node name="Order">
  <attribute_node name="key">
    <column name="order_key"/>
  </attribute_node>
          <element_node name="Customer">
    <element_node name="Name">
                <text_node>
      <column name="customer_name"/>
      </text_node>
    </element_node>
    <element_node name="Email">
                <text_node>
      <column name="customer_email"/>
                </text_node>
    </element_node>
    </element_node>
  <element_node name="Part">
    <attribute_node name="color">
       <column name="color"/>
    </attribute_node>
    <element_node name="key">
                <text_node>
      <column name="part_key"/>
      </text_node>
    <element_node name="Quantity">
                <text_node>
      <column name="quantity"/>
      </text_node>
    </element_node>
     <element_node name="ExtendedPrice">
```

```
                    <text_node>
                <column name="price"/>
              </text_node>
            </element_node>
            <element_node name="Tax">
                    <text_node>
                <column name="tax"/>
              </text_node>
            </element_node>
            <element_node name="Shipment" multi-occurence="YES">
                      <element_node name="ShipDate">
                        <text_node>
                  <column name="date"/>
                </text_node>
            </element_node>
                <element_node name="ShipMode">
                        <text_node>
                  <column name="mode"/>
                </text_node>
            </element_node>
            </element_node> <!-- end Shipment -->
          </element_node> <!-- end Part -->
        </element_node> <!-- end Order -->
        </root_node>
```

11. 在最後一個 </element_node> 標示後面，請記得加上 </root_node> 結束標示。
12. 在 </root_node> 標示後面，請記得加上 </Xcollection> 結束標示。
13. 在 </Xcollection> 標示後面，請記得加上 </DAD> 結束標示。
14. 將檔案儲存爲 getstart_xcollection.dad

您可比較剛才建立的檔案與範例檔 c:\dxx\samples\dad\getstart_xcollection.dad。這個檔案是製作 XML 文件需要的 DAD 檔的工作複本。範例檔包含一些路徑陳述式，可能需要將這些陳述式變更爲符合您的環境，才能順利執行它們。

在應用程式中，若會經常使用 XML 集合製作文件，那麼您可啓用此集合定義集合名稱。在執行儲存程序時指定集合名稱 (不是 DAD 檔名) 的時候，啓用此集合會將它登錄到 XML_USAGE 表格中並可增進效能。在這些課程中，您不會啓用此集合。若要更加瞭解啓用集合，請參閱 第96頁的『啓用 XML 集合』。

製作 XML 文件

在本步驟中，您使用 dxxGenXML() 儲存程序製作 DAD 檔指定的 XML 文件。本儲存程序傳回文件作爲 XMLVARCHAR UDT。

製作 XML 文件：

1. 從「DB2 指令視窗」，輸入下列指令執行儲存程序：

```
getstart_stp.cmd
```

已製作 XML 文件而且儲存在 RESULT_TAB 表格中。

您可在下列檔案中查看一些儲存程序範例，您可以在本步驟中使用這些範例：

- c:\dxx\samples\c\tests2x.sqc 說明如何使用內含的 SQL 呼叫儲存程序，以及產生 texts2x 可執行檔， getstart_stp.cmd 使用該可執行檔。
- c:\dxx\samples\cli\sql2xml.c 說明如何使用 CLI 呼叫儲存程序。

2. 使用 XML Extender 擷取函數 Content()，從表格將 XML 文件匯出至檔案：

```
DB2 CONNECT TO SALES_DB

DB2 SELECT db2xml.Content(db2xml.xmlvarchar(doc),
   'c:\dxx\samples\cmd\getstart.xml') FROM RESULT_TAB
```

另外，您可執行下列指令匯出檔案：

```
getstart_exportXML.cmd
```

本課程說明如何使用 DB2 儲存程序的結果集特性取得一個或多個已製作的 XML 文件，讓您提取每一列以取得每一個文件。取得文件的每一列之後，您可將文件匯出至檔案，這是示範這個特性的最簡單方法。關於提取資料的更有效方法，請參閱 c:\dxx\samples\cli 中的 CLI 範例。

清除教學指導環境

若要清除教學指導環境，您可以執行 getstart_clean.cmd 檔。這個檔案：

- 停用 XML 直欄 ORDER
- 捨棄在教學指導建立的表格
- 從 DTD 參照表刪除 DTD

本指令檔不停用或捨棄 SALES_DB 資料庫；資料庫仍然可以與 XML Extender 一起使用。若在本章中沒有完成這兩個課程，那麼您可能會收到錯誤訊息。您可忽略這些錯誤訊息。

清除教學指導環境：

1. 從「DB2 指令視窗」，執行下列指令：

```
getstart_clean.cmd
```

2. 若要停用資料庫，您可從「DB2 指令視窗」執行下列 XML Extender 指令：

```
dxxadm disable_db SALES_DB
```

這個指令捨棄管理控制表格 DTD_REF 和 XML_USAGE，移除 XML Extender 提供的使用者定義類型和函數。

3. 若要捨棄資料庫，您可從「DB2 指令視窗」執行下列指令：

```
db2 drop database SALES_DB
```

這個指令捨棄 SALES_DB。

第2篇 管理

這個部份說明如何執行 XML Extender 的管理作業。

第3章 準備使用 XML Extender：管理

本章說明設置與規劃 XML Extender 的需求。

設置需求

下列各節說明 XML Extender 的設置需求。

軟體需求

XML Extender 可用於 AIX、Windows NT 及 Sun Solaris。

軟體需求：XML Extender 要求使用 DB2 Universal Database 版本 6.1 或更新版本。若使用 DB2 UDB 6.1，您必須有 Fixpack 2。

可選用的軟體：

- 若是建構文字搜尋，需要 DB2 Universal Database Text Extender 版本 6.1 或更新版本
- 若是 XML Extender 管理工具：
  - DB2 UDB JDBC 驅動程式 1.22 (可用於 DB2 UDB 版本 6.1 或更新版本)
  - JDK 1.1.7 或 JRE 1.1.7 (可用於 DB2 UDB Control Center)
  - JFC 1.1 與 Swing 1.1 (可用於 DB2 UDB Control Center)

安裝需求

請參閱作業系統的 README 檔以執行下列作業：

- 連結 XML Extender 至 DB2 UDB 資料庫。

  基於安全理由，您必須將 XML Extender 連結到每一個資料庫。 關於如何完成連結的詳細資訊，請參閱第180頁的『開始之前』，或參閱範例

  DXX_INSTALL\samples\cmd\getstart_prep.cmd
- 檢視 UNIX 的設定指示。
- 建立供 XML 存取的資料庫。

授權需求

您需要 DB2ADM 權限才能執行管理作業。

管理工具

XML Extender 提供三個管理方法：XML Extender 管理精靈、 XML Extender 管理指令及 XML Extender *儲存程序*。

- 管理精靈會引導您完成管理作業，它是建議的管理方法。 這個工具的用法說明於第61頁的『第4章 管理 XML 資料』中的管理作業。
- 管理指令 **dxxadm** 對不同管理作業提供選項。 這個指令的用法說明於第61頁的『第4章 管理 XML 資料』及第135頁的『第7章 XML Extender 管理指令：**dxxadm**』中的管理作業。
- 管理儲存程序也提供各種管理作業的選項。 這些儲存程序說明於第180頁的『管理儲存程序』中。

管理規劃

規劃使用 XML 文件的應用程式時，您必須先做下列設計決策：

- 是否利用資料庫中的資料來編排 XML 文件
- 是否儲存預存的 XML 文件，以及要將這些文件在直欄中儲存爲完整的 XML 文件或分解成一般 DB2 資料

在訂定這些決策之後，即可開始規劃其它的管理作業：

- 是否驗證 XML 文件
- 是否對 XML 直欄資料編排索引來快速搜尋及擷取
- 如何將 XML 文件結構對映到 DB2 關聯式表格

根據您的應用程式需求來使用 XML Extender。如同第3頁的『第1章 XML Extender 簡介』中所述，您可以利用現存的 DB2 資料來編排 XML 文件，然後將這些 XML 文件儲存到 DB2 中，作爲完整文件或 DB2 資料。 這些儲存和存取方法有不同規劃需求。下列各節分別討論這些規劃注意事項。

選擇存取與儲存方法

XML Extender 提供兩種存取與儲存方法，來使用 DB2 作爲 XML 儲存庫：XML 直欄與 XML 集合。您必須先決定哪一個方法最符合應用程式存取與操作 XML 資料的需求。

XML 直欄

儲存及擷取整個 XML 文件爲 DB2 直欄資料。XML 資料由 XML 直欄代表。

XML 集合

將 XML 文件分解成關聯式表格的集合，或從關聯式表格的集合來組合 XML 文件。

應用程式本質可決定所要使用的存取類型與儲存方法，以及如何建構 XML 資料。下列實務說明每一個存取與儲存方法最適當的狀況。

何時使用 XML 直欄

下列狀況中會使用到 XML 直欄：

- XML 文件已存在或來自某些外部來源，並且您偏好以原始 XML 格式儲存文件。基於資料完整性及歸檔與審核目的，將文件儲存在 DB2 中。
- 一般而言，XML 文件可讀取但不能更新。
- 您要使用檔名資料類型將 DB2 外部的 XML 文件儲存到本端或遠端檔案系統，以及在管理及搜尋作業中使用 DB2。
- 您需要根據 XML 元素或屬性值來安排搜尋範圍，並且您知道哪些元素或屬性常當作搜尋引數。
- 文件含有具大量文字區塊的元素，並且在保持整個文件的完整性時，您要使用 DB2 Text Extender 來建構文字搜尋。

何時使用 XML 集合

在下列狀況中使用 XML 集合：

- 您的現存關聯式表格中含有資料，而您要根據一定的 DTD 來組合 XML 文件。
- 您的 XML 文件需要以資料集合 (正確對映到關聯式表格) 來儲存。
- 您需要使用不同對映方法建立關聯式資料的不同概略表。
- 您有一些來自其它資料來源的 XML 文件。您關心的是資料而非標示，並且要在資料庫中儲存純粹資料。您要彈性地決定將資料儲存在某些現存的表格或新表格中。
- XML 文件的小型子集需要時常更新，且更新效能很重要。
- 您需要儲存整個進入 XML 文件的資料，但通常僅要擷取它們的一個子集。
- 您的 XML 文件超出二十億位元組，您必須分解它們。

您可以透過這兩個存取與儲存方法，利用 DAD 檔來使 XML 資料與 DB2 表格產生關聯。 第44頁的圖7顯示 DAD 如何指定存取與儲存方法。

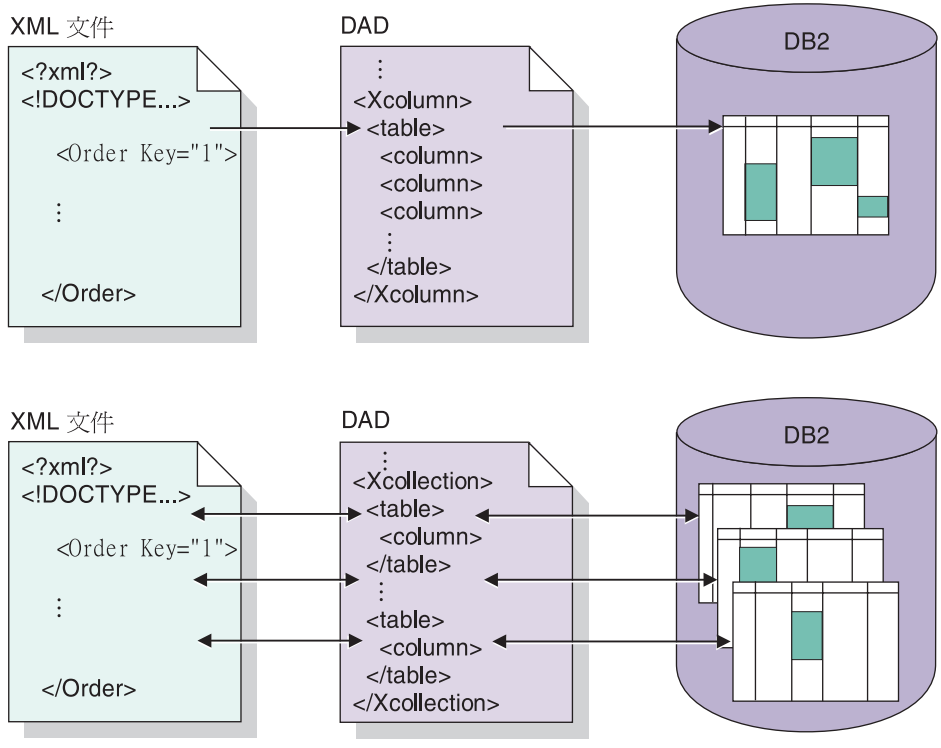


圖 7. DAD 檔將 XML 文件結構對映至 DB2，並指定存取與儲存方法。

DAD 檔是管理 XML Extender 的一個重要部份。它定義鍵值檔的位置，就像 DTD 一樣，並且指定 XML 文件結構如何與您的 DB2 資料相關。最重要的是，它定義您應用程式中使用的存取與儲存方法。

規劃 XML 直欄

下列各節將爲您說明 XML 直欄的規劃作業。

驗證

選擇存取與儲存方法之後，您可以決定是否*驗證*資料及可以指定 DAD。您可以使用 DTD 來驗證 XML 資料。利用 DTD 來確保 XML 文件的有效性，它可讓您對 XML 資料執行結構搜尋。 DTD 儲存於 DTD 儲存庫中。

建議事項：除非您儲存 XML 文件的目的是歸檔，否則請使用 DTD 來驗證 XML 資料。驗證之前，請確定 XML Extender 儲存庫中有 DTD。 如何插入 DTD 到儲存庫中，請參閱第64頁的『將 DTD 存入 DTD 儲存庫中』。

您可以使用不同的 DTD 來驗證同一個 XML 直欄中的文件。換言之，您可以使用具有類似結構、類似元素及屬性的文件來呼叫不同的 DTD。 要參照多重 DTD，請遵循下列準則：

- DOCTYPE 定義中 XML 文件的系統 ID 必須使用完整路徑名稱來指定 DTD 檔。
- 您必須指定「是」才能在 DAD 檔中進行驗證。
- 至少要有一個 DTD 儲存於 DTD_REF 表格中。所有 DTD 都可儲存於這個表格中。
- DTD 應有一個共用結構，這些 DTD 的唯一差異在於子元素。
- DAD 檔應指定該直欄中文件參照的所有 DTD 共用之元素或屬性。

重要事項：請決定是否在插入 XML 資料到 DB2 之前驗證。 XML Extender 不支援已插入 DB2 的資料驗證。

注意事項：

- 您不需要 DTD 來儲存或保存 XML 文件。
- 您必須使用 DTD 來執行結構搜尋。
- 驗證 XML 資料可能會使效能稍微受到影響。
- 您可以使用多重 DTD，但只能對共用元素及屬性編排索引。

XML 使用者定義的類型

您可以在 XML 直欄中將 XML 文件儲存爲 UDT。可用的 UDT，請參閱表4。

表 4. XML Extender UDT

使用者定義的類型直欄	來源資料類型	用法說明
XMLVARCHAR	VARCHAR(*varchar_len*)	將整個 XML 文件儲存成 DB2 內的 VARCHAR。
XMLCLOB	CLOB(*clob_len*)	將整個 XML 文件在 DB2 內儲存爲 *CLOB*。
XMLFILE	VARCHAR(1024)	在 DB2 內儲存 XML 文件的檔名，以及將 XML 文件儲存到 DB2 伺服器本端的檔案。

輔助表格

規劃輔助表格時，您必須考慮如何組織表格，要建立多少表格以及是否建立輔助表格的預設概略表。這些決定有部份是根據一些問題而定：元素和屬性是否可以出現數次，以及查詢效能的需求。

多次出現的項目: 一個文件有多次出現的多重位置時，XML Extender 會在每一個輔助表格新增類型 INTEGER 的直欄 DXX_SEQNO，追蹤多次出現的元素順序。有了 DXX_SEQNO 之後，您可以在 SQL 查詢指定 ORDER BY DXX_SEQNO 擷取元素列示，這些元素使用與原始 XML 文件相同的順序。

預設概略表和查詢效能: 啓用 XML 直欄時，您可指定一個預設的唯讀概略表，該表格使用名稱是 ROOT ID 的唯一 ID 來合併應用程式表格與輔助表格。使用預設概略表，您可以透過查詢輔助表格搜尋 XML 文件。例如，若有應用程式表格 SALES_TAB 和輔助表格 ORDER_TAB、PART_TAB、SHIP_TAB：

```
SELECT sales_person FROM sales_order_view
   WHERE price > 2500.00
```

SQL 陳述式傳回 SALES_TAB 中的銷售員姓名，這些銷售員有訂單儲存在 ORDER 直欄，而且 PRICE 超過 2500.00。

查詢預設概略表的優點是該表格提供應用程式表格和輔助表格的虛擬單一檢視畫面。不過，建立愈多輔助表格，查詢成本就愈高。 因此，唯有輔助表格直欄總數不大時才建議建立預設概略表。應用程式可建立自己的概略表，合併重要輔助表格直欄。

XML 直欄資料的索引

是否要將 XML 直欄文件編排索引是一個重要的規劃決策。這個決策應根據您多久需要存取資料，以及結構搜尋期間效能的重要性來訂定。

當使用含有整個 XML 文件的 XML 直欄時，您可以建立輔助表格來包含 XML 元素或屬性值的直欄，然後在這些直欄建立索引。您必須判斷哪些是需要建立索引的元素及屬性。

XML 直欄索引容許透過資料庫引擎的本體 DB2 索引支援，對一般資料類型 (如整數、小數或日期) 的經常性查詢資料進行索引編排。 XML Extender 可從 XML 文件擷取 XML 元素或屬性的值，然後將它們儲存於*輔助表格*內，讓您在這些輔助表格建立索引。

您可以透過位置路徑來指定輔助表格的每一個直欄，位置路徑可識別 XML 元素或屬性及 SQL 資料類型。 第47頁的圖8顯示輔助表格內的某 XML 直欄。

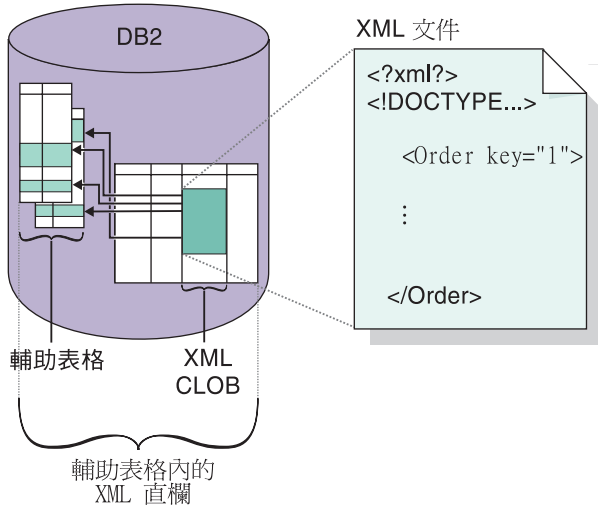


圖 8. XML 直欄及輔助表格

當您在 XML 直欄中儲存 XML 文件時，XML Extender 會自動移入輔助表格。

若要快速搜尋，請使用 DB2 *B-tree 索引*技術，在這些直欄建立索引。建立索引的方法視不同的作業系統而不同，XML Extender 支援這些方法。

注意事項：

- 對 XML 文件中具有*多次出現的項目*的元素或屬性而言，由於 XML 文件的複雜結構，您必須爲每一個具有多次出現的項目之 XML 元素或屬性，建立個別的輔助表格。

  例如，您可能要在 `/Order/Part/ExtendedPrice` 建立索引，及指定 `/Order/Part/ExtendedPrice` 爲資料類型 REAL。在這種情況中，XML Extender 會將 PRICE 直欄中的 `/Order/Part/ExtendedPrice` 的值儲存在輔助表格。
- 您可以在某 XML 直欄建立多重索引。在前一個範例中，您可以在兩個輔助表格中建立兩個直欄，一個用於 `ExtendedPrice`，另一個用於 `ShipDate`。
- 您可以使用 ROOT ID、應用程式表格中的主要鍵的直欄名稱，以及唯一識別字 (使應用程式表格與全部輔助表格相關) 使輔助表格與應用程式表格相關。您可決定是否要應用程式表格的主要鍵作爲 ROOT ID，雖然它無法作爲組合鍵。這是建議的作法。

  如果應用程式表格中沒有單一主要鍵，或基於某些理由您不想使用它， XML Extender 會變更應用程式表格來新增直欄 DXXROOT_ID，此直欄儲存插入時建立的唯一 ID。 所有輔助表格都有內含唯一 ID 的 DXXROOT_ID 直欄。 若主

要鍵作爲 ROOT ID，那麼全部輔助表格就有一個直欄的名稱和類型與應用程式表格的主要鍵直欄相同，並儲存主要鍵的值。

- 如果對 DB2 Text Extender 啓用 XML 直欄，您亦可使用 Text Extender 的結構式文字特性。 Text Extender 具有 *"區段搜尋"* 支援，它延伸了慣用全文搜尋的能力，容許搜尋字組符合於位置路徑指定的特定文件內容中。在一般 SQL 資料類型上，*結構式文字索引*可搭配 XML Extender的索引使用。

位置路徑

*位置路徑*是一連串 *XML 標示*，用來識別 XML 元素或屬性。使用位置路徑的 XML Extender 是在下列狀況下使用：

- 當取出 UDF 來識別要取出的元素及屬性時
- 在 DAD 中編排 XML 直欄的方法索引時，指定 XML 元素或屬性和 DB2 直欄之間的對映檔
- Text Extender 用來執行結構式文字搜尋

圖9顯示位置路徑的範例，以及與 XML 文件結構的關係。

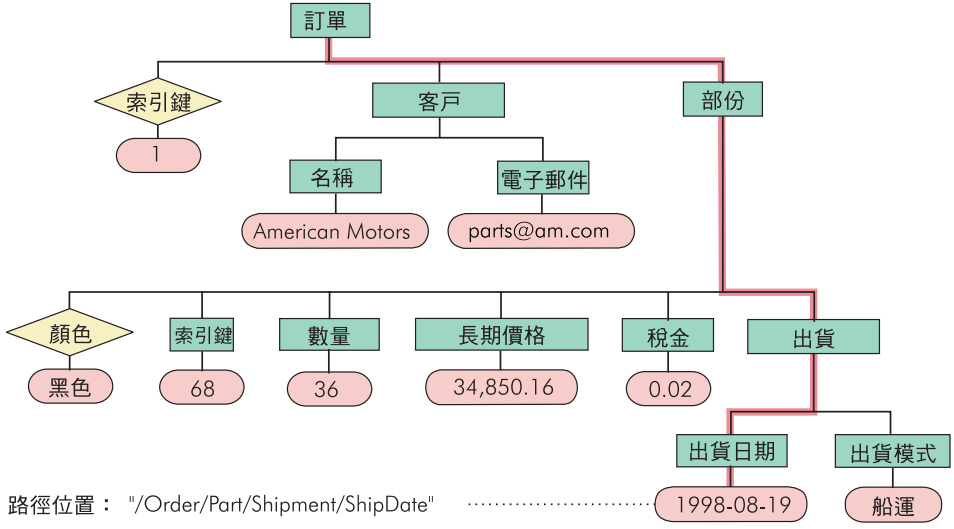


圖 9. 將文件儲存爲 DB2 表格直欄中的結構化 XML 文件

位置路徑語法: 以下列示說明 XML Extender 支援的位置路徑語法。

1. */* 代表 XML *root 元素*。

2. */tag1*
代表 root 下方的元素 *tag1*。

3. */tag1/tag2/..../tagn*

   代表一個元素，以名稱 *tagn* 做爲從 root、*tag1*、*tag2* 一直到 *tagn*-1 之降序鏈的子項。

4. *//tagn*

   代表名稱爲 *tagn* 的任何元素，其中雙斜線 (//) 表示零或多個任意的標示。

5. */tag1//tagn*

   代表名稱爲 *tagn* 的任何元素，該元素是 root 下方名稱爲 *tag1* 之元素的子項，其中雙斜線 (//) 表示零或多個任意的標示。

6. */tag1/tag2/@attr1*

   代表名稱爲 *tag2* 之元素的屬性 *attr1*，該元素是 root 下方 *tag1* 元素的子項。

7. */tag1/tag2[@attr1="5"]*

   代表名稱爲 *tag2* 的元素，其屬性 *attr1* 的值是 5。*tag2* 是 root 下方名稱爲 *tag1* 之元素的子項。

8. */tag1/tag2[@attr1="5"]/.../tagn*

   代表名稱爲 *tagn* 的元素，該元素是從 root、 *tag1*、*tag2* 一直到 *tagn*-1 之降序鏈的子項，其中 *tag2* 之屬性 *attr1* 的值是 5。

萬用字元：您可以用星號替代位置路徑中的一個元素以符合任何字串。

簡式位置路徑: *簡式位置路徑* 是一連串元素類型名稱，以單斜線 (/) 連接。屬性值是在其元素類型後面並以方括弧括住。 第48頁的『位置路徑語法』中 3 及 6 的語法說明，解釋簡式位置路徑。表5說明語法。

表 5. 簡式位置路徑語法

主旨	位置路徑	說明
XML 元素	*/tag1/tag2/..../tagn-1/tagn*	以元素名稱 *tagn* 及其母項所識別的某元素內容
XML 屬性	*/tag_1/tag_2/..../tag_n-1/tag_n/@attr1*	以 *tagn* 及其母項所識別之元素的某個屬性名稱 *attr1*

XML Extender 限制: 在 DAD 中定義元素或屬性時，XML Extender 有使用位置路徑的限制條件。 因爲 XML Extender 在元素或屬性及 DB2 直欄之間使用一對一的對映，所以需要特殊規則的位置路徑。 第50頁的表6說明位置路徑的限制。關於位置路徑支援的直欄中指定的數目，請參照第48頁的『位置路徑語法』中的語法說明。

表 6. XML Extender 對於使用位置路徑的限制

位置路徑的用法	支援的位置路徑
DAD 中的元素	3，6 (第49頁的表5中說明的簡式位置路徑)
取出 UDF	1-9
Text Extender 的搜尋 UDF	1-9

DAD 檔

以 XML 直欄而言，DAD 主要是指定 XML 直欄中儲存的多少個文件要編排索引。 DAD 是一個常駐於從屬站的 XML 格式化文件。 如果您選擇使用 DTD 驗證 XML 文件，則 DAD 檔會與該 DTD 產生關聯。 DAD 檔具有資料類型 CLOB。

XML 直欄的 DAD 檔包含 XML 標頭，其指定 DAD 檔與 DTD 在從屬站的目錄路徑，並且提供要儲存於輔助表格中任何 XML 資料的對映來編排索引。

若要指定 XML 直欄存取與儲存方法，可以使用 DAD 檔中的下列標示。

<Xcolumn>

指定儲存及擷取 XML 資料爲 DB2 直欄中，針對 XML 資料啓用的完整 XML 文件。

XML 型直欄爲 XML Extender 的 UDT。應用程式可將該直欄併入任何 *使用者表格*中。 您主要透過 SQL 陳述式及 XML Extender 的 UDF 來存取 XML 直欄資料。

您可以使用 XML Extender 管理精靈或編輯器來建立及更新 DAD。

規劃 XML 集合

規劃 XML 集合時，要從 DB2 資料編製文件或將 XML 文件分解成爲 DB2 資料或兩者都要，對此您有不同的考量。下列各節說明 XML 集合的規劃議題，以及說明編製和分解的注意事項。

驗證

選擇存取與儲存方法之後，您可以決定是否驗證資料。 您可以使用 DTD 來驗證 XML 資料。利用 DTD 來確保 XML 文件的有效性，它可讓您對 XML 資料執行結構搜尋。 DTD 儲存於 DTD 儲存庫中。

建議事項： 使用 DTD 來驗證 XML 資料。 驗證之前，請確定 XML Extender 儲存庫中有 DTD。 如何插入 DTD 到儲存庫中，請參閱第64頁的『將 DTD 存入 DTD 儲存庫中』。 根據您要編製或分解 XML 文件，DTD 需求會有不同。

- 以編製而言，您只能對一個 DTD 驗證建立的 XML 文件。所要使用的 DTD 指定於 DAD 檔中。
- 以分解而言，您可以利用不同的 DTD 來驗證編製的文件。換言之，您可以使用相同的 DAD 檔來分解文件，但呼叫的是不同的 DTD。若要參照多重 DTD，請遵循下列準則：
  - 至少要有一個 DTD 儲存於 DTD_REF 表格中。所有 DTD 都可儲存於這個表格中。
  - DTD 應有一個共用結構，這些 DTD 的差異在於子元素。
  - 您必須在 DAD 檔中指定驗證。
  - XML 文件的系統 ID 必須使用完整路徑名稱來指定 DTD 檔。
  - DAD 檔包含如何分解文件的規格，所以您可以僅指定分解的共用元素及屬性。 DTD 唯一的元素與屬性不能分解。

重要事項：請決定是否在插入 XML 資料到 DB2 之前驗證 XML 資料。 XML Extender 不支援已插入 DB2 的資料驗證。

注意事項：

- 當使用 XML 爲交換格式時，您應該使用 DTD。
- 驗證 XML 資料可能會使效能稍微受到影響。
- 當使用多重 DTD 來分解時，您僅可以分解共用元素與屬性。
- 當使用一個 DTD 時，您可以分解所有元素及屬性。
- 您僅可以使用一個 DTD 來編製。

DAD 檔

針對 XML 集合，DAD 檔會將 XML 文件結構對映至 DB2 表格，您可以在 DB2 表格中編製文件或分解文件。

例如，如果您的 XML 文件中有元素 <Tax>，則必須將 <Tax> 對映至稱爲 TAX 的直欄。您可以定義 XML 資料與 DAD 中關聯式資料之間的關係。

當啓用集合或在 XML 集合*儲存程序*中使用 DAD 檔時，指定 DAD 檔。 DAD 是一個常駐於從屬站的 XML 格式化文件。如果您選擇使用 DTD 驗證 XML 文件，則 DAD 檔會與該 DTD 產生關聯。 當使用爲 XML Extender 儲存程序的輸入參數時，DAD 檔具有資料類型 CLOB。

若要指定 XML 集合存取與儲存方法，可以使用 DAD 檔中的下列標示：

<Xcollection>

指定將 XML 資料從 XML 文件分解成關聯式表格的集合，或從關聯式表格的集合來組合成 XML 文件。

XML 集合是一組含有 XML 資料的關聯式表格之虛擬名稱。應用程式可啓用任何使用者表格的 XML 集合。這些使用者表格可以是大型業務資料的現存表格，或 XML Extender 最近建立的表格。您主要可以透過 XML Extender 提供的儲存程序來存取 XML 集合資料。

DAD 檔利用下列節點種類來定義 XML 文件樹狀結構：

root_node

指定文件的 Root 元素。

element_node

定義一個元素，它同時可以是 Root 元素或子項元素。

text_node

代表元素的 CDATA 文字。

attribute_node

代表元素的屬性。

第53頁的圖10顯示 DAD 檔中使用的對映片斷。節點將 XML 文件內容對映到關聯式表格中的表格直欄。

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dtd\dad.dtd">
<DAD>
  ...
<Xcollection>
    <SQL_stmt>
       ...
</SQL_stmt>
      <prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE DAD SYSTEM "c:\dxx\sample\dtd\getstart.dtd"</doctype>
      <root_node>
   <element_node name="Order">          --> 定義元素 <次序>
     <attribute_node name="key">        --> 定義屬性 "鍵"
        <column name="order_key"/>      --> 定義直欄名稱， "order_key"，
                                                元素及屬性對映到該直欄
             </attribute_node>
     <element_node name="Customer">     --> 定義 <次序> 的子項元素為
  <Customer>
       <text_node>                      --> 對該元素指定 CDATA 文字
  <Customer>
         <column name="customer">       --> 定義直欄的名稱， "客戶"，
                                                子項元素對映到該直欄
       </text_node>
    </element_node>
       ...
    </element_node>

      ...
      <root_node>
  </Xcollection>
</DAD>
```

圖 10. 節點定義

本範例中，SQL 陳述式中前兩個直欄有對映到它們的元素及屬性。

您可以使用 XML Extender 管理精靈或編輯器來建立及更新 DAD 檔。

XML 集合的對映方法

如果您是使用 XML 集合，則必須選取*對映方法*，該方法定義 XML 資料如何在關聯式資料庫中顯示。由於 XML 集合必須符合 XML 文件中使用的階層式結構及關聯式結構，所以您應該瞭解如何比較這兩種結構。 第54頁的圖11顯示階層式結構如何對映到關聯式表格直欄。

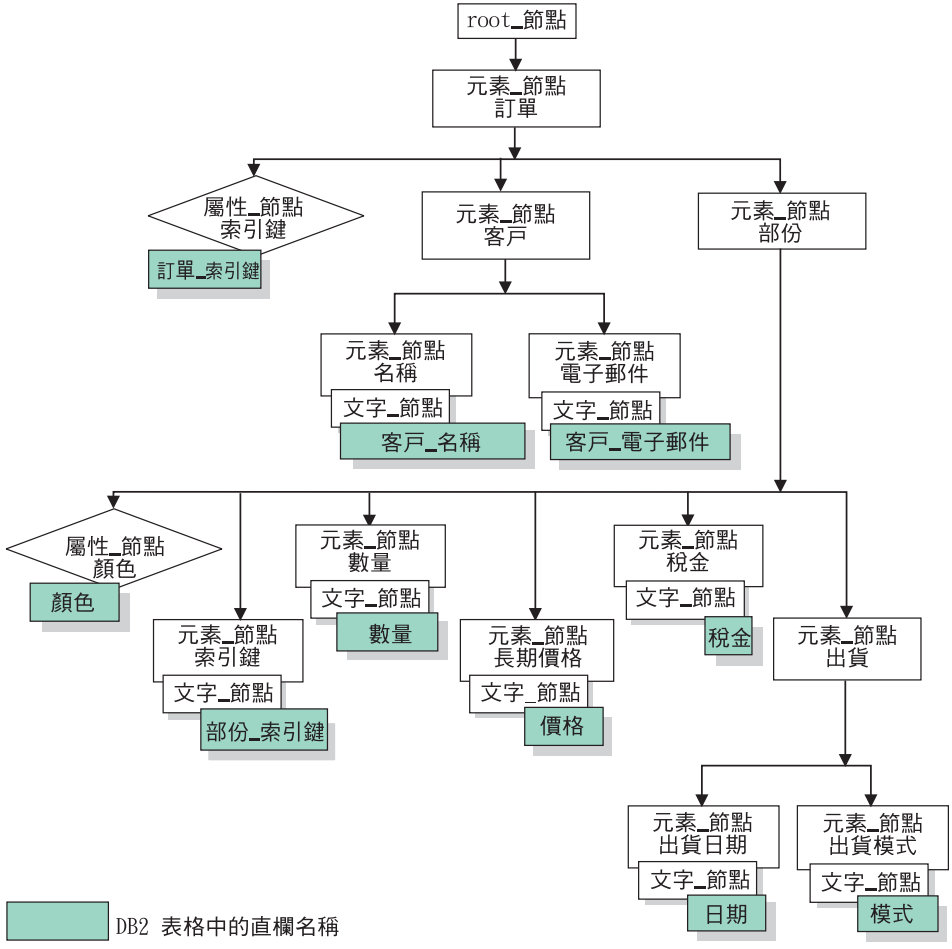


圖 11. XML 文件結構對映至關聯式表格直欄

當編製或分解位於多重關聯式表格中的 XML 文件時，XML Extender 使用對映方法。 XML Extender 提供精靈來協助您建立 DAD 檔。不過，建立 DAD 檔之前，您必須瞭解 XML 資料如何對映至 XML 集合。

對映方法的類型: 對映方法指定於 DAD 檔的 <Xcollection> 元素中。 XML Extender 提供兩類型對映方法：*SQL 對映*與*關聯式資料庫 (RDB_node) 對映*。這兩個方法使用 XSLT 模型來定義 XML 文件的階層。

SQL 對映

容許透過單一 SQL 陳述式及 *XSLT 資料模型*，簡單又直接地從關聯式資料對映至 XML 文件。 SQL 對映使用於組合；它不用於分解。 SQL 對映使用 DAD 檔中的 SQL_stmt 元素定義。SQL_stmt 的內容爲有效的 SQL

陳述式。 SQL_stmt 將 SELECT 子句中的直欄對映至 XML 文件中使用的 XML 元素或屬性。當爲編製 XML 文件定義時，可使用 SQL 陳述式的 SELECT 子句中之直欄名稱，來定義 *attribute_node* 的值或 *text_node* 的內容。FROM 子句可定義含有資料的表格； WHERE 子句可指定*合併*與搜尋*條件*。

SQL 對映提供 DB2 使用者利用 SQL 來對映資料的能力。當使用 SQL 對映時，您必須能夠合併一個 SELECT 陳述式中的所有表格來產生查詢。如果一個 SQL 陳述式不夠，可考慮使用 RDB_node 對映。若要同時連結所有表格，建議在這些表格之間要有*主要鍵*與*外來鍵*關係。

RDB_node 對映

定義 XML 元素內容或 XML 屬性值的位置，讓 XML Extender 可決定儲存或擷取 XML資料的位置。

RDB_node 包含用於表格、選用性直欄及選用性條件的一或多個節點定義。這些表格與直欄用來定義 XML 資料儲存在資料庫的方法。 該條件指定選取 XML 資料的準則，或合併 XML 集合表格的方法。

若要定義對映方法，可以建立一個含有 <Xcollection> 元素的 DAD。 第56頁的圖 12顯示一段含 XML 集合 SQL 對映的範例 DAD 檔，XML 集合 SQL 對映可將三個關聯式表格中的資料組合成一組 XML 文件。

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dtd\dad.dtd">
<DAD>
  <dtdid>c:\dxx\samples\dad\getstart.dtd</dtdid>
   <validation>YES</validation>
  <Xcollection>
    <SQL_stmt>
       SELECT o.order_key, customer, p.part_key, quantity, price, tax, date,
              mode, comment
                FROM order_tab o, part_tab p,
               table(select substr(char(timestamp(generate_unique())),
               as ship_id, date, mode, from ship_tab) as s
        WHERE p.price > 2500.00 and s.date > "1996-06-01" AND
              p.order_key = o.order_key and s.part_key = p.part_key
</SQL_stmt>
      <prolog>?xml version="1.0"?</prolog>
    <doctype>!DOCTYPE DAD SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
      <root_node>
   <element_node name="Order">
  <attribute_node name="key">
<column_name="order_key"/>
  </attribute_node>
  <element_node name="Customer">
  <text_node>
  <column name="customer"/>
  </text_node>
  <element_node>

...

  </element_node><!-end Part->
   </element_node><!-end Order->
      </root_node>
  </Xcollection>
</DAD>
```

圖 12. SQL 對映方法

XML Extender 提供數個儲存程序來管理 XML 集合中的資料。這些儲存程序支援兩種對映類型，但要求 DAD 檔遵循『對映方法需求』中描述的規則。

對映方法需求: 下列各節將說明各種 XML 集合對映方法的需求。

使用 SQL 對映的需求

在這個對映方法中，您必須在 DAD <Xcollection> 元素中指定 SQL_stmt 元素。 SQL_stmt 應包含單一 SQL 陳述式，此 SQL 陳述式可將多重關聯式表格與查詢*述詞*合併。 此外，需要下列子句：

- **SELECT 子句**
  - 請確定直欄名稱是唯一的。如果兩個表格有相同的直欄名稱，請使用 AS 關鍵字來分別建立它們的別名。

– 同時組合相同表格的直欄，以及使用關聯式表格的邏輯階層性層次。這表示組合表格是根據 (表格對映到 XML 文件階層式結構的) 重要性層次。在 SELECT 子句中，高階表格的直欄應比低階表格的直欄先處理。下列範例示範表格之間的階層性關係：

```
SELECT o.order_key, customer, p.part_key, quantity, price, tax,
       ship_id, date, mode
```

在這個範例中，表格 ORDER_TAB 的 order_key 及客戶，具有最高關聯式層次，因爲它們位在 XML 文件較高的階層性樹狀結構上。表格 SHIP_TAB 中的 ship_id、date 及 mode，位於最低關聯式層次上。

– 使用單一直欄候選鍵來起始每一個層次。如果表格中沒有這樣的鍵值，查詢應使用表格表示式及內建函數 generate_unique() 來對該表格建立一個鍵值。上述範例中，o.order_key 是 ORDER_TAB 的主要鍵，而 part_key 是 PART_TAB 的主要鍵。 它們會出現在要選取的所屬直欄群組開端。因爲 SHIP_TAB 表格沒有主要鍵，所以必須建立一個，在這種情形下是 ship_id。 在 SHIP_TAB 表格群組中它是第一個直欄。使用 FROM 子句來產生主要鍵直欄，如下列範例所示。

- **FROM 子句**
  – 使用表格表示式及內建函數 generate_unique()，對沒有主要單一鍵的表格產生單一鍵。例如：

```
FROM order_tab as o, part_tab as p,
        table(select substr(char(timestamp(generate_unique())),16) as
        ship_id, date, mode from ship_tab) as s
```

在這個範例中，generate_unique() 函數被強制轉型成 CHAR 資料類型 TIMESTAMP，並且提供了別名 ship_id。

  – 若需要區分直欄，請使用別名。例如，您可以使用 o 來代表 ORDER_TAB，使用 p 來代表 PART_TAB，以及使用 s 來代表 SHIP_TAB。

- **WHERE 子句**
  – 指定主要鍵與外來鍵關係，作爲同時連結集合中表格的結合條件。例如：

```
WHERE p.price > 2500.00 AND s.date > "1996-06-01" AND
      p.order_key = o.order_key AND s.part_key = p.part_key
```

  – 在述詞中指定其它任何搜尋條件。可使用任何驗證述詞。

- **ORDER BY 子句**
  – 在 SQL_stmt 尾端定義 ORDER BY 子句。

- 確定此直欄名稱符合 SELECT 子句中的直欄名稱。
- 指定資料庫的實體關係設計中，唯一識別實體的識別字或直欄名稱。可利用表格表示式及內建函數 generate_unique 或使用者定義的函數 (UDF) 來建立識別字。
- 維護實體階層的由上往下次序。ORDER BY 子句中指定的直欄，必須是每一個實體列示的第一個直欄。保持順序以確保建立的 XML 文件不會有不正確的複製。
- 不要根據綱目或表格名稱定義 ORDER BY 中的直欄。

雖然 SQL_stmt 具有前置需求，但其仍具備強大功能；因爲只要述詞中的表示式使用表格中的直欄，您就可以在 WHERE 子句中指定任何述詞。

使用 RDB_node 對映的需求

使用這個對映方法時，請不要在 DAD 檔的 <Xcollection> 元素中使用元素 SQL_stmt。 您可以在 *element_node* 以及每一個 attribute_node 與 text_node 的每一個頂端節點中，使用 RDB_node 元素。

- **用於頂端 element_node 的 RDB_node**

  DAD 檔中*頂端 element_node* 代表 XML 文件的 Root 元素。請指定頂端 element_node 的 RDB_node，如下所示：

  - 指定與 XML 文件相關的全部表格。例如，下列對映在 element_node <Order> (它是頂端 element_node) 的 RDB_node 中指定三個表格：

```
<element_node name="Order">
        <RDB_node>
                    <table name="order_tab"/>
                    <table name="part_tab"/>
                       <table name="ship_tab"/>
                       <condition>
                order_tab.order_key = part_tab.order_key AND
                    part_tab.part_key = ship_tab.part_key
                       </condition>
                    </RDB_node>
```

  - 如果您正在分解、或正在啓用由 DAD 檔指定的 XML 集合，那麼您必須爲每一個表格指定其主要鍵。主要鍵可以由單一直欄或多重直欄組成，稱爲組合鍵。 可藉由新增屬性鍵至 RDB_node 的表格元素來指定主要鍵。 提供組合鍵時，鍵值屬性是由空格區隔的鍵值直欄名稱來指定。 例如：

```
<table name="part_tab" key="part_key, price"/>
```

    編製文件時會忽略指定用來分解的資訊。

– 使用 orderBy 屬性，將包含多次出現的元素或屬性之 XML 文件，重新編製回它們的原始結構。 此屬性可讓您指定直欄的名稱，該直欄是用來保留文件次序的鍵值。 orderBy 屬性是 DAD 檔中表格元素的一部份，且爲可選用的屬性。

您必須明確拼出表格名稱與直欄名稱。

- **用於每一個 attribute_node 及 text_node 的 RDB_node**

  在這個對映方法中，資料會常駐於每一個 element_node 的 attribute_node 及 text_node 中。因此，XML Extender 必須知道尋找資料所需的資料庫位置。您必須對每一個 attribute_node 及 text_node 指定 RDB_node，根據表格、直欄及查詢條件通知儲存程序取得資料。您必須指定表格與直欄值；條件值爲選用的。

  – 指定含有直欄資料的表格名稱。表格名稱必須包含於頂端 element_node 的 RDB_node 中。 在這個範例中，針對元素 <Price> 的 text_node，表格指定爲 PART_TAB。

  ```
  <element_node name="Price">
                    <text_node>
          <RDB_node>
                       <table name="part_tab"/>
                          <column name="price"/>
                          <condition>
                             price > 2500.00
                          </condition>
                       </RDB_node>
                    </text_node>
      </element_node>
  ```

  – 指定含有元素文字的資料之直欄名稱。前一個範例中，直欄被指定爲 PRICE。

  – 若要使用查詢條件產生 XML 文件，請指定條件。在上述範例中，條件被指定爲 `price > 2500.00`。 僅符合條件的資料會出現在建立的 XML 文件中。 此條件必須是有效的 WHERE 子句。

  – 如果您正在分解文件、或正在啓用由 DAD 檔指定的 XML 集合，那麼您必須爲每一個 attribute_node 及 text_node 指定其直欄類型。這可確保啓用 XML 集合期間建立新表格時，每一個直欄都有正確的資料類型。 直欄類型可藉由新增屬性類型至直欄元素來指定。 例如，

  ```
  <column name="order_key" type="integer"/>
  ```

  編製文件時會忽略指定用來分解的資訊。

透過 RDB_node 對映方式，您不必提供 SQL 陳述式。 不過，要在 RDB_node 元素中放置複數查詢條件，可能會更困難。 例如，使用 *union*、表示式或作業，其功能性比使用 SQL 對 XML 方法稍微差一點。

分解表格大小需求

分解使用 RDB_node 對映，透過將元素值和屬性值解壓縮到表格列來指定 XML 文件如何分解成 DB2 表格。每一個 XML 文件中的值皆儲存在一或多個 DB2 表格。每一個表格最多儲存每一份文件中分解出來的 1024 列。

例如，如果 XML 文件分解成 5 個表格，則每一個表格最多可儲存該文件的 1024 列。如果表格含有多重文件的列，則每一份文件最多 1024 列。如果表格有 20 份文件，則最多 20,480 列，亦即每一份文件 1024 列。

使用重複出現的元素 (在 XML 結構中位置路徑出現一次以上的元素) 會影響列數。例如，一份文件含有一個元素 <Part> 出現 20 次，則文件可能在一個表格中分解成 20 列。使用出現多次的元素時，請考慮這個表格大小限制。

第4章 管理 XML 資料

XML Extender 管理作業包括 XML 啓用您的資料庫和對表格直欄，以及將 XML 資料對映至 DB2 關聯式結構。XML Extender 提供數個管理工具，您可以自行開發管理作業的應用程式或僅使用精靈來執行管理作業。您可使用下列工具來完成 XML Extender 的管理作業：

- XML Extender 管理精靈
- **dxxadm** 指令
- XML Extender 管理儲存程序

本章說明管理精靈和 **dxxadm** 指令相關的管理作業。管理儲存程序說明於第180頁的『管理儲存程序』。

爲了完成本章的作業，您應該熟悉第42頁的『管理規劃』中說明的概念和規劃作業。

下列各節說明 XML Extender 管理作業：

1. 『啓動管理精靈』
2. 第63頁的『啓用 XML 資料庫』
3. 第64頁的『將 DTD 存入 DTD 儲存庫中』
4. 第66頁的『定義 XML 直欄或集合』
5. 第66頁的『使用 XML 直欄』
6. 第76頁的『使用 XML 集合』

啓動管理精靈

本節包含關於設置和呼叫 XML Extender 管理精靈的資訊。

設置管理精靈

確定您已根據作業系統的 README 檔案，執行管理精靈的安裝與架構步驟。包括確定您已執行 bind 陳述式，且在 CLASSPATH 陳述式中包含必要的軟體。

呼叫管理精靈

請遵循下列步驟，呼叫 XML Extender 管理精靈。

1. 呼叫精靈。

對於 Windows NT：

在桌面上按兩下 XML Extender 管理精靈圖示。

關於 AIX、Sun Solaris 及 Linux：

執行 `dxxadmin` 檔案。

管理精靈的「登入」視窗開啓。

當您呼叫 XML Extender 管理精靈時，即顯示「登入」視窗。請登入您在使用 XML 資料時所要採用的資料庫。XML Extender 連接至現行的案例。

2. 在**位址**欄位中，輸入您所連接之 IBM DB2 UDB 資料來源的完整 JDBC 位址。位址的語法如下：

   對於網路架構：

   ```
   jdbc:db2://server_name:port_number/database_name
   ```

   其中：

   server_name
   XML Extender 所在的伺服器名稱。

   port_number
   用來連接伺服器的埠號。若要決定埠號，請在伺服器機器的 DB2 指令行輸入下列指令：

   ```
   db2jstrt port#
   ```

   database_name
   您要連接及儲存 XML 文件的資料庫。

   例如，

   ```
   jdbc:db2://host1.ibm.com:8080/sales_db
   ```

   對於獨立式架構：

   ```
   jdbc:db2:database_name
   ```

   其中：

   database_name
   您要連接及儲存 XML 文件的資料庫。

   例如，

   ```
   jdbc:db2:sales_db
   ```

3. 在**使用者 ID** 和**通行碼**欄位中，輸入或驗證您所連接之資料庫的 DB2 使用者 ID 和通行碼。

4. 在 **JDBC 驅動程式**欄位中，使用下列值，驗證指定之位址的 JDBC 驅動程式名稱：

   對於網路架構：

   ```
   COM.ibm.db2.jdbc.net.DB2DRIVER
   ```

   對於獨立式架構：

   ```
   COM.ibm.db2.jdbc.app.DB2DRIVER
   ```

5. 按一下**完成**以連接至精靈並進階至「發射台」視窗。

「發射台」視窗提供對五個管理精靈的存取權限。使用精靈，請：

- 啓用資料庫
- 新增 DTD 至 DTD 儲存庫
- 使用 DAD 檔案：
  - XML 直欄
  - XML 集合
- 使用 XML 直欄
- 使用 XML 集合

啓用 XML 資料庫

若要從含有 XML Extender DB2 中儲存或擷取 XML 文件，您必須啓用 XML 資料庫。 XML Extender 啓用您連接的資料庫，使用現行案例。

爲 XML 啓用資料庫時，XML Extender 會執行下列動作：

- 建立所有使用者定義類型 (UDT) 和使用者定義函數 (UDF)
- 建立控制表格，並在控制表格中移入 XML Extender 需要的 meta 資料
- 建立 db2xml 綱目，指定必需的專用權

  XML 函數的完整名稱是 *schema-name.function-name*，其中 *schema-name* 是一個識別字，代表 SQL *物件*的邏輯群組。 您可依照喜好，對 UDF 或 UDT使用完整名稱。您亦可在參照 UDF 或 UDT 時省略綱目名稱；此時，DB2 使用函數路徑來決定您要的函數或資料類型。

使用管理精靈

使用下列步驟，啓用資料庫來處理 XML 資料：

1. 設定和啓動管理精靈。詳細資訊，請參閱第61頁的『啓動管理精靈』。
2. 從「發射台」視窗中，按一下**啓用資料庫** 以啓用現行資料庫。

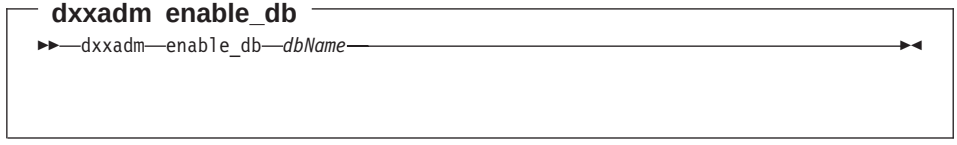
   若資料庫已啓用，則只能選取**停用資料庫**。

啓用資料庫後，會返回「發射台」視窗。

從 DB2 指令 Shell

從指令行輸入 **dxxadm**，指定要啓用的資料庫。

語法：



參數：

dbName
要啓用的資料庫名稱。

範例： 啓用一個現存的資料庫，稱爲 SALES_DB。

```
dxxadm enable_db SALES_DB
```

將 DTD 存入 DTD 儲存庫中

您可使用 DTD 來驗證 XML 直欄或 XML 集合中的 XML 資料。 DTD 可驗證 XML 直欄及定義 DAD 檔案；DAD 檔案使用於 XML 結構式搜尋以及集合的組合和分解。

全部 DTD 皆儲存於 DTD 儲存庫中，是一個稱爲 DTD_REF 的 DB2 表格。此表格有一個綱目名稱 db2xml。DTD_REF 表格中的每一個 DTD 有一個唯一的 ID。當您啓用資料庫供 XML 操作時，XML Extender 建立 DTD_REF 表格。

關於使用 DTD 的詳細資訊，請參閱第44頁的『規劃 XML 直欄』和第50頁的『規劃 XML 集合』。

您可從 DB2 指令 Shell 中插入 DTD，或使用管理精靈來插入 DTD。

使用管理精靈

使用下列步驟來插入 DTD：

1. 設定和啓動管理精靈。詳細資訊，請參閱第61頁的『啓動管理精靈』。
2. 從「發射台」視窗中按一下**匯入 DTD**，將現存的 DTD 檔案匯入現行資料庫的 DTD 儲存庫內。即顯示「匯入 DTD」視窗。

3. 將 DTD 檔名鍵入 **DTD 檔名**欄位中，或按一下 **...** 以瀏覽現存的 DTD 檔。
4. 將 DTD ID 鍵入 **DTD ID** 欄位中。

   DTD ID 是 DTD 的識別字，可以是指定本端系統上 DTD 位置的路徑。DTDID 必須符合 DAD 檔案中指定的 <DTDID>元素值。
5. 另外，可選擇在**作者**欄位中鍵入 DTD 作者的名稱。

   若在 DTD 中指定作者名稱，則 XML Extender 會自動顯示作者名稱。
6. 按一下**完成**，將 DTD 插入 DTD 儲存庫表格 (DB2XML.DTD_REF) 並返回「發射台」。

從 DB2 指令 Shell

使用表7中的綱目，對 DTD_REF 表格發出 SQL INSERT 陳述式：

表 7. DTD_REF DTD 表格的綱目

直欄名稱	資料類型	說明
DTDID	VARCHAR(128)	主要鍵 (唯一的且不是 NULL)。驗證時，主要鍵是用來識別 DTD，它必須與每一個 XML 文件中 DOCTYPE 行上的 SYSTEM ID 相同。在 DAD 檔案中指定主要鍵時，DAD 檔案必須遵循 DTD 所定義的綱目。
CONTENT	XMLCLOB	DTD 的內容。
USAGE_COUNT	INTEGER	資料庫中使用此 DTD 來定義 DAD 的 XML 直欄和 XML 集合的數目。
AUTHOR	VARCHAR(128)	DTD 的作者，使用者可選擇輸入的資訊。
CREATOR	VARCHAR(128)	執行第一個插入動作的使用者 ID。
UPDATOR	VARCHAR(128)	執行前次更新的使用者 ID。

例如：

```
DB2 INSERT into db2xml.dtd_ref values('c:\dxx\samples\dtd\getstart.dtd',
    db2xml.XMLClobFromFile('c:\dxx\samples\dtd\getstart.dtd'), 0, 'user1',
    'user1', 'user1')
```

XML 集合的重要事項： DTDID 是一個路徑，指定本端系統上的 DTD 位置。DTDID 必須符合 DAD 檔案中指定的 <DTDID>元素值。

定義 XML 直欄或集合

下列各節說明如何設定和定義用於 XML 直欄或集合的資料庫，以及如何準備必需的資料對映綱目。

這幾節包括：

- 『使用 XML 直欄』
- 第76頁的『使用 XML 集合』

使用 XML 直欄

若要設定 XML 直欄，您必須定義 DAD 檔案來存取 XML 資料，並啓用直欄來處理 XML 表格中的 XML 資料。建立 DAD 時的一個重要觀念爲瞭解位置路徑語法，因爲它是用來將您要索引的元素及屬性值對映至 DB2 表格內的。請參閱第48頁的『位置路徑』以學習更多關於位置路徑及其語法的資訊。

建立或編輯 DAD 檔案

指定 DAD 檔案時，您要爲需要搜尋的資料定義屬性和索引元素。 XML Extender 使用此資訊來建立輔助表格，讓您能夠編排資料的索引來快速擷取資料。關於建立 DAD 檔案的規劃問題，請參閱第50頁的『DAD 檔』。

開始之前

- 瞭解 XML 資料的階層式結構，才能夠定義索引元素和屬性，達到檢索和快速搜尋的目標。
- 準備 XML 文件的 DTD 並將它插入 DTD_REF 表格中。驗證作業需要此步驟。

使用管理精靈

使用下列步驟來建立 DAD 檔案：

1. 設定和啓動管理精靈。詳細資訊，請參閱第61頁的『啓動管理精靈』。
2. 在「發射台」視窗中按一下**使用 DAD 檔案**，編輯或建立 XML DAD 檔案。即開啓「指定 DAD 檔」視窗。
3. 選擇編輯現存的 DAD 檔案或建立新的 DAD 檔案。
   - **編輯現存的 DAD：**
     a. 按一下 **...** 以瀏覽下拉功能表中的現存 DAD 檔，或將 DAD 檔名鍵入**檔名**欄位中。
     b. 驗證精靈是否可辨識指定的 DAD 檔案。
        – 若精靈可辨識指定的 DAD 檔案，則可選取**下一步**，**類型**欄位中會顯示 XML 直欄。

– 若精靈無法辨識指定的 DAD 檔案，則無法選取**下一步**。在**檔名**欄位中重新鍵入 DAD 檔案，或按一下**開啓**，再次瀏覽現存的 DAD 檔案。繼續進行，直到可選取**下一步**爲止。

c. 按一下**下一步**。

- **建立新的 DAD：**

a. 將**檔名**欄位留白。

b. 從**類型**功能表中，按一下 **XML 直欄**。

c. 按一下**下一步**。

4. 從「選取驗證」視窗中，選擇是否使用 DTD 來驗證 XML 文件。
   - 若要驗證：

     a. 按一下**使用 DTD 驗證 XML 文件**。

     b. 從 **DTD ID** 功能表中，選取要用來驗證的 DTD。

     若您沒有爲資料庫匯入任何 DTD 到 DTD 儲存庫內，您將無法驗證 XML 文件。
   - 按一下**「不」使用 DTD 驗證 XML 文件**以繼續而不驗證您的 XML 文件。
5. 按一下**下一步**。
6. 從「輔助表格」視窗中，選擇新增新的輔助表格、編輯現存的輔助表格、或移除現存的輔助表格。
   - **若要新建一個輔助表格或輔助表格直欄：**

     若要新建一個輔助表格，請於表格中定義直欄。 在輔助表格的每一個直欄中完成下列步驟。

     a. 在「輔助表格」視窗中，完成**明細**方框的欄位。

        1) **表格名稱：**鍵入含有直欄的表格名稱。 例如：

           ```
           ORDER_SIDE_TAB
           ```

        2) **直欄名稱：**鍵入直欄的名稱。例如：

           ```
           CUSTOMER_NAME
           ```

        3) **類型：**從功能表中選取直欄的類型。 例如：

           ```
           XMLVARCHAR
           ```

        4) **長度 (僅限 VARCHAR 類型)**：鍵入最大 VARCHAR 字元數。 例如：

           ```
           30
           ```

        5) **路徑：**鍵入元素或屬性的位置路徑。例如：

           ```
           /ORDER/CUSTOMER/NAME
           ```

關於位置路徑語法，請參閱 第48頁的『位置路徑』。

6) **出現多次**：從功能表中選取**否**或**是**。

指出此元素或屬性的位置路徑是否可在一個文件中使用多次。

重要事項 若要爲直欄指定多次出現的項目，在包含直欄的輔助表格中只能指定一個直欄。

b. 按一下**新增**以新增直欄。

c. 繼續新增、編輯、或爲輔助表格除去直欄， 或按一下**下一步**。

- **若要編輯現存的輔助表格直欄：**

  您可以藉由變更現存直欄的定義來更新輔助表格。

  a. 按一下您要編輯的輔助表格及直欄名稱。

  b. 編輯**明細**方框的欄位。

  c. 按一下**變更**以儲存變更。

  d. 繼續新增、編輯、或爲每一個輔助表格除去直欄，或按一下**下一步**。

- **若要除去現存的輔助表格直欄：**

  a. 按一下您要除去的輔助表格及直欄。

  b. 按一下**移除**。

  c. 繼續新增、編輯、或移除輔助表格直欄，或按一下**下一步**。

- **移除現存的輔助表格：**

  若要除去整個輔助表格，請刪除表格中的每一個直欄。

  a. 按一下您要除去之表格中的每一個輔助表格直欄。

  b. 按一下**移除**。

  c. 繼續新增、編輯、或除去輔助表格直欄，或按一下**下一步**。

7. 在「指定 DAD」視窗的**檔名**欄位中，鍵入已修改之 DAD 檔案的輸出檔名。
8. 按一下**完成**以儲存 DAD 檔並返回「發射台」視窗。

從 DB2 指令 Shell

DAD 檔案是一個可用任何文字編輯器建立的 XML 檔案。

使用下列步驟來建立 DAD 檔案：

1. 開啓一個文字編輯器。
2. 使用下列語法建立 DAD 檔案表頭：

```
<?xml version="1.0"?>
<!D CTYPE DAD SYSTEM "path\dtd\dad.dtd"> --> DAD 檔 的 DTD 路徑和檔案名稱
```

3. 插入 <DAD></DAD> 標示。

4. 在 <DAD> 標示中，可選用性地指定 DTDID 識別字，將 DAD 檔案與 XML 文件 DTD 結合起來進行驗證：

```
<dtdid>path\dtd_name.dtd</dtdid> --> the path and file name of the DTD
                                     路徑和檔案名稱
```

   DTD ID 是驗證的必要條件，且必須符合將 DTD 插入 DTD 參照表 (db2xml.DTD_REF) 時所使用的 DTD ID 值。

5. 指定是否要驗證 (亦即，使用 DTD 來確定 XML 文件是有效的 XML 文件)。例如：

```
<validation>YES</validation>   --> 指定 YES 或 NO
```

   若您指定 YES，您在上一個步驟中必須已指定 DTDID，且在 DTD_REF 表格中插入 DTD。

6. 使用 <Xcolumn> 元素，定義 XML 直欄的存取和儲存方法。

```
<Xcolumn>
  </Xcolumn>
```

7. 定義每一個輔助表格，以及要製成索引來進行結構式搜尋的重要元素和屬性。對每一個表格執行下列步驟。下列步驟採用第233頁的『DAD 檔：XML 直欄』中的範例 DAD 檔案：

   a. 插入 <TABLE></TABLE> 標示和名稱屬性。

```
<table name="order_tab">
     </table>
```

   b. 在 <TABLE> 標示後面，爲表格中的每一個直欄插入一個 <COLUMN> 標示及其屬性。

      - ***name***：直欄的名稱
      - ***type***：直欄的類型
      - ***path***：元素或屬性的位置路徑。關於位置路徑語法，請參閱 第48頁的『位置路徑』。
      - ***multi_occurrence***：指出此元素或屬性是否可在一個文件中使用多次。

```
<table ...>
         <column name="order_key"
          type="integer"
            path="/Order/@key"
          multi_occurrence="NO"/>
         <column name="customer"
          type="varchar(50)"
          path="/Order/Customer/Name"
          multi_occurrence="NO"/>
     </table>
```

8. 在最後一個直欄定義後面，請記得加上 </TABLE> 結束標示。

9. 在最後一個 </TABLE> 標示後面，請記得加上 </Xcolumn> 結束標示。
10. 在 </Xcolumn> 標示後面，請記得加上 </DAD> 結束標示。

建立或變更 XML 表格

若要在表格中儲存完整的 XML 直欄，您必須建立或變更表格，讓表格包含 XML 使用者定義類型 (UDT) 的直欄。這種表格稱爲 *XML 表格*，亦即可包含 XML 文件的表格。 表格可以是變更過的表格或新的表格。當表格包含 XML 類型的直欄時，您可以啓用 XML 直欄。

您可以使用管理精靈或 DB2 指令 Shell 來變更含有 XML 類型直欄的現存表格。

使用管理精靈

1. 設定和啓動管理精靈。詳細資訊，請參閱第61頁的『啓動管理精靈』。
2. 在「發射台」視窗中，按一下**使用 XML 直欄**。即開啓「選取作業」視窗。
3. 按一下**新增 XML 直欄**。 即開啓「新增 XML 直欄」視窗。
4. 從**表格名稱**下拉功能表中選取表格名稱，或鍵入所要變更之表格的名稱。例如：

   ```
   SALES_DB
   ```

5. 在**直欄名稱**欄位上鍵入直欄的名稱，以便新增到表格中。 例如：

   ```
   ORDER
   ```

6. 從**直欄類型**下拉功能表中選取直欄的 UDT。例如：

   ```
   XMLVARCHAR
   ```

7. 按一下**完成**，新增 XML 類型的直欄。

從 DB2 指令 Shell

在 CREATE TABLE 或 ALTER TABLE 陳述式的直欄子句中，建立或變更含有 XML 類型直欄的表格。

範例： 在銷售應用程式中，您可能想在 SALES_TAB 應用程式表格的 ORDER 直欄中，儲存 XML 格式的行項目訂單。 此表格亦包含 INVOICE_NUM 和 SALES_PERSON 直欄。 因爲只是小訂單，所以使用 XMLVARCHAR 類型來儲存。 主要鍵是 INVOICE_NUM。下列 CREATE TABLE 陳述式建立含有 XML 類型直欄的表格：

```
CREATE TABLE sales_tab(
             invoice_num  char(6) NOT NULL PRIMARY KEY,
             sales_person     varchar(20),
             order            XMLVarchar);
```

啓用 XML 直欄

若要在 DB2 資料庫中儲存 XML 文件，您必須對 XML 啓用直欄。啓用直欄後，即可開始編排索引，以便進行快速搜尋。您可使用 XML Extender 管理精靈或 DB2 指令 Shell 來啓用直欄。直欄必須是 XML 類型。

當 XML Extender 啓用 XML 直欄時，動作如下：

- 讀取 DAD 檔案，並可選擇：
  - 根據 DAD 檔案的 DTD 來驗證 DAD 檔案。
  - 若有指定，可從 DTD_REF 表格中擷取 DTDID。
  - 建立輔助表格來處理 XML 直欄的索引。
  - 準備直欄來包含 XML 資料。
- 若已定義 XML 表格或輔助表格，可選用性地建立其*預設概略表*。
- 若尚未指定 ROOT ID 值，則指定一個。

在啓用 XML 直欄之後，您可以

- 在輔助表格中建立索引
- 在 XML 直欄中插入 XML 文件
- 在 XML 直欄中查詢、更新或搜尋 XML 文件。

開始之前

建立或變更含有 XML UDT 直欄的 DB2 表格來建立 XML 表格。

使用管理精靈

使用下列步驟來啓用 XML 直欄：

1. 設定和啓動管理精靈。詳細資訊，請參閱第61頁的『啓動管理精靈』。
2. 在「發射台」視窗中按一下**使用 XML 直欄**，以檢視 XML Extender 直欄的相關作業。出現「選取作業」視窗。
3. 按一下**啓用直欄**，然後按一下**下一步**以啓用資料庫中的現存表格直欄。
4. 從**表格名稱**欄位中選取含有 XML 直欄的表格。 例如：

   ```
   SALES_TAB
   ```

5. 從**直欄名稱**欄位中選取要啓用的直欄。 例如：

   ```
   ORDER
   ```

   直欄必須存在並爲 XML 類型。

6. 將 DAD 路徑和檔案名稱鍵入 **DAD 檔名**欄位中，或按一下 **...** 以瀏覽現存的 DAD 檔。 例如：

   ```
   c:\dxx\samples\dad\getstart.dad
   ```

7. 可選用性地在**表格空間**欄位中，鍵入現存的表格空間的名稱。

   表格空間包含 XML Extender 所建立的輔助表格。 若指定表格空間，則輔助表格會建立於指定的表格空間中。若未指定表格空間，則輔助表格會建立於預設的表格空間中。

8. 可選用性地在**預設概略表**欄位中鍵入預設概略表的名稱。

   指定時，在啓用直欄的同時，將自動建立預設概略表，並合併 XML 表格及所有相關的輔助表格。

9. 可選用性地在 **Root ID** 欄位中，鍵入應用程式表格中主要鍵的直欄名稱。建議執行此動作。

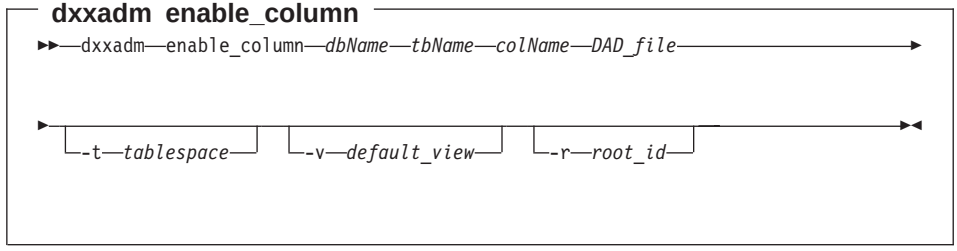
   XML Extender 使用 ROOT ID 值做爲唯一的識別字，用來結合所有相關的輔助表格和應用程式表格。若未指定，則 XML Extender 將 DXXROOT_ID 直欄新增到應用程式表格中，並且產生一個識別字。

10. 按一下**完成**以啓用 XML 直欄、建立輔助表格並返回「發射台」視窗。
    - 若順利啓用直欄，則顯示**啓用直欄成功**訊息。
    - 若未順利啓用直欄，則顯示錯誤方框。更正輸入欄位值直到已順利啓用直欄爲止。

從 DB2 指令 Shell

若要啓用 XML 直欄，請輸入下列指令：

語法：



參數：

dbName
資料庫的名稱。

tbName
表格的名稱，表格含有要啓用的直欄。

colName
要啓用的 XML 直欄名稱。

DAD_file
檔案的名稱，檔案含有文件存取定義 (DAD)。

tablespace
先前建立的表格空間，表格空間含有 XML Extender 所建立的輔助表格。如果沒有指定，則會使用預設的表格空間。

default_view
可選用的。XML Extender 所建立之預設概略表的名稱，用來結合應用程式表格和所有相關的輔助表格。

root_id 可選用的。應用程式表格中主要鍵的直欄名稱，這是用來關關所有輔助表格和應用程式的唯一識別字。 XML Extender 使用 *root_id* 值做爲唯一的識別字，用來結合所有相關的輔助表格和應用程式表格。 建議您指定 ROOT ID。若未指定 ROOT ID，則 XML Extender 將 DXXROOT_ID 直欄新增到應用程式表格，並且產生一個識別字。

限制： 若應用程式表格含有一個直欄名稱 DXXROOT_ID，但此直欄不含 *root_id* 的值，您必須指定 *root_id* 參數；否則會發生錯誤。

範例： 下列範例使用 DB2 指令 Shell 來啓用直欄。 DAD 檔案和 XML 文件位於第231頁的『附錄B. 範例』。

```
dxxadm enable_column SALES_DB sales_tab order getstart.dad
           -v sales_order_view -r invoice_num
```

本範例中，在表格 SALES_DB.SALES_TAB 中啓用直欄 ORDER。 DAD 檔案是 `getstart.dad`，預設概略表是 sales_order_view， ROOT ID 是 INVOICE_NUM。

在本範例中，SALES_TAB 表格包含下列綱目：

直欄名稱	INVOICE_NUM	SALES_PERSON	ORDER
資料類型	CHAR(6)	VARCHAR(20)	XMLVARCHAR

下列輔助表格係根據 DAD 規格所建立：

ORDER_SIDE_TAB:

直欄名稱	ORDER_KEY	CUSTOMER	INVOICE_NUM
資料類型	INTEGER	VARCHAR(50)	CHAR(6)
路徑表示式	`/Order/@key`	`/Order/Customer/Name`	N/A

PART_SIDE_TAB:

直欄名稱	PART_KEY	PRICE	INVOICE_NUM
資料類型	INTEGER	DOUBLE	CHAR(6)
路徑表示式	/Order/Part/@key	/Order/Part/ExtendedPrice	N/A

SHIP_SIDE_TAB:

直欄名稱	DATE	INVOICE_NUM
資料類型	DATE	CHAR(6)
路徑表示式	/Order/Part/Shipment/ShipDate	N/A

因爲 Root ID 由應用程式表格中的主要鍵 INVOICE_NUM 所指定，所以全部輔助表格皆含有相同類型的 INVOICE_NUM 直欄。啓用直欄之後，INVOICE_NUM 的值即插入輔助表格中。若在啓用 XML 直欄 ORDER 時指定 *default_view* 參數，則會建立預設概略表 sales_order_view。此概略表使用下列陳述式來合併上述表格：

```
CREATE VIEW sales_order_view(invoice_num, sales_person, order,
                             order_key, customer, part_key, price, date)
     AS
     SELECT sales_tab.invoice_num, sales_tab.sales_person, sales_tab.order,
            order_tab.order_key, order_tab.customer,
            part_tab.part_key, part_tab.price,
            ship_tab.date
     FROM sales_tab, order_tab, part_tab, ship_tab
     WHERE sales_tab.invoice_num = order_tab.invoice_num
               AND sales_tab.invoice_num = part_tab.invoice_num
               AND sales_tab.invoice_num = ship_tab.invoice_num
```

若在 **enable_column** 指令中指定表格空間，則輔助表格會建立於指定的表格空間中。 若未指定表格空間，則在預設的表格空間中建立輔助表格。

處理輔助表格索引

在您啓用 XML 直欄並建立輔助表格之後，即可以處理輔助表格索引。輔助表格包含當您建立 DAD 檔案時所指定輸入直欄中的 XML 資料。處理這些表格索引可幫助您改進 XML 文件查詢的效能。

開始之前

- 建立爲 XML 文件結構指定輔助表格的 DAD 檔案。
- 使用 DAD 檔案啓用 XML 直欄；該檔案建立輔助表格。

DB2 CREATE INDEX 指令

使用 DB2 CREATE INDEX 指令。

範例：

下列範例在四個輔助表格中建立索引：

```
DB2 CREATE INDEX KEY_IDX
      ON ORDER_SIDE_TAB(ORDER_KEY)

DB2 CREATE INDEX CUSTOMER_IDX
      ON ORDER_SIDE_TAB(CUSTOMER)

DB2 CREATE INDEX PRICE_IDX
      ON PART_SIDE_TAB(PRICE)

DB2 CREATE INDEX DATE_IDX
      ON SHIP_SIDE_TAB(DATE)
```

停用 XML 直欄

若須爲 XML 直欄更新 DAD 檔案，或者，要刪除 XML 直欄或含有直欄的表格時，請停用一個直欄。在停用直欄之後，您可以重新啓用含有已更新的 DAD 檔案的直欄，刪除直欄，或其它作業。您可使用 XML Extender 管理精靈或 DB2 指令 Shell 來停用直欄。

當 XML Extender 啓用 XML 直欄時，動作如下：

- 刪除 XML_USAGE 表格中的直欄的項目。
- 捨棄與該直欄相關的輔助表格。

重要事項： 如果您未事先停用直欄而捨棄一個含有 XML 直欄的表格，則 XML Extender 無法捨棄任何與 XML 直欄相關的輔助表格，它可能會造成非預期的結果。

開始之前

確定停用存在於現行 DB2 資料庫中的 XML 欄。

使用管理精靈

使用下列步驟來停用 XML 直欄：

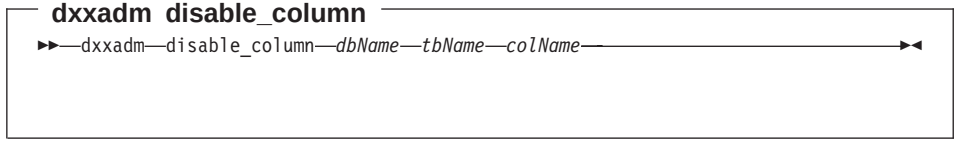
1. 設定和啓動管理精靈。詳細資訊，請參閱第61頁的『啓動管理精靈』。
2. 在「發射台」視窗中按一下**使用 XML 直欄**，以檢視 XML Extender 直欄的相關作業。出現「選取作業」視窗。
3. 按一下**停用直欄**，然後按一下**下一步**以停用資料庫中的現存表格直欄。
4. 從**表格名稱**欄位中選取含有 XML 直欄的表格。
5. 從**直欄名稱**欄位中選取要停用的直欄。

6. 按一下**完成**。
   - 若順利停用直欄，則顯示**停用直欄成功**訊息。
   - 若未順利停用直欄，則顯示錯誤方框。更正輸入欄位值直到已順利停用直欄爲止。

從 DB2 指令 Shell

若要停用 XML 直欄，請輸入下列指令：

語法：



參數：

dbName
資料庫的名稱。

tbName
表格的名稱，表格含有要停用的直欄。

colName
要停用的 XML 直欄名稱。

範例： 下列範例使用 DB2 指令 Shell 來停用直欄。 DAD 檔案和 XML 文件位於第231頁的『附錄B. 範例』。

```
dxxadm disable_column SALES_DB sales_tab order
```

本範例中，在表格 SALES_DB.SALES_TAB 中啓用直欄 ORDER。

當此直欄停用後，即捨棄輔助表格。

使用 XML 集合

設置 XML 集合需要建立一個對映方法，以及選擇性地使用結合 DB2 表格和 DAD 檔案的虛擬名稱來啓用此集合。

雖然不一定要啓用 XML 集合，但啓用後可提高效能。

為對映方法建立或編輯 DAD 檔案

使用 XML 集合時，必須建立一個 DAD 檔案。DAD 檔案定義 XML 資料和多重關聯式表格之間的關係。 XML Extender 使用 DAD 檔案的目的：

- 採用關聯式資料來組成 XML 文件
- 將 XML 文件拆解成關聯式資料

您可使用兩種方法來對映 XML 表格和 DB2 表格之間的資料， SQL 對映和 RDB_node 對映：

SQL 對映

使用 SQL 陳述式元素，對於包含 XML 資料的表格和直欄，指定 SQL 查詢。 SQL 對映僅可用於製作 XML 文件。

RDB_node 對映

使用一個 XML Extender專用元素，稱爲「關聯式資料庫」節點或 RDB_node，來指定表格、直欄、條件及 XML 資料的次序。 RDB_node 對映支援的對映比 SQL 陳述式更複雜。 RDB_node 對映可同時用於製作及分解 XML 文件。

這兩種對映方法皆使用 *XPath 資料模型*，請參閱第51頁的『DAD 檔』。

開始之前

- 對映 DB2 表格和 XML 文件之間的關係。此步驟應該包括對映 XML 文件的階層，以及指定文件中的資料如何對映至 DB2 表格。
- 如果您計劃驗證 XML 文件，請將您爲 XML 文件製作或分解的 DTD 插入 DTD 參照表 (db2xml.DTD_REF) 中。

製作與 SQL 對映的 XML 文件

當您製作 XML 文件及要使用 SQL 時，請使用 SQL 對映。

使用管理精靈: 使用下列步驟來建立使用 XML 集合 SQL 對映的 DAD 檔案

若要使用 SQL 對映建立 DAD 檔案組合：

當您製作 XML 文件，並要使用 SQL 陳述式來定義表格及直欄以衍生 XML 文件資料時，請使用 SQL 對映。

1. 設定和啓動管理精靈。詳細資訊，請參閱第61頁的『啓動管理精靈』。
2. 在「發射台」視窗中按一下**使用 DAD 檔案**。即顯示「指定 DAD」視窗。
3. 選擇編輯現存的 DAD 檔案或建立新的 DAD 檔案。

   建立新的 DAD 檔案：

   a. 將**檔名**欄位留白。

b. 從**類型**功能表中選取**XML 集合 SQL 對映**。

c. 按一下**下一步**，開啓「選取驗證」視窗。

編輯現存的 DAD 檔案：

a. 將 DAD 檔名鍵入**檔名**欄位中，或按一下 **...** 以瀏覽現存的 DAD 檔。

b. 驗證精靈是否可辨識指定的 DAD 檔案。

- 若精靈可辨識指定的 DAD 檔案，則可選取**下一步**，**類型**欄位中會顯示 XML 集合 SQL 對映。
- 若精靈無法辨識指定的 DAD 檔案，則無法選取**下一步**。重新鍵入 DAD 檔名，或按一下 **...** 以再次瀏覽現存的 DAD 檔。 更正輸入欄位值直到可選取**下一步**爲止。

c. 按一下**下一步**，開啓「選取驗證」視窗。

4. 在「選取驗證」視窗中，選擇是否使用 DTD 來驗證 XML 文件。
   - 若要驗證：
     a. 按一下**使用 DTD 驗證 XML 文件**。
     b. 從 **DTD ID** 功能表中，選取要用來驗證的 DTD。

     若您沒有爲資料庫匯入任何 DTD 到 DTD 儲存庫內，您將無法驗證 XML 文件。
   - 按一下**「不」使用 DTD 驗證 XML 文件**以繼續而不驗證您的 XML 文件。
5. 按一下**下一步**，開啓「指定文字」視窗。
6. 在**前言**欄位中鍵入前言名稱，以指定所要製作之 XML 文件的前言。

   ```
   <?xml version="1.0"?>
   ```

   若編輯現存的 DAD，**前言**欄位會自動顯示前言。
7. 在「指定文字」視窗的**文件類型**欄位中，鍵入 XML 文件的文件類型，以指向 XML 文件的 DTD。 例如：

   ```
   !DOCTYPE DAD SYSTEM "c:\dxx\samples\dtd\getstart.dtd"
   ```

   若編輯現存的 DAD，**文件類型**欄位會自動顯示文件類型。
8. 按一下**下一步**，開啓「指定 SQL 陳述式」視窗。
9. 在 **SQL 陳述式**欄位中鍵入有效的 SQL SELECT 陳述式。 例如：

   ```
   SELECT o.order_key, customer_name, customer_email, p.part_key, color, quantity,
   price, tax, ship_id, date, mode from order_tab o, part_tab p,
                    table(select substr(char(timestamp(generate_unique())),16)
     as ship_id, date, mode, part_key from ship_tab) s
         WHERE o.order_key = 1 and
               p.price > 20000 and
   ```

```
        p.order_key = o.order_key and
        s.part_key = p.part_key
ORDER BY order_key, part_key, ship_id
```

若編輯現存的 DAD，**SQL 陳述式**欄位會自動顯示 SQL 陳述式。

10. 按一下**測試 SQL**，測試 SQL 陳述式的有效性。
    - 若 SQL 陳述式有效，**範例結果**欄位會顯示範例結果。
    - 若 SQL 陳述式無效，**範例結果**欄位會顯示錯誤訊息。 錯誤訊息指示您更正 SQL SELECT 陳述式再重試。
11. 按一下**按一下**，開啓「SQL 對映」視窗。
12. 在「SQL 對映」視窗的左邊欄位中，按一下對映的來源元素或屬性。

    將 XML 文件中的元素和屬性，對映至與 DB2 資料相對應的元素和屬性節點。 這些節點提供一個從 XML 資料至 DB2 資料的路徑。

    - **若要新增 Root 節點：**
      a. 選取 **Root** 圖示。
      b. 按一下**新建元素**以定義一個新的節點。
      c. 在**明細**方框中，指定**節點類型**爲**元素**。
      d. 在**節點名稱**欄位中輸入頂端層次節點的名稱。
      e. 按一下**新增**以建立新節點。

         您已建立 Root 節點或元素，它是對映中所有其它元素或屬性節點的父節點。 現在即可在此節點新增子項元素及屬性。
    - **若要新增子項元素或屬性節點：**
      a. 按一下在欄位左邊的父節點以新增子項元素或屬性。

         如果您未選取父節點，則無法選取**新建元素**。
      b. 按一下**新建元素**。
      c. 從**明細**方框的**節點類型**功能表中，選取節點類型。

         節點類型功能表只顯示目前對映有效的節點類型：

         元素 表示定義於 XML 文件相關 DTD 中的 XML 元素。用於結合 XML 元素與 DB2 表格中的直欄。 元素節點可能有屬性節點、子節點或文字節點。 底層節點具有文字節點，並在樹狀檢視畫面中具有直欄名稱。

         屬性 表示定義於 XML 文件相關 DTD 中的 XML 屬性。用於結合 XML 屬性與 DB2 表格中的直欄。 屬性節點具有文字節點，並在樹狀檢視畫面中具有相關的直欄名稱。

文字 爲元素或屬性節點指定文字內容，其內容對映至關聯表格。文字節點在樹狀檢視畫面中具有相關的直欄名稱。

表格 爲對映至關聯表格的元素或屬性值指定表格名稱。

直欄 爲對映至關聯表格的元素或屬性值指定直欄名稱。

條件 指定直欄條件。

d. 在**明細**方框的**節點名稱**欄位中，鍵入節點名稱。 例如：

```
Order
```

e. 若指定**屬性**、**元素**或**文字**給底層元素作爲「節點類型」，從**明細**框的**直欄**欄位中，請選取直欄。 例如：

```
Customer_Name
```

限制： 無法使用管理精靈來建立新直欄。 若指定**直欄**作爲節點類型，您只可以選取 DB2 資料庫中已存在的直欄。

f. 按一下**新增**以新建節點。

以後若要修改節點，請在左邊欄位中按一下節點，在**明細**方框中做必要的修改。 按一下**變更**以更新元素。

也可以對節點新增子項元素或屬性，請高亮度標示節點並重複新增程序。

g. 繼續編輯 SQL 對映，或按一下**下一步**，開啓「指定 DAD」視窗。

- **移除節點：**
  a. 按一下左邊欄位中的一個節點。
  b. 按一下**移除**。
  c. 繼續編輯 SQL 對映，或按一下**下一步**，開啓「指定 DAD」視窗。

請注意，如果您除去底層，另一個元素將成爲底層節點，而可能需要爲它定義直欄名稱。

13. 在「指定 DAD」視窗的**檔名**欄位中，鍵入已修改之 DAD 檔案的輸出檔名。
14. 按一下**完成**，以返回「發射台」視窗。

從 DB2 指令 Shell: 當您製作 XML 文件及要使用 SQL 時，請使用 SQL 對映表示法。

DAD 檔案是一個可用任何文字編輯器建立的 XML 檔案。下列步驟顯示第232頁的『文件存取定義檔』範例附錄中的片斷。完整的資訊和內容，請參閱這些範例。

1. 開啓一個文字編輯器。
2. 建立 DAD 標頭：

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "path\dad.dtd"> --> DAD 的 DTD 路徑和檔案名稱
```

3. 插入 <DAD></DAD> 標示。
4. 在 <DAD> 標示後面，指定 DTD ID 用來結合 DAD 檔案與 XML 文件 DTD。

   ```
   <dtdid>path\dtd_name.dtd> --> 應用程式的 DTD 路徑和檔案名稱
   ```

5. 指定是否要驗證 (亦即，使用 DTD 來確定 XML 文件是有效的 XML 文件)。例如：

   ```
   <validation>NO</validation>   --> 指定 YES 或 NO
   ```

6. 使用 <Xcollection> 元素，定義 XML 集合的存取和儲存方法。存取和儲存方法定義 XML 文件將包含自儲存於 DB2 表格之資料衍生而來的內容。

   ```
   <Xcollection>
     </Xcollection>
   ```

7. 請指定一或多個 SQL 陳述式，在 DB2 表格中查詢或插入資料。請參閱第56頁的『對映方法需求』，以取得相關指引。例如，指定一個如下的 SQL 查詢：

   ```
     <SQL_stmt>
   SELECT o.order_key, customer_name, customer_email, p.part_key, color, quantity,
   price, tax, ship_id, date, mode from order_tab o, part_tab p,
                    table(select substr(char(timestamp(generate_unique())),16)
     as ship_id, date, mode, part_key from ship_tab) s
         WHERE o.order_key = 1 and
               p.price > 20000 and
               p.order_key = o.order_key and
               s.part_key = p.part_key
         ORDER BY order_key, part_key, ship_id
    </SQL_stmt>
   ```

8. 新增下列前言資訊：

   ```
    <prolog>?xml version="1.0"?</prolog>
   ```

   這行文字必須完全相同。

9. 新增 <doctype></doctype> 標示。例如：

   ```
    <doctype>!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
   ```

10. 使用 <root_node></root_node> 標示來定義 Root 節點。在 root_node 中，您可指定組成 XML 文件的元素和屬性。
11. 將 XML 文件中的元素和屬性，對映至與 DB2 資料相對應的元素和屬性節點。 這些節點提供一個從 XML 資料至 DB2 資料的路徑。

    a. 爲 XML 文件中每一個對映至 DB2 表格直欄的元素，定義一個 <element_node>。

       ```
       <element_node name="name"></element_node>
       ```

element_node 可以有下列節點。

- attribute_node
- child element_node
- text_node

b. 爲 XML 文件中每一個對映至 DB2 表格直欄的屬性，定義一個 <attribute_node>。關於 SQL 對映，請參閱本節開頭的範例 DTD，關於 DAD 檔案的完整語法，請參閱第225頁的『附錄A. DAD 檔案的 DTD』中的 DAD 檔案之 DTD。

例如，您的元素 <Order> 需要一個屬性鍵值。鍵值被儲存於直欄 PART_KEY 中。

DAD 檔案： 在 DAD 檔案中，建立鍵值的屬性節點，指出要儲存 1 這個值的表格。

```
<attribute_node name="key">
                    <column name="part_key"/>
             </attribute_node>
```

製作 XML 文件： 自 PART_KEY 直欄中取出鍵值。

```
<Order key="1">
```

12. 爲每一個自 DB2 表格衍生內容之元素或屬性建立 <text_node>。文字節點具有 <column> 元素，用來指定提供內容的直欄。

例如，您可能有一個含有自名爲 TAX 直欄中取出之值的 XML 元素 <Tax>：

DAD 元素：

```
<element_node name="Tax">
      <text_node>
         <column name="tax"/>
      </text_node>
</element_node>
```

直欄名稱必須位於 SQL 陳述式中的 DAD 檔案開頭。

製作 XML 文件：

```
<Tax>0.02</Tax>
```

自直欄 TAX 中衍生之值 0.02。

13. 在最後一個 </element_node> 標示後面，請記得加上 </root_node> 結束標示。
14. 在 </root_node> 標示後面，請記得加上 </Xcollection> 結束標示。
15. 在 </Xcollection> 標示後面，請記得加上 </DAD> 結束標示。

製作與 RDB_node 對映的 XML 文件

使用 RDB_node 對映來製作 XML 結構的 XML 文件。

這個方法使用 <RDB_node> 來指定元素或屬性節點的 DB2 表格、直欄及條件。<RDB_node> 使用下列元素：

- <table>: 定義與元素相對應的表格
- <column>：定義含有相對應元素的直欄
- <condition>：選用性地指定直欄的條件

<RDB_node> 中使用的子項元素係根據節點的環境定義而定，且使用下列規則：

若節點類型是：	RDB 子項元素使用情形：		
	表格	直欄	條件[1]
Root 元素	Y	N	Y
屬性	Y	Y	選用的
文字	Y	Y	選用的

(1) 若爲多重表格則需要

使用管理精靈: 若要建立組合用的 DAD，請使用 RDB_node 對映：

1. 設定和啓動管理精靈。詳細資訊，請參閱第61頁的『啓動管理精靈』。
2. 在「發射台」視窗中按一下**使用 DAD 檔案**。出現「指定 DAD」視窗。
3. 選擇編輯現存的 DAD 檔案或建立新的 DAD。

   編輯現存的 DAD：

   a. 將 DAD 檔名鍵入**檔名**欄位中，或按一下 **...** 以瀏覽現存的 DAD。
   b. 驗證精靈是否可辨識指定的 DAD 檔案。
      - 若精靈可辨識指定的 DAD 檔案，則可選取**下一步**，**類型**欄位中會顯示 XML 集合 RDB 節點對映。
      - 若精靈無法辨識指定的 DAD 檔案，則無法選取**下一步**。 重新將 DAD 檔名鍵入**檔名**欄位中，或按一下 **...** 以再次瀏覽現存的 DAD 檔。 繼續這些步驟，直到可選取**下一步**爲止。
   c. 按一下**下一步**，開啓「選取驗證」視窗。

   建立新的 DAD：

   a. 將**檔名**欄位留白。
   b. 從**類型**功能表中選取 XML 集合 RDB_node 對映。
   c. 按一下**下一步**，開啓「選取驗證」視窗。

4. 在「選取驗證」視窗中，選擇是否使用 DTD 來驗證 XML 文件。
   - 若要驗證：
     a. 按一下**使用 DTD 驗證 XML 文件**。
     b. 從 **DTD ID** 功能表中，選取要用來驗證的 DTD。

     若您沒有爲資料庫匯入任何 DTD 到 DTD 儲存庫內，您將無法驗證 XML 文件。
   - 按一下**「不」使用 DTD 驗證 XML 文件**以繼續而不驗證您的 XML 文件。
5. 按一下**下一步**，開啓「指定文字」視窗。
6. 在「指定文字」視窗的**前言**欄位中，鍵入前言名稱。

   ```
   <?xml version="1.0"?>
   ```

   若編輯現存的 DAD，**前言**欄位會自動顯示前言。
7. 在「指定文字」視窗的 **Doctype** 欄位中，輸入 XML 文件的文件類型。

   若編輯現存的 DAD，**文件類型**欄位會自動顯示文件類型。
8. 按一下**下一步**，開啓「RDB 對映」視窗。
9. 在「RDB 對映」視窗的左邊欄位中，按一下要對映的元素或屬性節點，即可選取它。

   將 XML 文件中的元素和屬性，對映至與 DB2 資料相對應的元素和屬性節點。這些節點提供一個從 XML 資料至 DB2 資料的路徑。
10. **若要新增 Root 節點：**
    a. 選取 **Root** 圖示。
    b. 按一下**新建元素**以定義一個新的節點。
    c. 在**明細**方框中，指定**節點類型**爲**元素**。
    d. 在**節點名稱**欄位中輸入頂端層次節點的名稱。
    e. 按一下**新增**以建立新節點。

       您已建立 Root 節點或元素，它是對映中所有其它元素或屬性節點的父節點。 Root 節點有一些表格子項元素和一個結合條件。
    f. 爲集合中的每一個表格新增表格節點。
       1) 強調顯示 Root 節點名稱，並選取**新建元素**。
       2) 在**明細**方框中，指定**表格**做爲**節點類型**。
       3) 從**表格名稱**中選取表格的名稱。表格必須已存在。
       4) 按一下**新增**來新增表格節點。
       5) 對每一個表格重複這些步驟。
    g. 爲表格節點新增一個結合條件。

1) 強調顯示 Root 節點名稱，並選取**新建元素**。
2) 在**明細**方框中，指定**條件**做爲**節點類型**。
3) 在**條件**欄位中，使用下列語法輸入結合條件：

```
table_name.table_column = table_name.table_column AND
table_name.table_column = table_name.table_column ...
```

4) 按一下**新增**來新增條件。

11. **若要新增元素或屬性節點：**
    a. 按一下在欄位左邊的父節點以新增子項元素或屬性。
    b. 按一下**新建元素**。 如果您未選取父節點，則無法選取**新建元素**。
    c. 從**明細**方框的**節點類型**功能表中，選取節點類型。

       節點類型功能表只顯示目前對映有效的節點類型。**元素**或**屬性**。
    d. 在**節點名稱**欄位中指定節點名稱。
    e. 按一下**新增**以新建節點。
    f. **將元素或屬性節點的內容對映到關聯式表格：**
       1) 指定文字節點。
          a) 按一下父節點。
          b) 按一下**新建元素**。
          c) 在**節點類型**欄位中，選取**文字**。
          d) 選取**新增**以新增節點。
       2) 新增表格節點。
          a) 選取您剛才建立的文字節點，按一下**新建元素**。
          b) 在 **節點類型**欄位中，選取**表格**，並指定元素的表格名稱。
          c) 按一下**新增**以新增節點。
       3) 新增直欄節點。
          a) 再次選取文字節點，然後按一下**新建元素**。
          b) 在**節點類型**欄位中，選取**直欄**，並指定元素的直欄名稱。
          c) 按一下**新增**以新增節點。

          限制： 無法使用管理精靈來建立新直欄。 若您指定「直欄」做爲節點類型，您只能選取 DB2 資料庫中已存在的直欄。
       4) 選用性地新增直欄的條件。
          a) 再次選取文字節點，然後按一下**新建元素**。
          b) 在**節點類型**欄位中，選取**條件**，使用下列語法指定條件：

             operator LIKE|<|>|= *value*

c) 按一下**新增**以新增節點。

g. 繼續編輯 RDB 對映，或按一下**下一步**，開啓「指定 DAD」視窗。

12. **移除節點：**
    a. 按一下左邊欄位中的一個節點。
    b. 按一下**移除**。
    c. 繼續編輯 RDB_node 對映，或按一下**下一步**，開啓「指定 DAD」視窗。
13. 在「指定 DAD」視窗的**檔名**欄位中，鍵入已修改之 DAD 的輸出檔名。
14. 按一下**完成**以移除節點並返回「發射台」視窗。

從 DB2 指令 Shell: DAD 檔案是一個可用任何文字編輯器建立的 XML 檔案。下列步驟顯示第232頁的『文件存取定義檔』範例附錄中的片斷。完整的資訊和內容，請參閱這些範例。

1. 開啓一個文字編輯器。
2. 建立 DAD 標頭：

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "path\dad.dtd"> --> DAD 的 DTD 路徑和檔案名稱
```

3. 插入 <DAD></DAD> 標示。
4. 在 <DAD> 標示後面，指定 DTD ID 用來結合 DAD 檔案與 XML 文件 DTD。

```
<dtdid>path\dtd_name.dtd> --> 應用程式的 DTD 路徑和檔案名稱
```

5. 指定是否要驗證 (亦即，使用 DTD 來確定 XML 文件是有效的 XML 文件)。例如：

```
<validation>NO</validation>   --> 指定 YES 或 NO
```

6. 使用 <Xcollection> 元素，定義 XML 集合的存取和儲存方法。存取及儲存方法定義 XML 資料儲存在 DB2 表格集合中。

```
<Xcollection>
  </Xcollection>
```

7. 新增下列前言資訊：

```
<prolog>?xml version="1.0"?</prolog>
```

這行文字必須完全相同。

8. 新增 <doctype></doctype> 標示。例如：

```
<doctype>!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
```

9. 使用 <root_node> 來定義 Root 節點。在 root_node 中，您可指定組成 XML 文件的元素和屬性。

10. 將 XML 文件中的元素和屬性，對映至與 DB2 資料相對應的元素和屬性節點。 這些節點提供一個從 XML 資料至 DB2 資料的路徑。

    a. 定義 Root element_node。element_node 包含：

       - 一個 RDB_node，它以結合條件指定 table_nodes 來指定集合
       - 子項元素
       - 屬性

       若要指定表格節點及條件：

       1) 請建立 RDB_node 元素：例如：

```
<RDB_node>
                </RDB_node>
```

       2) 爲表格定義一個 <table_node>，每一個表格含有在 XML 文件中要包含的資料。例如，如果您有 ORDER_TAB、PART_TAB 及 SHIP_TAB 這三個表格，其中含有要放入文件中的直欄資料，請爲每一個表格建立一個表格節點。例如：

```
        <RDB_node>
<table name="ORDER_TAB">
<table name="PART_TAB">
<table name="SHIP_TAB"></RDB_node>
```

       3) 當您計劃啓用此集合時，您可以選擇是否爲每一個表格指定鍵值直欄。組合時不一定要求此鍵值屬性；但是，當您啓用集合時，DAD 檔通常需要同時支援組合與分解。例如：

```
        <RDB_node>
<table name="ORDER_TAB" key="order_key">
<table name="PART_TAB" key="part_key">
<table name="SHIP_TAB" key="date mode">
                </RDB_node>
```

       4) 爲集合中的表格建立結合條件。語法：

```
expression = expression AND
expression = expression
```

          例如：

```
        <RDB_node>
<table name="ORDER_TAB">
<table name="PART_TAB">
<table name="SHIP_TAB">
<condition>
                order_tab.order_key = part_tab.order_key AND
                    part_tab.part_key = ship_tab.part_key
</condition>
                </RDB_node>
```

    b. 爲 XML 文件中每一個對映至 DB2 表格直欄的元素，定義一個 <element_node> 標示。例如：

```
<element_node name="name">
    </element_node>
```

元素節點可具有下列其中一種元素類型：

- <text_node>：指定元素具有 DB2 表格的內容；元素沒有子項元素
- <attribute_node>：指定屬性。屬性節點定義於下一個步驟。

text_node 包含一個 <RDB_node> 將內容對映到 DB2 表格和直欄名稱。

RDB_nodes 使用於底層元素，這些元素含有要對映到 DB2 表格的內容。RDB_node 具有下列子項元素：

- <table>: 定義與元素相對應的表格
- <column>：定義含有相對應元素的直欄且指定直欄類型與類型屬性
- <condition>：選用性地指定直欄的條件

例如，您可能有一個 XML 元素 <Tax> 對映至一個稱爲 TAX 的直欄：

XML 文件：

```
<Tax>0.02</Tax>
```

本例中，您使用 0.02 做爲 TAX 直欄中的值。

```
          <element_node name="Tax">
                <text_node>
       <RDB_node>
                   <table name="part_tab"/>
        <column name="tax"/>
                   </RDB_node>
                </text_node>
   </element_node>
```

在本範例中，<RDB_node> 指定 <Tax> 元素的值是文字值，資料儲存於 PART_TAB 表格的 TAX 直欄中。關於 RDB_node 對映，請參閱第232頁的『文件存取定義檔』中的範例 DAD 檔案；關於 DAD 檔案的完整語法，請參閱第225頁的『附錄A. DAD 檔案的 DTD』中的 DAD 檔案之 DTD。

c. 當您計劃啓用此集合時，您可以選擇是否新增類型屬性至每一個 <column> 元素。組合時不一定要求此類型屬性；但是，當您啓用集合時，DAD 檔通常需要同時支援組合與分解。例如：

```
<column name="tax" type="real"/>
```

d. 爲 XML 文件中每一個對映至 DB2 表格直欄的屬性，定義一個 <attribute_node>。例如：

```
<attribute_node name="key">
             </attribute_node>
```

attribute_node 有一個 <RDB_node> 能夠將屬性值對映到 DB2 表格和直欄。 <RDB_node> 具有下列子項元素：

- <table>: 定義與元素相對應的表格
- <column>：定義含有相對應元素的直欄
- <condition>：選用性地指定直欄的條件

例如，元素 <Order> 可以有 key 這個屬性。鍵值需要儲存於直欄 PART_KEY 中。 在 DAD 檔案中，爲鍵值建立一個 <attribute_node>，指出儲存數值的表格。

DAD 檔案

```
<attribute_node name="key">
      <RDB_node>
  <table name="part_tab">
                  <column name="part_key"/>
      <RDB_node>
            </attribute_node>
```

製作 XML 文件：

```
<Order key="1">
```

11. 在最後一個 </element_node> 標示後面，請記得加上 </root_node> 結束標示。
12. 在 </root_node> 標示後面，請記得加上 </Xcollection> 結束標示。
13. 在 </Xcollection> 標示後面，請記得加上 </DAD> 結束標示。

使用 RDB_node 對映分解 XML 文件

使用 RDB_node 對映來分解 XML 文件。這個方法使用 <RDB_node> 來指定元素或屬性節點的 DB2 表格、直欄及條件。 <RDB_node> 使用下列元素：

- <table>: 定義與元素相對應的表格
- <column>：定義含有相對應元素的直欄
- <condition>：選用性地指定直欄的條件

<RDB_node> 中使用的子項元素係根據節點的環境定義而定，且使用下列規則：

若節點類型是：	RDB 子項元素使用情形：		
	表格	直欄	條件[1]
Root 元素	Y	N	Y
屬性	Y	Y	選用的
文字	Y	Y	選用的

(1) 若爲多重表格則需要

使用管理精靈： 若要建立分解 DAD：

1. 設定和啓動管理精靈。詳細資訊，請參閱第61頁的『啓動管理精靈』。
2. 在「發射台」視窗中按一下**使用 DAD 檔案**。即顯示「指定 DAD」視窗。
3. 選擇編輯現存的 DAD 檔案或建立新的 DAD。

   編輯現存的 DAD：

   a. 將 DAD 檔名鍵入**檔名**欄位中，或按一下 **...** 以瀏覽現存的 DAD。

   b. 驗證精靈是否可辨識指定的 DAD 檔案。

      - 若精靈可辨識指定的 DAD 檔案，則可選取**下一步**，**類型**欄位中會顯示 XML 集合 RDB 節點對映。
      - 若精靈無法辨識指定的 DAD 檔案，則無法選取**下一步**。 重新將 DAD 檔名鍵入**檔名**欄位中，或按一下 **...** 以再次瀏覽現存的 DAD 檔。 繼續這些步驟，直到可選取**下一步**爲止。

   c. 按一下**下一步**，開啓「選取驗證」視窗。

   建立新的 DAD：

   a. 將**檔名**欄位留白。

   b. 從**類型**功能表中選取 XML 集合 RDB_node 對映。

   c. 按一下**下一步**，開啓「選取驗證」視窗。

4. 在「選取驗證」視窗中，選擇是否使用 DTD 來驗證 XML 文件。

   - 若要驗證：

     a. 按一下**使用 DTD 驗證 XML 文件**。

     b. 從 **DTD ID** 功能表中，選取要用來驗證的 DTD。

     若您沒有爲資料庫匯入任何 DTD 到 DTD 儲存庫內，您將無法驗證 XML 文件。

   - 按一下**「不」使用 DTD 驗證 XML 文件**以繼續而不驗證您的 XML 文件。

5. 按一下**下一步**，開啓「指定文字」視窗。
6. 若只要分解 XML 文件，請忽略**前言**欄位。 若組合與分解時都使用 DAD 檔，在「指定文字」視窗的**前言**欄位中，請鍵入前言名稱。 若將 XML 文件分解爲 DB2 資料，則不需要前言。

   ```
   <?xml version="1.0"?>
   ```

   若編輯現存的 DAD，**前言**欄位會自動顯示前言。
7. 若只要分解 XML 文件，請忽略**文件類型**欄位。 如果您在組合和分解時都用到 DAD 檔案，請在 **Doctype** 欄位中鍵入 XML 文件的文件類型。

   若編輯現存的 DAD，**文件類型**欄位會自動顯示文件類型。

8. 按一下**下一步**，開啓「RDB 對映」視窗。
9. 在「RDB 對映」視窗的左邊欄位中，按一下要對映的元素或屬性節點，即可選取它。

   將 XML 文件中的元素和屬性，對映至與 DB2 資料相對應的元素和屬性節點。這些節點提供一個從 XML 資料至 DB2 資料的路徑。

10. **若要新增 Root 節點：**
    a. 選取 **Root** 圖示。
    b. 按一下**新建元素**以定義一個新的節點。
    c. 在**明細**方框中，指定**節點類型**爲**元素**。
    d. 在**節點名稱**欄位中輸入頂端層次節點的名稱。
    e. 按一下**新增**以建立新節點。

       您已建立 Root 節點或元素，這是該對映中所有其它元素或屬性節點的父節點。 Root 節點有一些表格子項元素和一個結合條件。

    f. 爲集合中的每一個表格新增表格節點。
       1) 強調顯示 Root 節點名稱，並選取**新建元素**。
       2) 在**明細**方框中，指定**表格**做爲**節點類型**。
       3) 從**表格名稱**中選取表格的名稱。表格必須已存在。
       4) 在**表格鍵值**欄位中，指定表格的鍵值直欄。
       5) 按一下**新增**來新增表格節點。
       6) 對每一個表格重複這些步驟。
    g. 爲表格節點新增一個結合條件。
       1) 強調顯示 Root 節點名稱，並選取**新建元素**。
       2) 在**明細**方框中，指定**條件**做爲**節點類型**。
       3) 在**條件**欄位中，使用下列語法輸入結合條件：

          ```
          table_name.table_column = table_name.table_column AND
          table_name.table_column = table_name.table_column ...
          ```

       4) 按一下**新增**來新增條件。

    現在即可在此節點新增子項元素及屬性。

11. **若要新增元素或屬性節點：**
    a. 按一下在欄位左邊的父節點以新增子項元素或屬性。

       若不選取父節點，則無法選取**新建**。

    b. 按一下**新建元素**。
    c. 從**明細**方框的**節點類型**功能表中，選取節點類型。

節點類型功能表只顯示目前對映有效的節點類型。**元素**或**屬性**。

d. 在**節點名稱**欄位中指定節點名稱。

e. 按一下**新增**以新建節點。

f. **將元素或屬性節點的內容對映到關聯式表格：**

1) 指定文字節點。

a) 按一下父節點。

b) 按一下**新建元素**。

c) 在**節點類型**欄位中，選取**文字**。

d) 選取**新增**以新增節點。

2) 新增表格節點。

a) 選取您剛才建立的文字節點，按一下**新建元素**。

b) 在 **節點類型**欄位中，選取**表格**，並指定元素的表格名稱。

c) 按一下**新增**以新增節點。

3) 新增直欄節點。

a) 再次選取文字節點，然後按一下**新建元素**。

b) 在**節點類型**欄位中，選取**直欄**，並指定元素的直欄名稱。

c) 在**類型**欄位中，指定直欄的基本資料類型，指出該直欄必須爲何種類型才能儲存未標示的資料。

d) 按一下**新增**以新增節點。

限制： 無法使用管理精靈來建立新直欄。 若您指定「直欄」做爲節點類型，您只能選取 DB2 資料庫中已存在的直欄。

4) 選用性地新增直欄的條件。

a) 再次選取文字節點，然後按一下**新建元素**。

b) 在**節點類型**欄位中，選取**條件**，使用下列語法指定條件：

operator LIKE|<|>|= *value*

c) 按一下**新增**以新增節點。

如果要修改這些節點，您可選取節點，變更**明細**方框中的欄位，然後按一下**變更**。

g. 繼續編輯 RDB 對映，或按一下**下一步**，開啓「指定 DAD」視窗。

12. **移除節點：**

a. 按一下左邊欄位中的一個節點。

b. 按一下**移除**。

c. 繼續編輯 RDB_node 對映，或按一下**下一步**，開啓「指定 DAD」視窗。

13. 在「指定 DAD」視窗的**檔名**欄位中，鍵入已修改之 DAD 的輸出檔名。
14. 按一下**完成**以移除節點並返回「發射台」視窗。

從 DB2 指令 Shell: DAD 檔案是一個可用任何文字編輯器建立的 XML 檔案。下列步驟顯示第232頁的『文件存取定義檔』範例附錄中的片斷。完整的資訊和內容，請參閱這些範例。

1. 開啓一個文字編輯器。
2. 建立 DAD 標頭：

   ```
   <?xml version="1.0"?>
   <!DOCTYPE DAD SYSTEM "path\dad.dtd"> --> DAD 的 DTD 路徑和檔案名稱
   ```
3. 插入 <DAD></DAD> 標示。
4. 在 <DAD> 標示後面，指定 DTD ID 用來結合 DAD 檔案與 XML 文件 DTD。

   ```
   <dtdid>path\dtd_name.dtd> --> 應用程式的 DTD 路徑和檔案名稱
   ```
5. 指定是否要驗證 (亦即，使用 DTD 來確定 XML 文件是有效的 XML 文件)。例如：

   ```
   <validation>NO</validation>   --> 指定 YES 或 NO
   ```
6. 使用 <Xcollection> 元素，定義 XML 集合的存取和儲存方法。存取及儲存方法定義 XML 資料儲存在 DB2 表格集合中。

   ```
   <Xcollection>
     </Xcollection>
   ```
7. 新增下列前言資訊：

   ```
   <prolog>?xml version="1.0"?</prolog>
   ```

   這行文字必須完全相同。
8. 新增 <doctype></doctype> 標示。例如：

   ```
   <doctype>!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
   ```
9. 使用 <root_node></root_node> 標示來定義 root_node。在 root_node 中，您可指定組成 XML 文件的元素和屬性。
10. 在 <root_node> 標示後面，將 XML 文件中的元素和屬性，對映至與 DB2 資料相對應的元素和屬性節點。 這些節點提供一個從 XML 資料至 DB2 資料的路徑。
    a. 定義最上層 Root element_node。element_node 包含：
       - 表格節點，以結合條件來指定集合。
       - 子項元素
       - 屬性

若要指定表格節點及條件：

1) 請建立 RDB_node 元素：例如：

```
        <RDB_node>
                    </RDB_node>
```

2) 爲表格定義一個 <table_node>，每一個表格含有在 XML 文件中要包含的資料。例如，如果您有 ORDER_TAB、PART_TAB 及 SHIP_TAB 這三個表格，其中含有要放入文件中的直欄資料，請爲每一個表格建立一個表格節點。例如：

```
        <RDB_node>
<table name="ORDER_TAB">
<table name="PART_TAB">
<table name="SHIP_TAB"></RDB_node>
```

3) 爲集合中的表格建立結合條件。語法：

```
expression = expression AND
expression = expression ...
```

例如：

```
        <RDB_node>
<table name="ORDER_TAB">
<table name="PART_TAB">
<table name="SHIP_TAB">
<condition>
                order_tab.order_key = part_tab.order_key AND
                    part_tab.part_key = ship_tab.part_key
</condition>
                    </RDB_node>
```

4) 爲每一個表格指定一個主要鍵。主要鍵由一個直欄或多個直欄組成，稱爲組合鍵。若要指定主要鍵，請新增一個屬性鍵到 RDB_node 的表格元素中。下列範例在 root element_node Order 的 RDB_node 中，爲每一個表格定義一個主要鍵：

```
<element_node name="Order">
        <RDB_node>
         <table name="order_tab" key="order_key"/>
         <table name="part_tab" key="part_key price"/>
         <table name="ship_tab" key="date mode"/>
                    <condition>
               order_tab.order_key = part_tab.order_key AND
                   part_tab.part_key = ship_tab.part_key
                    </condition>
        <RDB_node>
```

製作 XML 文件時，指定的分解資訊會被忽略。

分解時要求此鍵值屬性，尤其是當您啓用集合時，因爲 DAD 檔通常需要同時支援組合與分解。

b. 爲 XML 文件中每一個對映至 DB2 表格直欄的元素，定義一個 <element_node> 標示。例如：

```
<element_node name="name">
    </element_node>
```

元素節點可具有下列其中一種元素類型：

- <text_node>：指定元素具有 DB2 表格的內容；本例中，元素沒有子項元素。
- <attribute_node>：指定屬性；屬性節點定義於下一個步驟
- 子項元素

text_node 包含一個 RDB_node，它能夠將內容對映到 DB2 表格和直欄名稱。

RDB_nodes 使用於底層元素，這些元素含有要對映到 DB2 表格的內容。RDB_node 具有下列子項元素：

- <table>: 定義與元素相對應的表格
- <column>：定義含有相對應元素的直欄
- <condition>：選用性地指定直欄的條件

例如，您可能有一個 XML 元素 <Tax> 想用來儲存 TAX 直欄中未標示的內容：

XML 文件：

```
<Tax>0.02</Tax>
```

本例中，您要將 0.02 儲存在 TAX 直欄中。

在 DAD 檔案中，您指定一個 <RDB_node> 將 XML 元素對映到 DB2 表格和直欄。

DAD 檔案：

```
          <element_node name="Tax">
                <text_node>
       <RDB_node>
                   <table name="part_tab"/>
        <column name="tax"/>
                   </RDB_node>
                </text_node>
   </element_node>
```

<RDB_node> 指定 <Tax> 元素的值是文字值，資料是儲存於 PART_TAB 表格的 TAX 直欄中。

c. 爲 XML 文件中每一個對映至 DB2 表格直欄的屬性，定義一個 <attribute_node>。例如：

```
<attribute_node name="key">
              </attribute_node>
```

attribute_node 有一個 RDB_node 將屬性值對映到 DB2 表格和直欄。RDB_node 具有下列子項元素：

- <table>: 定義與元素相對應的表格
- <column>：定義含有相對應元素的直欄
- <condition>：選用性地指定直欄的條件

例如，您的元素 <Order> 可能有一個屬性鍵值。鍵值需要儲存於直欄 PART_KEY 中。

XML 文件：

```
<Order key="1">
```

在 DAD 檔案中，建立鍵值的 attribute_node，指出要儲存 1 這個值的表格。

DAD 檔案：

```
<attribute_node name="key">
      <RDB_node>
  <table name="part_tab">
                  <column name="part_key"/>
      <RDB_node>
            </attribute_node>
```

11. 爲每一個 attribute_node 和 text_node 的 RDB_node 指定直欄類型。這可確定每一個要儲存未標示資料的直欄都有正確的資料類型。若要指定直欄類型，請新增屬性類型到直欄元素。 下列範例將直欄類型定義爲 INTEGER：

```
<attribute_node name="key">
      <RDB_node>
                  <table name="order_tab"/>
      <column name="order_key" type="integer"/>
                  </RDB_node>
            </attribute_node>
```

12. 在最後一個 </element_node> 標示後面，請記得加上 </root_node> 結束標示。
13. 在 </root_node> 標示後面，請記得加上 </Xcollection> 結束標示。
14. 在 </Xcollection> 標示後面，請記得加上 </DAD> 結束標示。

啓用 XML 集合

啓用 XML 集合會剖析 DAD 檔案來識別 XML 文件相關的表格和直欄，在 XML_USAGE 表格中記錄控制資訊。在下列情況下，啓用 XML 集合是選用的：

- 分解 XML 文件，在新的 DB2 表格中儲存資料
- 使用多重 DB2 表格的現存資料來製作 XML 文件

若使用相同 DAD 檔案來製作和分解，您可啓用集合來處理組合和分解。

您可透過 XML Extender 管理精靈、使用 **dxxadm** 指令加上 enable_collection 選項，或使用 XML Extender 儲存程序 dxxEnableCollection()，啓用 XML 集合。

使用管理精靈

使用下列步驟來啓用 XML 集合。

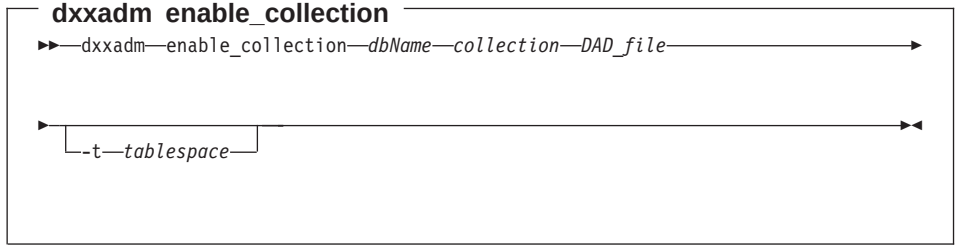
1. 設定和啓動管理精靈。詳細資訊，請參閱第61頁的『啓動管理精靈』。
2. 在「發射台」視窗中按一下**使用 XML 集合**。出現「選取作業」視窗。
3. 按一下**啓用集合**，再按**下一步**。 出現「啓用集合」視窗。
4. 從下拉功能表的**集合名稱**欄位中，選取您要啓用的集合名稱。
5. 將 DAD 檔名鍵入 **DAD 檔名**欄位中，或按一下 **...** 以瀏覽現存的 DAD 檔。
6. 可選用性地在**表格空間**欄位中，鍵入先前建立之表格空間的名稱。

   表格空間將包含爲了分解而產生的新 DB2 表格。
7. 按一下**完成**來啓用集合，返回「發射台」視窗。
   - 若順利啓用集合，則顯示**啓用集合成功**訊息。
   - 若未順利啓用集合，則顯示錯誤訊息。繼續執行先前步驟，直到順利啓用集合爲止。

從 DB2 指令 Shell

若要啓用 XML 集合，請輸入 **dxxadm** 指令：

語法：



參數：

dbName

資料庫的名稱。

collection

XML 集合的名稱。 此值用來做爲 XML 集合儲存程序的參數。

DAD_file

檔案的名稱，檔案含有文件存取定義 (DAD)。

tablespace

一個現存的表格空間，其中包括爲了分解而產生的新 DB2 表格。 如果沒有指定，則會使用預設的表格空間。

範例： 下列範例使用 DB2 指令 Shell，在資料庫 SALES_DB 中啓用一個稱爲 sales_ord 的集合。 DAD 檔案使用 SQL 對映，可以在第233頁的『DAD 檔：XML 集合 - SQL 對映』找到。

```
dxxadm enable_collection SALES_DB sales_ord getstart.dad
```

在您啓用 XML 集合之後，您可使用 XML Extender 儲存程序來製作或分解 XML 文件。

停用 XML 集合

停用 XML 集合會移除 XML_USAGE 表格中的記錄，該記錄識別表格和直欄爲集合的一部份。但不會捨棄任何資料表。當您想要更新 DAD 且需要重新啓用集合時，或想要捨棄集合時，您可停用一個集合。

您可透過 XML Extender 管理精靈、使用 **dxxadm** 指令加上 disable_collection 選項，或使用 XML Extender 儲存程序 dxxDisableCollection()，停用 XML 集合。

使用管理精靈

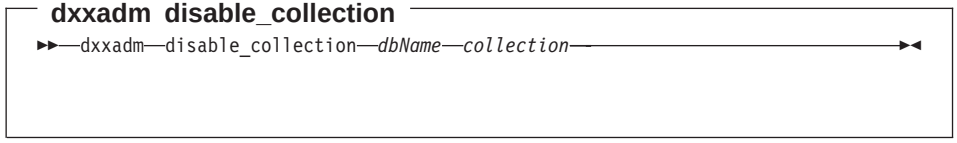
使用下列步驟來停用 XML 集合。

1. 設定和啓動管理精靈。詳細資訊，請參閱第61頁的『啓動管理精靈』。
2. 在「發射台」視窗中按一下**使用 XML 集合**，檢視 XML Extender 集合的相關作業。出現「選取作業」視窗。
3. 按一下**停用 XML 集合**，再按**下一步**，停用 XML 集合。 出現「停用集合」視窗。
4. 在**集合名稱**欄位中，鍵入您要停用的集合名稱。
5. 按一下**完成**來停用集合，返回「發射台」視窗。
   - 若順利停用集合，則顯示**停用集合成功**訊息。
   - 若未順利停用集合，則顯示錯誤方框。繼續執行先前步驟，直到順利停用集合爲止。

從 DB2 指令 Shell

若要停用 XML 集合，請輸入 **dxxadm** 指令：

語法：



參數：

dbName
資料庫的名稱。

collection
XML 集合的名稱。 此值用來做爲 XML 集合儲存程序的參數。

範例：

```
dxxadm disable_collection SALES_DB sales_ord
```

停用 XML 資料庫

當您想要清除 XML Extender 環境、捨棄 XML Extender UDT、UDF、儲存程序、以及管理支援表格時，您可停用資料庫。 XML Extender 使用現行案例來停用您所連接的資料庫。

當您停用 XML 的資料庫時， XML Extender 會對資料庫採取下列動作：

- 刪除所有使用者定義類型 (UDT) 和使用者定義函數 (UDF)
- 刪除 XML Extender 的控制表格和 meta 資料
- 刪除 db2xml 綱目。

開始之前

對於要停用的資料庫，停用任何 XML 直欄或集合。

使用管理精靈

使用下列步驟，停用 XML 資料的資料庫：

1. 設定和啓動管理精靈。詳細資訊，請參閱第61頁的『啓動管理精靈』。
2. 從「發射台」視窗中按一下**停用資料庫**來停用現行資料庫。

如果目前未啓用資料庫，則只能選取**啓用資料庫**。

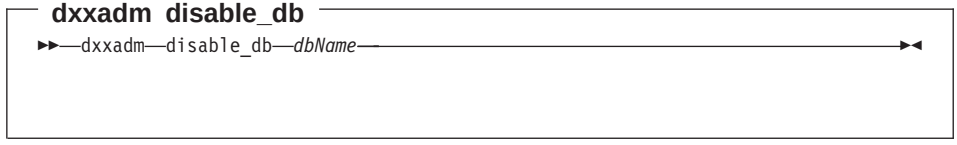
停用資料庫後，就會返回「發射台」視窗。

從 DB2 指令 Shell

在指令行上輸入 **dxxadm**，指定要停用的資料庫。

語法：

dxxadm disable_db



參數：

dbName

要停用的資料庫名稱。

範例： 停用一個現存的資料庫，稱爲 SALES_DB。

```
dxxadm disable_db SALES_DB
```

第3篇 程式設計

這個部份說明管理 XML 資料的程式設計技術。

第5章 管理 XML 直欄資料

使用 XML 直欄時，請儲存整個 XML 文件作爲直欄資料。存取及儲存方法可讓您在索引及搜尋文件、從文件中擷取資料以及更新文件時，同時保持 XML 文件原封不動。XML 直欄以 XML 文件在 DB2 中使用的原來格式，包含這些文件作爲直欄資料。啓用 XML 的資料庫之後，您可使用下列使用者定義類型 (UDT)：

XMLCLOB
在 DB2 中，儲存爲字元大型物件 (CLOB) 的 XML 文件內容

XMLVARCHAR
在 DB2 中，儲存爲 VARCHAR 的 XML 文件內容

XMLFile
儲存在本端檔案系統上的一個檔案的 XML 文件

您可使用 XML UDT 建立或變更應用程式表格作爲直欄資料類型。 這些表格稱爲 XML 表格。 若要瞭解如何建立或變更 XML 的表格，請參閱第70頁的『建立或變更 XML 表格』。

啓用 XML 的直欄之後，您可開始管理 XML 直欄內容。 建立 XML 直欄之後，您可執行下列管理作業：

- 在 DB2 中儲存 XML 文件
- 從 DB2 擷取 XML 資料或文件
- 更新 XML 文件
- 刪除 XML 資料或文件

若要執行這些作業，您可使用下列兩種方法：

- *預設強制轉型函數*，該函數將 SQL 基本類型轉換成 XML UDT
- XML Extender 提供的使用者定義函數 (UDF)

本書針對每一個作業說明這兩種方法。

UDT 及 UDF 名稱

DB2 函數的完整名稱：*schema-name.function-name*，其中 *schema-name* 是一個識別字，這個識別字對 SQL 物件提供邏輯分組。 XML Extender UDF 的綱目名稱是 DB2XML。 DB2XML 綱目名稱也是 XML Extender UDT 的限定元。 本書只提供函數名稱的參照。

函數路徑是排序的綱目名稱列示。DB2 使用列示中的綱目名稱次序，分辨函數及 UDT 的參照。您可透過指定 SQL 陳述式 SET CURRENT FUNCTION PATH 以指定函數路徑。 此動作在 CURRENT FUNCTION PATH 特別暫存區中設定函數路徑。

就 XML Extender 而言，建議將 db2xml 綱目新增至函數路徑。此動作可讓您輸入 XML Extender UDF 及 UDT 名稱，不必使用 db2xml 作爲它們的字首。下例說明如何將 db2xml 綱目新增至函數路徑：

```
SET CURRENT FUNCTION PATH = db2xml, CURRENT FUNCTION PATH
```

重要事項：若以 db2xml 的身份登入，請勿在函數路徑中新增 db2xml 作爲第一個綱目；以 db2xml 身份登入時，db2xml 會自動設定成第一個綱目。此動作會產生錯誤狀況，因爲函數路徑會以兩個 db2xml 綱目爲開頭。

儲存資料

使用 XML Extender 時，您可將完整的 XML 文件插入 XML 直欄。 若定義輔助表格，那麼 XML Extender 會自動更新這些表格。 直接儲存 XML 文件時，XML Extender 會將基本類型儲存爲 XML 類型。

作業概觀：

1. 確定已建立或更新 DAD 檔。
2. 決定在儲存文件時使用什麼資料類型。
3. 選擇一種方法在 DB2 表格中儲存資料 (強制轉型函數或 UDF)。
4. 指定一個 SQL INSERT 陳述式，該陳述式指定 XML 表格及直欄以包含 XML 文件。

XML Extender 提供兩種方法儲存 XML 文件：預設強制轉型函數及儲存 UDF。表8 顯示何時使用每一種方法。

表 8. XML Extender 儲存函數

基本類型	儲存在 DB2 作為...		
	XMLVARCHAR	**XMLCLOB**	**XMLFILE**
VARCHAR	XMLVARCHAR()	N/A	XMLFileFromVarchar()
CLOB	N/A	XMLCLOB()	XMLFileFromCLOB()
FILE	XMLVarcharFromFile()	XMLCLOBFromFile()	XMLFILE

使用預設強制轉型函數

就每個 UDT 而言，*預設強制轉型函數*將 SQL 基本類型強制轉型爲

UDT。您可在 VALUES 子句中使用 XML Extender 提供的強制轉型函數插入資料。 表9顯示提供的強制轉型函數：

表 9. XML Extender 預設強制轉型函數

用於 SELECT 子句中的強制轉型	回覆類型	說明
XMLVARCHAR(VARCHAR)	XMLVARCHAR	從 VARCHAR 的記憶體緩衝區輸入
XMLCLOB(CLOB)	XMLCLOB	從 CLOB 的記憶體緩衝區或 CLOB 定位器輸入
XMLFILE(VARCHAR)	XMLFILE	僅儲存檔名

範例：下面陳述式將強制轉型的 VARCHAR 類型插入 XMLVARCHAR 類型：

```
INSERT INTO sales_tab
VALUES('123456', 'Sriram Srinivasan', db2xml.XMLVarchar(:xml_buff))
```

使用儲存 UDF：

就每個 XML Extender UDT 而言，儲存 UDF 可以從不是它的基本類型的資源將資料匯入 DB2。例如，若要將 XML 檔文件匯入 DB2 作爲 XMLCLOB，那麼您可使用函數 XMLCLOBFromFile()。

表10 顯示 XML Extender 提供的儲存函數。

表 10. XML Extender 儲存 UDF

儲存使用者定義的函數	回覆類型	說明
XMLVarcharFromFile()	XMLVARCHAR	從伺服器上的某個檔案讀取 XML 文件，然後傳回 XMLVARCHAR 類型值。
XMLCLOBFromFile()	XMLCLOB	從伺服器上的某個檔案讀取 XML 文件，然後傳回 XMLCLOB 類型值。
XMLFileFromVarchar()	XMLFILE	從記憶體讀取 XML 文件作爲 VARCHAR，將它寫入外部檔，然後傳回 XMLFILE 類型的值，該值是檔名。
XMLFileFromCLOB()	XMLFILE	從記憶體讀取 XML 文件作爲 CLOB 或 CLOB 定位器，將它寫入外部檔，然後傳回 XMLFILE 類型的值，該值是檔名。

範例：下列陳述式使用 XMLCLOBFromFile() 函數作爲 XMLCLOB，在 XML 表格儲存記錄。

```
EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
    VALUES( '1234', 'Sriram Srinivasan,
      XMLCLOBFromFile('c:\dxx\samples\cmd\getstart.xml'))
```

前述範例從 `c:\dxx\samples\cmd\getstart.xml` 檔案將 XML 物件匯入 SALES_TAB 表格的 ORDER 直欄。

擷取資料

使用 XML Extender 時，您可擷取整個文件或元素及屬性內容。直接擷取 XML 直欄時，XML Extender 會傳回 UDT 作爲直欄類型。 擷取資料的明細，請參閱下列各節：

- 『擷取整個文件』
- 第108頁的『擷取元素內容及屬性值』

XML Extender 提供兩種方法以擷取資料：預設強制轉型函數及 Content() 超載 UDF。 表11顯示何時使用每一種方法。

表 11. XML Extender 擷取函數

XML 類型	從 DB2 擷取作為...		
	VARCHAR	**CLOB**	**FILE**
XMLVARCHAR	VARCHAR	N/A	Content()
XMLCLOB	N/A	XMLCLOB	Content()
XMLFILE	N/A	Content()	FILE

擷取整個文件

作業概觀：

1. 確定 XML 表格有儲存 XML 文件，然後決定要擷取什麼資料。
2. 選擇一種方法擷取 DB2 表格中的資料 (強制轉型函數或 UDF)。
3. 若使用超載 Content() UDF 則是用來決定哪一種資料類型與擷取的資料相關，以及決定要匯出哪一種資料類型。
4. 指定 SQL 查詢，該查詢指定擷取 XML 文件的 XML 表格及直欄。

XML Extender 提供兩種方法擷取資料：

使用預設強制轉型函數

使用 DB2 提供的預設強制轉型函數，讓 UDT 將 XML UDT 轉換成 SQL

基本類型然後操作它。您可在 SELECT 陳述式中使用 XML Extender 提供的強制轉型函數以擷取資料。表12顯示提供的強制轉型函數：

表 12. XML Extender 預設強制轉型函數

用於選取子句中的強制轉型	回覆類型	說明
varchar(XMLVARCHAR)	VARCHAR	VARCHAR 中的 XML 文件
clob(XMLCLOB)	CLOB	CLOB 中的 XML 文件
varchar(XMLFile)	VARCHAR	VARCHAR 中的 XML 檔名

範例：下列範例擷取 XMLVARCHAR 並將它儲存在記憶體作爲 VARCHAR 資料類型：

```
EXEC SQL SELECT db2xml.varchar(order) from sales_tab
```

使用 Content() 超載 UDF

使用 Content() UDF 從外部儲存體將文件內容擷取到記憶體，或從內部儲存體將文件匯出到 DB2 伺服器上的*外部檔*。

例如，您可能有一個儲存爲 XMLFILE 的 XML 文件 且您希望以記憶體操作，您可以使用 Content() UDF，其可採用 XMLFILE 資料類型以作爲輸入及傳回的 CLOB。

Content() UDF 執行兩個不同擷取函數，這根據指定的資料類型而定。其中包括：

從外部儲存體擷取某個文件，然後將該文件放在記憶體

將文件儲存爲外部檔時，您可使用 Content() 將 XML 文件擷取到記憶體緩衝區或 CLOB 定位器. 使用下列函數語法，其中 *xmlobj* 是查詢的 XML 直欄：

XMLFILE to CLOB：從檔案擷取資料並匯出到 CLOB 定位器。

內容(*xmlobj* XMLFile)

從內部記憶體擷取文件，並將它匯出到外部檔

您亦可以使用 Content()，來擷取 XML 文件，該文件儲存於 DB2 中有如 XMLCLOB 資料類型，並將它匯出至資料庫伺服器檔案系統中的檔案。它傳回 VARCHAR 類型的檔案名稱。 使用下列函數語法，其中 *xmlobj* 是查詢的 XML 直欄且*filename* 是外部檔。*XML type* 可以是 XMLVARCHAR 或 XMLCLOB 資料類型。

外部檔的 *XML type*：擷取 XML 內容，該內容儲存爲一種 XML 資料類型，並將它匯出到外部檔。

內容(*xmlobj* XML type, *filename* varchar(512))

其中：

xmlobj 是 XML 直欄名稱，從該直欄擷取 XML 內容； *xmlobj* 可以是類型 XMLVARCHAR 或 XMLCLOB。

filename
是在其中儲存 XML 資料的檔案名稱。

在下面範例中，有*內含的 SQL* 的一小段 C 程式會說明如何從檔案將 XML 文件擷取到記憶體。本範例假設 BOOK 直欄是 XMLFILE 類型。

```
EXEC SQL BEGIN DECLARE SECTION;
             SQL TYPE IS CLOB_LOCATOR xml_buff;
          EXEC SQL END DECLARE SECTION;
          EXEC SQL CONNECT TO SALES_DB
          EXEC SQL DECLARE c1 CURSOR FOR
              SELECT Content(order) from sales_tab
                        EXEC SQL OPEN c1;
          do {
             EXEC SQL FETCH c1 INTO :xml_buff;
             if (SQLCODE != 0) {
                break;
                }
             else {
               /* do whatever you need to do with the XML doc in buffer */
               }
             }
          EXEC SQL CLOSE c1;
          EXEC SQL CONNECT RESET;
```

擷取元素內容及屬性值

您可從一個或多個 XML 文件擷取 (取出) *元素*或*屬性*值的內容 (單一文件或集合文件搜尋)。XML Extender 對每一種 SQL 資料類型提供一些使用者定義的取出函數，您可在 SQL SELECT 子句指定這些函數。

擷取元素及屬性的內容及值有助於應用程式的開發，因爲您可存取 XML 資料作爲關聯式資料。例如，您可能有 1000 個 XML 文件，這些文件儲存在 SALES_TAB 表格的 ORDER 直欄中。您可使用下列 SQL 陳述式，在 SELECT 子句中以 extracting UDF 擷取此資訊，來擷取所有已訂購項目的客戶姓名：

```
SELECT extractVarchar(Order, '/Order/Customer/Name') from sales_order_view
   WHERE price > 2500.00
```

在本範例中，extracting UDF 從 ORDER 直欄擷取 <customer> 元素作爲 VARCHAR 資料類型。位置路徑爲 /Order/Customer/Name (關於位置路徑語法，請參閱 第48頁的『位置路徑』)。此外，使用 WHERE 子句減少回覆值數目，該子句指定僅具有次元素 <ExtendedPrice> <customer> 元素內容大於 2500.00 的值。

取出元素內容或屬性值：使用下列語法作爲表格或*純量函數*，使用列示在第109頁的表13的 extracting UDF。

extract*retrieved_datatype*(*xmlobj*, *path*)

其中：

retrieved_datatype

是從 extracting 函數傳回的資料類型；它可以是下列其中一種類型：

- INTEGER
- SMALLINT
- DOUBLE
- REAL
- CHAR
- VARCHAR
- CLOB
- DATE
- TIME
- TIMESTAMP
- FILE

xmlobj 是 XML 直欄名稱，從該直欄取出元素或屬性。本直欄必須定義成下列其中一種 XML 使用者定義的類型：

- XMLVARCHAR
- XMLCLOB as LOCATOR
- XMLFILE

path 是 XML 文件中的元素或屬性的位置路徑 (如 /Order/Customer/Name)。關於位置路徑語法，請參閱 第48頁的『位置路徑』。

表13 顯示 extracting 函數，這些函數使用純量及表格格式：

表 13. XML Extender extracting 函數

純量函數	表格函數	傳回的直欄名稱 (表格函數)	回覆類型
extractInteger()	extractIntegers()	returnedInteger	INTEGER
extractSmallint()	extractSmallints()	returnedSmallint	SMALLINT
extractDouble()	extractDoubles()	returnedDouble	DOUBLE
extractReal()	extractReals()	returnedReal	REAL
extractChar()	extractChars()	returnedChar	CHAR
extractVarchar()	extractVarchars()	returnedVarchar	VARCHAR

表 13. XML Extender extracting 函數 (繼續)

純量函數	表格函數	傳回的直欄名稱 (表格函數)	回覆類型
extractCLOB()	extractCLOBs()	returnedCLOB	CLOB
extractDate()	extractDates()	returnedDate	DATE
extractTime()	extractTimes()	returnedTime	TIME
extractTimestamp()	extractTimestamps()	returnedTimestamp	TIMESTAMP

更新 XML 資料

透過 XML Extender，您可取代 XML 直欄資料以更新整個 XML 文件，或者更新指定的元素值或屬性值。

作業概觀：

1. 確定 XML 表格有儲存 XML 文件，然後決定要擷取什麼資料。
2. 選擇一種方法更新 DB2 表格中的資料 (強制轉型函數或 UDF)。
3. 指定 SQL 查詢，該查詢指定要更新的 XML 表格及直欄。

重要事項：更新爲 XML 啓用的直欄時， XML Extender 會自動更新輔助表格以反映變更。不過，在藉由變更對應的 XML 元素或屬性值來更新儲存在 XML 直欄的原始 XML 文件之前，不要直接更新這些表格。 因爲直接更新會產生資料不一致的問題。

更新 XML 文件：

使用下列其中一種方法：

使用預設強制轉型函數

就每一個使用者定義的類型 (UDT) 而言，預設強制轉型函數是要將 SQL 基本類型強制轉型成 UDT。您可使用 XML Extender 提供的強制轉型函數更新 XML 文件。第105頁的表9顯示一些提供的強制轉型函數，並假設 ORDER 直欄是由 XML Extender 提供的另一個 UDT 所建立。

範例：假設 `xml_buf` 是定義成 VARCHAR 類型的主變數，從強制轉型的 VARCHAR 類型更新 XMLVARCHAR 類型。

```
UPDATE sales_tab VALUES('123456', 'Sriram Srinivasan',
  db2xml.XMLVarchar(:xml_buff))
```

使用儲存 UDF

就每個 XML Extender UDT 而言，儲存 UDF 是要從不是它的基本類型的資源將資料匯入 DB2。您可使用儲存 UDF 透過取代 XML 文件來更新整個文件。

範例：下例範例使用 XMLVarcharFromFile() 函數更新 XML 文件：

```
UPDATE sales_tab
       set order = XMLVarcharFromFile('c:\dxx\samples\cmd\getstart.xml')
   WHERE sales_person = 'Sriram Srinivasan'
```

前述範例從 `c:\dxx\samples\cmd\getstart.xml` 檔案將 XML 物件更新至 SALES_TAB 表格的 ORDER 直欄。

XML Extender 提供的儲存函數列示，請參閱第105頁的表10。

更新 XML 文件的特定元素及屬性：

使用 Update() UDF 指定特定變更，而不是更新整個文件。使用 UDF 時，請指定一個位置路徑以及要取代的元素值或屬性值 (由位置路徑代表)。 (關於位置路徑語法，請參閱 第48頁的『位置路徑』。) 您不必編輯 XML 文件：XML Extender 會自動變更。

`Update(`*xmlobj*`, `*path*`, `*value*`)`

在本範例中：

xmlobj 是要更新元素值或屬性值的 XML 直欄名稱。

path 是要更新元素或屬性的位置路徑。關於位置路徑語法，請參閱 第48頁的『位置路徑』。

值 是要更新的值。

範例：下列陳述式使用 Update() UDF，更新字串 IBM 的 <Customer> 元素值：

```
UPDATE sales_tab
       set order = Update(order, '/Order/Customer/Name', 'IBM')
   WHERE sales_person = 'Sriram Srinivasan'
```

搜尋 XML 文件

搜尋 XML 資料類似擷取 XML 資料：這兩種技術擷取資料供進一步操作，但它們搜尋時是使用 WHERE 子句定義述詞作爲擷取準則。

XML Extender 提供數種方法在 XML 直欄中搜尋 XML 文件，這根據應用程式的需求而定。 它提供搜尋文件結構的功能，並根據元素內容及屬性值傳回結果。 您可搜尋 XML 直欄及它的輔助表格畫面，直接搜尋輔助表格以增進效能，或使用具

有 WHERE 子句的 extracting UDF。此外，您可使用 DB2 Text Extender 並在結構內容中搜尋直欄資料是否有某個字串。

透過 XML Extender 您可在輔助表格直欄使用索引執行高速搜尋，這些直欄包含那些從 XML 文件取出的 XML 元素內容或屬性值。指定某個元素或屬性的資料類型之後，您可以在 SQL 中搜尋一般資料類型或執行範圍搜尋。例如，在採購訂單範例中，您可搜尋長期價格超過 2500.00 的全部訂單。

此外，您可使用 DB2 UDB Text Extender 執行結構式文字搜尋或全文搜尋。例如，您可製作一個含有 XML 格式之履歷的 RESUME 直欄。您可能想要有 Java 技能的全部應徵者姓名。您可使用 DB2 Text Extender 在 XML 文件中搜尋 <skill> 元素含有字串 JAVA 的全部履歷。

下列各節說明搜尋方法：

- 『根據結構搜尋 XML 文件』
- 第114頁的『使用 Text Extender 執行結構式文字搜尋』

根據結構搜尋 XML 文件

使用 XML Extender 搜尋特性，您可根據文件結構 (亦即根據元素及屬性) 在直欄中搜尋 XML 資料。若要搜尋直欄資料，請根據文件元素及屬性的相配情形，以數種方法使用 SELECT 陳述式並傳回*結果集*。您可使用下列方法搜尋直欄資料：

- 使用直接查詢搜尋輔助表格
- 從*合併概略表*搜尋
- 使用 extracting UDF 搜尋
- 搜尋多次出現的元素或屬性

這些方法說明於下列各節，並在下列實務範例中使用範例。 應用程式表格 SALES_TAB 有一個名稱是 ORDER 的 XML 直欄。 本直欄有三個輔助表格：ORDER_SIDE_TAB、PART_SIDE_TAB 及 SHIP_SIDE_TAB。 啓用 ORDER 直欄之後指定預設概略表 sales_order_view，然後使用下列 CREATE VIEW 陳述式合併這些表格：

```
CREATE VIEW sales_order_view(invoice_num, sales_person, order,
                             order_key, customer, part_key, price, date)
     AS
     SELECT sales_tab.invoice_num, sales_tab.sales_person, sales_tab.order,
            order_side_tab.order_key, order_side_tab.customer,
            part_side_tab.part_key, ship_side_tab.date
     FROM sales_tab, order_side_tab, part_side_tab, ship_side_tab
     WHERE sales_tab.invoice_num = order_side_tab.invoice_num
               AND sales_tab.invoice_num = part_side_tab.invoice_num
               AND sales_tab.invoice_num = ship_side_tab.invoice_num
```

使用直接查詢搜尋輔助表格

輔助表格編製索引時，具有次查詢搜尋的直接查詢對結構式搜尋提供最佳效能。您可使用查詢或次查詢正確地搜尋輔助表格。

範例：下列陳述式使用查詢及次查詢直接搜尋輔助表格：

```
SELECT sales_person from sales_tab
       WHERE invoice_num in
          (SELECT invoice_num from part_side_tab
           WHERE price > 2500.00)
```

在本範例中，invoice_num 是 SALES_TAB 表格中的主要鍵。

從合併概略表搜尋

您可使 XML Extender 建立預設概略表，該概略表使用唯一 ID 合併應用程式表格與輔助表格。您可使用本預設概略表，或合併應用程式表格與輔助表格的任何概略表，搜尋直欄資料並查詢輔助表格。本方法提供應用程式表格及它的輔助表格的單一虛擬概略表。不過，建立愈多輔助表格，查詢成本就愈高。

要訣：建立自己的概略表時，您可使用 *root_id* 或 DXXROOT_ID (由 XML Extender 建立) 合併表格。

範例：下列陳述式搜尋概略表

```
SELECT sales_person from sales_order_view
   WHERE price > 2500.00
```

SQL 陳述式從合併概略表 sales_order_view 表格傳回 sales_person 值，該表格有一些價格大於 2500.00 的行項目訂單。

使用 extracting UDF 搜尋

沒有爲應用程式表格建立索引或輔助表格時，您也可使用 XML Extender 的 extracting UDF 搜尋元素及屬性。使用 extracting UDF 掃描 XML 資料的成本很高，應該只與 WHERE 子句一起使用，來限制搜尋包含的 XML 文件數目。

範例：下列陳述式使用 extracting XML Extender UDF 搜尋：

```
SELECT sales_person from sales_tab
     WHERE extractVarchar(order, '/Order/Customer/Name')
    like '%IBM%'
AND invoice_num > 100
```

在本範例中，extracting UDF 使用 IBM 值取出 </Order/Customer/Name> 元素。

搜尋多次出現的元素或屬性

搜尋多次出現的元素或屬性時，請使用 DISTINCT 子句防止重複值。

範例：下列陳述式使用 DISTINCT 子句搜尋：

```
SELECT sales_person from sales_tab
       WHERE invoice_num in
              (SELECT DISTINCT invoice_num from part_side_tab
          WHERE price > 2500.00)
```

在本範例中，DAD 檔指定 /Order/Part/Price 有多次出現的項目，並且爲它建立一個輔助表格 PART_SIDE_TAB。 PART_SIDE_TAB 表格可能有一個以上，具有相同 invoice_num 的橫列。 使用 DISTINCT 只傳回唯一值。

使用 Text Extender 執行結構式文字搜尋

搜尋 XML 文件結構時，XML Extender 會搜尋那些轉換成一般資料類型的元素及屬性值，但不搜尋文字。 您可使用 DB2 UDB Text Extender 在爲 XML 啓用的直欄執行結構或全文搜尋。 在 DB2 UDB 版本 6.1 或更新版中，Text Extender 支援 XML 文件搜尋。

結構式文字搜尋

搜尋那些根據 XML 文件樹狀結構的字串。 例如，若有 /Order/Customer/Name 的文件結構，而且想要在 <Customer> 次元素中搜尋字串 『IBM』，那麼您可使用結構式文字搜尋。 本文件在 <Comment> 次元素中可能也有 IBM 這個字串，或作爲產品的零件名稱。 在指定的元素中結構式文字只搜尋此字串。 在本範例中，只找到在 </Order/Customer/Name> 次元素中有 IBM 的文件；不會傳回在其它元素中有 IBM 但在 </Order/Customer/Name> 次元素中沒有 IBM 的文件。

全文搜尋

搜尋文件結構中的字串，不考慮元素或屬性。 使用前一個範例時，會傳回有 IBM 這個字串的全部文件，而不管 IBM 字串出現在哪裡。

若要使用 Text Extender 搜尋，您必須按照下列說明安裝 DB2 Text Extender 並啓用資料庫及表格。若要瞭解如何使用 Text Extender 搜尋，請參閱 *DB2 Universal Database Text Extender Administration and Programming* 中有關使用 Text Extender 的 UDF來搜尋這一章。

啓用 Text Extender 的 XML 直欄

假設您有一個已啓用 XML 的資料庫，請執行下列步驟啓用 Text Extender 來搜尋已啓用 XML 的直欄內容。例如，資料庫名稱是 SALES_DB，表格名稱是 ORDER， XML 直欄名稱是 XVARCHAR 及 XCLOB：

1. 如何安裝 Text Extender，請參閱 Extenders CD 上的 install.txt 檔。

2. 從下列其中一個位置輸入 **txstart** 指令：
   - 在 UNIX 作業系統上，從案例擁有者的指令提示輸入此指令。
   - 在 Windows NT 上，從指定 DB2INSTANCE 的指令視窗輸入此指令。
3. 開啓 Text Extender 指令行視窗。本步驟假設您有名稱是 SALES_DB 的資料庫，以及名稱是 ORDER 的表格，該表格有兩個 XML 直欄，名稱分別是 XVARCHAR 及 XCLOB。 您可能需要在 dxx\samples\c 中執行範例程式。
4. 連接到資料庫。在 **db2tx** 指令提示下，鍵入：

   ```
   'connect to SALES_DB'
   ```
5. 啓用 Text Extender 的資料庫。

   從 **db2tx** 指令提示，鍵入：

   ```
   'enable database'
   ```
6. 啓用 Text Extender 的 XML 表格中的直欄，定義 XML 文件的資料類型、語言、字碼頁以及有關此直欄的其它資訊。
   - 就 VARCHAR 直欄 XVARCHAR 而言，請鍵入：

     ```
     'enable text column order xvarchar function db2xml.varchartovarchar handle
     varcharhandle ccsid 850 language us_english format xml indextype precise
         indexproperty sections_enabled
     documentmodel (Order) updateindex update'
     ```
   - 就 CLOB 直欄 XCLOB 而言，請鍵入：

     ```
     'enable text column order xclob function db2xml.clob handle clobhandle
      ccsid 850 language us_english indextype precise updateindex update'
     ```
7. 檢查索引狀態。
   - 就 XVARCHAR 直欄而言，請鍵入：

     ```
     get index status order handle varcharhandle
     ```
   - 就 XCLOB 直欄而言，請鍵入：

     ```
     get index status order handle clobhandle
     ```
8. 在名稱是 desmodel.ini 的文件模型 ini 檔案中定義 XML 文件模型。本檔案位於：UNIX 的 /db2tx/txins000， Windows NT 的 \instance\\db2tx\txins000，以及起始設定檔中的一些區段。例如，就 textmodel.ini 而言：

   ```
   ;list of document models
   [MODELS]
   modelname=Order

   ; an 'Order' document model definition
   ; left side = section name identifier
   ; right side = section name tag

   [Order]
   Order = /Order
   ```

```
Order/Customer/Name = /Order/Customer/Name
Order/Customer/Email = /Order/Customer/Email
Order/Part/@color = /Order/Part/@color
Order/Part/Shipment/ShipMode = /Order/Part/Shipment/ShipMode
```

使用 Text Extender 搜尋文字

Text Extender 的搜尋功能很適合與 XML Extender 文件結構式搜尋一起使用。建議的方法是建立一個搜尋文件元素或屬性的查詢，並使用 Text Extender 搜尋元素內容或屬性值。

範例：下列陳述式搜尋具有 Text Extender 的 XML 文件文字。在 DB2 指令視窗上，鍵入：

```
'connect to SALES_DB'
'select xvarchar from order where db2tx.contains(varcharhandle,
 'model Order section(Order/Customer/Name) "Motors"')=1'
'select xclob from order where db2tx.contains(clobhandle,
 'model Order section(Order/Customer/Name) "Motors"')=1'
```

Text Extender Contains() UDF 搜尋。

本範例不含需要使用 Text Extender 搜尋直欄資料的所有步驟。若要瞭解 Text Extender 搜尋概念及功能，請參閱 *DB2 Universal Database Text Extender Administration and Programming* 中有關使用 Text Extender 的 UDF 來搜尋這一章。

刪除 XML 文件

使用 SQL DELETE 陳述式從 XML 直欄刪除 XML 文件列。您可指定一些 WHERE 子句修正要刪除哪些文件。

範例：下列陳述式刪除 <ExtendedPrice> 的值大於 2500.00 的全部文件。

```
DELETE from sales_tab
       WHERE invoice_num in
          (SELECT invoice_num from part_side_tab
           WHERE price > 2500.00)
```

第6章 管理 XML 集合資料

XML 集合是一組關聯式表格，這些表格含有對映 XML 文件的資料。這個存取及儲存方法可讓您從現存資料製作 XML 文件、分解 XML 文件，以及使用 XML 作爲交換方法。

關聯式表格可以是分解 XML 文件時 XML Extender 建立的新表格，或是已含有資料的現存表格，此資料配合 XML Extender 使用以建立應用程式的 XML 文件。這些表格中的直欄資料不含 XML 標示；它含有分別與元素及屬性相關的內容及值。 儲存程序作爲儲存、擷取、更新、搜尋及刪除 XML 集合資料的存取及儲存方法。

管理 XML 集合的相關資訊，請參閱下列各節：

- 『製作 DB2 資料的 XML 文件』
- 第125頁的『分解 XML 文件成爲 DB2 資料』

大小參數，*UDB_SIZE*

XML Extender 儲存程序語法使用參數 *UDB_SIZE* 來指定某些字元參數的大小。 根據執行中的 DB2 UDB 版本決定 *UDB_SIZE* 值。 下面列示顯示不同 DB2 UDB 版本的欄位大小值。 當宣告 XML Extender 儲存程序時，請使用正確值。

DB2 UDB 版本	大小
6.1	32
7.1	64

製作 DB2 資料的 XML 文件

組合是指從 XML 集合中的關聯式資料產生的一組 XML 文件。您可使用儲存程序製作 XML 文件。若要使用這些儲存程序，您必須建立 DAD 檔，該檔案指定 XML 文件與 DB2 表格結構之間的對映。儲存程序使用 DAD 檔製作 XML 文件。關於如何建立 DAD 檔，請參閱第50頁的『規劃 XML 集合』。

開始之前

- 將 XML 文件結構對映到含有元素內容及屬性值的關聯式表格。
- 選取對映方法：SQL 對映或 RDB_node 對映。

- 準備 DAD 檔。完整明細，請參閱第50頁的『規劃 XML 集合』。
- 選用性地啓用 XML 集合。

製作 XML 文件

XML Extender 提供兩個儲存程序 (dxxGenXML() 及 dxxRetrieveXML()) 製作 XML 文件。

dxxGenXML()

本儲存程序用於那些偶而才更新的應用程式，或用於那些不想要管理 XML 資料成爲額外負荷的應用程式。儲存程序 dxxGenXML() 不需要已啓用的集合；它使用 DAD 檔來代替。

儲存程序 dxxGenXML() 使用儲存在 XML 集合表格的資料建構 XML 文件， DAD 檔中的 <Xcollection> 元素指定這些文件。本儲存程序將 XML 文件當作一個橫列插入*結果表格*中。您亦可開啓結果表格上的游標，然後提取結果集。 應該由應用程式建立結果表格，且該表格一定有一個 VARCHAR、CLOB、XMLVARCHAR 或 XMLCLOB 類型的直欄。此外，若在 DAD 檔中將驗證元素指定成 YES，XML Extender 就會新增 INTEGER 類型的 DXX_VALID 直欄，然後插入值 1 代表有效 XML 文件，以及插入值 0 代表無效文件。

儲存程序 dxxGenXML() 也可讓您指定在結果表格中建立的最大列數。此動作縮短處理時間。儲存程序傳回表格中的實際列數，以及任何回覆碼及訊息。

分解的對應儲存程序是 dxxShredXML()；，它也使用 DAD 作爲輸入參數而且不需要啓用 XML 集合。

製作 XML 集合：dxxGenXML()

使用下列儲存程序宣告將儲存程序呼叫嵌入應用程式：

```
dxxGenXML(CLOB(100K)     DAD,             /* input */
              char(UDB_SIZE) resultTabName,   /* input */
              integer        overrideType,    /* input */
              varchar(1024)  override,        /* input */
              integer        maxRows,         /* input */
              integer        numRows,         /* output */
            long           returnCode,      /* output */
            varchar(1024)  returnMsg)       /* output */
```

完整語法及範例，請參閱第188頁的『dxxGenXML()』。

範例：下例範例製作 XML 文件：

```
#include "dxx.h"
#include "dxxrc.h"
```

```
EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
     SQL TYPE is CLOB(100K) dad;              /* DAD */
     SQL TYPE is CLOB_FILE  dadFile;          /* dad file */
         char    result_tab[32];  /* name of the result table */
              char    override[2];      /* override, will set to NULL*/
              short   overrideType;     /* defined in dxx.h */
         short   max_row;          /* maximum number of rows */
         short   num_row;          /* actual number of rows */
         long    returnCode;       /* return error code */
              char    returnMsg[1024];  /* error message text */
         short                dad_ind;
         short   rtab_ind;
         short   ovtype_ind;
         short   ov_inde;
         short   maxrow_ind;
         short   numrow_ind;
         short                returnCode_ind;
         short                returnMsg_ind;

          EXEC SQL END DECLARE SECTION;

         /* create table */
         EXEC CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

     /* read data from a file to a CLOB */
     strcpy(dadfile.name,"c:\dxx\samples\dad\getstart_xcollection.dad");
     dadfile.name_length = strlen("c:\dxx\samples\dad\getstart_xcollection.dad");
     dadfile.file_options = SQL_FILE_READ;
     EXEC SQL VALUES (:dadfile) INTO :dad;
         strcpy(result_tab,"xml_order_tab");
              override[0] = '\0';
              overrideType = NO_OVERRIDE;
         max_row = 500;
         num_row = 0;
         returnCode = 0;
         msg_txt[0] = '\0';
         collection_ind = 0;
        dad_ind = 0;
         rtab_ind = 0;
              ov_ind = -1;
         ovtype_ind = 0;
         maxrow_ind = 0;
         numrow_ind = -1;
         returnCode_ind = -1;
         returnMsg_ind = -1;

        /* Call the store procedure */
     EXEC SQL CALL dxxGenXML(:dad:dad_ind;
                :result_tab:rtab_ind,
           :overrideType:ovtype_ind,:override:ov_ind,
                :max_row:maxrow_ind,:num_row:numrow_ind,
                      :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

呼叫儲存程序之後的結果表格含有 250 列，因爲在 DAD 檔中指定的 SQL 查詢建立 250 個 XML 文件。

dxxRetrieveXML()

本儲存程序用於那些經常更新的應用程式。因爲一直重複相同作業，所以增進效能相當重要。在儲存程序中啓用 XML 集合並使用集合名稱增進效能。

除了儲存程序 dxxRetrieveXML() 使用啓用的 XML 集合名稱而不是 DAD 檔名以外，本儲存程序與儲存程序 dxxGenXML() 的運作方式相同。啓用 XML 集合之後，某個 DAD 檔會儲存在 XML_USAGE 表格中。因此，XML Extender 會擷取 DAD 檔，此後 dxxRetrieveXML() 與 dxxGenXML() 相同。

dxxRetrieveXML() 容許用於組合及分解的相同 DAD 檔。本儲存程序也可用於擷取分解的 XML 文件。

分解的對應儲存程序是 dxxInsertXML()；它也使用啓用的 XML 集合名稱。

製作 XML 集合：dxxRetrieveXML()

使用下列儲存程序宣告將儲存程序呼叫嵌入應用程式：

```
dxxRetrieveXML(char(UDB_SIZE) collectionName,  /* input */
               char(UDB_SIZE) resultTabName,   /* input */
               integer        overrideType,    /* input */
               varchar(1024)  override,        /* input */
               integer        maxRows,         /* input */
               integer        numRows,         /* output */
             long           returnCode,       /* output */
             varchar(1024)  returnMsg)        /* output */
```

完整的語法及範例，請參閱 第191頁的『dxxRetrieveXML()』。

範例：下例是 dxxRetrieveXML() 的呼叫範例。 假設使用 XML_ORDER_TAB 名稱建立結果表格，而且該表格有一個 XMLVARCHAR 類型的直欄。

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
         char    collection[32];  /* dad buffer */
         char    result_tab[32];  /* name of the result table */
              char    override[2];      /* override, will set to NULL*/
              short   overrideType;     /* defined in dxx.h */
         short   max_row;         /* maximum number of rows */
         short   num_row;         /* actual number of rows */
         long    returnCode;      /* return error code */
              char    returnMsg[1024];  /* error message text */
         short   dadbuf_ind;
         short   rtab_ind;
         short   ovtype_ind;
         short   ov_inde;
         short   maxrow_ind;
         short   numrow_ind;
         short                  returnCode_ind;
         short                  returnMsg_ind;
```

```
 EXEC SQL END DECLARE SECTION;

/* create table */
EXEC SQL CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

/* initialize host variable and indicators */
strcpy(collection,"sales_ord");
strcpy(result_tab,"xml_order_tab");
     override[0] = '\0';
     overrideType = NO_OVERRIDE;
max_row = 500;
num_row = 0;
returnCode = 0;
msg_txt[0] = '\0';
collection_ind = 0;
rtab_ind = 0;
     ov_ind = -1;
ovtype_ind = 0;
maxrow_ind = 0;
numrow_ind = -1;
returnCode_ind = -1;
returnMsg_ind = -1;

/* Call the store procedure */
EXEC SQL CALL dxxRetrieve(:collection:collection_ind;
       :result_tab:rtab_ind,
 :overrideType:ovtype_ind,:override:ov_ind,
       :max_row:maxrow_ind,:num_row:numrow_ind,
             :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

動態置換 DAD 檔中的值

就動態查詢而言，您可使用這兩個可選用的參數置換 DAD 檔中的條件： *override* 及 *overrideType*。根據來自 *overrideType* 的輸入，應用程式可置換 SQL 對映的 <SQL_stmt> 標示值，或置換 DAD 中的 RDB_node 對映之 RDB_nodes 中的條件。

這些參數有下列值及規則：

overrideType

本參數是必要的輸入參數 (IN)，它標示 *override* 參數的類型。 *overrideType* 有下列值：

NO_OVERRIDE

指定不置換 DAD 檔中的條件。

SQL_OVERRIDE

指定以 SQL 陳述式置換 DAD 檔中的條件。

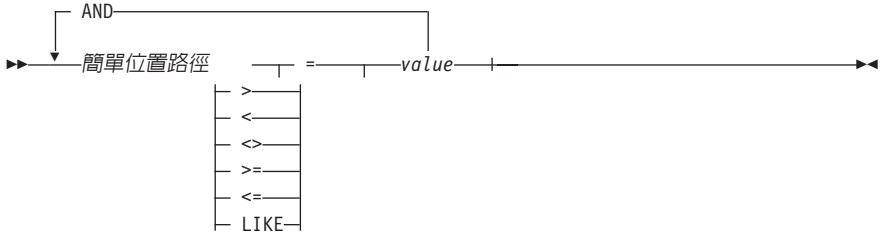
XML_OVERRIDE

指定以 XPath 型條件置換 DAD 檔中的條件。

置換

本參數是可選用的輸入參數 (IN)，它指定 DAD 檔的置換條件。輸入值語法與 *overrideType* 中指定的值相對應。

- 若指定 NO_OVERRIDE，那麼輸入值是 NULL 字串。
- 若指定 SQL_OVERRIDE，那麼輸入值是有效 SQL 陳述式。 ***必要的：***若使用 SQL_OVERRIDE 及 SQL 陳述式，那麼您必須在 DAD 檔中使用 SQL 對映方法。 輸入 SQL 陳述式置換 DAD 檔中的 <SQL_stmt> 元素指定之 SQL 陳述式。
- 若使用 XML_OVERRIDE，那麼輸入值是含有一個或多個表示式的字串。***必要的：***若使用 XML_OVERRIDE 及一個表示式，那麼您必須在 DAD 檔中使用 RDB_node 對映方法。輸入 XML 表示式置換在 DAD 檔中指定的 RDB_node 條件。此表示式使用下列語法：



其中：

簡單位置路徑

使用 XPATH 定義之語法的簡單位置路徑；關於語法，請參閱 第49頁的表5。

運算子

可以使用空白來將運算子與表示式的其它部份分隔。

值 數值或單引號字串。

您可以在運算子前後選擇性加上空白；對 LIKE 運算子而言，空白是必要的。

指定 XML_OVERRIDE 值之後，指定的表示式會置換 text_node 或 attribute_node 中符合簡單位置路徑的 RDB_node 條件。

XML_OVERRIDE 不是完全地 XPath 順從。簡式位置路徑僅用來識別對映至直欄的元素或屬性。

範例：

下列範例使用 SQL_OVERRIDE 及 XML_OVERRIDE 顯示動態置換。本書中大部份的儲存程序範例使用 NO_OVERRIDE。

範例：使用 SQL_OVERRIDE 的儲存程序。

```
include "dxx.h"
include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
         char    collection[32];  /* dad buffer */
         char    result_tab[32];  /* name of the result table */
         char    override[256];   /* override, SQL_stmt */
         short   overrideType;    /* defined in dxx.h */
         short   max_row;         /* maximum number of rows */
         short   num_row;         /* actual number of rows */
         long    returnCode;      /* return error code */
              char    returnMsg[1024];  /* error message text */
         short   rtab_ind;
         short   ovtype_ind;
         short   ov_inde;
         short   maxrow_ind;
         short   numrow_ind;
         short                  returnCode_ind;
         short                  returnMsg_ind;

          EXEC SQL END DECLARE SECTION;

         /* create table */
         EXEC CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

         /* initialize host variable and indicators */
         strcpy(collection,"sales_ord");
         strcpy(result_tab,"xml_order_tab");
         sprintf(override,"%s %s %s %s %s %s %s",
                   "SELECT o.order_key, customer, p.part_key, quantity, price,",
                   "tax, ship_id, date, mode ",
                   "FROM order_tab o, part_tab p,",
                   "table(select substr(char(timestamp(generate_unique())),16",
                   "as ship_id,date,mode from ship_tab)as s",
                   "WHERE p.price > 50.00 and s.date >'1998-12-01' AND",
                   "p.order_key = o.order_key and s.part_key = p.part_key");
        overrideType = SQL_OVERRIDE;
         max_row = 500;
         num_row = 0;
         returnCode = 0;
         msg_txt[0] = '\0';
         collection_ind = 0;
         rtab_ind = 0;
         ov_ind = 0;
         ovtype_ind = 0;
         maxrow_ind = 0;
```

```
         numrow_ind = -1;
         returnCode_ind = -1;
         returnMsg_ind = -1;

        /* Call the store procedure */
         EXEC SQL CALL dxxRetrieve(:collection:collection_ind;
                :result_tab:rtab_ind,
                :overrideType:ovtype_ind,:override:ov_ind,
                :max_row:maxrow_ind,:num_row:numrow_ind,
                      :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

在本範例中，DAD 檔中的 <xcollection> 元素必須有 <SQL_stmt> 元素。 *override* 參數置換 <SQL_stmt> 值，其做法是將價格變成大於 50.00，將日期變成大於 1998-12-01。

範例：使用 XML_OVERRIDE 的儲存程序。

```
include "dxx.h"
include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
         char    collection[32];  /* dad buffer */
         char    result_tab[32];  /* name of the result table */
         char    override[256];   /* override, SQL_stmt */
         short   overrideType;    /* defined in dxx.h */
         short   max_row;         /* maximum number of rows */
         short   num_row;         /* actual number of rows */
         long    returnCode;      /* return error code */
              char    returnMsg[1024];  /* error message text */
         short   dadbuf_ind;
         short   rtab_ind;
         short   ovtype_ind;
         short   ov_inde;
         short   maxrow_ind;
         short   numrow_ind;
         short                  returnCode_ind;
         short                  returnMsg_ind;

          EXEC SQL END DECLARE SECTION;

         /* create table */
         EXEC CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

         /* initialize host variable and indicators */
         strcpy(collection,"sales_ord");
         strcpy(result_tab,"xml_order_tab");
         sprintf(override,"%s %s",
                   "/Order/Part/Price > 50.00 AND ",
                   "Order/Part/Shipment/ShipDate > '1998-12-01'");
        overrideType = XML_OVERRIDE;
         max_row = 500;
         num_row = 0;
         returnCode = 0;
```

```
msg_txt[0] = '\0';
collection_ind = 0;
rtab_ind = 0;
ov_ind = 0;
ovtype_ind = 0;
maxrow_ind = 0;
numrow_ind = -1;
returnCode_ind = -1;
returnMsg_ind = -1;

/* Call the store procedure */
EXEC SQL CALL dxxRetrieve(:collection:collection_ind;
       :result_tab:rtab_ind,
       :overrideType:ovtype_ind,:override:ov_ind,
       :max_row:maxrow_ind,:num_row:numrow_ind,
             :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

在本範例中，DAD 檔中的 <collection> 元素有 Root element_node 的 RDB_node。*置換* 值是以 XML-content 爲依據。XML Extender 將簡單位置路徑轉換成對映的 DB2 直欄。

分解 XML 文件成為 DB2 資料

分解 XML 文件就是瓦解 XML 文件內的資料，然後將資料儲存到關聯式表格中。The XML Extender 提供儲存程序將來源 XML 文件的 XML 資料分解到關聯式表格中。若要使用這些儲存程序，那麼您必須建立 DAD 檔，該檔案指定 XML 文件與 DB2 表格結構之間的對映。儲存程序使用 DAD 檔分解 XML 文件。 如何建立 DAD 檔，請參閱第50頁的『規劃 XML 集合』。

為分解啓用 XML 集合

在大部份狀況中，您必須啓用 XML 集合之後才能使用儲存程序。 在下列狀況中，您需要啓用 XML 集合：

- 將 XML 文件分解成一些新表格時，必須啓用某 XML 集合，因爲在啓用該集合之後 XML Extender 會建立該集合中的全部表格。
- 保留那些多次出現的元素及屬性的順序很重要。 XML Extender 只保留在啓用集合時建立的表格中，多次出現的元素或屬性的順序。將 XML 文件分解成現存的關聯式表格時，不保證保留此順序。

若要在資料庫已有這些表格時自動傳送 DAD 檔，那麼您不必啓用 XML 集合。

分解表格大小限制

分解使用 RDB_node 對映，透過將元素值和屬性值解壓縮到表格列來指定 XML 文件如何分解成 DB2 表格。每一個 XML 文件中的值皆儲存在一或多個 DB2 表格。每一個表格最多儲存每一份文件中分解出來的 1024 列。

例如，如果 XML 文件分解成 5 個表格，則每一個表格最多可儲存該文件的 1024 列。如果表格含有多重文件的列，則每一份文件最多 1024 列。如果表格有 20 份文件，則最多 20,480 列，亦即每一份文件 1024 列。

使用重複出現的元素 (在 XML 結構中位置路徑出現一次以上的元素) 會影響列數。例如，一份文件含有一個元素 <Part> 出現 20 次，則文件可能在一個表格中分解成 20 列。使用出現多次的元素時，請考慮這個表格大小限制。

開始之前

- 將 XML 文件結構對映到含有元素內容及屬性值的關聯式表格。
- 準備 DAD 檔案，使用 RDB_node 對映。詳細資訊，請參閱第50頁的『規劃 XML 集合』。
- 選用性地啓用 XML 集合。

分解 XML 文件

XML Extender 提供兩個儲存程序 (dxxShredXML() 及 dxxInsertXML) 分解 XML 文件。

dxxShredXML()

本儲存程序用於那些偶而才更新的應用程式，或用於那些不想要管理 XML 資料成爲額外負荷的應用程式。儲存程序 dxxShredXML() 不需要啓用的集合；它使用 DAD 檔。

儲存程序 dxxShredXML() 使用兩個輸入參數，一個 DAD 檔及要分解的 XML 文件；它傳回兩個輸出參數：一個回覆碼及一個傳回訊息。

儲存程序 dxxShredXML() 根據輸入 DAD 檔中的 <Xcollection> 規格，將 XML 文件插入 XML 集合。假設用於 DAD 檔的 <Xcollection> 的表格已存在，而且假設直欄符合在 DAD 對映中指定的資料類型。若上述假設不是眞的，就會傳回錯誤訊息。然後儲存程序 dxxShredXML() 會分解 XML 文件，而且將無標示的 XML 資料插入在 DAD 檔中指定的表格。

分解的對應儲存程序是 dxxGenXML()；它也使用 DAD 作爲輸入參數而且不必啓用 XML 集合。

分解 XML 集合：dxxShredXML()

使用下列儲存程序宣告將儲存程序呼叫嵌入應用程式：

```
dxxShredXML(CLOB(100K)    DAD,            /* input */
            CLOB(1M)        xmlobj,         /* input */
            long            returnCode,     /* output */
            varchar(1024)   returnMsg)      /* output */
```

完整語法及範例，請參閱第195頁的『dxxShredXML()』。

範例：下列是呼叫 dxxShredXML() 的範例：

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
         SQL TYPE is CLOB      dad;             /* DAD*/
         SQL TYPE is CLOB_FILE dadFile;         /* DAD file*/
         SQL TYPE is CLOB      xmlDoc;          /* input XML document */
         SQL TYPE is CLOB_FILE xmlFile;          /* input XML file */
         long                  returnCode;       /* error code */
              char    returnMsg[1024];  /* error message text */
         short                 dad_ind;
         short                 xmlDoc_ind;
         short                 returnCode_ind;
         short                 returnMsg_ind;
          EXEC SQL END DECLARE SECTION;

         /* initialize host variable and indicators */
        strcpy(dadFile.name,"c:\dxx\samples\dad\getstart_xcollection.dad");
        dadFile.name_length=strlen("c:\dxx\samples\dad\getstart_xcollection.dad");
        dadFile.file_option=SQL_FILE_READ;
        strcpy(xmlFile.name,"c:\dxx\samples\cmd\getstart_xcollection.xml");
        xmlFile.name_length=strlen("c:\dxx\samples\cmd\getstart_xcollection.xml");
        xmlFile.file_option=SQL_FILE_READ;
        SQL EXEC VALUES (:dadFile) INTO :dad;
        SQL EXEC VALUES (:xmlFile) INTO :xmlDoc;
         returnCode = 0;
         returnMsg[0] = '\0';
        dad_ind = 0;
        xmlDoc_ind = 0;
         returnCode_ind = -1;
         returnMsg_ind = -1;

        /* Call the store procedure */
        EXEC SQL CALL db2xml!dxxShredXML(:dad:dad_ind;
                     :xmlDoc:xmlDoc_ind,
                     :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

dxxInsertXML()

本儲存程序用於那些經常更新的應用程式。因爲一直重複相同作業，所以增進效能相當重要。在儲存程序中啓用 XML 集合並使用集合名稱增進效能。除了儲存程序 dxxInsertXML() 使用啓用的 XML 集合作爲它的第一個輸入參數以外，儲存程序 dxxInsertXML() 與 dxxShredXML() 的運作方式相同。

儲存程序 dxxInsertXML() 將 XML 文件插入啓用的 XML 集合，該文件與某個 DAD 檔相關。 DAD 檔含有集合表格及對映的規格。根據 <Xcollection> 中的規格檢查或建立集合表格。然後儲存程序 dxxInsertXML() 根據對映分解 XML 文件，而且將無標示的 XML 資料插入已命名的 XML 集合的表格。

分解的對應儲存程序是 dxxRetrieveXML()；它也使用啓用的 XML 集合名稱。

分解 XML 集合：dxxInsertXML()

使用下列儲存程序宣告將儲存程序呼叫嵌入應用程式：

```
dxxInsertXML(char(UDB_SIZE) collectionName,   /* input */
             CLOB(1M)       xmlobj,           /* input */
             long           returnCode,       /* output */
             varchar(1024)  returnMsg)        /* output */
```

完整語法及範例，請參閱第197頁的『dxxInsertXML()』。

範例：下列是呼叫 dxxInsertXML() 的範例：

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
        char                collection[64];   /* name of an XML collection */
        SQL TYPE is CLOB_FILE xmlFile;        /* input XML file */
        SQL TYPE is CLOB      xmlDoc;         /* input XML doc */
        long                returnCode;       /* error code */
             char    returnMsg[1024];  /* error message text */
        short               collection_ind;
        short               xmlDoc_ind;
        short               returnCode_ind;
        short               returnMsg_ind;
         EXEC SQL END DECLARE SECTION;

        /* initialize host variable and indicators */
        strcpy(collection,"sales_ord")
        strcpy(xmlobj.name,"c:\dxx\samples\cmd\getstart_xcollection.xml");
        xmlobj.name_length=strlen("c:\dxx\samples\cmd\getstart_xcollection.xml");
        xmlobj.file_option=SQL_FILE_READ;
        SQL EXEC VALUES (:xmlFile) INTO (:xmlDoc);
        returnCode = 0;
        returnMsg[0] = '\0';
        collection_ind = 0;
        xmlobj_ind = 0;
        returnCode_ind = -1;
        returnMsg_ind = -1;

       /* Call the store procedure */
        EXEC SQL CALL db2xml!dxxInsertXML(:collection:collection_ind;
                    :xmlDoc:xmlDoc_ind,
                    :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

存取 XML 集合

您可更新、刪除、搜尋及擷取 XML 集合。不過請記得，使用 XML 集合的目的是爲了在資料庫表格中，儲存或擷取無標示的單純資料。現存資料庫表格中的資料與進來的 XML 文件無關；更新、刪除及搜尋作業實際上是由這些表格的一般 SQL 存取權組成。若從進來的 XML 文件分解此資料，那麼原始 XML 文件就會消失。

XML Extender 提供從 XML 集合概略表操作此資料的功能。使用 UPDATE 及 DELETE SQL 陳述式時，您可修改用於製作 XML 文件的資料，因而更新 XML 集合。

注意事項：

- 若要更新文件，請勿刪除含有表格主要鍵的橫列，該橫列是其它集合表格的外來鍵列。刪除主要鍵列及外來鍵列時，會刪除文件。

- 若要取代或刪除元素及屬性值，您可在低階表格刪除及插入橫列，而不刪除文件。
- 若要刪除文件，請刪除製作頂層 element_node (在 DAD 中指定該 element_node) 的橫列。

更新 XML 集合中的資料

XML Extender 可讓您更新儲存在 XML 集合表格中的無標示資料。更新 XML 集合表格值時，會更新 XML 元素文字或 XML 屬性值。此外，更新可從多次出現的元素或屬性刪除資料案例。

從 SQL 的觀點，變更元素值或屬性值是更新作業，刪除元素或屬性案例是刪除作業。從 XML 的觀點，只要 Root element_node 的元素文字或屬性值存在，XML 文件就存在，所以算是更新作業。

需求：若要更新 XML 集合中的資料，請遵循下列規則。

- 現存表格有主要鍵-外來鍵關係時，在集合表格之中指定這個關係。若現存表格沒有這個關係，請確定可以合併這些表格中的一些直欄。
- 包括 DAD 檔中指定的結合條件：
  - 就 SQL 對映而言，在 <SQL_stmt> 元素中
  - 就 RDB_node 對映而言，在 Root 元素節點的 RDB_node 中

更新元素及屬性值

在 XML 集合中，元素文字及屬性值都對映資料庫表格中的直欄。 無論直欄資料是先前已存在或分解自進來的 XML 文件，您都要使用一般 SQL 更新技術取代此資料。

若要更新元素或屬性值，請在含有結合條件 (在 DAD 檔中指定該條件) 的 SQL UPDATE 陳述式指定 WHERE 子句。

例如：

```
UPDATE SHIP_TAB
    set MODE = 'BOAT'
  WHERE MODE='AIR' AND PART_KEY in
    (SELECT PART_KEY from PART_TAB WHERE ORDER_KEY=68)
```

在 SHIP_TAB 表格中，<ShipMode> 元素值是從 AIR 更新成 BOAT，其中鍵值是 68。

刪除元素及屬性案例

若要刪除多次出現的元素或屬性來更新已製作的 XML 文件，請使用 WHERE 子句刪除含有欄位值 (對應元素或屬性值) 的橫列。只要沒有刪除含有頂層 element_node 值的橫列，那麼刪除元素值就會被視爲更新 XML 文件。

例如，在下列 DELETE 陳述式中，您指定 <shipment> 元素的其中一個次元素的唯一值來刪除該元素。

```
DELETE from SHIP_TAB
   WHERE DATE='1999-04-12'
```

指定 DATE 值刪除符合本值的橫列。製作的文件本來含有兩個 <shipment> 元素，但現在只含有一個。

從 XML 集合刪除 XML 文件

您可刪除從集合製作的 XML 文件。這表示若有製作多重 XML 文件的 XML 集合，您就可以刪除其中一個已製作的文件。

刪除文件時，您會刪除表格中的一個橫列，該表格製作當初在 DAD 檔中指定的頂層 element_node。本表格含有頂層集合表格的主要鍵及低階表格的外來鍵。

例如，下列 DELETE 陳述式指定主要鍵直欄值。

```
DELETE from order_tab
   WHERE order_key=1
```

製作 XML 文件之後，ORDER_KEY 就是表格 ORDER_TAB 中的主要鍵而且是頂層 element_node。刪除本橫列即刪除在製作期間產生的一個 XML 文件。因此，從 XML 的觀點，會從 XML 集合刪除一個 XML 文件。

從 XML 集合擷取 XML 文件

從 XML 集合擷取 XML 文件類似從該集合製作文件。

若要擷取 XML 文件，請使用儲存程序 dxxRetrieveXML()。語法及範例，請參閱第191頁的『dxxRetrieveXML()』。

DAD 檔注意事項：將 XML 文件分解到某 XML 集合時，除非在 DAD 檔中指定多次出現的元素及屬性值的順序，否則會失去該順序。若要保留本順序，您應使用 RDB_node 對映方法。這個對映方法可讓您爲表格指定 orderBy 屬性，該表格在它的 RDB_node 中含有 Root 元素。

搜尋 XML 集合

本節根據下列目標說明搜尋 XML 集合：

- **使用搜尋準則建立 XML 文件：**

  本作業實際使用條件製作。您可使用下列搜尋準則指定搜尋準則：

  – 在 DAD 檔中的 text_node 及 attribute_node 中指定條件

  – 使用 dxxGenXML() 及 dxxRetrieveXML() 儲存程序時，指定*改寫*參數。

  例如，若使用 DAD 檔 order.dad 啓用 XML 集合 sales_ord，但您目前想要使用衍生自 Web 的套表資料置換價格，那麼您可按照下列方式置換 <SQL_stmt> DAD 元素值：

```
EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
       ...

          EXEC SQL END DECLARE SECTION;

        float    price_value;

        /* create table */
        EXEC SQL CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

        /* initialize host variable and indicators */
        strcpy(collection,"sales_ord");
        strcpy(result_tab,"xml_order_tab");
        overrideType = SQL_OVERRIDE;
        max_row = 20;
        num_row = 0;
        returnCode = 0;
        msg_txt[0] = '\0';
        override_ind = 0;
        overrideType_ind = 0;
        rtab_ind = 0;
        maxrow_ind = 0;
        numrow_ind = -1;
        returnCode_ind = -1;
        returnMsg_ind = -1;

        /* get the price_value from some place, such as form data */
        price_value = 1000.00         /* for example*/

        /* specify the overwrite */
        sprintf(overwrite,
               "SELECT o.order_key, customer, p.part_key, quantity, price,
                      tax, ship_id, date, mode
               FROM order_tab o, part_tab p,
                   table(select substr(char(timestamp(generate_unique())),16)
                   as ship_id, date, mode from ship_tab)as s
               WHERE p.price > %d and s.date >'1996-06-01' AND
                   p.order_key = o.order_key and s.part_key = p.part_key",
                   price_value);
```

```
/* Call the store procedure */
EXEC SQL CALL db2xml!dxxRetrieve(:collection:collection_ind,
        :result_tab:rtab_ind,
        :overrideType:overrideType_ind,:overwrite:overwrite_ind,
        :max_row:maxrow_ind,:num_row:numrow_ind,
              :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

對 price > ? 則置換 order.dad 中的 price > 2500.00 條件而言，其中 ? 是根據輸入變數 *price_value* 而定。

- **搜尋分解的 XML 資料：**

  您可使用一般 SQL 查詢作業搜尋集合表格。您可合併集合表格或使用子查詢，然後在文字直欄中執行結構式文字搜尋。有了結構式搜尋結果之後，您就可以應用該資料擷取或建立指定的 XML 文件。

第4篇 參照

這個部份提供 XML Extender 管理指令、使用者定義資料類型 (UDT)、使用者定義函數 (UDF) 及儲存程序的語法資訊。也針對問題與解決方案提供訊息文字。

第7章 XML Extender 管理指令：*dxxadm*

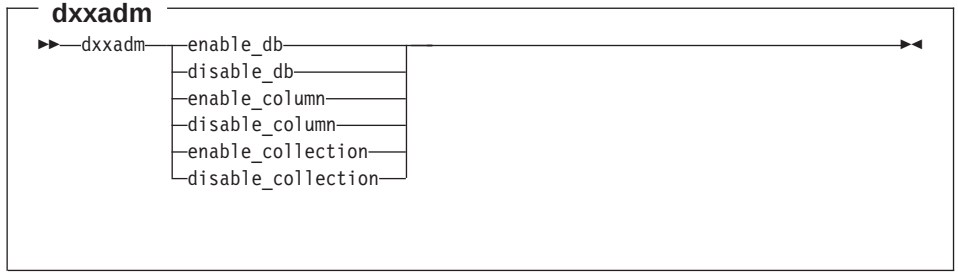
XML Extender 提供一個管理指令 **dxxadm** 以完成下列管理作業：使用不同指令選項，呼叫 **dxxadm** 執行下列 XML Extender 管理作業：

- 啓用或停用 XML Extender 的資料庫
- 啓用或停用 XML 直欄
- 啓用或停用 XML 集合

您亦可使用 XML Extender 管理精靈或儲存程序來執行每一個管理作業。

高階語法

下列語法圖提供 **dxxadm** 指令的高階語法。 下列區段提供每一個選項的說明。



管理指令選項

下列各節說明系統程式設計師可使用的 **dxxadm** 指令選項：

- 第136頁的『enable_db』
- 第137頁的『disable_db』
- 第138頁的『enable_column』
- 第140頁的『disable_column』
- 第142頁的『enable_collection』
- 第143頁的『disable_collection』

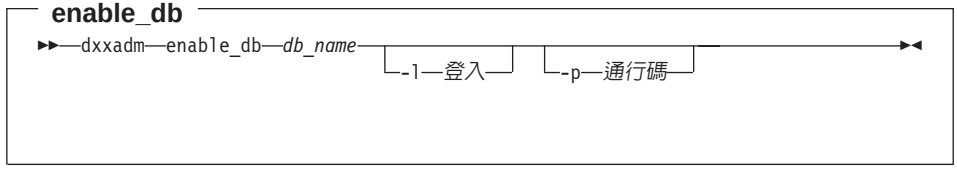
enable_db

目的

連接及啓用資料庫以便與 XML Extender 一起使用。啓用資料庫後，XML Extender 會建立下列物件：

- XML Extender 使用者定義類型 (UDT)。
- XML Extender 使用者定義函數 (UDF)。
- XML Extender DTD 參照表 DTD_REF，其中儲存 DTD 與每一個 DTD 的相關資訊。關於 DTD_REF 表格的完整說明，請參閱第199頁的『DTD 參照表』。
- XML Extender 用法表 XML_USAGE，其中儲存針對 XML 及每一個集合啓用的每一個直欄之共用資訊。 關於 XML_USAGE 表格的完整說明，請參閱第200頁的『XML 使用表』。

語法



參數

表 14. enable_db 參數

參數	說明
db_name	XML 資料所在的資料庫名稱。
-l 登入	當指定時用來連接資料庫的使用者 ID (可選用的)。如果沒有指定，則使用現行使用者 ID。
-p *通行碼*	當指定時用來連接資料庫的通行碼 (可選用的)。如果沒有指定，則使用現行通行碼。

範例

下列範例啓用資料庫 SALES_DB。

```
dxxadm enable_db SALES_DB
```

disable_db

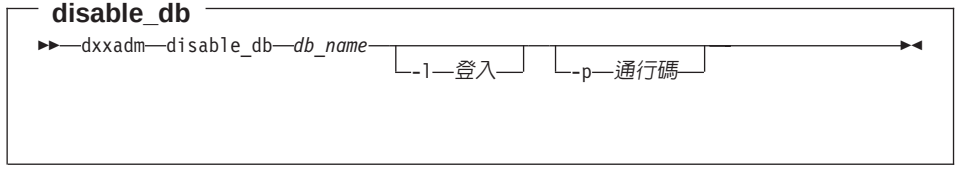
目的

連接至及停用已啓用 XML 的資料庫。當資料庫停用後，XML Extender 即無法再使用它。當 XML Extender 停用資料庫時，會捨棄下列物件：

- XML Extender 使用者定義類型 (UDT)。
- XML Extender 使用者定義函數 (UDF)。
- XML Extender DTD 參照表 DTD_REF，其中儲存 DTD 與每一個 DTD 的相關資訊。關於 DTD_REF 表格的完整說明，請參閱第199頁的『DTD 參照表』。
- XML Extender 用法表 XML_USAGE，其中儲存針對 XML 及每一個集合啓用的每一個直欄之共用資訊。 關於 XML_USAGE 表格的完整說明，請參閱第200頁的『XML 使用表』。

重要事項：試圖停用資料庫之前，您必須停用所有 XML 直欄。 XML Extender 無法停用含有爲 XML 啓用的直欄或集合的資料庫。

語法



參數

表 15. disable_db 參數

參數	說明
db_name	XML 資料所在的資料庫名稱
-l *登入*	當指定時用來連接資料庫的使用者 ID (可選用的)。如果沒有指定，則使用現行使用者 ID。
-p *通行碼*	當指定時用來連接資料庫的通行碼 (可選用的)。如果沒有指定，則使用現行通行碼。

範例

下列範例停用資料庫 SALES_DB。

```
dxxadm disable_db SALES_DB
```

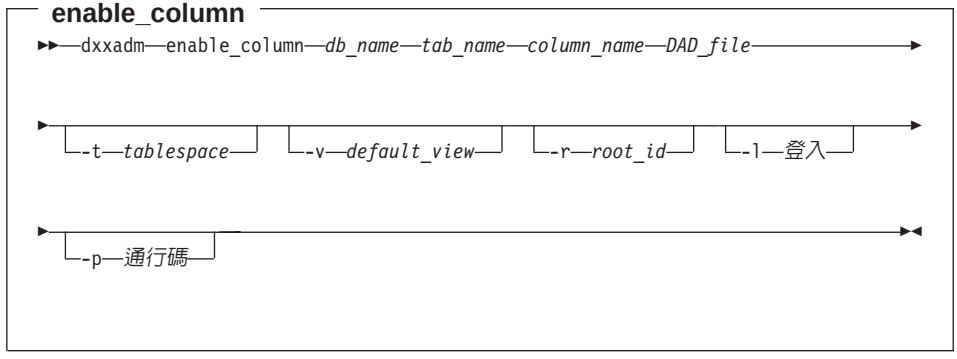
enable_column

目的

連接至資料庫並啓用 XML 直欄，使其包含 XML Extender UDT。啓用直欄時，XML Extender 會完成下列作業：

- 判斷 XML 表格是否有主要鍵；若沒有，XML Extender 就會變更 XML 表格並新增一個名稱是 DXXROOT_ID 的直欄。
- 使用含有 XML 表格中每一橫列之唯一識別字的直欄，以建立 DAD 檔中指定的輔助表格。 這個直欄爲使用者指定的 *root_id*，或 XML Extender 命名的 DXXROOT_ID。
- 選用性地使用您指定的名稱，針對 XML 表格及其輔助表格建立一個預設概略表。

語法



參數

表 16. enable_column 參數

參數	說明
db_name	XML 資料所在的資料庫名稱。
tab_name	XML 直欄所在的表格名稱。
column_name	XML 直欄的名稱。
DAD_file	將 XML 文件對映至 XML 直欄及輔助表格的 DAD 檔案名稱。
-t *表格空間*	包含與 XML 直欄相關之輔助表格的表格空間 (可選用的)。如果沒有指定，則會使用預設的表格空間。

表 16. enable_column 參數 (繼續)

參數	說明
-v *default_view*	結合 XML 直欄與輔助表格的預設概略表名稱 (可選用的)。
-r *root_id*	XML 直欄表格中當作輔助表格 root_id 使用的主要鍵名稱。root_id 爲可選用的。
-l *登入*	當指定時用來連接資料庫的使用者 ID (可選用的)。如果沒有指定，則使用現行使用者 ID。
-p *通行碼*	當指定時用來連接資料庫的通行碼 (可選用的)。如果沒有指定，則使用現行通行碼。

範例

下列範例啓用 XML 直欄。

```
dxxadm enable_column SALES_DB SALES_TAB ORDER -v sales_order_view -r INVOICE_NUMBER
```

disable_column

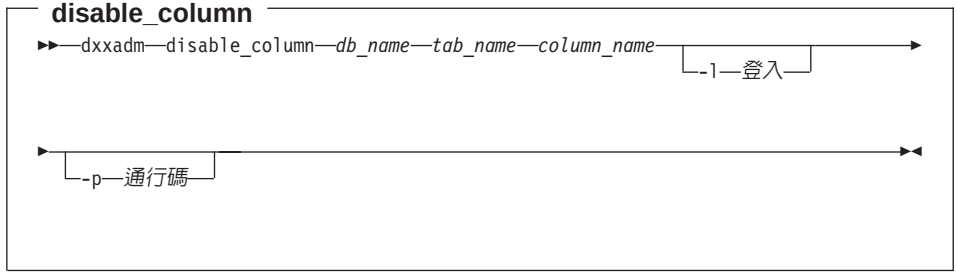
目的

連接至資料庫並停用已啓用的 XML 直欄。當此直欄停用後，即不再含有 XML 資料類型。 當已啓用的 XML 直欄停用後，會執行下列動作：

- 從 XML_USAGE 表格中刪除 XML 直欄用法登錄。
- 在 DTD_REF 表格中減少 USAGE_COUNT。
- 與這個直欄相關的所有觸發函式都會被捨棄。
- 與這個直欄相關的所有輔助表格都會被捨棄。

重要事項：捨棄 XML 表格之前，您必須先停用 XML 直欄。 如果 XML 表格已捨棄但未停用其 XML 直欄， XML Extender 會保留所建立的輔助表格及 XML_USAGE 表格中的 XML 直欄登錄。

語法



參數

表 17. disable_column 參數

參數	說明
db_name	資料所在的資料庫名稱。
tab_name	XML 直欄所在的表格名稱。
column_name	XML 直欄的名稱。
-l 登入	當指定時用來連接資料庫的使用者 ID (可選用的)。如果沒有指定，則使用現行使用者 ID。
-p *通行碼*	當指定時用來連接資料庫的通行碼 (可選用的)。如果沒有指定，則使用現行通行碼。

範例

下列範例停用已啓用的 XML 直欄。

```
dxxadm disable_column SALES_DB SALES_TAB ORDER
```

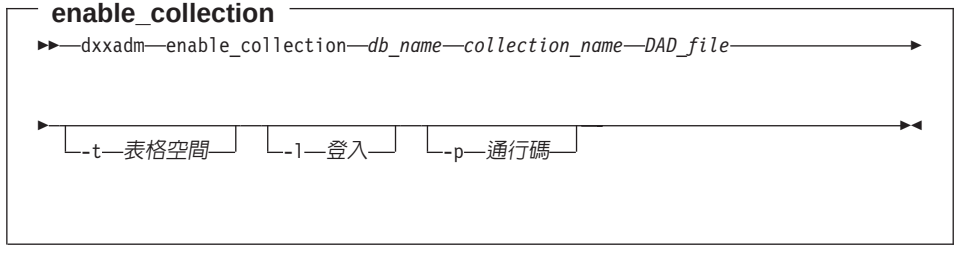
enable_collection

目的

連接至資料庫並根據指定的 DAD 啓用 XML 集合。 啓用集合時，XML Extender 會執行下列作業：

- 在 XML_USAGE 表格中建立 XML 集合用法登錄。
- 以 RDB_node 對映而言，如果資料庫中沒有表格，則建立 DAD 中指定的集合表格。

語法



參數

表 18. enable_collection 參數

參數	說明
db_name	資料所在的資料庫名稱。
collection_name	XML 集合的名稱。
DAD_file	將 XML 文件對映至集合中關聯式表格的 DAD 檔案名稱。
-t *表格空間*	與集合相關的表格空間名稱 (可選用的)。 如果沒有指定，則會使用預設的表格空間。
-l 登入	當指定時用來連接資料庫的使用者 ID (可選用的)。如果沒有指定，則使用現行使用者 ID。
-p *通行碼*	當指定時用來連接資料庫的通行碼 (可選用的)。如果沒有指定，則使用現行通行碼。

範例

下列範例啓用 XML 集合。

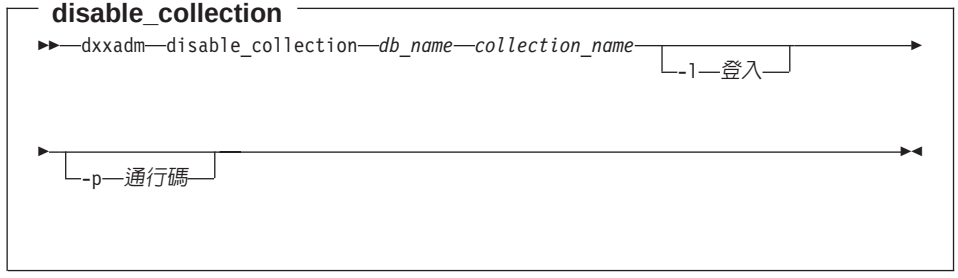
```
dxxadm enable_collection SALES_DB sales_ord getstart_xcollection.dad -t orderspace
```

disable_collection

目的

連接至資料庫並停用已啓用的 XML 集合。 集合名稱無法再用於組合 (dxxRetrieveXML) 及分解 (dxxInsertXML) 儲存程序。 當停用 XML 集合後，相關的集合登錄會從 XML_USAGE 表格中刪除。 請注意：停用集合並不會捨棄 enable_collection 步驟期間建立的集合表格。

語法



參數

表 19. disable_collection 參數

參數	說明
db_name	資料所在的資料庫名稱。
collection_name	XML 集合的名稱。
-l 登入	當指定時用來連接資料庫的使用者 ID (可選用的)。如果沒有指定，則使用現行使用者 ID。
-p *通行碼*	當指定時用來連接資料庫的通行碼 (可選用的)。如果沒有指定，則使用現行通行碼。

範例

下列範例停用 XML 集合。

```
dxxadm disable_collection SALES_DB sales_ord
```

第8章 XML Extender 使用者定義類型

XML Extender 使用者定義類型 (UDT) 是使用於 XML 直欄和 XML 集合的資料類型。 所有 UDT 皆有綱目名稱 db2xml。 XML Extender 建立 UDT 來儲存和擷取 XML 文件。 表20包含 UDT 的概觀。

表 20. XML Extender UDT

使用者定義的類型直欄	來源資料類型	用法說明
XMLVARCHAR	VARCHAR(*varchar_len*)	將整個 XML 文件儲存成 DB2 內的 VARCHAR。
XMLCLOB	CLOB(*clob_len*)	將整個 XML 文件儲存成 DB2 內的字元大型物件 (CLOB)。
XMLFILE	VARCHAR(512)	指定本端檔案伺服器的檔名。若對 XML 直欄指定 XMLFILE，則 XML Extender 將 XML 文件儲存於外部伺服器檔案中。無法使用 XMLFILE 來啓用 Text Extender。您需要負責確保檔案內容和 DB2 之間的完整性，也需要考慮爲了索引所建立的輔助表格。

其中，*varchar_len* 和 *clob_len* 視作業系統而定。

對於 DB2 UDB，*varchar_len* = 3K，*clob_len* = 2G。

這些 UDT 只用來指定應用程式直欄的類型；不適用於 XML Extender 所建立的輔助表格。

第9章 XML Extender 使用者定義函數

XML Extender 提供函數來儲存、擷取、搜尋及更新 XML 文件，以及取出 XML 元素或屬性。 XML 使用者定義函數 (UDF) 使用於 XML 直欄，不使用於 XML 集合。 所有 UDF 皆有綱目名稱 db2xml，但在 UDF 前面可以省略。

XML Extender 函數的四種類型：儲存函數、擷取函數、取出函數和更新函數。

儲存函數

儲存函數在 DB2 資料庫中插入 XML 文件。相關語法和範例，請參閱第148頁的『儲存函數』。

擷取函數

擷取函數從 DB2 資料庫的 XML 直欄中擷取 XML 文件。相關語法和範例，請參閱第153頁的『擷取函數』。

取出函數

取出函數可取出 XML 文件中的元素內容或屬性值，轉換成函數名稱所指定的資料類型。 XML Extender 針對各種 SQL 資料類型，提供一組取出函數。 相關語法和範例，請參閱第160頁的『取出函數』。

更新函數

Update() 函數修改元素內容或屬性值，傳回一份 XML 文件，其中含有位置路徑所指定的更新值。 Update() 函數可讓應用程式設計師指定要更新的元素或屬性。相關語法和範例，請參閱第176頁的『更新函數』。

表21提供 XML Extender 函數的摘要。

表 21. XML Extender 使用者定義函數

類型	函數
儲存函數	XMLVarcharFromFile()
	XMLCLOBFromFile()
	XMLFileFromVarchar()
	XMLFileFromCLOB()
擷取函數	Content()：從 XMLFile 中擷取到 CLOB
	Content()：從 XMLVarchar 中擷取到外部伺服器檔案
	Content()：從 XMLCLOB 中擷取到外部伺服器檔案

表 21. XML Extender 使用者定義函數 (繼續)

類型	函數
取出函數	extractInteger() 和 extractIntegers()
	extractSmallint() 和 extractSmallints()
	extractDouble() 和 extractDoubles()
	extractReal() 和 extractReals()
	extractChar() 和 extractChars()
	extractVarchar() 和 extractVarchars()
	extractCLOB() 和 extractCLOBs()
	extractDate() 和 extractDates()
	extractTime() 和 extractTimes()
	extractTimestamp() 和 extractTimestamps()
更新函數	Update()

儲存函數

儲存函數可在 DB2 資料庫中插入 XML 文件。 您可在 INSERT 或 SELECT 陳述式中直接使用 UDT 的預設強制轉型函數。 此外，XML Extender 提供 UDF 從 UDT 基本資料類型以外的來源中取得 XML 文件，再轉換成所要的 UDT。

XML Extender 提供下列儲存函數：

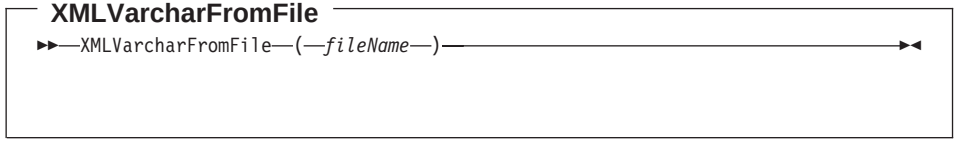
- 第149頁的『XMLVarcharFromFile()』
- 第150頁的『XMLCLOBFromFile()』
- 第151頁的『XMLFileFromVarchar()』
- 第152頁的『XMLFileFromCLOB()』

XMLVarcharFromFile()

目的

從伺服器檔案中讀取 XML 文件，傳回 XMLVARCHAR 類型的文件。

語法



參數

表 22. XMLVarcharFromFile 參數

參數	資料類型	說明
fileName	VARCHAR(512)	完整的伺服器檔名。

回覆類型

XMLVARCHAR

範例

下列範例從伺服器檔案中讀取 XML 文件，然後以 XMLVARCHAR 類型插入到 XML 直欄中。

```
EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
   VALUES('1234', 'Sriram Srinivasan',
           XMLVarcharFromFile('c:\dxx\samples\cmd\getstart.xml'))
```

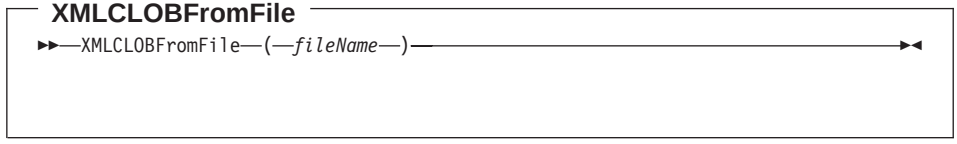
本範例中，在 SALES_TAB 表格內插入一筆記錄。 函數 XMLVarcharFromFile() 將一個檔案中的 XML 文件匯入 DB2 中，儲存成 XMLVARCHAR。

XMLCLOBFromFile()

目的

從伺服器檔案中讀取 XML 文件，傳回 XMLCLOB 類型的文件。

語法



參數

表 23. XMLCLOBFromFile 參數

參數	資料類型	說明
fileName	VARCHAR(512)	完整的伺服器檔名。

回覆類型

XMLCLOB as LOCATOR

範例

下列範例從伺服器檔案中讀取 XML 文件，然後以 XMLCLOB 類型插入到 XML 直欄中。

```
EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
   VALUES('1234', 'Sriram Srinivasan',
           XMLCLOBFromFile('c:\dxx\samples\cmd\getstart.xml'))
```

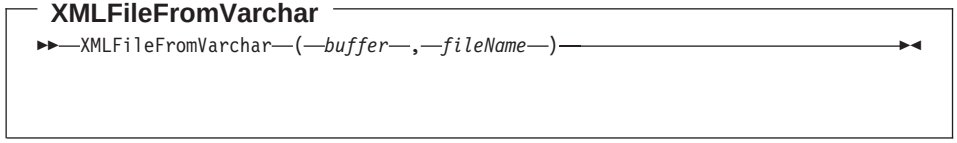
SALES_TAB 表格中的直欄 ORDER 定義爲 XMLCLOB 類型。前述範例顯示直欄 ORDER 如何插入 SALES_TAB 表格內。

XMLFileFromVarchar()

目的

從記憶體中以 VARCHAR 類型讀取 XML 文件，寫入外部伺服器檔案中，然後以 XMLFILE 類型傳回檔名和路徑。

語法



參數

表 24. XMLFileFromVarchar 參數

參數	資料類型	說明
buffer	VARCHAR(3K)	記憶體緩衝區。
fileName	VARCHAR(512)	完整的伺服器檔名。

回覆類型

XMLFILE

範例

下列範例以 VARCHAR 類型從記憶體中讀取 XML 文件，寫入外部伺服器檔案中，然後以 XMLFILE 類型將檔名和路徑插入一個 XML 直欄中。

```
EXEC SQL BEGIN DECLARE SECTION;
      struct { short len; char data[3000]; } xml_buff;
          EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
   VALUES('1234', 'Sriram Srinivasan',
       XMLFileFromVarchar(:xml_buf, 'c:\dxx\samples\cmd\getstart.xml'))
```

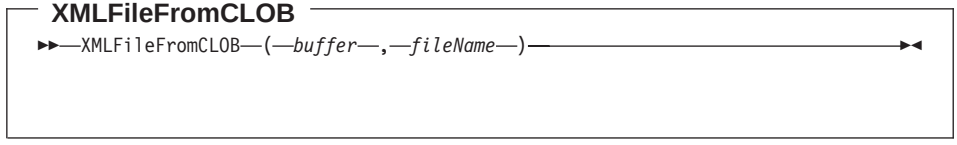
SALES_TAB 表格中的直欄 ORDER 定義爲 XMLFILE 類型。前述範例顯示若您在緩衝區有一份 XML 文件，則可儲存於伺服器檔案中。

XMLFileFromCLOB()

目的

以 CLOB 定位器類型讀取 XML 文件，寫入外部伺服器檔案中，以 XMLFILE 類型傳回檔名和路徑。

語法



參數

表 25. XMLFileFromCLOB() 參數

參數	資料類型	說明
buffer	CLOB as LOCATOR	含有 XML 文件的緩衝區。
fileName	VARCHAR(512)	完整的伺服器檔名。

回覆類型

XMLFILE

範例

下列範例以 CLOB 定位器類型讀取 XML 文件，寫入外部伺服器檔案中，然後以 XMLFILE 類型將檔名和路徑插入一個 XML 直欄中。

```
EXEC SQL BEGIN DECLARE SECTION;
             SQL TYPE IS CLOB_LOCATOR xml_buff;
         EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
   VALUES('1234', 'Sriram Srinivasan',
           XMLFileFromCLOB(:xml_buf, 'c:\dxx\samples\cmd\getstart.xml'))
```

SALES_TAB 表格中的直欄 ORDER 定義爲 XMLFILE 類型。如果緩衝區內有一份 XML 文件，您可將文件儲存在伺服器檔案中。

擷取函數

您可使用預設強制轉型函數，將 XML UDT 轉換成基本資料類型。 XML Extender 亦提供一個*超載函數* Content()，使用於擷取動作。

Content() 函數提供下列擷取類型：

- **從外部儲存體中擷取到記憶體指標**

  您可使用 Content()，將儲存爲外部伺服器檔案的 XML 文件擷取到記憶體緩衝區。. 您可使用第154頁的『Content()：從 XMLFILE 中擷取到 CLOB』 來達成此目的。

- **從內部記憶體中擷取到外部伺服器檔案**

  您亦可使用 Content()，擷取 DB2 內儲存的 XML 文件，儲存成 DB2 伺服器檔案系統上的伺服器檔案。下列 Content() 函數用來將資訊儲存在外部伺服器檔案中：

  – 第156頁的『Content()：從 XMLVARCHAR 中擷取到外部伺服器檔案』
  – 第158頁的『Content(): 從 XMLCLOB 中擷取到外部伺服器檔案』

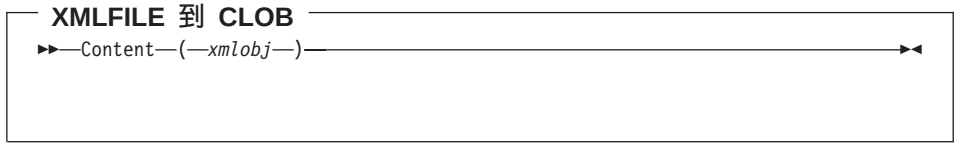
下一節中的範例假設您使用 DB2 指令 Shell，您不需要在每一個指令的開頭鍵入『DB2』。

Content()：從 XMLFILE 中擷取到 CLOB

目的

擷取資料伺服器檔案中的資料，儲存於 CLOB LOCATOR。

語法



參數

表 26. XMLFILE 到 CLOB 參數

參數	資料類型	說明
xmlobj	XMLFILE	XML 文件。

回覆類型

CLOB (*clob_len*) as LOCATOR

對於 DB2 UDB，*clob_len* 是 2G。

範例

下列範例擷取資料伺服器檔案中的資料，儲存於 CLOB 定位器。

```
EXEC SQL BEGIN DECLARE SECTION;
             SQL TYPE IS CLOB_LOCATOR xml_buff;
          EXEC SQL END DECLARE SECTION;

          EXEC SQL CONNECT TO SALES_DB

EXEC SQL DECLARE c1 CURSOR FOR

              SELECT Content(order) from sales_tab
   WHERE sales_person = 'Sriram Srinivasan'

                         EXEC SQL OPEN c1;

          do {
             EXEC SQL FETCH c1 INTO :xml_buff;
             if (SQLCODE != 0) {
                break;
     }
             else {
     /* do with the XML doc in buffer */
```

```
    }
  }

        EXEC SQL CLOSE c1;

        EXEC SQL CONNECT RESET;
```

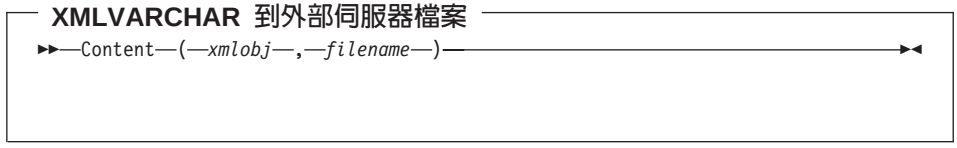
SALES_TAB 表格中的直欄 ORDER 是 XMLFILE 類型，因此 Content() UDF 擷取資料伺服器檔案中的資料並儲存於 CLOB 定位器。

Content()：從 XMLVARCHAR 中擷取到外部伺服器檔案

目的

擷取以 XMLVARCHAR 類型儲存的 XML 內容並儲存於外部伺服器檔案中。

語法



重要事項： 若指定名稱的檔案已存在， content 函數會置換其內容。

註:

參數

表 27. XMLVarchar 到外部伺服器檔案參數

參數	資料類型	說明
xmlobj	XMLVARCHAR	XML 文件。
filename	VARCHAR(512)	完整的伺服器檔名。

回覆類型

VARCHAR(512)

範例

下列範例擷取以 XMLVARCHAR 類型儲存的 XML 內容並儲存於外部伺服器檔案中。

```
CREATE table app1 (id int NOT NULL, order db2xml.XMLVarchar);
INSERT into app1 values (1, '<?xml version="1.0"?>
  <!DOCTYPE SYSTEM c:\dxx\samples\dtd\getstart.dtd"->
    <Order key="1">
        <Customer>
    <Name>American Motors</Name>
    <Email>parts@am.com</Email>
      </Customer>
      <Part color="black">
    <key>68</key>
    <Quantity>36</Quantity>
    <ExtendedPrice>34850.16</ExtendedPrice>
    <Tax>6.000000e-02</Tax>
    <Shipment>
```

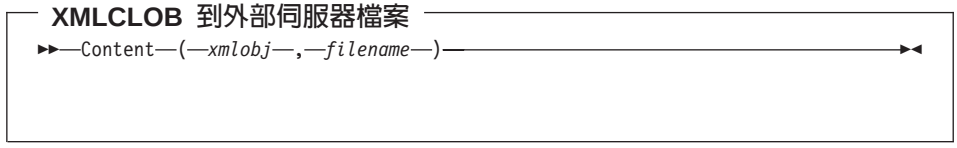
```
      <ShipDate>1998-08-19</ShipDate>
       <ShipMode>AIR   </ShipMode>
    </Shipment>
    <Shipment>
      <ShipDate>1998-08-19</ShipDate>
      <ShipMode>BOAT  </ShipMode>
    </Shipment>
    </Order>');

SELECT db2xml.Content(order, 'c:\dxx\samples\cmd\getstart_.dad')
  from app1 where ID=1;
```

Content(): 從 XMLCLOB 中擷取到外部伺服器檔案

目的

擷取以 XMLCLOB 類型儲存的 XML 內容並儲存於外部伺服器檔案中。

語法



重要事項： 若指定名稱的檔案已存在， content 函數會置換其內容。

參數

表 28. XMLCLOB 到外部伺服器檔案參數

參數	資料類型	說明
xmlobj	XMLCLOB as LOCATOR	XML 文件。
filename	VARCHAR(512)	完整的伺服器檔名。

回覆類型

VARCHAR(512)

範例

下列範例擷取以 XMLCLOB 類型儲存的 XML 內容並儲存於外部伺服器檔案中。

```
CREATE table app1 (id int NOT NULL, order db2xml.XMLCLOB not logged);

INSERT into app1 values (1, '<?xml version="1.0"?>
  <!DOCTYPE SYSTEM c:\dxx\samples\dtd\getstart.dtd"->
    <Order key="1">
        <Customer>
    <Name>American Motors</Name>
    <Email>parts@am.com</Email>
  </Customer>
  <Part color="black ">
    <key>68</key>
    <Quantity>36</Quantity>
    <ExtendedPrice>34850.16</ExtendedPrice>
    <Tax>6.000000e-02</Tax>
    <Shipment>
      <ShipDate>1998-08-19</ShipDate>
       <ShipMode>AIR   </ShipMode>
    </Shipment>
```

```
    <Shipment>
      <ShipDate>1998-08-19</ShipDate>
      <ShipMode>BOAT  </ShipMode>
    </Shipment>
    </Order>');

SELECT db2xml.Content(order, 'c:\dxx\samples\cmd\getstart.xml')
from app1 where ID=1;
```

取出函數

取出函數從 XML 文件中取出元素內容或屬性值，傳回所要求的 SQL 資料類型。XML Extender 針對各種 SQL 資料類型，提供一組取出函數。取出函數接受兩個輸入參數。第一個參數是 XML Extender UDT，可以是其中一個 XML UDT。第二個參數是位置路徑，用來指定 XML 元素或屬性。每一個取出函數皆傳回位置路徑所指定的值或內容。

因爲部份元素或屬性會出現多次，所以取出函數傳回一個純量或表格值；後者稱爲表格函數。

XML Extender 提供下列取出函數：

- 第161頁的『extractInteger() 和 extractIntegers()』
- 第163頁的『extractSmallint() 和 extractSmallints()』
- 第164頁的『extractDouble() 和 extractDoubles()』
- 第166頁的『extractReal() 和 extractReals()』
- 第167頁的『extractChar() 和 extractChars()』
- 第168頁的『extractVarchar() 和 extractVarchars()』
- 第170頁的『extractCLOB() 和 extractCLOBs()』
- 第171頁的『extractDate() 和 extractDates()』
- 第172頁的『extractTime() 和 extractTimes()』
- 第174頁的『extractTimestamp() 和 extractTimestamps()』

下一節中的範例假設您使用 DB2 指令 Shell，您不需要在每一個指令的開頭鍵入『DB2』。

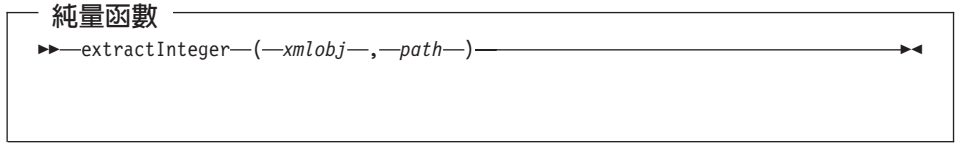
extractInteger() 和 extractIntegers()

目的

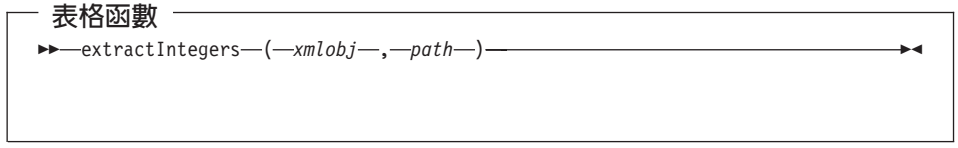
從 XML 文件中取出元素內容或屬性值，傳回 INTEGER 類型的資料。

語法

純量函數



表格函數



參數

表 29. extractInteger 和 extractIntegers 函數參數

參數	資料類型	說明
xmlobj	XMLVARCHAR、XMLFILE 或 XMLCLOB	直欄名稱。
path	VARCHAR	元素或屬性的位置路徑。

回覆類型

INTEGER

傳回的直欄名稱 (表格函數)

returnedInteger

範例

純量函數範例：

下列範例中，鍵值的屬性值 = ″1″ 時會傳回一個值。此值以 INTEGER 形式取出。

```
SELECT * from table(db2xml.extractInteger(Order,
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/[@key="1"]')) as x;
```

表格函數範例：

在下列範例中，訂購單的每一個訂購鍵值皆自動從 INTEGER 轉換成 DECIMAL 類型。

```
CREATE TABLE t1(decimal(3,2));
INSERT into t1
SELECT * from table(db2xml.extractIntegers(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/@key')) as x;
SELECT * from t1;
```

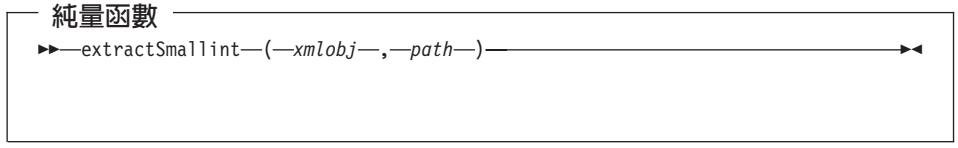
extractSmallint() 和 extractSmallints()

目的

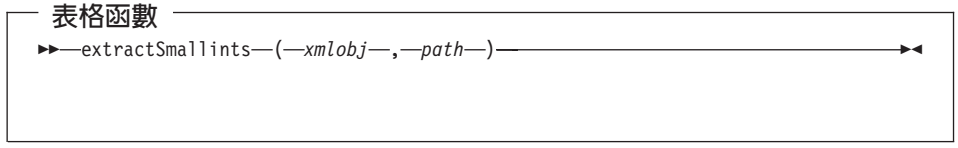
從 XML 文件中取出元素內容或屬性值，傳回 SMALLINT 類型的資料。

語法

純量函數



表格函數



參數

表 30. extractSmallint 和 extractSmallints 函數參數

參數	資料類型	說明
xmlobj	XMLVARCHAR、XMLFILE 或 XMLCLOB	直欄名稱。
path	VARCHAR	元素或屬性的位置路徑。

回覆類型

SMALLINT

傳回的直欄名稱 (表格函數)

returnedSmallint

範例

在下列範例中，所有訂購單中的 Quantity 值皆以 SMALLINT 形式取出

```
SELECT * from table(db2xml.extractSmallints(Order,
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/Quantity')) as x;
```

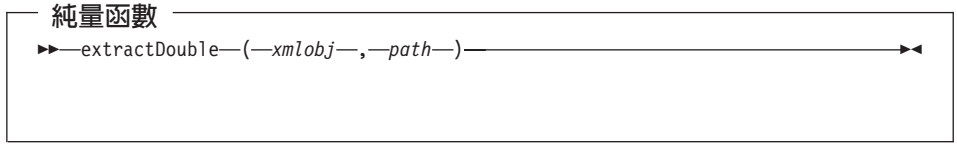
extractDouble() 和 extractDoubles()

目的

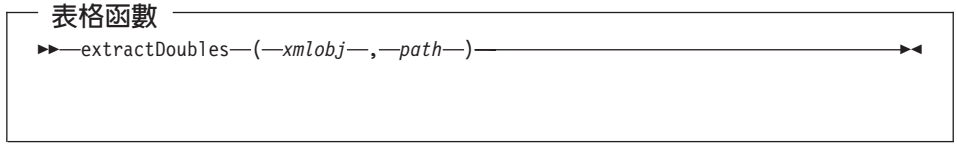
從 XML 文件中取出元素內容或屬性值，傳回 DOUBLE 類型的資料。

語法

純量函數



表格函數



參數

表 31. extractDouble 和 extractDoubles 函數參數

參數	資料類型	說明
xmlobj	XMLVARCHAR、XMLFILE 或 XMLCLOB	直欄名稱。
path	VARCHAR	元素或屬性的位置路徑。

回覆類型

DOUBLE

傳回的直欄名稱 (表格函數)

returnedDouble

範例

表格函數範例：

在下列範例中，訂購單中每一個零件的 ExtendedPrice 值皆以 DOUBLE 形式取出。

```
SELECT * from table(db2xml.extractDoubles(Order,
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/ExtendedPrice')) as x;
```

純量函數範例：

下列範例自動將訂購單中的售價從 DOUBLE 類型轉換成 DECIMAL。

```
CREATE TABLE t1(price, DECIMAL(5,2));
INSERT into t1 values (db2xml.extractDouble(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/ExtendedPrice'));
SELECT * from t1;
```

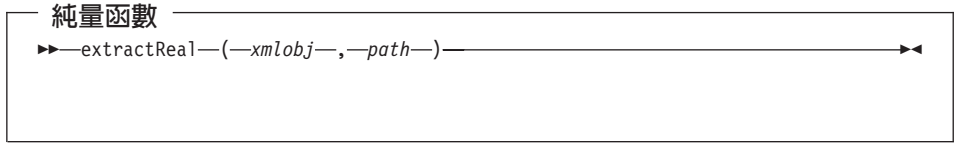
extractReal() 和 extractReals()

目的

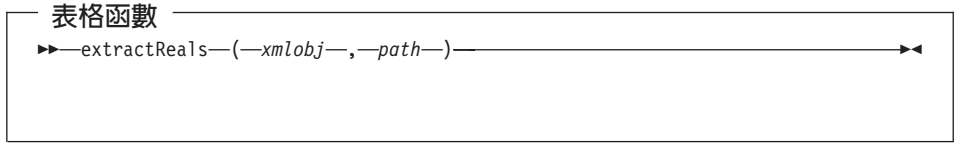
從 XML 文件中取出元素內容或屬性值，傳回 REAL 類型的資料。

語法

純量函數



表格函數



參數

表 32. extractReal 和 extractReals 函數參數

參數	資料類型	說明
xmlobj	XMLVARCHAR、XMLFILE 或 XMLCLOB	直欄名稱。
path	VARCHAR	元素或屬性的位置路徑。

回覆類型

REAL

傳回的直欄名稱 (表格函數)

returnedReal

範例

在下列範例中，Tax 的值以 REAL 形式取出。

```
SELECT * from table(db2xml.extractReals(Order,
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/Tax')) as x;
```

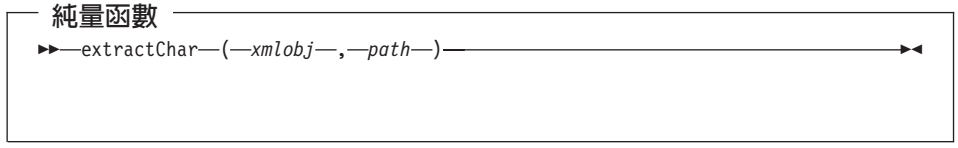
extractChar() 和 extractChars()

目的

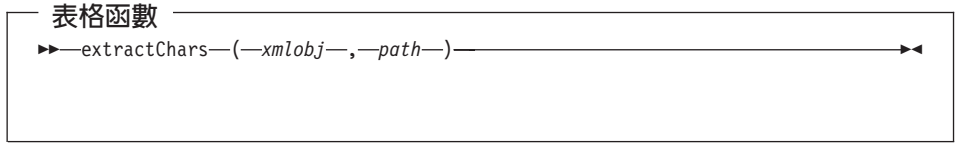
從 XML 文件中取出元素內容或屬性值，傳回 CHAR 類型的資料。

語法

純量函數



表格函數



參數

表 33. extractChar 和 extractChars 函數參數

參數	資料類型	說明
xmlobj	XMLVARCHAR、XMLFILE 或 XMLCLOB	直欄名稱。
path	VARCHAR	元素或屬性的位置路徑。

回覆類型

CHAR

傳回的直欄名稱 (表格函數)

returnedChar

範例

在下列範例中，Color 的值以 CHAR 形式取出。

```
SELECT * from table(db2xml.extractChars(Order,
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/@Color')) as x;
```

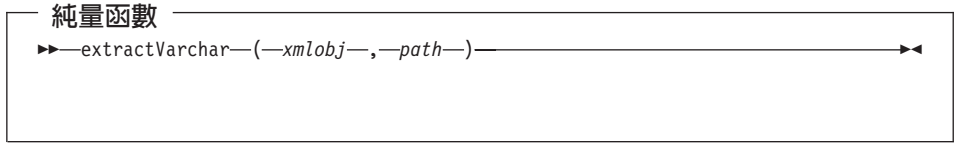
extractVarchar() 和 extractVarchars()

目的

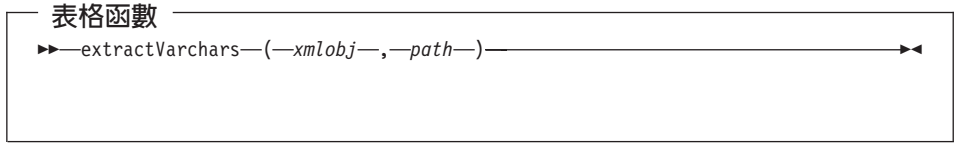
從 XML 文件中取出元素內容或屬性值，傳回 VARCHAR 類型的資料。

語法

純量函數



表格函數



參數

表 34. extractVarchar 和 extractVarchars 函數參數

參數	資料類型	說明
xmlobj	XMLVARCHAR、XMLFILE 或 XMLCLOB	直欄名稱。
path	VARCHAR	元素或屬性的位置路徑。

回覆類型

VARCHAR(4K)

傳回的直欄名稱 (表格函數)

returnedVarchar

範例

若一個資料庫含有 1000 份 XML 文件，而文件都儲存在 SALES_TAB 表格的直欄 ORDER 中，您可能想尋找訂購項目中 ExtenderPrice 大於 2500.00 的全部客戶。下列 SQL 陳述式在 SELECT 子句中使用取出 UDF：

```
SELECT extractVarchar(Order, '/Order/Customer/Name') from sales_order_view
   WHERE price > 2500.00
```

UDF extractVarchar() 使用直欄 ORDER 做爲輸入，使用位置路徑 /Order/Customer/Name 做爲選取識別字。 UDF 傳回客戶的姓名。 取出函數透過 WHERE 子句，只評估 ExtenderPrice 大於 2500.00 的訂單。

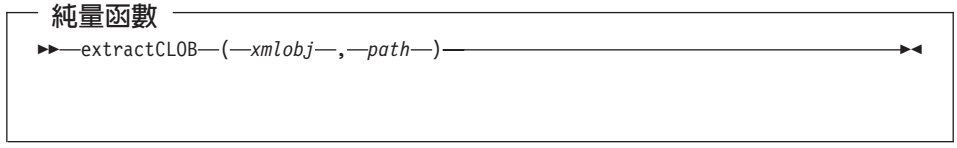
extractCLOB() 和 extractCLOBs()

目的

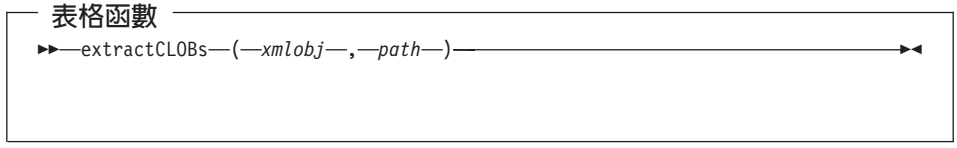
從 XML 文件中取出元素內容或屬性值，傳回 CLOB 類型的資料。

語法

純量函數



表格函數



參數

表 35. extractCLOB 和 extractCLOBs 函數參數

參數	資料類型	說明
xmlobj	XMLVARCHAR、XMLFILE 或 XMLCLOB	直欄名稱。
path	VARCHAR	元素或屬性的位置路徑。

回覆類型

CLOB(10K)

傳回的直欄名稱 (表格函數)

returnedCLOB

範例

在本範例中，所有的零件元素皆從一個採購單中取出。

```
SELECT returnedCLOB as part
   from table(db2xml.extractCLOBs(db2xml.XMLFile('c:\dxx\samples\xml\getstart.xml'),
   '/Order/Part')) as x;
```

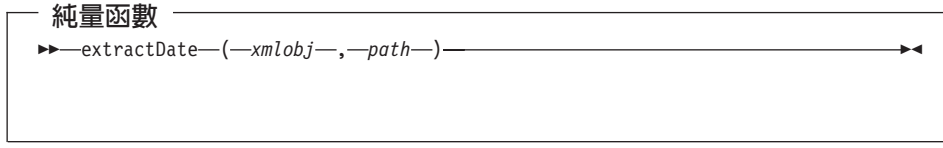
extractDate() 和 extractDates()

目的

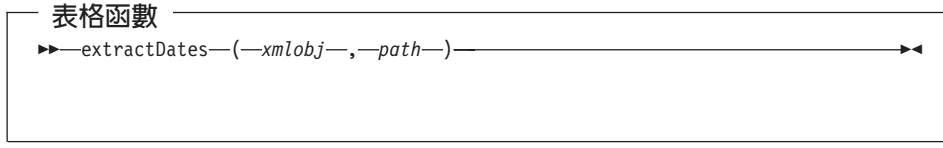
從 XML 文件中取出元素內容或屬性值，傳回 DATE 類型的資料。

語法

純量函數



表格函數



參數

表 36. extractDate 和 extractDates 函數參數

參數	資料類型	說明
xmlobj	XMLVARCHAR、XMLFILE 或 XMLCLOB	直欄名稱。
path	VARCHAR	元素或屬性的位置路徑。

回覆類型

DATE

傳回的直欄名稱 (表格函數)

returnedDate

範例

在下列範例中，ShipDate 的值以 DATE 形式取出。

```
SELECT * from table(db2xml.extractDates(Order,
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/Shipment/ShipDate')) as x;
```

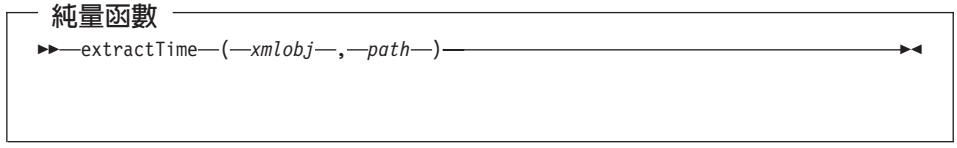
extractTime() 和 extractTimes()

目的

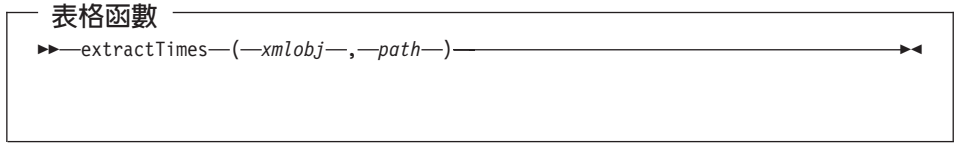
從 XML 文件中取出元素內容或屬性值，傳回 TIME 類型的資料。

語法

純量函數



表格函數



參數

表 37. extractTime 和 extractTimes 函數參數

參數	資料類型	說明
xmlobj	XMLVARCHAR、XMLFILE 或 XMLCLOB	直欄名稱。
path	VARCHAR	元素或屬性的位置路徑。

回覆類型

TIME

傳回的直欄名稱 (表格函數)

returnedTime

範例

本範例使用書籍範例檔。在 XML 檔案 `book.xml` 中搜尋書籍的定價時間，傳回 TIME 的值。

```
<?xml version="1.0">
<DOCTYPE book SYSTEM "c:\dxx\samples\book.dtd">
<book>
<chapter id="1" date="07/01/97">
```

```
<section>This is a section in Chapter One.</section>
<chapter id="2" date="01/02/1997">
<section>This is a section in Chapter Two.</section>
</chapter>
<price date="12/22/1998" time="11.12.13" timestamp="1998-12-22-11.12.13.888888">
38.281
</price>
</book>

select db2xml.extractTime(doc, '/book/price/@time') from t1
```

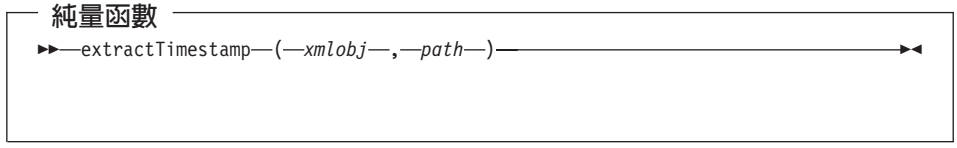
extractTimestamp() 和 extractTimestamps()

目的

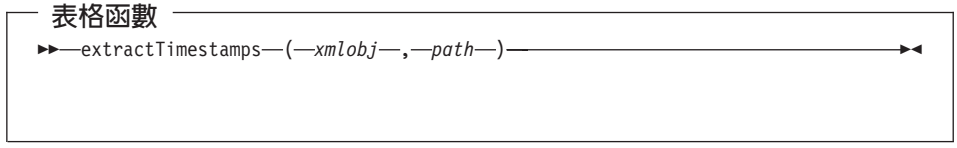
從 XML 文件中取出元素內容或屬性值，傳回 TIMESTAMP 類型的資料。

語法

純量函數



表格函數



參數

表 38. extractTimestamp 和 extractTimestamps 函數參數

參數	資料類型	說明
xmlobj	XMLVARCHAR、XMLFILE 或 XMLCLOB	直欄名稱。
path	VARCHAR	元素或屬性的位置路徑。

回覆類型

TIMESTAMP

傳回的直欄名稱 (表格函數)

returnedTimestamp

範例

本範例使用書籍範例檔。在 XML 檔案 `book.xml` 中搜尋每一本書的定價時間，以 TIMESTAMP 形式取出值。

```
<?xml version="1.0">
<DOCTYPE book SYSTEM "c:\dxx\samples\book.dtd">
<book>
<chapter id="1" date="07/01/97">
```

```
<section>This is a section in Chapter One.</section>
<chapter id="2" date="01/02/1997">
<section>This is a section in Chapter Two.</section>
</chapter>
<price date="12/22/1998" time="11.12.13" timestamp="1998-12-22-11.12.13.888888">
38.281
</price>
</book>

SELECT db2xml.extractTimestamp(doc, '/book/price/@timestamp') from t1
```

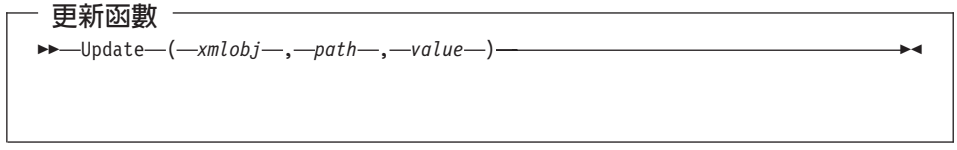
更新函數

您可使用預設的強制轉型函數，將 SQL 基本類型轉換成 XML UDT。 XML Extender 亦提供 Update() 函數來更新 XML 文件中指定的元素和屬性值。

目的

使用 XML UDT 的直欄名稱、位置路徑以及更新值的字串，傳回與第一個輸入參數相同的 XML UDT。您可使用 Update() 函數來指定要更新的元素或屬性。

語法



參數

表 39. UDF Update 參數

參數	資料類型	說明
xmlobj	XMLVARCHAR, XMLCLOB as LOCATOR	直欄名稱。
path	VARCHAR	元素或屬性的位置路徑。
value	VARCHAR	更新字串。

回覆類型

資料類型	回覆類型
XMLVARCHAR	XMLVARCHAR
XMLCLOB as LOCATOR	XMLCLOB

範例

下列範例更新業務員 Sriram Srinivasan 所負責的採購單。

```
UPDATE sales_tab
       set order = Update(order, '/Order/Customer/Name', 'IBM')
   WHERE sales_person = 'Sriram Srinivasan'
```

在本範例中，/Order/Customer/Name 的內容更新爲 IBM。

第10章 XML Extender 儲存程序

XML Extender 提供儲存程序來管理 XML 直欄與集合。 這些儲存程序可從 DB2 從屬站呼叫。 從屬站介面可嵌入 SQL、ODBC 或 JDBC。 關於如何呼叫儲存程序的詳細資料，請參閱 *DB2 UDB Administration Guide*中儲存程序一節。

儲存程序使用 db2xml 爲字首，這是 XML Extender 的 *綱目名稱*

XML Extender 提供三種類型的儲存程序：

- 第180頁的『管理儲存程序』，協助使用者完成管理作業
- 第187頁的『組合儲存程序』，利用現存資料庫表格的資料來產生 XML 文件
- 第194頁的『分解儲存程序』，瓦解或撕毀進入的 XML 文件，並將資料儲存於新建或現存的資料庫表格中

大小參數，*UDB_SIZE*

XML Extender 儲存程序語法使用參數 *UDB_SIZE* 來指定某些字元參數的大小。 根據執行中的 DB2 UDB 版本決定 *UDB_SIZE* 值。 下面列示顯示不同 DB2 UDB 版本的欄位大小值。 當宣告 XML Extender 儲存程序時，請使用正確值。

DB2 UDB 版本	大小
6.1	32
7.1	64

指定併入檔

請確定 XML Extender 外部標頭檔已併入呼叫儲存程序的程式中。標頭檔位於 `DXX_INSTALL/include` 目錄中。 `DXX_INSTALL` 是 XML Extender 的安裝目錄。它依作業系統而定。標頭檔如下：

dxx.h XML Extender 定義常數與資料類型

dxxrc.h XML Extender 回覆碼

包括這些標頭檔的語法如下：

```
#include "dxx.h"
#include "dxxrc.h"
```

請確定在含有編譯選項的 make 檔 (含編譯選項) 中指定了併入檔路徑。

呼叫 XML Extender 儲存程序

一般而言，使用下列語法來呼叫 XML Extender：

```
CALL db2xml!function_entry_point
```

其中：

db2xml
指定 XML Extender 儲存程序的程式庫。它儲存於 sqllib/function 目錄中。

function_entry_point
指定傳遞至儲存程序的引數。

在 CALL 陳述式中，傳遞至儲存程序的引數必須是主變數，而不是常數或表示式。主變數可以有 NULL 指示符。請參閱 DXX_INSTALL/samples/c 及 DXX_INSTALL/samples/cli 目錄下呼叫儲存程序的範例，以及本書的下列章節中：第36頁的『製作 XML 文件』 及 第117頁的『第6章 管理 XML 集合資料』。

在 DXX_INSTALL/samples/c 目錄中，所提供的 SQC 程式檔利用內含的 SQL 來呼叫 XML 集合儲存程序。在 DXX_INSTALL/samples/cli 目錄中，範例檔說明如何利用「呼叫層次介面 (CLI)」來呼叫儲存程序。

開始之前

將您的資料庫與 XML Extender 儲存程序及 DB2 CLI 連結檔連結。您可以使用範例指令檔 getstart_prep.cmd 來連結檔案。這個指令檔位於 DXX_INSTALL/samples/cmd 目錄中。

1. 連接到資料庫。例如：

   ```
   db2 "connect to SALES_DB"
   ```

2. 變更爲 DXX_INSTALL/bnd 目錄，然後將 XML Extender 連結到資料庫。

   ```
   db2 "bind @dxxbind.lst"
   ```

3. 變更爲 sqllib/bnd 目錄，然後將 CLI 連結到資料庫。

   ```
   db2 "bind @db2cli.lst"
   ```

4. 終止連接。

   ```
   db2 "terminate"
   ```

管理儲存程序

這些儲存程序使用於管理作業，如啓用或停用 XML 直欄或集合。 您可以透過 XML Extender 管理精靈及管理指令 **dxxadm** 來呼叫它們。

dxxEnableDB()

目的

啓用資料庫。啓用資料庫後，XML Extender 會建立下列物件：

- XML Extender 使用者定義類型 (UDT)。
- XML Extender 使用者定義函數 (UDF)。
- XML Extender DTD 參照表 DTD_REF，其中儲存 DTD 與每一個 DTD 的相關資訊。關於 DTD_REF 表格的完整說明，請參閱第199頁的『DTD 參照表』。
- XML Extender 用法表 XML_USAGE，其中儲存針對 XML 及每一個集合啓用的每一個直欄之共用資訊。 關於 XML_USAGE 表格的完整說明，請參閱第200頁的『XML 使用表』。

```
dxxEnableDB(char(UDB_SIZE) dbName,       /* input */
            long           returnCode,       /* output */
            varchar(1024)  returnMsg)        /* output */
```

參數

表 40. dxxEnableDB() 參數

參數	說明	IN/OUT 參數
dbName	資料庫名稱。	IN
returnCode	來自儲存程序的回覆碼。	OUT
returnMsg	傳回有關錯誤的訊息文字。	OUT

dxxDisableDB()

目的

停用資料庫。當 XML Extender 停用資料庫時，會捨棄下列物件：

- XML Extender 使用者定義類型 (UDT)。
- XML Extender 使用者定義函數 (UDF)。
- XML Extender DTD 參照表 DTD_REF，其中儲存 DTD 與每一個 DTD 的相關資訊。關於 DTD_REF 表格的完整說明，請參閱第199頁的『DTD 參照表』。
- XML Extender 用法表 XML_USAGE，其中儲存針對 XML 及每一個集合啓用的每一個直欄之共用資訊。 關於 XML_USAGE 表格的完整說明，請參閱第200頁的『XML 使用表』。

重要事項：試圖停用資料庫之前，您必須停用所有 XML 直欄。 XML Extender 無法停用含有針對 XML 啓用的直欄或集合之資料庫。

```
dxxDisableDB(char(UDB_SIZE)     dbName,       /* input */
             long           returnCode,       /* output */
             varchar(1024)  returnMsg)        /* output */
```

參數

表 41. dxxDisableDB() 參數

參數	說明	IN/OUT 參數
dbName	資料庫名稱。	IN
returnCode	來自儲存程序的回覆碼。	OUT
returnMsg	傳回有關錯誤的訊息文字。	OUT

dxxEnableColumn()

目的

啓用 XML 直欄。啓用直欄時，XML Extender 會完成下列作業：

- 判斷 XML 表格是否有主要鍵；如果沒有，XML Extender 會變更 XML 表格，新增一個名爲 DXXROOT_ID 的直欄。
- 使用含有 XML 表格中每一橫列之唯一識別字的直欄，以建立 DAD 檔中指定的輔助表格。 這個直欄爲使用者指定的 *root_id*，或 XML Extender 命名的 DXXROOT_ID。
- 選用性地使用您指定的名稱，針對 XML 表格及其輔助表格建立一個預設概略表。

```
dxxEnableColumn(char(UDB_SIZE) dbName,      /* input */
                char(UDB_SIZE) tbName,      /* input */
                   char(UDB_SIZE) colName,     /* input */
                   CLOB(100K)     DAD,         /* input */
                   char(UDB_SIZE) tablespace,  /* input */
                char(UDB_SIZE) defaultView, /* input */
                char(UDB_SIZE) rootID,      /* input */
             long           returnCode,       /* output */
             varchar(1024)  returnMsg)        /* output */
```

參數

表 42. dxxEnableColumn() 參數

參數	說明	IN/OUT 參數
dbName	資料庫名稱。	IN
tbName	含有 XML 直欄的表格名稱。	IN
colName	XML 直欄的名稱。	IN
DAD	包含 DAD 檔的 CLOB。	IN
tablespace	包含 (預設表格空間以外) 輔助表格的表格空間。 如果沒有指定，則會使用預設的表格空間。	IN
defaultView	結合應用程式表格與輔助表格的預設概略表名稱。	IN
rootID	應用程式表格中當作輔助表格 *root_id* 使用的單一主要鍵名稱。	IN
returnCode	來自儲存程序的回覆碼。	OUT
returnMsg	傳回有關錯誤的訊息文字。	OUT

dxxDisableColumn()

目的

停用已啓用 XML 的直欄。當 XML 直欄停用後，它即不再含有 XML 資料類型。

```
dxxDisableColumn(char(UDB_SIZE) dbName,      /* input */
                 char(UDB_SIZE) tbName,      /* input */
                    char(UDB_SIZE) colName,     /* input */
            long           returnCode,       /* output */
            varchar(1024)  returnMsg)        /* output */
```

參數

表 43. dxxDisableColumn() 參數

參數	說明	IN/OUT 參數
dbName	資料庫名稱。	IN
tbName	含有 XML 直欄的表格名稱。	IN
colName	XML 直欄的名稱。	IN
returnCode	來自儲存程序的回覆碼。	OUT
returnMsg	傳回有關錯誤的訊息文字。	OUT

dxxEnableCollection()

目的

啓用與應用程式表格相關的 XML 集合。

```
dxxEnableCollection(char(UDB_SIZE) dbName,      /* input */
                    char(UDB_SIZE) colName,     /* input */
                    CLOB(100K)     DAD,         /* input */
                    char(UDB_SIZE) tablespace,  /* input */
             long           returnCode,       /* output */
             varchar(1024)  returnMsg)        /* output */
```

參數

表 44. dxxEnableCollection() 參數

參數	說明	IN/OUT 參數
dbName	資料庫名稱。	IN
colName	XML 集合的名稱。	IN
DAD	包含 DAD 檔的 CLOB。	IN
tablespace	包含 (預設表格空間以外) 輔助表格的表格空間。 如果沒有指定，則會使用預設的表格空間。	IN
returnCode	來自儲存程序的回覆碼。	OUT
returnMsg	傳回有關錯誤的訊息文字。	OUT

dxxDisableCollection()

目的

停用已啓用 XML 的集合，把表格及直欄識別爲集合之一部份的標記移除。

```
dxxDisableCollection(char(UDB_SIZE) dbName,      /* input */
                     char(UDB_SIZE) colName,     /* input */
             long           returnCode,       /* output */
             varchar(1024)  returnMsg)        /* output */
```

參數

表 45. dxxDisableCollection() 參數

參數	說明	IN/OUT 參數
dbName	資料庫名稱。	IN
colName	XML 集合的名稱。	IN
returnCode	來自儲存程序的回覆碼。	OUT
returnMsg	傳回有關錯誤的訊息文字。	OUT

組合儲存程序

組合儲存程序 dxxGenXML() 及 dxxRetrieveXML() 利用現存資料庫表格的資料來建立 XML 文件。 dxxGenXML() 儲存程序選擇 DAD 檔作爲輸入；它不需要已啓用的 XML 集合。 dxxRetrieveXML() 儲存程序選擇已啓用的 XML 集合名稱作爲輸入。

dxxGenXML()

目的

使用 XML 集合表格 (它們由 DAD 檔中 <Xcollection> 指定) 中儲存的資料來建構 XML 文件，然後將每一個 XML 文件各以一個橫列插入結果表格中。 您亦可開啓結果表格上的游標，然後提取結果集。

爲了提供彈性，dxxGenXML() 也容許使用者指定結果表格中所要建立的最大橫列數。這可減少任何測試處理期間，應用程式等待結果的時間量。此儲存程序傳回表格中實際的橫列數及任何錯誤資訊，包括錯誤碼及錯誤訊息。

爲了支援動態查詢，dxxGenXML() 使用一個輸入參數，叫作 *override*。根據輸入 *overrideType*，應用程式可置換 SQL 對映的 SQL_stmt，或置換 DAD 檔中 RDB_node 對映的 RDB_node 內之條件。 輸入參數 *overrideType* 是用來闡明 *override* 的類型。 *override* 參數的詳細資訊，請參閱第121頁的『動態置換 DAD 檔中的值』。

```
dxxGenXML(CLOB(100K)     DAD,             /* input */
          char(UDB_SIZE) resultTabName,   /* input */
      integer        overrideType     /* input */
          varchar(1024)  override,        /* input */
          integer        maxRows,         /* input */
          integer        numRows,         /* output */
        long           returnCode,      /* output */
        varchar(1024)  returnMsg)       /* output */
```

參數

表 46. dxxGenXML() 參數

參數	說明	IN/OUT 參數
DAD	包含 DAD 檔的 CLOB。	IN
resultTabName	呼叫前應存在的結果表格名稱。此表格僅包含一個 XMLVARCHAR 或 XMLCLOB 類型的直欄。	IN
overrideType	指示下列 *override* 參數類型的旗號： • **NO_OVERRIDE**：不置換。 • **SQL_OVERRIDE**：由 SQL_stmt 置換。 • **XML_OVERRIDE**：由 XPath 型條件置換。	IN

表 46. dxxGenXML() 參數 (繼續)

參數	說明	IN/OUT 參數
override	置換 DAD 檔中的條件。輸入值視 *overrideType* 而定。 • **NO_OVERRIDE**：空字串。 • **SQL_OVERRIDE**：有效的 SQL 陳述式。若要使用這個 *overrideType*，DAD 檔中必須使用 SQL 對映。此輸入 SQL 陳述式會置換 DAD 檔中的 SQL_stmt。 • **XML_OVERRIDE**：包含以雙引號括住的一或多個表示式之字串，這些表示式以 "AND" 區隔。若要使用這個 *overrideType*，DAD 檔中必須使用 RDB_node 對映。	IN
maxRows	結果表格中最大橫列數。	IN
numRows	結果表格中實際產生的列數。	OUT
returnCode	來自儲存程序的回覆碼。	OUT
returnMsg	傳回有關錯誤的訊息文字。	OUT

範例

下列範例假設結果表格是使用 XML_ORDER_TAB 名稱建立的，而且該表格具有一個 XMLVARCHAR 類型的直欄。

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
     SQL TYPE is CLOB(100K) dad;              /* DAD */
     SQL TYPE is CLOB_FILE  dadFile;          /* dad file */
         char    result_tab[32];  /* name of the result table */
              char    override[2];      /* override, will set to NULL*/
              short   overrideType;     /* defined in dxx.h */
         short   max_row;         /* maximum number of rows */
         short   num_row;         /* actual number of rows */
         long    returnCode;      /* return error code */
              char    returnMsg[1024];  /* error message text */
```

```
    short                dad_ind;
    short   rtab_ind;
    short   ovtype_ind;
    short   ov_inde;
    short   maxrow_ind;
    short   numrow_ind;
    short                returnCode_ind;
    short                returnMsg_ind;

     EXEC SQL END DECLARE SECTION;

    /* create table */
    EXEC CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

/* read data from a file to a CLOB */
         strcpy(dadfile.name,"e:\dxx\dad\litem3.dad");
         dadfile.name_length = strlen("e:\dxx\dad\litem3.dad");
dadfile.file_options = SQL_FILE_READ;
EXEC SQL VALUES (:dadfile) INTO :dad;
    strcpy(result_tab,"xml_order_tab");
         override[0] = '\0';
         overrideType = NO_OVERRIDE;
    max_row = 500;
    num_row = 0;
    returnCode = 0;
    msg_txt[0] = '\0';
    collection_ind = 0;
   dad_ind = 0;
    rtab_ind = 0;
         ov_ind = -1;
    ovtype_ind = 0;
    maxrow_ind = 0;
    numrow_ind = -1;
    returnCode_ind = -1;
    returnMsg_ind = -1;

   /* Call the store procedure */
EXEC SQL CALL dxxGenXML(:dad:dad_ind;
           :result_tab:rtab_ind,
     :overrideType:ovtype_ind,:override:ov_ind,
           :max_row:maxrow_ind,:num_row:numrow_ind,
                 :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

dxxRetrieveXML()

目的

啓用要同時用於組合及分解的同一個 DAD 檔。 儲存程序 dxxRetrieveXML() 也提供一個方法來擷取分解的 XML 文件。 作爲輸入時，dxxRetrieveXML() 會建立一個緩衝區，其中包含 DAD 檔、建立的結果表格名稱，以及要傳回的最大橫列數。 它傳回結果表格的結果集、結果集中的實際橫列數、錯誤碼及訊息文字。

爲了支援動態查詢，dxxRetrieveXML() 使用一個輸入參數，叫作 *override*。根據輸入 *overrideType*，應用程式可置換 SQL 對映的 SQL_stmt，或置換 DAD 檔中 RDB_node 對映的 RDB_node 內之條件。輸入參數 *overrideType* 是用來闡明 *override* 的類型。 *override* 參數的詳細資訊，請參閱第121頁的『動態置換 DAD 檔中的值』。

dxxRetrieveXML() 的 DAD 檔需求與 dxxGenXML() 的需求一樣。 唯一的差異是 DAD 並不是 dxxRetrieveXML() 的輸入參數，而是已啓用的 XML 集合名稱。

```
dxxRetrieveXML(char(UDB_SIZE) collectionName,  /* input */
               char(UDB_SIZE) resultTabName,   /* input */
               integer        overrideType,    /* input */
               varchar(1024)  override,        /* input */
               integer        maxRows,         /* input */
               integer        numRows,         /* output */
             long           returnCode,       /* output */
             varchar(1024)  returnMsg)        /* output */
```

參數

表 47. dxxRetrieveXML() 參數

參數	說明	IN/OUT 參數
collectionName	已啓用的 XML 集合名稱。	IN
resultTabName	呼叫前應存在的結果表格名稱。此表格僅包含一個 XMLVARCHAR 或 XMLCLOB 類型的直欄。	IN
overrideType	指示下列 *override* 參數類型的旗號： • **NO_OVERRIDE**：不置換。 • **SQL_OVERRIDE**：由 SQL_stmt 置換。 • **XML_OVERRIDE**：由 XPath 型條件置換。	IN

表 47. *dxxRetrieveXML() 參數 (繼續)*

參數	說明	IN/OUT 參數
override	置換 DAD 檔中的條件。輸入值視 *overrideType* 而定。 • **NO_OVERRIDE**：空字串。 • **SQL_OVERRIDE**：有效的 SQL 陳述式。若要使用這個 *overrideType*，DAD 檔中必須使用 SQL 對映。此輸入 SQL 陳述式會置換 DAD 檔中的 SQL_stmt。 • **XML_OVERRIDE**：包含以雙引號括住的一或多個表示式之字串，這些表示式以 ″AND″ 區隔。若要使用這個 *overrideType*，DAD 檔中必須使用 RDB_node 對映。	IN
maxRows	結果表格中最大橫列數。	IN
numRows	結果表格中實際產生的橫列數。	OUT
returnCode	來自儲存程序的回覆碼。	OUT
returnMsg	傳回有關錯誤的訊息文字。	OUT

範例

下列爲 dxxRetrieveXML() 的呼叫範例。本範例中，結果表格是使用 XML_ORDER_TAB 名稱建立的，而且該表格具有一個 XMLVARCHAR 類型的直欄。

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
         char    collection[32];  /* dad buffer */
         char    result_tab[32];  /* name of the result table */
              char    override[2];      /* override, will set to NULL*/
              short   overrideType;     /* defined in dxx.h */
         short   max_row;         /* maximum number of rows */
         short   num_row;         /* actual number of rows */
         long    returnCode;      /* return error code */
              char    returnMsg[1024];  /* error message text */
```

```
short   dadbuf_ind;
short   rtab_ind;
short   ovtype_ind;
short   ov_inde;
short   maxrow_ind;
short   numrow_ind;
short                 returnCode_ind;
short                 returnMsg_ind;

 EXEC SQL END DECLARE SECTION;

/* create table */
EXEC SQL CREATE TABLE xml_order_tab (xmlorder XMLVarchar);

/* initialize host variable and indicators */
strcpy(collection,"sales_ord");
strcpy(result_tab,"xml_order_tab");
     override[0] = '\0';
     overrideType = NO_OVERRIDE;
max_row = 500;
num_row = 0;
returnCode = 0;
msg_txt[0] = '\0';
collection_ind = 0;
rtab_ind = 0;
     ov_ind = -1;
ovtype_ind = 0;
maxrow_ind = 0;
numrow_ind = -1;
returnCode_ind = -1;
returnMsg_ind = -1;

/* Call the store procedure */
EXEC SQL CALL dxxRetrieve(:collection:collection_ind;
       :result_tab:rtab_ind,
 :overrideType:ovtype_ind,:override:ov_ind,
       :max_row:maxrow_ind,:num_row:numrow_ind,
             :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

分解儲存程序

分解儲存程序 dxxInsertXML() 及 dxxShredXML()，用來瓦解或撕毀進入的 XML 文件，並將資料儲存於新建或現存的資料庫表格中。 dxxInsertXML() 儲存程序選擇已啓用的 XML 集合名稱作爲輸入。 dxxShredXML() 儲存程序選擇 DAD 檔作爲輸入；它不需要已啓用的 XML 集合。

dxxShredXML()

目的

dxxShredXML() 儲存程序是 dxxGenXML() 的配對儲存程序。爲了 dxxShredXML() 的運作，DAD 檔中指定的所有表格都必須存在，同時 DAD 中指定的所有直欄及它們的資料類型要與現存表格一致。儲存程序 dxxShredXML() 不需要結合表格之間的主要鍵與外來鍵關係，它是啓用集合處理期間由 XML Extender 所建立的。不過，表格中必須有 root element_node 的 RDB_node 中所指定之結合條件直欄。

```
dxxShredXML(CLOB(100K)    DAD,            /* input */
            CLOB(1M)       xmlobj,           /* input */
            long           returnCode,       /* output */
            varchar(1024)  returnMsg)        /* output */
```

參數

表 48. dxxShredXML() 參數

參數	說明	IN/OUT 參數
DAD	包含 DAD 檔的 CLOB。	IN
xmlobj	XMLCLOB 類型中的 XML 文件物件。	IN
returnCode	來自儲存程序的回覆碼。	OUT
returnMsg	傳回有關錯誤的訊息文字。	OUT

範例

下列爲呼叫 dxxShredXML() 的一個範例。

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
         SQL TYPE is CLOB      dad;             /* DAD*/
         SQL TYPE is CLOB_FILE dadFile;         /* DAD file*/
         SQL TYPE is CLOB      xmlDoc;          /* input XML document */
         SQL TYPE is CLOB_FILE xmlFile;          /* input XML file */
         long                  returnCode;       /* error code */
              char    returnMsg[1024];  /* error message text */
         short                 dad_ind;
         short                 xmlDoc_ind;
         short                 returnCode_ind;
         short                 returnMsg_ind;
          EXEC SQL END DECLARE SECTION;

         /* initialize host variable and indicators */
              strcpy(dadFile.name,"c:\dxx\samples\dad\getstart_xcollection.dad");
              dadFile.name_length=strlen("c:\dxx\samples\dad\getstart_xcollection.dad");
        dadFile.file_option=SQL_FILE_READ;
```

```
      strcpy(xmlFile.name,"c:\dxx\samples\cmd\getstart.xml");
      xmlFile.name_length=strlen("c:\dxx\samples\cmd\getstart.xml");
xmlFile.file_option=SQL_FILE_READ;
SQL EXEC VALUES (:dadFile) INTO :dad;
SQL EXEC VALUES (:xmlFile) INTO :xmlDoc;
 returnCode = 0;
 returnMsg[0] = '\0';
dad_ind = 0;
xmlDoc_ind = 0;
 returnCode_ind = -1;
 returnMsg_ind = -1;

/* Call the store procedure */
EXEC SQL CALL db2xml!dxxShredXML(:dad:dad_ind;
              :xmlDoc:xmlDoc_ind,
              :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

dxxInsertXML()

目的

使用兩個輸入參數：已啓用的 XML 集合名稱及要分解的 XML 文件；傳回兩個輸出參數：回覆碼及傳回訊息。

```
dxxInsertXML(char(UDB_SIZE) collectionName,   /* input */
             CLOB(1M)       xmlobj,           /* input */
             long           returnCode,       /* output */
             varchar(1024)  returnMsg)        /* output */
```

參數

表 49. dxxInsertXML() 參數

參數	說明	IN/OUT 參數
collectionName	已啓用的 XML 集合名稱。	IN
xmlobj	CLOB 類型中的 XML 文件物件。	IN
returnCode	來自儲存程序的回覆碼。	OUT
returnMsg	傳回有關錯誤的訊息文字。	OUT

範例

下列範例中，dxxInsertXML() 呼叫分解輸入 XML 文件 e:\xml\order1.xml，並根據 DAD 檔中指定的對映 (啓用該呼叫的對應)，將資料插入 SALES_ORDER 集合表格中。

```
#include "dxx.h"
#include "dxxrc.h"

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
         char                  collection[64];   /* name of an XML collection */
              SQL TYPE is CLOB_FILE xmlDoc;          /* input XML document */
         long                  returnCode;       /* error code */
              char    returnMsg[1024];  /* error message text */
         short                 collection_ind;
         short                 xmlDoc_ind;
         short                 returnCode_ind;
         short                 returnMsg_ind;
          EXEC SQL END DECLARE SECTION;

         /* initialize host variable and indicators */
         strcpy(collection,"sales_ord")
              strcpy(xmlobj.name,"c:\dxx\samples\cmd\getstart.xml");
              xmlobj.name_length=strlen("c:\dxx\samples\cmd\getstart.xml");
         xmlobj.file_option=SQL_FILE_READ;
         returnCode = 0;
         returnMsg[0] = '\0';
```

```
 collection_ind = 0;
 xmlobj_ind = 0;
 returnCode_ind = -1;
 returnMsg_ind = -1;

/* Call the store procedure */
 EXEC SQL CALL db2xml!dxxInsertXML(:collection:collection_ind;
                      :xmlobj:xmlobj_ind,
             :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

第11章 管理支援表

啓用資料庫之後，就會建立 DTD 參照表、DTD_REF 及 XML_USAGE 表格。DTD_REF 表格含有關於 DTD 的全部資訊。XML_USAGE 表格儲存每一個已啓用 XML 的直欄的共用資訊。

大小參數，*UDB_SIZE*

XML Extender 使用參數 *UDB_SIZE* 指定某些字元參數的大小。 根據執行中的 DB2 UDB 版本決定 *UDB_SIZE* 值。 下面列示顯示不同 DB2 UDB 版本的欄位大小值。

DB2 UDB 版本	大小
6.1	32
7.1	64

DTD 參照表

XML Extender 也作爲 XML DTD 儲存庫。資料庫啓用 XML 之後，就會建立 DTD 參照表 DTD_REF。 本表格的每一橫列代表一個具有其它 meta 資料資訊的 DTD。 使用者可存取本表格以及插入自己的 DTD。 DTD_REF 表格中的 DTD 是用來驗證 XML 文件，以及協助應用程式定義 DAD 檔。 它的綱目名稱爲 db2xml。 DTD_REF 表格可在表50中顯示直欄。

表 50. DTD_REF 表格

直欄名稱	資料類型	說明
DTDID	VARCHAR	主要鍵 (唯一但不是 NULL)。它是用來識別 DTD。在 DAD 檔中指定它時，DAD 檔必須遵循 DTD 定義的綱目。
CONTENT	XMLCLOB	DTD 的內容。
USAGE_COUNT	INTEGER	資料庫中的 XML 直欄數目及 XML 集合數目，使用 DTD 定義自己的 DAD 檔。
AUTHOR	VARCHAR	DTD 作者，使用者輸入的可選用的 '資訊。
CREATOR	VARCHAR	執行第一個插入動作的使用者 ID。 CREATOR 直欄是可選用的。

表 50. DTD_REF 表格 (繼續)

直欄名稱	資料類型	說明
UPDATOR	VARCHAR	執行前次更新的使用者 ID。 UPDATOR 直欄是可選用的。

限制：唯有 USAGE_COUNT 是零時，應用程式才能修改 DTD。

XML 使用表

儲存每一個已啓用 XML 的直欄的共用資訊。XML_USAGE 表格的綱目名稱是 db2xml，它的主要鍵是 (*table_name*，*col_name*)。 XML_USAGE 表格是在資料庫以 表51 所列出的直欄啓用時建立的。

表 51. XML_USAGE 表格

直欄名稱	資料類型	說明
table_schema	CHAR(*UDB_SIZE*+1)	就 XML 直欄而言，含有 XML 直欄的使用者表格綱目名稱。就 XML 集合而言，″DXX_COLL″ 值可作爲預設的綱目名稱。
表格_名稱	CHAR(*UDB_SIZE*+1)	就 XML 直欄而言，含有 XML 直欄的使用者表格名稱。就 XML 集合而言，一個 ″DXX_COLLECTION″ 值，它定義此實體爲一集合。
col_name	CHAR(*UDB_SIZE*+1)	XML 直欄名稱或 XML 集合名稱。它是組合鍵以及表格_名稱的一部份。
DTDID	CHAR(*UDB_SIZE*+1)	定義 DAD 檔的 DTD_REF 表中的 DTD ID。它是外來鍵。
DAD	CLOB(100k)	與此直欄相關的 DAD 檔內容。
default_view	CHAR(*UDB_SIZE*+1)	儲存預設檢視名稱 (若有的話)。
trigger_suffix	CHAR(8)	非 NULL。用於專用觸發函式名稱。
驗證	INTEGER	1 代表是，0 代表否。

限制：唯有 USAGE_COUNT 是零時，應用程式才能修改 DTD。

第12章 診斷資訊

程式中所有內含的 SQL 陳述式和 DB2 指令行介面 (CLI) 呼叫 (包括那些呼叫 DB2 XML Extender 使用者定義函數 (UDF) 的陳述式和呼叫)，均產生訊息碼，指出內含的 SQL 陳述式或 DB2 CLI 呼叫是否順利執行。

您的程式可擷取這些訊息碼的補充資訊。它包括 SQLSTATE 資訊和錯誤訊息。您可使用此診斷資訊來找出和修正程式中的問題。

有時無法輕易地診斷出問題的來源。 在這些情況下，您可能需要將資訊提供給「軟體支援中心」提供者，以隔離並修正問題。XML Extender 提供一個追蹤機能，可記錄 XML Extender 活動。 對於 IBM 軟體支援中心而言，追蹤資訊具有參考價值。 您應該依照 IBM 軟體支援中心的指示，使用追蹤機能。

本章說明如何存取此診斷資訊。內容包括：

- 如何處理 XML Extender UDF 回覆碼。
- 如何控制追蹤

本章亦列示和說明 XML Extender 可能傳回的 SQLSTATE 訊息碼和錯誤訊息。

處理 UDF 回覆碼

內含的 SQL 陳述式傳回 SQLCA 結構中 SQLCODE、SQLWARN、及 SQLSTATE 欄位內的訊息碼。此結構定義於 SQLCA INCLUDE 檔案中。(關於 SQLCA 結構和 SQLCA INCLUDE 檔案的詳細資訊，請參閱 *DB2 Application Development Guide*。)

DB2 CLI 呼叫傳回 SQLCODE 和 SQLSTATE 值，您可使用 SQLError 函數來擷取。 (關於使用 SQLError 函數來擷取錯誤資訊的詳細資訊，請參閱 *CLI Guide and Reference*。)

SQLCODE 值 0 表示陳述式已順利執行 (可能附帶警告狀況)。正數的 SQLCODE 值表示陳述式已順利執行，但發生警告狀況。 (內含的 SQL 陳述式傳回警告的相關資訊，警告是以 SQLWARN 欄位中的 0 或正數的 SQLCODE 值表示。) 負數的 SQLCODE 值表示發生錯誤。

DB2 在每一個 SQLCODE 值上結合一則訊息。若 XML Extender UDF 發現警告或錯誤狀況，則傳遞相關資訊到 DB2 來併入 SQLCODE 訊息內。

SQLSTATE 值包含訊息碼來補充 SQLCODE 訊息。關於 XML Extender 傳回的每一個 SQLSTATE 訊息碼說明，請參閱『SQLSTATE 訊息碼』。

呼叫 DB2 XML Extender UDF 的內含的 SQL 陳述式和 DB2 CLI 呼叫，可傳回這些 UDF 專用的 SQLCODE 訊息和 SQLSTATE 值，但 DB2 以對於其它內含的 SQL 陳述式或 DB2 CLI 呼叫的相同方式來傳回這些值。因此，您存取這些值的方式，與沒有啓動 DB2 XML Extender UDF 的內含的 SQL 陳述式或 DB2 CLI 呼叫相同。

關於 XML Extender 傳回的 SQLSTATE 值和相關訊息的訊息碼，請參閱『SQLSTATE 訊息碼』。關於每一則訊息的資訊，請參閱第207頁的『訊息』。

處理儲存程序回覆碼

XML Extender 提供回覆碼來協助解決儲存程序的問題。當您收到儲存程序的回覆碼時，請檢查下列檔案，找出回覆碼的相對應 XML Extender 錯誤訊息碼和常數符號。

```
DXX_INSTALL/include/dxxrc.h
```

您可參考第207頁的『訊息』中的錯誤訊息碼，使用說明內的診斷資訊。

SQLSTATE 訊息碼

表52列示和說明 XML Extender 傳回的 SQLSTATE 值。每一個 SQLSTATE 值的說明包含該符號意義。表格亦列示每一個 SQLSTATE 值相關的訊息碼。關於每一則訊息的資訊，請參閱第207頁的『訊息』。

表 52. SQLSTATE 碼及相關訊息碼

SQLSTATE	訊息碼	說明
00000	DXX*nnnn*I	未發生錯誤。
01HX0	DXXD003W	XML 文件遺失路徑表示式中指定的屬性或元素。
38X00	DXXC000E	XML Extender 無法開啓指定的檔案。
38X01	DXXA072E	XML Extender 試圖在啓用資料庫之前自動連結它。
	DXXC001E	XML Extender 找不到指定的檔案。
38X02	DXXC002E	XML Extender 無法從指定的檔案讀取資料。

表 52. SQLSTATE 碼及相關訊息碼 (繼續)

SQLSTATE	訊息碼	說明
38X03	DXXC003E	XML Extender 無法將資料寫入檔案。
	DXXC011E	XML Extender 無法將資料寫入追蹤控制項檔案。
38X04	DXXC004E	XML Extender 無法操作指定的定位器。
38X05	DXXC005E	檔案大小大於 XMLVarchar 大小，且 XML Extender 無法從該檔案匯入全部資料。
38X06	DXXC006E	檔案大小大於 XMLCLOB 大小，且 XML Extender 無法從該檔案匯入全部資料。
38X07	DXXC007E	「LOB 定位器」中的位元組數不等於檔案大小。
38X08	DXXD001E	純量解壓縮函數使用多次出現的位置路徑。純量函數僅可使用沒有多次出現項目的位置路徑。
38X09	DXXD002E	路徑表示式的語法不正確。
38X10	DXXG002E	XML Extender 無法從作業系統配置記憶體。
38X11	DXXA009E	這個儲存程序是 XML 直欄專用。
38X12	DXXA010E	當試圖啓用直欄時， XML Extender 在「文件存取定義 (DAD)」檔中找不到 DTDID (它是針對 DTD 指定的識別字)。
38X14	DXXD000E	試圖將無效的文件儲存到表格中。驗證失敗。
38X15	DXXA056E	DAD 檔中的驗證元素錯誤或遺失。
	DXXA057E	DAD 檔中輔助表格的名稱屬性錯誤或遺失。
	DXXA058E	DAD 檔中直欄的名稱屬性錯誤或遺失。
	DXXA059E	DAD 檔中直欄的類型屬性錯誤或遺失。
	DXXA060E	DAD 檔中直欄的路徑屬性錯誤或遺漏。

表 52. SQLSTATE 碼及相關訊息碼 (繼續)

SQLSTATE	訊息碼	說明
	DXXA061E	DAD 檔中直欄的 multi_occurrence 屬性錯誤或遺失。
	DXXQ000E	DAD 檔中遺失必要的元件。
38X16	DXXG004E	某必要參數的 NULL 值傳至 XML 儲存程序。
38X17	DXXQ001E	DAD 中的 SQL 陳述式或置換它的 SQL 陳述式無效。建立 XML 文件需要 SELECT 陳述式。
38X18	DXXG001E	XML Extender 發現內部錯誤。
	DXXG006E	使用 CLI 時，XML Extender 發現內部錯誤。
38X19	DXXQ002E	系統記憶體或磁碟空間不足。沒有足夠的空間來保存產生的 XML 文件。
38X20	DXXQ003W	使用者定義的 SQL 查詢所產生之 XML 文件數超過指定的最大值。僅傳回指定的文件數目。
38X21	DXXQ004E	指定的直欄不是 SQL 查詢結果的其中一個直欄。
38X22	DXXQ005E	SQL 查詢與 XML 的對映不正確。
38X23	DXXQ006E	DAD 檔中 attribute_node 元素沒有名稱屬性。
38X24	DXXQ007E	DAD 檔中 attribute_node 元素沒有直欄元素或 RDB_node。
38X25	DXXQ008E	DAD 檔中 text_node 元素沒有直欄元素。
38X26	DXXQ009E	系統型錄中找不到指定的結果表格。
38X27	DXXQ010E	attribute_node 或 text_node 的 RDB_node 必須有一個表格。
	DXXQ011E	attribute_node 或 text_node 的 RDB_node 必須有一個直欄。
	DXXQ017E	XML Extender 建立的 XML 文件太大，無法填入結果表格的直欄中。
38X28	DXXQ012E	處理程序 DAD 時，XML Extender 找不到期望的元素。

表 52. SQLSTATE 碼及相關訊息碼 (繼續)

SQLSTATE	訊息碼	說明
	DXXQ016E	所有表格皆須定義於 DAD 檔中頂端元素的 RDB_node 中。次元素表格必須符合頂端元素中定義的表格。此 RDB_node 中的表格名稱不在頂端元素中。
38X29	DXXQ013E	DAD 檔中元素表格或直欄都必須有名稱。
	DXXQ015E	文件存取定義 (DAD) 檔中的條件元素之條件格式無效。
38X30	DXXQ014E	DAD 檔中 element_node 元素沒有名稱屬性。
	DXXQ018E	在對映 SQL 至 XML 的 DAD 檔中，SQL 陳述式內遺失 ORDER BY 子句。
38X31	DXXQ019E	在 DAD 檔中 objids 元素沒有對映 SQL 至 XML 的直欄元素。
38X36	DXXA073E	當使用者嘗試啓用它時，資料庫尚未連結。
38X37	DXXG007E	伺服器作業系統語言環境與 DB2 字碼頁不一致。
38X38	DXXG008E	在字碼頁表格中，找不到伺服器作業系統語言環境。
38x33	DXXG005E	這個版次不支援此參數，未來版次會支援此參數。
38x34	DXXG000E	指定的檔名無效。

訊息

XML Extender 提供錯誤訊息來協助分析問題。

錯誤訊息

XML Extender 在完成作業或偵測到錯誤時，將產生下列訊息。

DXXA000I 啓用直欄 <*column_name*>。請稍候。

解說: 這是參考訊息。

使用者回應: 不需要任何動作。

DXXA001S 建置版 <*build_ID*> 的檔案 <*file_name*> 中的第 <*line_number*> 行發生異常錯誤。

解說: 發生異常錯誤。

使用者回應: 如果這個錯誤還是發生，請洽您的「軟體服務提供者」。報告錯誤時，請務必附上所有訊息文字、追蹤檔以及如何使問題重現的說明。

DXXA002I 連接至資料庫 <*database*>。

解說: 這是參考訊息。

使用者回應: 不需要任何動作。

DXXA003E 無法連接至資料庫 <*database*>。

解說: 指定的資料庫可能不存在或已損毀。

使用者回應:

1. 請確定所指定的是正確的資料庫。
2. 請確定資料庫存在且爲可存取的。
3. 判斷資料庫是否已損毀。 如果已損毀，請聯絡資料庫管理者從備份回復它。

DXXA004E 無法啓用資料庫 <*database*>。

解說: 資料庫可能已啓用或損毀。

使用者回應:

1. 判斷資料庫是否已啓用。
2. 判斷資料庫是否已損毀。如果已損毀，請聯絡資料庫管理者從備份回復它。

DXXA005I 啓用資料庫 <*database*>。請稍候。

解說: 這是參考訊息。

使用者回應: 不需要任何動作。

DXXA006I 已順利啓用資料庫 <*database*>。

解說: 這是參考訊息。

使用者回應: 不需要任何動作。

DXXA007E 無法停用資料庫 <*database*>。

解說: XML Extender 無法停用含有任何 XML 直欄或集合的資料庫。

使用者回應: 請備份任何重要資料、停用任何 XML 直欄或集合，然後更新或捨棄任何表格，直到資料庫內沒有任何 XML資料類型爲止。

DXXA008I 停用直欄 <*column_name*>。請稍候。

解說: 這是參考訊息。

使用者回應: 不需要任何動作。

DXXA009E DAD 檔中並未指定 Xcolumn 標示。

解說: 這個儲存程序是 XML 直欄專用。

使用者回應: 請確定 DAD 檔中已正確指定了 Xcolumn 標示。

DXXA010E 試圖尋找 DTD ID <*dtdid*> 失敗。

解說: 當試圖啓用直欄時， XML Extender 在「文件存取定義 (DAD)」檔中找不到 DTDID (它是針對 DTD 指定的識別字)。

使用者回應: 請確定 DAD 檔中指定了正確的 DTDID 值。

DXXA011E 將記錄插入 DB2XML.XML_USAGE 表格中失敗。

解說: 試圖啓用直欄時，XML Extender 無法在 DB2XML.XML_USAGE 表格中插入記錄。

使用者回應: 請確定 DB2XML.XML_USAGE 表格存在，並且該表格中尚無相同名稱的記錄。

DXXA012E 試圖更新 DB2XML.DTD_REF 表格失敗。

解說: 當試圖啓用直欄時，XML Extender 無法更新 DB2XML.DTD_REF 表格。

使用者回應: 請確定 DB2XML.DTD_REF 表格存在。 判斷表格是否受損，或管理使用者 ID 是否有正確的權限來更新表格。

DXXA013E 試圖變更表格 <*table_name*> 失敗。

解說: 當試圖啓用直欄時，XML Extender 無法變更指定的表格。

使用者回應: 請檢查變更該表格是否需要專用權。

DXXA014E 指定的 root ID 直欄：<*root_id*> 不是表格 <*table_name*> 的單一主要鍵。

解說: 指定的 root ID 不是一個鍵值，或不是表格 *table_name* 的單一鍵。

使用者回應: 請確定指定的 root ID 是表格的單一主要鍵。

DXXA015E 直欄 DXXROOT_ID 已存在於表格 <*table_name*> 中。

解說: 直欄 DXXROOT_ID 存在，但不是由 XML Extender 建立。

使用者回應: 啓用直欄時，使用不同的直欄名稱對 root ID 選項指定一主要直欄。

DXXA016E 輸入表格 <*table_name*> 不存在。

解說: XML Extender 在系統型錄中找不到指定的表格。

使用者回應: 請確定表格存在於資料庫中且已正確指定。

DXXA017E 輸入直欄 <*column_name*> 不存在於指定的表格 <*table_name*> 中。

解說: XML Extender 在系統型錄中找不到直欄。

使用者回應: 請確定直欄存在於使用者表格中。

DXXA018E 未對 XML 資料啓用指定的直欄。

解說: 試圖停用直欄時，XML Extender 在 DB2XML.XML_USAGE 表格中找不到直欄，表示直欄尚未啓用。如果直欄不是 XML 型直欄，則不需要停用。

使用者回應: 不需要任何動作。

DXXA019E 啓用直欄的必要輸入參數為 NULL。

解說: enable_column() 儲存程序的必要輸入參數爲 NULL。

使用者回應: 檢查 enable_column() 儲存程序的全部輸入參數。

DXXA020E 表格 <*table_name*> 中找不到直欄。

解說: 當試圖建立預設概略表時，XML Extender 在指定的表格中找不到直欄。

使用者回應: 請確定直欄與表格名稱皆已正確指定。

DXXA021E 無法建立預設概略表 <*default_view*>。

解說: 當試圖啓用直欄時，XML Extender 無法建立指定的概略表。

使用者回應: 請確定預設概略表名稱是唯一的。如果概略表的名稱已存在，請爲預設概略表指定唯一名稱。

DXXA022I 直欄 <*column_name*> 已啓用。

解說: 這是參考訊息。

使用者回應: 不需要回應。

DXXA023E 找不到 DAD 檔。

解說: 當試圖停用直欄時，XML Extender 找不到「文件存取定義 (DAD)」檔。

使用者回應: 請確定所指定的是正確的資料庫名稱、表格名稱或直欄名稱。

DXXA024E 存取系統型錄表格時，XML Extender 發現內部錯誤。

解說: XML Extender 無法存取系統型錄表格。

使用者回應: 請確定資料庫處於穩定狀態下。

DXXA025E 無法捨棄預設概略表 <*default_view*>。

解說: 當試圖停用直欄時，XML Extender 無法捨棄預設概略表。

使用者回應: 請確定 XML Extender 的管理使用者 ID 具有捨棄預設概略表的必要專用權。

DXXA026E 無法捨棄輔助表格 <*side_table*>。

解說: 當試圖停用直欄時，XML Extender 無法捨棄指定的表格。

使用者回應: 請確定 XML Extender 的管理使用者 ID 具有捨棄表格的必要專用權。

DXXA027E 無法停用直欄。

解說: 由於內部觸發函式失敗，導致 XML Extender 無法停用直欄。可能的原因：

使用者回應: 使用追蹤機能來建立追蹤檔，並嘗試更正問題。如是這個問題還是發生，請洽您的「軟體服務提供者」並提供追蹤檔。

DXXA028E 無法停用直欄。

解說: 由於內部觸發函式失敗，導致 XML Extender 無法停用直欄。可能的原因：

使用者回應: 使用追蹤機能來建立追蹤檔，並嘗試更正問題。如是這個問題還是發生，請洽您的「軟體服務提供者」並提供追蹤檔。

DXXA029E 無法停用直欄。

解說: 由於內部觸發函式失敗，導致 XML Extender 無法停用直欄。可能的原因：

使用者回應: 使用追蹤機能來建立追蹤檔，並嘗試更正問題。如是這個問題還是發生，請洽您的「軟體服務提供者」並提供追蹤檔。

DXXA030E 無法停用直欄。

解說: 由於內部觸發函式失敗，導致 XML Extender 無法停用直欄。可能的原因：

使用者回應: 使用追蹤機能來建立追蹤檔，並嘗試更正問題。如是這個問題還是發生，請洽您的「軟體服務提供者」並提供追蹤檔。

DXXA031E 無法將應用程式表格中的 DXXROOT_ID 欄位值重設為 NULL。

解說: 當試圖停用直欄時，XML Extender 無法將應用程式表格中的 DXXROOT_ID 值設定爲 NULL。

使用者回應: 請確定 XML Extender 的管理使用者 ID 具有變更應用程式表格的必要專用權。

DXXA032E DB2XML.XML_USAGE 表格中的 USAGE_COUNT 減量失敗。

解說: 當試圖停用直欄時，XML Extender 無法將 USAGE_COUNT 直欄值減 1。

使用者回應: 請確定 DB2XML.XML_USAGE 表格存在，並且 XML Extender 的管理使用者 ID 具有更新表格的必要專用權。

DXXA033E 試圖從 DB2XML.XML_USAGE 表格中刪除橫列失敗。

解說: 當試圖停用直欄時，XML Extender 無法在 DB2XML.XML_USAGE 表格中刪除其相關的橫列。

使用者回應: 請確定 DB2XML.XML_USAGE 表格

存在，並且 XML Extender 的管理使用者 ID 具有更新此表格的必要專用權。

DXXA034I XML Extender 已順利停用直欄 <*column_name*>。

解說: 這是參考訊息

使用者回應: 不需要任何動作。

DXXA035I XML Extender 正停用資料庫 <*database*>。請稍候。

解說: 這是參考訊息。

使用者回應: 不需要任何動作。

DXXA036I XML Extender 已順利停用資料庫 <*database*>。

解說: 這是參考訊息。

使用者回應: 不需要任何動作。

DXXA037E 指定的表格空間名稱超過 18 個字元。

解說: 表格空間名稱不能超過 18 個英數字元。

使用者回應: 請指定少於 18 個字元的名稱。

DXXA038E 指定的預設概略表名稱超過 18 個字元。

解說: 預設概略表名稱不能超過 18 個英數字元。

使用者回應: 請指定少於 18 個字元的名稱。

DXXA039E 指定的 ROOT_ID 名稱超過 18 個字元。

解說: ROOT_ID 名稱不能超過 18 個英數字元。

使用者回應: 請指定少於 18 個字元的名稱。

DXXA046E 無法建立輔助表格 <*side_table*>。

解說: 當試圖啓用直欄時，XML Extender 無法建立指定的輔助表格。

使用者回應: 請確定 XML Extender 的管理使用者 ID 具有建立輔助表格的必要專用權。

DXXA047E 無法啓用直欄。

解說: 由於內部觸發函式失敗，導致 XML Extender 無法啓用直欄。可能的原因：

使用者回應: 使用追蹤機能來建立追蹤檔，並嘗試更正問題。如是這個問題還是發生，請洽您的「軟體服務提供者」並提供追蹤檔。

DXXA048E 無法啓用直欄。

解說: 由於內部觸發函式失敗，導致 XML Extender 無法啓用直欄。可能的原因：

使用者回應: 使用追蹤機能來建立追蹤檔，並嘗試更正問題。如是這個問題還是發生，請洽您的「軟體服務提供者」並提供追蹤檔。

DXXA049E 無法啓用直欄。

解說: 由於內部觸發函式失敗，導致 XML Extender 無法啓用直欄。可能的原因：

使用者回應: 使用追蹤機能來建立追蹤檔，並嘗試更正問題。如是這個問題還是發生，請洽您的「軟體服務提供者」並提供追蹤檔。

DXXA050E 無法啓用直欄。

解說: 由於內部觸發函式失敗，導致 XML Extender 無法啓用直欄。可能的原因：

使用者回應: 使用追蹤機能來建立追蹤檔，並嘗試更正問題。如是這個問題還是發生，請洽您的「軟體服務提供者」並提供追蹤檔。

DXXA051E 無法停用直欄。

解說: 由於內部觸發函式失敗，導致 XML Extender 無法停用直欄。可能的原因：

使用者回應: 使用追蹤機能來建立追蹤檔，並嘗試更正問題。如是這個問題還是發生，請洽您的「軟體服務提供者」並提供追蹤檔。

DXXA052E 無法停用直欄。

解說: 由於內部觸發函式失敗，導致 XML Extender 無法停用直欄。可能的原因：

使用者回應: 使用追蹤機能來建立追蹤檔，並嘗試更正問題。如是這個問題還是發生，請洽您的「軟體服務提供者」並提供追蹤檔。

DXXA053E 無法啓用直欄。

解說: 由於內部觸發函式失敗，導致 XML Extender 無法啓用直欄。可能的原因：

使用者回應: 使用追蹤機能來建立追蹤檔，並嘗試更正問題。如是這個問題還是發生，請洽您的「軟體服務提供者」並提供追蹤檔。

DXXA054E 無法啓用直欄。

解說: 由於內部觸發函式失敗，導致 XML Extender 無法啓用直欄。可能的原因：

使用者回應: 使用追蹤機能來建立追蹤檔，並嘗試更正問題。如是這個問題還是發生，請洽您的「軟體服務提供者」並提供追蹤檔。

DXXA056E DAD 檔中的驗證值 <*validation_value*> 無效。

解說: DAD 檔中的驗證元素錯誤或遺失。

使用者回應: 請確定 DAD 檔中已正確指定了驗證元素。

DXXA057E DAD 中的輔助表格名稱 <*side_table_name*> 無效。

解說: DAD 檔中輔助表格的名稱屬性錯誤或遺失。

使用者回應: 請確定 DAD 檔中已正確指定了輔助表格的名稱屬性。

DXXA058E DAD 檔中的直欄名稱 <*column_name*> 無效。

解說: DAD 檔中直欄的名稱屬性錯誤或遺失。

使用者回應: 請確定 DAD 檔中已正確指定了直欄的名稱屬性。

DXXA059E DAD 檔中直欄 <*column_name*> 的類型 <*column_type*> 無效。

解說: DAD 檔中直欄的類型屬性錯誤或遺失。

使用者回應: 請確定 DAD 檔中已正確指定了直欄的類型屬性。

DXXA060E 在 DAD 檔案中，<*column_name*> 的路徑屬性 <*location_path*> 無效。

解說: DAD 檔中直欄的路徑屬性錯誤或遺漏。

使用者回應: 請確定 DAD 檔案中已正確指定直欄的路徑屬性。

DXXA061E DAD 檔中 <*column_name*> 的 multi_occurrence 屬性 <*multi_occurrence*> 無效。

解說: DAD 檔中直欄的 multi_occurrence 屬性錯誤或遺失。

使用者回應: 請確定 DAD 檔中已正確指定了直欄的 multi_occurrence 屬性。

DXXA062E 無法擷取表格 <*table_name*> 中 <*column_name*> 的直欄號碼。

解說: XML Extender 無法從系統型錄擷取表格 *table_name* 中 *column_name* 的直欄號碼。

使用者回應: 請確定應用程式表格已定義妥當。

DXXA063I 啓用集合 <*collection_name*>。請稍候。

解說: 這是參考訊息。

使用者回應: 不需要任何動作。

DXXA064I 停用集合 <*collection_name*>。請稍候。

解說: 這是參考訊息。

使用者回應: 不需要任何動作。

DXXA065E 呼叫儲存程序 <*procedure_name*> 失敗。

解說: 請檢查共用程式庫 db2xml，看看許可權是否正確。

使用者回應: 請確定從屬站有權執行儲存程序。

DXXA066I XML Extender 已順利停用集合 <*collection_name*>。

解說: 這是參考訊息。

使用者回應: 不需要回應。

DXXA067I XML Extender 已順利啓用集合 <*collection_name*>。

解說: 這是參考訊息。

使用者回應: 不需要回應。

DXXA068I XML Extender 已順利開啓追蹤功能。

解說: 這是參考訊息。

使用者回應: 不需要回應。

DXXA069I XML Extender 已順利關閉追蹤功能。

解說: 這是參考訊息。

使用者回應: 不需要回應。

DXXA070W 已啓用資料庫。

解說: 已對啓用的資料庫執行啓用資料庫指令

使用者回應: 不需要任何動作。

DXXA071W 已停用資料庫。

解說: 已對停用的資料庫執行停用資料庫指令

使用者回應: 不需要任何動作。

DXXA072E XML Extender 找不到連結檔案。啓用資料庫之前必須先連結資料庫。

解說: XML Extender 試圖在啓用資料庫之前自動連結它。

使用者回應: 啓用資料庫之前必須先連結資料庫。

DXXA073E 尚未連結資料庫。啓用資料庫之前，請先連結資料庫。

解說: 當使用者嘗試啓用它時，資料庫尚未連結。

使用者回應: 啓用資料庫之前必須先連結資料庫。

DXXA074E 錯誤參數類型。儲存程序預期一個 STRING 參數。

解說: 儲存程序預期一個 STRING 參數。

使用者回應: 將輸入參數宣告爲 STRING 類型。

DXXA075E 錯誤參數類型。輸入參數應該為 LONG 類型。

解說: 儲存程序預期輸入參數爲 LONG 類型。

使用者回應: 將輸入參數宣告爲 LONG 類型。

DXXA076E　XML Extender 無法追蹤案例 ID。

解說: 無法啓動案例 ID 所提供的追蹤。

使用者回應: 請確定案例 ID 爲有效的 AS/400 使用者 ID。

DXXC000E　無法開啓指定的檔案。

解說: XML Extender 無法開啓指定的檔案。

使用者回應: 請確定應用程式使用者 ID 具有檔案的讀寫權。

DXXC001E　找不到指定的檔案。

解說: XML Extender 找不到指定的檔案。

使用者回應: 請確定檔案已存在，且正確指定路徑。

DXXC002E　無法讀取檔案。

解說: XML Extender 無法從指定的檔案讀取資料。

使用者回應: 請確定應用程式使用者 ID 有檔案的讀取許可權。

DXXC003E　無法寫入指定的檔案。

解說: XML Extender 無法將資料寫入檔案。

使用者回應: 請確定應用程式使用者 ID 具有檔案的寫入權，或檔案系統具有足夠的空間。

DXXC004E　無法操作「LOB 定位器」：rc=<*locator_rc*>。

解說: XML Extender 無法操作指定的定位器。

使用者回應: 請確定「LOB 定位器」的設定正確。

DXXC005E　輸入檔大小大於 XMLVarchar 大小。

解說: 檔案大小大於 XMLVarchar 大小，且 XML Extender 無法從該檔案匯入全部資料。

使用者回應: 請使用 XMLCLOB 直欄類型。

DXXC006E　輸入檔超出 DB2 LOB 限制。

解說: 檔案大小大於 XMLCLOB 大小，且 XML Extender 無法從該檔案匯入全部資料。

使用者回應: 將檔案分解成較小物件，或使用 XML 集合。

DXXC007E　無法將資料從檔案擷取到「LOB 定位器」。

解說: 「LOB 定位器」中的位元組數不等於檔案大小。

使用者回應: 請確定「LOB 定位器」的設定正確。

DXXC008E　無法移除檔案 <*file_name*>。

解說: 檔案發生共用存取違規，或仍然開啓中。

使用者回應: 請關閉檔案，或停止任何保留該檔案的程序。您可能必須停止和重新啓動 DB2。

DXXC009E　無法建立檔案至 <*directory*> **目錄。**

解說: XML Extender 無法在目錄 *directory* 中建立檔案。

使用者回應: 請確定目錄已存在、應用程式使用者 ID 具有該目錄的寫入許可權、以及檔案系統有足夠的空間來存放檔案。

DXXC010E　寫入檔案 <*file_name*> **發生錯誤。**

解說: 寫入檔案 *file_name* 時發生錯誤。

使用者回應: 請確定檔案系統有足夠的空間來存放檔案。

DXXC011E　無法寫入追蹤控制檔。

解說: XML Extender 無法將資料寫入追蹤控制項檔案。

使用者回應: 請確定應用程式使用者 ID 具有檔案的寫入權，或檔案系統具有足夠的空間。

DXXC012E 無法建立暫存檔。

解說: 無法在系統 temp 目錄中建立檔案。

使用者回應: 請確定應用程式使用者 ID 對於檔案系統 temp 目錄具有寫入許可權，或檔案系統有足夠的空間來存放檔案。

DXXD000E 拒絕無效的 XML 文件。

解說: 試圖將無效的文件儲存到表格中。驗證失敗。

使用者回應: 請使用可檢視隱藏及無效字元的編輯器來檢查文件及其 DTD。要抑制這個錯誤發生，請關閉 DAD 檔的驗證。

DXXD001E <*location_path*> **出現多次。**

解說: 純量解壓縮函數使用多次出現的位置路徑。純量函數僅可使用沒有多次出現項目的位置路徑。

使用者回應: 使用表格函數 (在純量函數名稱的尾端加上一個 's')。

DXXD002E 在搜尋路徑的位置 <*position*> **附近發生語法錯誤。**

解說: 路徑表示式的語法不正確。

使用者回應: 請更正查詢的搜尋路徑引數。請參考文件來取得路徑表示式的語法。

DXXD003W 找不到路徑。傳回 NULL。

解說: XML 文件遺失路徑表示式中指定的屬性或元素。

使用者回應: 請驗證指定的路徑是否正確。

DXXG000E 檔名 <*file_name*> **無效。**

解說: 指定的檔名無效。

使用者回應: 請指定一個正確的檔名，然後重試。

DXXG001E 在建置版 <*build_ID*> **檔案** <*file_name*> **的第** <*line_number*> **行發生內部錯誤。**

解說: XML Extender 發現內部錯誤。

使用者回應: 請洽您的「軟體服務提供者」。報告錯誤時，請務必附上所有訊息、追蹤檔以及如何使錯誤重現的說明。

DXXG002E 系統的記憶體不足。

解說: XML Extender 無法從作業系統配置記憶體。

使用者回應: 請關閉部份應用程式，然後重試。如果問題仍然存在，請參考作業系統文件來取得協助。某些作業系統會要求您重新啓動系統來更正問題。

DXXG004E 無效 NULL 參數。

解說: 某必要參數的 NULL 值傳至 XML 儲存程序。

使用者回應: 請檢查引數列示中用於儲存程序呼叫的所有必要參數。

DXXG005E 不支援參數。

解說: 這個版次不支援此參數，未來版次會支援此參數。

使用者回應: 請將這個參數設定爲 NULL。

DXXG006E 內部錯誤 CLISTATE=<*clistate*>**, RC=**<*cli_rc*>**, 建置** <*build_ID*>**, 檔案** <*file_name*>**, 行號** <*line_number*> **CLIMSG=**<*CLI_msg*>。

解說: 使用 CLI 時，XML Extender 發現內部錯誤。

使用者回應: 請洽您的「軟體服務提供者」。這個錯誤可能是由於使用者輸入錯誤所造成。報告錯誤時，請務必附上所有輸出訊息、追蹤日誌以及如何重現問題的說明。可能的話，請一併提供

DAD、XML 文件及表格定義。

DXXG007E 語言環境 <*locale*> 與 DB2 字碼頁 <*code_page*> 不一致。

解說: 伺服器作業系統語言環境與 DB2 字碼頁不一致。

使用者回應: 請更正伺服器作業系統語言環境，並重新啓動 DB2。

DXXG008E 不支援的語言環境 <*locale*>。

解說: 在字碼頁表格中，找不到伺服器作業系統語言環境。

使用者回應: 請更正伺服器作業系統語言環境，並重新啓動 DB2。

DXXQ000E DAD 檔中遺失 <*Element*>。

解說: DAD 檔中遺失必要的元件。

使用者回應: 請將遺失的元件加入 DAD 檔中。

DXXQ001E XML 產生的無效 SQL 陳述式。

解說: DAD 中的 SQL 陳述式或置換它的 SQL 陳述式無效。建立 XML 文件需要 SELECT 陳述式。

使用者回應: 請更正 SQL 陳述式。

DXXQ002E 無法建立儲存體空間來保存 XML 文件。

解說: 系統記憶體或磁碟空間不足。沒有足夠的空間來保存產生的 XML 文件。

使用者回應: 請限制要建立的文件數目。藉由從 DAD 檔中移除某些不必要的元件及屬性節點，以縮減每一個文件的大小。

DXXQ003W 結果超出最大值。

解說: 使用者定義的 SQL 查詢所產生之 XML 文件數超過指定的最大值。僅傳回指定的文件數目。

使用者回應: 不需要任何動作。 如果所有文件皆爲必要，請指定零作爲最大文件數目。

DXXQ004E 直欄 <*column_name*> 不在查詢的結果中。

解說: 指定的直欄不是 SQL 查詢結果的其中一個直欄。

使用者回應: 請變更 DAD 檔中指定的直欄名稱，使其成爲 SQL 查詢結果的其中一個直欄。另外，變更 SQL 查詢，讓查詢結果中有指定的直欄。

DXXQ004W DAD 中找不到 DTD ID。

解說: 在 DAD 中，VALIDATION 是 YES，但 DTDID 元素尚未指定。不執行驗證檢查。

使用者回應: 不需要任何動作。 如果需要驗證，請在 DAD 檔案中指定 DTDID 元素。

DXXQ005E 錯誤關聯式對映。元素 <*element_name*> 位於比它的子項直欄 <*column_name*> 更低的層次上。

解說: SQL 查詢與 XML 的對映不正確。

使用者回應: 請確定 SQL 查詢結果中的直欄在關聯式階層中是採由上往下的次序。並請確定有單一直欄候選鍵來起始每一個層次。如果表格中沒有這樣的鍵值，查詢應使用表格表示式及 DB2內建函數 generate_unique() 來對該表格產生一個鍵值。

DXXQ006E attribute_node 元素未命名。

解說: DAD 檔中 attribute_node 元素沒有名稱屬性。

使用者回應: 請確定 DAD 檔中每一個 attribute_node 都有一個名稱。

DXXQ007E attribute_node <*attribute_name*> 沒有直欄元素或 RDB_node。

解說: DAD 檔中 attribute_node 元素沒有直欄元素或 RDB_node。

使用者回應: 請確定 DAD 中每一個 attribute_node

都有一個直欄元素或 RDB_node。

DXXQ008E text_node 元素沒有直欄元素。

解說: DAD 檔中 text_node 元素沒有直欄元素。

使用者回應: 請確定 DAD 中每一個 text_node 都有一個直欄元素。

DXXQ009E 結果表格 <*table_name*> 不存在。

解說: 系統型錄中找不到指定的結果表格。

使用者回應: 請在呼叫儲存程序前建立結果表格。

DXXQ010E DAD 檔中，<*node_name*> 的 RDB_node 沒有表格。

解說: attribute_node 或 text_node 的 RDB_node 必須有一個表格。

使用者回應: 請在 DAD 檔中爲 attribute_node 或 text_node 指定 RDB_node 的表格。

DXXQ011E DAD 檔中，<*node_name*> 的 RDB_node 元素沒有直欄。

解說: attribute_node 或 text_node 的 RDB_node 必須有一個直欄。

使用者回應: 請在 DAD 檔中爲 attribute_node 或 text_node 指定 RDB_node 的直欄。

DXXQ012E DAD 發生錯誤。

解說: 處理程序 DAD 時，XML Extender 找不到期望的元素。

使用者回應: 檢查 DAD 是否爲有效的 XML 文件，是否含有 DAD DTD 所需的全部元素。請參考 XML Extender 出版文件中有關 DAD DTD 的部份。

DXXQ013E DAD 檔中表格或直欄元素沒有名稱。

解說: DAD 檔中元素表格或直欄都必須有名稱。

使用者回應: 請指定 DAD 檔中表格或直欄元素的名稱。

DXXQ014E element_node 元素沒有名稱。

解說: DAD 檔中 element_node 元素沒有名稱屬性。

使用者回應: 請確定 DAD 檔中每一個 element_node 元素都有一個名稱。

DXXQ015E 無效的條件格式。

解說: 文件存取定義 (DAD) 檔中的條件元素之條件格式無效。

使用者回應: 請確定條件格式是有效的。

DXXQ016E RDB_node 中的表格名稱未定義於 DAD 檔的頂端元素中。

解說: 所有表格皆須定義於 DAD 檔中頂端元素的 RDB_node 中。次元素表格必須符合頂端元素中定義的表格。此 RDB_node 中的表格名稱不在頂端元素中。

使用者回應: 請確定 RDB 節點的表格是定義於 DAD 檔的頂端元素中。

DXXQ017E 結果表格 <*table_name*> 中的直欄太小。

解說: XML Extender 建立的 XML 文件太大，無法填入結果表格的直欄中。

使用者回應: 請捨棄結果表格。建立另一個含有較大直欄的結果表格。重新執行儲存程序。

DXXQ018E SQL 陳述式中遺失 ORDER BY 子句。

解說: 在對映 SQL 至 XML 的 DAD 檔中，SQL 陳述式內遺失 ORDER BY 子句。

使用者回應: 請編輯 DAD 檔。新增含有實體識別直欄的 ORDER BY 子句。

DXXQ019E 在 DAD 檔中，元素 objids 沒有直欄元素。

解說: 在 DAD 檔中 objids 元素沒有對映 SQL 至 XML 的直欄元素。

使用者回應: 請編輯 DAD 檔。新增鍵值直欄以作爲元素 objids 的次元素。

DXXQ020I XML 已順利建立。

解說: 所要求的 XML 文件已從資料庫順利建立。

使用者回應: 不需要任何動作。

DXXQ021E 表格 <*table_name*> 不含直欄 <*column_name*>。

解說: 資料庫中，表格沒有指定的直欄。

使用者回應: 請在 DAD 中指定另一個直欄名稱，或新增指定的直欄到表格資料庫中。

DXXQ022E <*table_name*> 的直欄 <*column_name*> 應該具有 <*type_name*> 類型。

解說: 直欄的類型錯誤。

使用者回應: 請更正文件存取定義 (DAD) 中直欄的類型。

DXXQ023E <*table_name*> 的直欄 <*column_name*> 長度不可超過 <*length*>。

解說: DAD 中定義的直欄長度太長。

使用者回應: 請在文件存取定義 (DAD) 中更正直欄長度。

DXXQ024E 無法建立表格 <*table_name*>。

解說: 無法建立指定的表格。

使用者回應: 請確定建立表格的使用者 ID 具有在資料庫中建立表格的必要權限。

DXXQ025I XML 已順利分解。

解說: XML 文件已順利分解並儲存在集合中。

使用者回應: 不需要任何動作。

DXXQ026E XML 資料 <*xml_name*> 太大，無法納入直欄 <*column_name*> 中。

解說: XML 文件中指定的資料片段太大，無法填入指定的直欄中。

使用者回應: 請使用 ALTER TABLE 陳述式來增加直欄的長度，或編輯 XML 文件來減少資料的大小。

DXXQ028E 在 XML_USAGE 表格中找不到集合 <*collection_name*>。

解說: XML_USAGE 表格中找不到集合的記錄。

使用者回應: 請驗證您是否已啓用集合。

DXXQ029E 對於集合 <*collection_name*>，在 XML_USAGE 表格中找不到 DAD。

解說: XML_USAGE 表格中找不到集合的 DAD 記錄。

使用者回應: 請確定您已正確地啓用集合。

DXXQ030E 錯誤的 XML 置換語法。

解說: 儲存程序中的 XML_override 值指定不正確。

使用者回應: 請確定 XML_override 的語法正確。

DXXQ031E 表格名稱不可超過 DB2 所容許的最大長度。

解說: DAD 中的條件元素所指定的表格名稱太長。

使用者回應: 請在文件存取定義 (DAD) 中更正表格名稱的長度。

DXXQ032E 直欄名稱不可超過 DB2 所容許的最大長度。

解說: DAD 中的條件元素所指定的直欄名稱太長。

使用者回應: 請在文件存取定義 (DAD) 中更正直欄名稱的長度。

DXXQ033E <*identifier*> 的開頭出現無效的識別字

解說: 該字串不是有效的 DB2 SQL 識別字。

使用者回應: 請在 DAD 中更正字串，以符合 DB2 SQL 識別字的規則。

DXXQ034E 位於 DAD 頂端 RDB_node 的無效條件元素：<*condition*>

解說: 條件元素必須是有效的 WHERE 子句，其中含有連結 AND 所連接的結合條件。

使用者回應: 請參閱 XML Extender 文件，以更正 DAD 中結合條件的語法。

DXXQ035E 位於 DAD 頂端 RDB_node 的無效結合條件：<*condition*>

解說: 如果 DAD 指定多重表格，頂端 RDB_node 的條件元素中的直欄名稱，必須符合表格名稱的規定。

使用者回應: 請參閱 XML Extender 文件，以更正 DAD 中結合條件的語法。

DXXQ036E 在 DAD 條件標示下指定的綱目名稱大於容許的長度。

解說: 在 DAD 的條件標示下剖析文字時，偵測到一個錯誤。此條件文字包含一個由過長的綱目名稱所定義的 ID。

使用者回應: 更正 DAD 中的條件標示文字。

DXXQ037E 無法建立 <*element*> 與多次出現的項目。

解說: 元素節點及其敘述未對映到資料庫中，但是其 multi_occurrence 值等於 YES。

使用者回應: 請將 multi_occurrence 設定爲 NO，或是在其中一個敘述中建立 RDB_node，以更正 DAD。

診斷追蹤

XML Extender 提供一個追蹤機能，可記錄 XML Extender 伺服器活動。您應該依照 IBM 軟體支援中心的指示，使用追蹤機能。

追蹤機能在一個伺服器檔案中記錄各種事件的資訊，例如進入或離開一個 XML Extender 元件，或 XML Extender 元件傳回錯誤碼。因爲記錄許多事件的資訊，所以應該在必要時才使用追蹤機能，例如，需要調查錯誤狀況時。此外，使用追蹤機能時，應該限制作用中的應用程式數目。限制作用中應用程式數目，較易於找出問題的原因。

使用 **dxxtrc** 指令來控制追蹤。您可在 AIX、Windows NT 或 Solaris 伺服器的指令行上發出此指令。您必須具備 SYSADM、SYSCTRL 或 SYSMINT 權限才能發出此指令。

使用 **dxxtrc** 指令來完成下列動作：

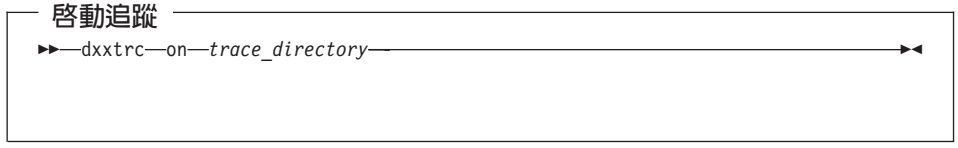
- 啓動追蹤
- 停止追蹤
- 重新製作追蹤資訊的格式，使它易於閱讀
- 顯示追蹤狀態

啓動追蹤

目的

記錄 XML Extender 伺服器活動。若要啓動追蹤，請在 **dxxtrc** 上使用 on 選項，加上含有追蹤檔的目錄名稱。 當追蹤開啓時，檔案dxx*INSTANCE*.trc 會被置放在指定的目錄。*INSTANCE*爲 DB2INSTANCE 的值。每一個 DB2 案例都有它自己的日誌檔。

語法



參數

表 53. 追蹤參數

參數	說明
trace_directory	dxx*INSTANCE*.trc 所在位置的目錄名稱。因爲追蹤是以 DB2 案例爲基礎，所以目錄應該放在 DB2 案例的 SQLLIB 之下。

範例

本範例顯示在 AIX 系統上使用資料庫 SALES_DB 時，開啓追蹤機能。 由於 DB2INSTANCE 爲 db2inst1，追蹤檔 dxxdb2inst1.trc 會被放置在/home/db2inst1/sqllib/log 目錄中。

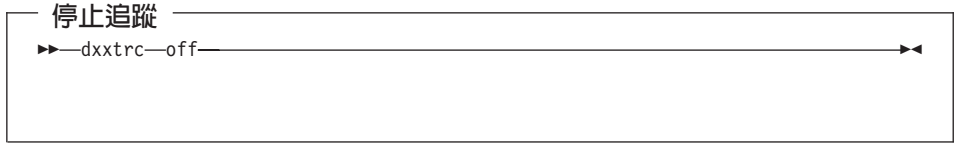
```
dxxtrc on /home/db2inst1/sqllib/log
```

停止追蹤

目的

關閉追蹤。不記載追蹤資訊。因爲執行追蹤會影響效能，所以建議在生產環境下關閉追蹤。

語法



範例

本範例顯示關閉追蹤機能。

```
dxxtrc off
```

第5篇 附錄與後記

附錄A. DAD 檔案的 DTD

本節說明文件存取定義 (DAD) 檔案的文件類型宣告 (DTD)。 DAD 檔案本身是一個樹狀結構的 XML 文件且需要 DTD。DTD 檔名是 `dxxdad.dtd`。 第226頁的圖13顯示 DAD 檔案的 DTD。 圖後面提供此檔案的元素說明。

```
<?xml encoding="US-ASCII"?>

  <!ELEMENT DAD (dtdid?, validation, (Xcolumn | Xcollection))>
  <!ELEMENT dtdid (#PCDATA)>
  <!ELEMENT validation (#PCDATA)>
  <!ELEMENT Xcolumn (table*)>
  <!ELEMENT table (column*)>
 <!ATTLIST table name CDATA #REQUIRED
                              key CDATA #IMPLIED
                              orderBy CDATA #IMPLIED>
  <!ELEMENT column EMPTY>
<!ATTLIST column
                     name CDATA #REQUIRED
                     type  CDATA #IMPLIED
                     path  CDATA #IMPLIED
                     multi_occurrence CDATA #IMPLIED>
  <!ELEMENT Xcollection (SQL_stmt?, objids?, prolog, doctype, root_node)>
  <!ELEMENT SQL_stmt (#PCDATA)>
  <!ELEMENT objids (column+)>
  <!ELEMENT prolog (#PCDATA)>
  <!ELEMENT doctype (#PCDATA | RDB_node)*>
  <!ELEMENT root_node (element_node)>
  <!ELEMENT element_node (RDB_node*,
                                       attribute_node*,
                                       text_node?,
                                       element_node*,
                                       namespace_node*,
                                       process_instruction_node*,
                                       comment_node*)>
  <!ATTLIST element_node
                     name CDATA #REQUIRED
                      ID CDATA #IMPLIED
                     multi_occurrence CDATA "NO"
                     BASE_URI CDATA #IMPLIED>
  <!ELEMENT attribute_node (column | RDB_node)>
  <!ATTLIST attribute_node
                     name CDATA #REQUIRED>
  <!ELEMENT text_node (column | RDB_node)>
  <!ELEMENT RDB_node (table+, column?, condition?)>
  <!ELEMENT condition (#PCDATA)>
  <!ELEMENT comment_node (#PCDATA)>
  <!ELEMENT namespace_node (EMPTY)>
  <!ATTLIST namespace_node
                      name CDATA #IMPLIED
                     value CDATA #IMPLIED>
  <!ELEMENT process_instruction_node (#PCDATA)>
```

圖 13. 文件存取定義 (DAD) 的 DTD

DAD 檔案具有四個主要元素：

- DTDID
- validation

- Xcolumn
- Xcollection

Xcolumn 和 Xcollection 包含子項元素和屬性，協助將 XML 資料對映至 DB2 的關聯式表格。下面列示說明主要元素及其子項元素和屬性。語法範例取自第226頁的圖13。

DTDID 元素

指定 DTD_REF 表格中儲存的 DTD 之 ID。DTDID 指向 DTD，用以驗證 XML 文件或指示 XML 集合表格和 XML 文件之間的對映。必須爲 XML 集合指定 DTDID。唯有當您想建立輔助表格來檢索元素或屬性或驗證輸入 XML 文件時，才需要 XML 直欄。 DTDID 必須與 XML 文件之 doctype 中指定的 SYSTEM ID 相同。

語法： `<!ELEMENT dtdid (#PCDATA)>`

validation 元素

指出是否以 DAD 的 DTA 來驗證 XML 文件。若指定 YES，則亦必須指定 DTDID。

語法： `<!ELEMENT validation(#PCDATA)>`

Xcolumn 元素

定義 XML 直欄的索引方法。由零個或多個表格組成。

語法： `<!ELEMENT Xcolumn (table*)>`Xcolumn 具有一個子項元素 table。

table 元素

定義一或多個關聯式表格，用以索引 XML 直欄中儲存之文件的元素或屬性。

語法：

```
<!ELEMENT table (column+)>
 <!ATTLIST table name CDATA #REQUIRED
key CDATA #IMPLIED
orderBy CDATA #IMPLIED>
```

table 元素具有一個屬性：

name 屬性

指定輔助表格的名稱

table 元素具有一個子項元素：

key 屬性

表格的主要單一鍵

orderBy 屬性

直欄的名稱，在產生 XML 文件時，用來決定重複元素文字或屬性值的順序。

column 元素

指定表格的直欄，直欄含有指定類型的位置路徑值。

語法：

```
<!ATTLIST column
                    name CDATA #REQUIRED
                    type  CDATA #IMPLIED
                    path  CDATA #IMPLIED
                    multi_occurrence CDATA #IMPLIED>
```

column 元素具有下列屬性：

name 屬性

指定直欄的名稱。位置路徑的別名，用來定義元素或屬性

type 屬性

定義直欄的資料類型。可以是任何 SQL 資料類型。

path 屬性

顯示 XML 元素或屬性的位置路徑，必須是表格 3.1.a (修正鏈結) 中指定的簡式位置路徑。

multi_occurrence 屬性

指定此元素或屬性是否可在一個 XML 文件中出現多次。值包括 YES 或 NO。

Xcollection

定義 XML 文件和 XML 關聯式表格集合之間的對映。

語法： <!ELEMENT Xcollection(SQL_stmt*, prolog, doctype, root_node)>Xcollection 具有下列子項元素：

SQL_stmt

指定 XML Extender 用來定義集合的 SQL 陳述式。特別說明，此陳述式從 XML 集合表格中選取 XML 資料，使用資料來建立集合中的 XML 文件。 此元素的值必須是有效的 SQL 陳述式。僅用於組合，且只容許一個 SQL_stmt。 對於分解，SQL_stmt 可指定一個以上的值來執行必需的表格建立和插入動作。

語法： <!ELEMENT SQL_stmt #PCDATA >

prolog

XML 前言的文字。相同的前言使用於整個集合的全部文件。prolog 的值是固定的。

語法： `<!ELEMENT prolog #PCDATA>`

doctype

定義 XML 文件類型定義的文字。

語法： `<!ELEMENT doctype #PCDATA | RDB_node>`doctype 可透過下列其中一種方式來指定：

- 定義一個明確值。此值使用於整個集合的全部文件。
- 使用分解時，請指定子項元素 RDB_node，可對映及儲存成表格的直欄資料。

doctype 具有一個子項元素：

RDB_node

尚未提供。

root_node

定義虛擬 root 節點。root_node 必須有一個只能使用一次的必要子項元素 element_node。 root_node 下的 element_node，實際上就是 XML 文件的 root_node。

語法： `<!ELEMENT root_node(element_node)>`

element_node

代表一個 XML 元素。定義於集合所指定的 DTD 中。對於 RDB_node 對映，root element_node 必須有一個 RDB_node，爲本身及其所有子節點指定含有 XML 資料的全部表格。可包含零個或多個 attribute_nodes 和子項 element_nodes，以及零個或一個 text_node。對於 root 元素以外的元素，不需要 RDB_node。

語法：

element_node 由下列子項元素所定義：

RDB_node

(可選用的) 指定 XML 資料的表格、直欄及條件。只需要爲 RDB_node 對映來定義元素的 RDB_node。在此情況下，必須指定一或多個表格。因爲元素內容由 text_node 所指定，所以不需要直欄。條件是可選用的，視 DTD 和查詢條件而定。

子節點 (可選用的) element_node 亦可包括下列子節點：

element_node

代表現行 XML 元素的子項元素

attribute_node
代表現行 XML 元素的屬性

text_node
代表現行 XML 元素的 CDATA 文字

attribute_node
代表 XML 屬性。定義 XML 屬性和關聯式表格中直欄資料之間的對映。

語法：

attribute_node 必須有 name 屬性的定義，以及 column 或 RDB_node 子項元素的定義。 attribute_node 具有下列屬性：

name 屬性名稱。

attribute_node 具有下列子項元素：

Column
使用於 SQL 對映。此直欄必須指定於 SQL_stmt 的 SELECT 子句。

RDB_node
使用於 RDB_node 對映。此節點定義該屬性和關聯式表格中直欄資料之間的對映。必須指定表格和直欄。條件是可選用的。

text_node
代表 XML 元素的文字內容。定義 XML 元素內容和關聯式表格中直欄資料之間的對映。

語法： 必須由 column 或 RDB_node 子項元素來定義：

Column
SQL 對映所需的。在此情況下，直欄必須指定於 SQL_stmt 的 SELECT 子句中。

RDB_node
RDB_node 對映所需的。此節點定義該文字內容和關聯式表格中直欄資料之間的對映。必須指定 table 和 column。條件是可選用的。

附錄B. 範例

本附錄顯示與本書範例一起使用的範例物件。

- 『XML DTD』
- 『XML 文件： getstart.xml』
- 第232頁的『文件存取定義檔』
  - 第233頁的『DAD 檔：XML 直欄』
  - 第233頁的『DAD 檔：XML 集合 - SQL 對映』
  - 第235頁的『DAD 檔：XML - RDB_node 對映』

XML DTD

下列 DTD 用於本書全文所參考的 getstart.xml 文件，並顯示於 第232頁的圖15中。

```
<!xml encoding="US-ASCII"?>

<!ELEMENT Order (Customer, Part+)>
<!ATTLIST Order key CDATA #REQUIRED>
<!ELEMENT Customer (Name, Email)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Email (#PCDATA)>
<!ELEMENT Part (key,Quantity,ExtendedPrice,Tax, Shipment+)>
<!ELEMENT key (#PCDATA)>
<!ELEMENT Quantity (#PCDATA)>
<!ELEMENT ExtendedPrice (#PCDATA)>
<!ELEMENT Tax (#PCDATA)>
<!ATTLIST Part color CDATA #REQUIRED>
<!ELEMENT Shipment (ShipDate, ShipMode)>
<!ELEMENT ShipDate (#PCDATA)>
<!ELEMENT ShipMode (#PCDATA)>
```

圖 14. 範例 XML DTD： getstart.dtd

XML 文件： getstart.xml

下列 XML 文件 getstart.xml 爲本書全部範例所用的範例 XML 文件。它包含構成採購單的 XML 標示。

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd">
    <Order key="1">
  <Customer>
    <Name>American Motors</Name>
    <Email>parts@am.com</Email>
  </Customer>
  <Part color="black ">
    <key>68</key>
    <Quantity>36</Quantity>
    <ExtendedPrice>34850.16</ExtendedPrice>
    <Tax>6.000000e-02</Tax>
    <Shipment>
      <ShipDate>1998-08-19</ShipDate>
      <ShipMode>BOAT  </ShipMode>
    </Shipment>
    <Shipment>
      <ShipDate>1998-08-19</ShipDate>
       <ShipMode>AIR   </ShipMode>
    </Shipment>
  </Part>
  <Part color="red   ">
    <key>128</key>
    <Quantity>28</Quantity>
    <ExtendedPrice>38000.00</ExtendedPrice>
    <Tax>7.000000e-02</Tax>
    <Shipment>
      <ShipDate>1998-12-30</ShipDate>
      <ShipMode>TRUCK </ShipMode>
    </Shipment>
  </Part>
</Order>
```

圖 15. 範例 XML 文件： getstart.xml

文件存取定義檔

下列各節包含使用 XML 直欄或 XML 集合存取模式，將 XML 資料對映至 DB2 關聯式表格的文件存取定義 (DAD) 檔。

- 第233頁的『DAD 檔：XML 直欄』
- 第233頁的『DAD 檔：XML 集合 - SQL 對映』 顯示使用 SQL 對映的 XML 集合之 DAD 檔。
- 第235頁的『DAD 檔：XML - RDB_node 對映』 顯示使用 RDB_node 對映的 XML 集合之 DAD。

DAD 檔：XML 直欄

這個 DAD 檔含有定義表格、輔助表格及直欄來包含 XML 資料的 XML 直欄對映。

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "c:\dxx\dad.dtd">
<DAD>
   <dtdid>c:\dxx\samples\dtd\getstart.dtd</dtdid>
   <validation>YES</validation>

   <Xcolumn>
     <table name="order_side_tab">
         <column name="order_key"
          type="integer"
            path="/Order/@key"
          multi_occurrence="NO"/>
  <column name="customer"
          type="varchar(50)"
          path="/Order/Customer/Name"
          multi_occurrence="NO"/>
     </table>
     <table name="part_side_tab">
         <column name="price"
          type="decimal(10,2)"
          path="/Order/Part/ExtendedPrice"
            multi_occurrence="YES"/>
     </table>
     <table name="ship_side_tab">
         <column name="date"
          type="DATE"
          path="/Order/Part/Shipment/ShipDate"
            multi_occurrence="YES"/>
     </table>

   </Xcolumn>

</DAD>
```

圖 16. XML 直欄的範例 DAD 檔

DAD 檔：XML 集合 - SQL 對映

這個 DAD 檔含有指定 DB2 表格、直欄及條件來包含 XML 資料的 SQL 陳述式。

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dxx\dtd\dad.dtd">
<DAD>
<validation>NO</validation>
<Xcollection>
<SQL_stmt>SELECT o.order_key, customer_name, customer_email, p.part_key, color, quantity,
   price, tax, ship_id, date, mode from order_tab o, part_tab p,
                    table(select substr(char(timestamp(generate_unique())),16)
     as ship_id, date, mode, part_key from ship_tab) s
         WHERE o.order_key = 1 and
               p.price > 20000 and
               p.order_key = o.order_key and
               s.part_key = p.part_key
         ORDER BY order_key, part_key, ship_id</SQL_stmt>
      <prolog>?xml version="1.0"?</prolog>
      <doctype>!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
```

圖 17. 使用 SQL 對映的 XML 集合之範例 DAD 檔 (1/2)

```
      <root_node>
<element_node name="Order">
  <attribute_node name="key">
                  <column name="order_key"/>
              </attribute_node>
           <element_node name="Customer">
    <element_node name="Name">
      <text_node><column name="customer_name"/></text_node>
    </element_node>
    <element_node name="Email">
      <text_node><column name="customer_email"/></text_node>
    </element_node>
    </element_node>
  <element_node name="Part">
    <attribute_node name="color">
      <column name="color"/>
    </attribute_node>
    <element_node name="key">
      <text_node><column name="part_key"/></text_node>
    </element_node>
<element_node name="Quantity">
      <text_node><column name="quantity"/></text_node>
    </element_node>
    <element_node name="ExtendedPrice">
      <text_node><column name="price"/></text_node>
    </element_node>
    <element_node name="Tax">
      <text_node><column name="tax"/></text_node>
    </element_node>
    <element_node name="Shipment" multi_occurence="YES">
              <element_node name="ShipDate">
        <text_node><column name="date"/></text_node>
    </element_node>
      <element_node name="ShipMode">
        <text_node><column name="mode"/></text_node>
    </element_node>
    </element_node>
    </element_node>
    </element_node>
</root_node>
  </Xcollection>
</DAD>
```

圖 17. 使用 SQL 對映的 XML 集合之範例 DAD 檔 (2/2)

DAD 檔：XML - RDB_node 對映

這個 DAD 檔使用 <RDB_node> 元素，定義 DB2 表格、直欄及條件來包含 XML 資料。

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "c:\dxx\dtd\dad.dtd">
<DAD>
   <dtdid>c:\dxx\samples\dtd\getstart.dtd</dtdid>
   <validation>YES</validation>
<Xcollection>
      <prolog>?xml version="1.0"?</prolog>
      <doctype>!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
      <root_node>
<element_node name="Order">
        <RDB_node>
                     <table name="order_tab"/>
                     <table name="part_tab"/>
                        <table name="ship_tab"/>
                        <condition>
                order_tab.order_key = part_tab.order_key AND
                     part_tab.part_key = ship_tab.part_key
                        </condition>
                     </RDB_node>
<attribute_node name="key">
        <RDB_node>
                     <table name="order_tab"/>
                  <column name="order_key"/>
                     </RDB_node>
               </attribute_node>
            <element_node name="Customer">
                  <text_node>
        <RDB_node>
                     <table name="order_tab"/>
                     <column name="customer"/>
                     </RDB_node>
                  </text_node>
    </element_node>
  <element_node name="Part">
        <RDB_node>
                     <table name="part_tab"/>
                        <table name="ship_tab"/>
                        <condition>
                     part_tab.part_key = ship_tab.part_key
                        </condition>
                     </RDB_node>
<attribute_node name="key">
        <RDB_node>
                     <table name="part_tab"/>
                     <column name="part_key"/>
                     </RDB_node>
    </attribute_node>
```

圖 18. 使用 RDB_node 對映的 XML 集合之範例 DAD 檔 (1/3)

```
       <element_node name="Quantity">
             <text_node>
    <RDB_node>
                <table name="part_tab"/>
                <column name="quantity"/>
                </RDB_node>
             </text_node>
</element_node>
<element_node name="ExtendedPrice">
             <text_node>
    <RDB_node>
                <table name="part_tab"/>
                   <column name="price"/>
                   <condition>
                      price > 2500.00
                   </condition>
                </RDB_node>
             </text_node>
</element_node>
<element_node name="Tax">
             <text_node>
    <RDB_node>
                <table name="part_tab"/>
                <column name="tax"/>
                </RDB_node>
             </text_node>
</element_node>
```

圖 18. 使用 RDB_node 對映的 XML 集合之範例 DAD 檔 (2/3)

```
            <element_node name="shipment">
          <RDB_node>
                          <table name="ship_tab"/>
                          <condition>
                       part_key = part_tab.part_key
                          </condition>
                       </RDB_node>
                  <element_node name="ShipDate">
                     <text_node>
          <RDB_node>
                          <table name="ship_tab"/>
                          <column name="date"/>
                          <condition>
                             date > "1966-01-01"
                          </condition>
                       </RDB_node>
          </text_node>
      </element_node>
        <element_node name="ShipMode">
                     <text_node>
          <RDB_node>
                          <table name="ship_tab"/>
                          <column name="mode"/>
                       </RDB_node>
          </text_node>
      </element_node>
                <element_node name="Comment">
                   <text_node>
          <RDB_node>
                          <table name="ship_tab"/>
                          <column name="comment"/>
                       </RDB_node>
          </text_node>
      </element_node>
              </element_node> <! -- end of element Shipment>
            </element_node> <! -- end of element Part --->
          </element_node> <! -- end of element Order --->
</root_node>
  </Xcollection>
</DAD>
```

圖 18. 使用 RDB_node 對映的 XML 集合之範例 DAD 檔 (3/3)

附錄C. 字碼頁注意事項

主/從 DB2 資料庫環境提供一些檢核措施，對於存取 XML 文件的每一個從屬站或伺服器，確保 XML 文件有適當地編碼。確定從屬站自 DB2 中擷取 XML 文件的實際字碼頁與編碼宣告相符是非常重要事項 (XML 文件中的編碼宣告)，如圖19中所述。

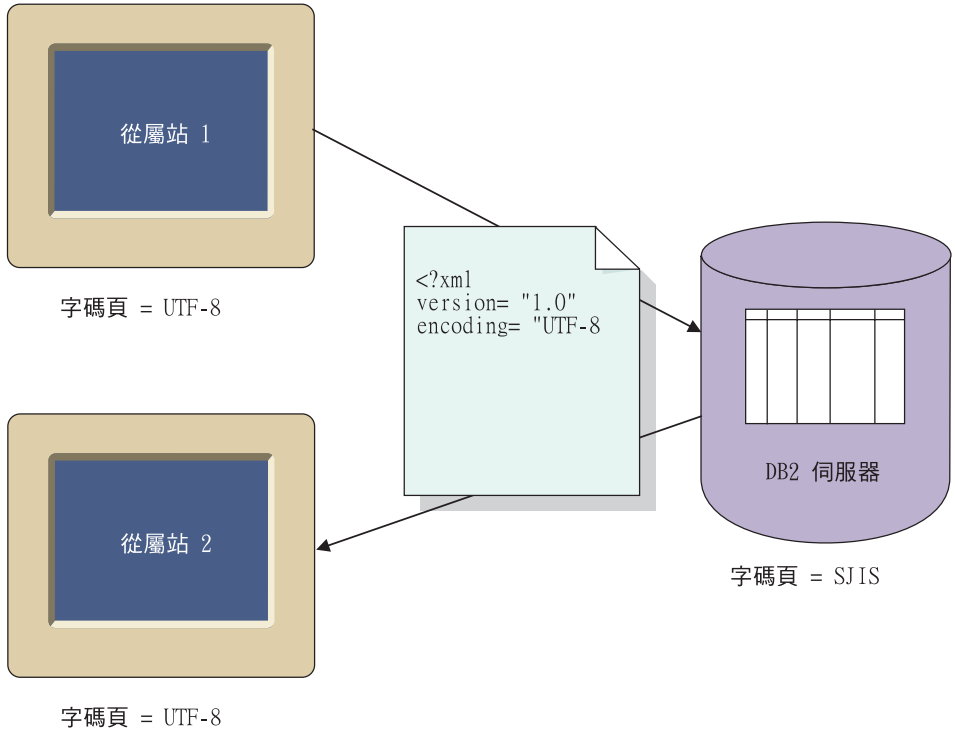


圖 19. 從屬站具有符合的字碼頁

當 DB2 接收或傳送 XML 文件時，它並未檢查編碼宣告。而是檢查從屬站的字碼頁是否符合 DB2 伺服器的字碼頁。 如果不相同，DB2 會轉換 XML 文件的資料，以符合下列各項的字碼頁：

- DB2 伺服器，插入文件到 DB2 表格時
- DB2 伺服器，將文件分解成一或多個 DB2 表格時
- 從屬站，向從屬站呈現文件時

當使用 XML Extender 時，建議您所有的從屬站及伺服器機器都必須具有符合排除資料轉換的字碼頁。

字碼頁不同可能造成下列狀況：

- 轉換期間遺失資料，尤其當原始字碼頁是 Unicode 但目標字碼頁不是 Unicode 時，最容易發生。
- 如果擷取 XML 文件的從屬站，其字碼頁不同於文件的宣告編碼，則文件的宣告編碼和實際的文件編碼將不再一致。

圖20顯示不一致的從屬站字碼頁環境。

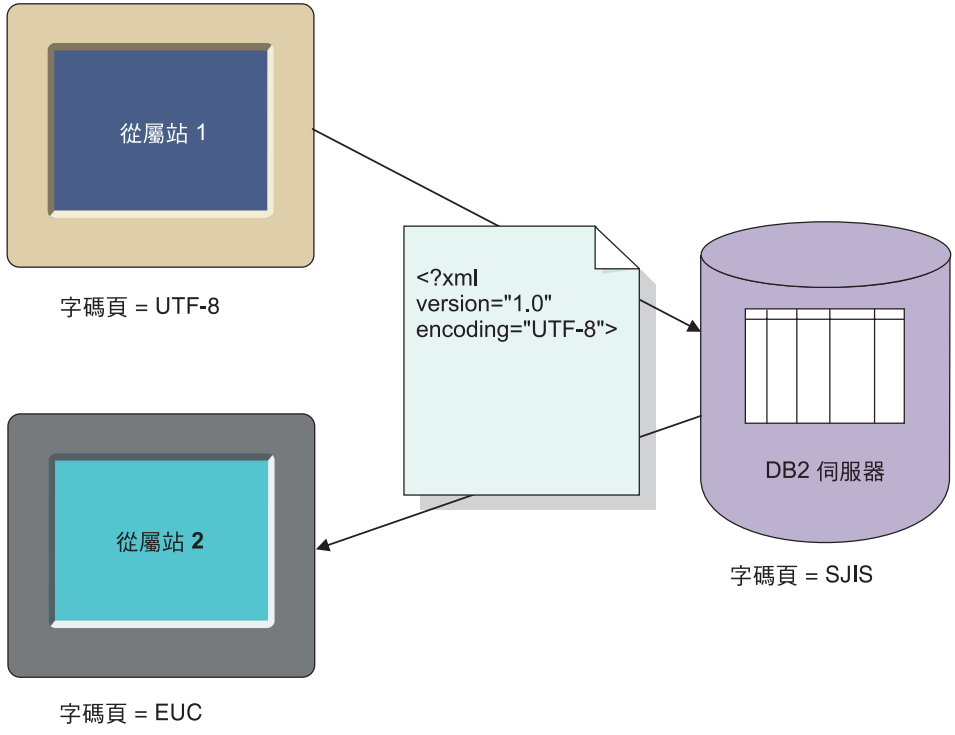


圖 20. 從屬站的字碼頁不符

編碼宣告

編碼宣告會宣告 XML 文件的編碼，且顯示於 XML 宣告陳述式中。DB2 及 XML Extender 會忽略編碼宣告值，但此值必須與從屬站字碼頁一致。

編碼宣告的預設值爲 UTF-8，未註明的編碼宣告表示文件位於 UTF-8。

確定文件編碼和從屬站一致是非常重要的，因爲如果實體包含的編碼宣告不同於宣告中的指定，則 XML 工具 (例如剖析器) 會產生錯誤。 XML Extender 假設 XML 文件使用資料庫伺服器編碼，剖析時會置換 XML 文件宣告的編碼。

宣告編碼值：

在 XML 文件宣告中，指定具有從屬站字碼頁名稱的編碼宣告。例如：

```
<?xml version="1.0" encoding="UTF-8" ?>
```

XML 直欄和集合注意事項

XML Extender 依下列方式處理 XML 直欄和集合的文件編碼：

- **對於 XML 直欄和集合**
  - XML Extender 不依賴 XML 文件中的正確編碼宣告設定，而是假設 XML 文件的字碼頁以及 DAD 和任何 DTD 一定符合下列字碼頁：
    - DB2 從屬站，文件位於 DB2 從屬站系統時
    - DB2 伺服器，文件位於 DB2 伺服器時
- **對於集合 - 組合**

  XML Extender 以伺服器字碼頁建立文件，複製 DAD 中指定的編碼宣告。傳送文件時，DB2 會將字碼頁轉換成從屬站字碼頁。

支援的字碼頁設定值

下列表格說明當您在多個從屬站和伺服器之間轉送文件時，可能遭遇的實際狀況。

支援下列字碼頁組合：

字碼頁都相同

文件編碼符合宣告的編碼，在從屬站和伺服器之間傳送文件需要轉換。 第242頁的表54 顯示範例情節。

表 54. 字碼頁相配

XML 文件宣告和編碼	XML Extender 從屬站	DB2 伺服器			範例
		OS 語言環境	資料庫字碼頁	文件編碼	
SJIS	SJIS (Windows 或 AIX)	SJIS (IBM-943)	943(SJIS)	SJIS	Windows NT 從屬站和伺服器，或 Windows NT 從屬站和 AIX 伺服器
UTF-8	UTF-8	UTF-8	UTF-8	UTF-8	Sun 從屬站和伺服器，或 AIX 從屬站和伺服器

從屬站和 XML 文件的字碼頁相同；伺服器的字碼頁不同

此編碼符合宣告的編碼，在從屬站和伺服器之間傳送文件不需要轉換，但從屬站和 XML 文件宣告的編碼一致。 表55 顯示範例情節

表 55. 從屬站的字碼頁相配，伺服器上則不相配

XML 文件宣告和編碼	XML Extender 從屬站	DB2 伺服器			範例
		OS 語言環境	資料庫字碼頁	文件編碼	
SJIS	SJIS (Windows 或 AIX)	UTF-8	UTF-8	UTF-8	Windows NT 從屬站和伺服器，或 Windows NT 從屬站和 AIX 伺服器
UTF-8	UTF-8	EUC(IBM-954)	954(EUC)	EUC	Sun 從屬站和伺服器，或 AIX 從屬站和伺服器

因爲各從屬站字碼頁彼此一致，與 XML 文件編碼也一致，所以伺服器可以有不同的字碼頁來執行轉換，而不會讓 XML 文件編碼陳述式與其編碼和從屬站不一致。

不支援下列字碼頁實務範例。

混合字碼頁：多重從屬站的字碼頁不一致，或和 XML 文件不一致，且也不同於伺服器。 DB2 根據是否接收文件或是否呈現至從屬站中，以轉換資料至從屬站或伺服器字碼頁。因爲從屬站字碼頁和 XML 文件的編碼不一致，所以 XML 文件的編碼不符合其編碼宣告。 表56 顯示範例情節。

表 56. 字碼頁不相配

XML 文件宣告和編碼	XML Extender client1	XML Extender client2	DB2 伺服器			範例
			OS 語言環境	資料庫字碼頁	文件編碼	
UTF-8	SJIS (Windows NT 或 AIX)	EUC (Windows NT 或 AIX)	UTF-8	UTF-8	資料損毀	Windows NT 從屬站和伺服器，或 Windows NT 從屬站和 AIX 伺服器
UTF-8	SJIS (Sun 或 AIX)	EUC (Sun 或 AIX)	EUC(IBM-954)	954(EUC)	資料損毀	Sun 從屬站和伺服器，或 AIX 從屬站和伺服器

提示和秘訣

當從屬站使用的字碼頁可能和 XML 文件的宣告編碼發生衝突時，建議您阻止 DB2 轉換。下列秘訣提供一些建議，可確保 XML 文件編碼和從屬站字碼頁一致。

使用 XML 工具來處理文件之前，請採納下列其中一項建議：

- 將文件轉換成宣告的編碼字碼頁
- 如果工具提供置換機能，請置換宣告的編碼
- 使用環境變數 DB2CODEPAGE，強制設定從屬站字碼頁爲一個已知的值。詳細資訊，請參閱 DB2 文件中有關作業系統的部份。

  使用此方法時，請確定所有變數和檔案皆爲指定的字碼頁，不只是 XML 文件而已。
- 使用 DB2 CREATE DATABASE 指令的 USING CODESET 選用性參數，強制設定資料庫字碼頁爲一個已知值。詳細資訊，請參閱 DB2 CREATE DATABASE 文件中有關作業系統的部份。

注意事項

本資訊是針對 IBM 在美國所提供之產品與服務所開發出來的，而在其他國家中，IBM 不見得會提供本書中所討論的各項產品、服務或功能。要知道目前在您所在地區是否可用到這些產品與服務時，請諮詢當地的 IBM 業務代表。本書在提及 IBM 的產品、程式或服務時，不表示只能使用 IBM 的產品、程式或服務。 只要未侵犯 IBM 的智慧財產權，任何功能相當的產品、程式或服務都可以取代 IBM 的產品、程式或服務。 不過，任何非 IBM 產品、程式或服務在運作上的評價與驗證，其責任屬於使用者。

IBM 可能有含蓋本書中說明主題之專利或暫准專利應用。使用者不因取得本書而享有本書內容之專利權。您可以用書面方式來查詢授權，來函請寄到：

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

若要查詢有關雙位元組 (DBCS) 資訊的特許權限事宜，請聯絡當地國家的 IBM 智慧財產部門，或者用書面信函寄到：

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

下列段落不適用於英國，或與該國之法律條款抵觸，即不適用： IBM 以 『交附時之現況』 供應本書，而不提供任何明示或默示之保證，包括但不限於如默示保證之不違反、適售性或符合客戶之特殊使用目的。若有些地區在某些交易上並不允許排除上述保證，則該聲明無效。

本資訊可能會有技術上或排版印刷上的錯誤。因此，IBM 會定期修訂；並將修訂後的內容納入新版中。同時，IBM 會隨時改進並 (或) 變動本書中所提及的產品及 (或) 程式，而不另行通知。

本程式之獲授權者若希望取得相關資料，以便使用下列資訊： (1) 獨立建立的程式與其他程式 (包括此程式) 之間的資訊更換 (2) 相互使用已交換之資訊，若有任何問題應聯絡：

IBM Corporation
J74/G4

555 Bailey Avenue
P.O. Box 49023
San Jose, CA 95161-9023
U.S.A.

上述資料之取得有其特殊條件，在某些情況下必須付費方得使用。

IBM 基於雙方之間的 IBM 客戶合約、 IBM International Program 授權合約或任何同等合約中的條款約定，提供本書中所說的授權程式與其所有適用的授權材料。

有關非 IBM 產品的資訊皆取自該產品的提供者、其出版聲明或其他公開的可用來源。 IBM 尚未測試那些產品，也無法確認執行效能的精確度、相容性或任何與非 IBM 產品相關的要求。如果您對非 IBM 產品的性能有任何的疑問，請逕向該產品的供應商查詢。

有關 IBM 未來動向的任何陳述，僅代表 IBM 的目標而已，並可能於未事先聲明的情況下有所變動或撤回。

著作權授權：

本資訊包含原始語言的範例應用程式，用以說明各種作業平台上的程式設計技術。您可以基於研發、使用、銷售或散布符合範例程式的作業平台應用程式介面等目的，以任何形式複製、修改及散布這些範例程式而不必向 IBM 付費。這些範例應用程式並未經過徹底的測試。因此，IBM 不提供明示或默示其可靠性、有用性或特定效用的保證。

商標

下列專有名詞是 IBM Corporation 在美國或 (以及) 其它國家的商標：

DB2	Net.Data
DB2 Extenders	OS/2
DB2 Universal Database	OS/390
IBM	OS/400
IMS	VTAM

Java 以及所有以 Java 爲基礎的商標及標章是 Sun Microsystems, Inc. 在美國及 (或) 其它國家的商標或註冊商標。

Microsoft、Windows、Windows NT 以及 Windows 標章是 Microsoft Corporation 在美國及 (或) 其它國家的商標。

UNIX 是在美國及 (或) 其它國家的註冊商標，由 X/Open Company Limited 獨家授權。

其它公司、產品及服務名稱，可能是其它公司的商標或服務標誌。

名詞解釋

一劃

一致資源定位器 (URL). 命名 HTTP 伺服器並選用性地命名目錄及檔名的一種位址，例如：http://www.ibm.com/data/db2/extenders。

三劃

大型物件 (LOB). 一連串位元組，其長度最多達 2 GB。LOB 可以有三種：*二進位大型物件* (BLOB)、*字元大型物件 (CLOB)* 或 *雙位元組大型物件 (DBCLOB)*。

四劃

中間資料表. 請參閱*管理支援表*。

元素. 請參閱 *XML 元素*。

內含的 SQL. 寫在應用程式內的 SQL 陳述式。請參閱*靜態 SQL*。

分割區. 儲存體的固定大小劃分。

分解. 將一些 XML 文件分成 XML 集合中的關聯式表格的集合。

文件存取定義 (DAD). 用來定義 XML 直欄的索引方法或 XML 集合的對映方法。它可用來啓用 XML 集合的 XML Extender 直欄，這個集合是格式化的 XML。

文件製作. 在 XML 集合中從關聯式資料建立 XML 文件。

文件類型定義 (DTD). XML 元素及屬性的一組宣告。DTD 定義 XML 文件要使用什麼元素，依什麼順序使用這些元素，以及哪些元素可含有其它元素。您可使 DTD 與文件存取定義 (DAD) 檔相關，以驗證 XML 文件。

五劃

主要鍵. 一個唯一鍵，它是表格的一部份定義。主要鍵是參照限制定義的預設母鍵。

可延伸的樣式表語言 (XSL). 用來表達樣式表的語言。 XSL 含有兩個部份：轉換 XML 文件的語言，以及指定格式化語意的 XML 字彙。

可延伸的樣式表語言轉換 (XSLT). 用來將 XML 文件轉換成其它 XML 文件的語言。 XSLT 專門作爲 XSL 的一部份，它是 XML 的樣式表語言。

外來鍵. 某一個鍵，它是參照限制的一部份定義，由相依表格的一個或多個直欄組成。

外部檔. 存在於 DB2 的外部檔案系統的檔案。

本端檔案系統. 存在於 DB2 的一種檔案系統

六劃

全文搜尋. 使用 DB2 Text Extender 隨處搜尋字串，而不管文件結構。

合併. 一種關聯式作業，該作業可讓您根據符合的欄位值，從兩個以上的表格擷取資料。

合併概略表. ″CREATE VIEW″ 陳述式建立的 DB2 概略表，該概略表合併一個以上的表格。

多次出現的項目. 在文件中是否可以重複使用直欄元素或屬性的指示。在 DAD 中指定多次出現的項目。

字元大型物件 (CLOB). 單一位元組字元字串，字串可多達 2 GB。CLOB 有一個相關的字碼頁。含有單一位元組字元的文字物件以 CLOB 形式儲存在 DB2 資料庫。

存取及儲存方法. 使用兩種主要存取及儲存方法，使 XML 文件與 DB2 資料庫相關：XML 直欄及 XML 集合。亦請參閱 *XML 直欄*及 *XML 集合*。

有效文件. 一種有相關 DTD 的 XML 文件。若要文件有效，那麼 XML 文件不可違反在它的 DTD 指定的語法規則。

次查詢. 一個完整 SELECT 陳述式，使用於 SQL 陳述式的搜尋條件內。

七劃

位置路徑. 位置路徑是一連串 XML 標示，識別 XML 元素或屬性。位置路徑識別 XML 文件結構，指出元素或屬性的環境定義。單斜線 (*/*) 路徑指出環境定義是整個文件。 取出的 UDF 使用位置路徑識別要取出的元素及屬性。定義 XML 直欄的索引方法時，DAD 檔也使用位置路徑指定 XML 元素，或屬性，與 DB2 直欄之間的對映。此外，Text Extender 使用位置路徑進行結構式文字搜尋。

形式完善的文件. 不含 DTD 的 XML 文件。雖然是 XML 規格，具有效 DTD 的文件也必須是形式完善的文件。

八劃

使用者定義的函數 (UDF). 定義給 DBMS 的一種函數，而且以後在 SQL 查詢中可以參照該函數。它可以是下列其中一項：

- 一種外部函數，在該函數中使用一種程式設計語言撰寫該函數主體，該程式設計語言的引數是純量值，而且爲每一個呼叫產生純量結果。
- 一種來源函數，由 DBMS 已知道的另一個內建函數或使用者定義的函數執行該函數。本函數可以是純量函數或直欄 (聚集) 函數，而且從一組值傳回單一個值 (例如，MAX 或 AVG)。

使用者定義的類型 (UDT). 一種資料類型，它不是資料庫管理程式的原始資料類型，且由使用者建立該資料類型。請參閱*特殊類型*。

使用者表格. 爲應用程式建立而且供應用程式使用的表格。

定位器. 可以用來尋找物件的指標。在 DB2 中，大型物件區塊 (LOB) 定位器是尋找 LOB 的資料類型。

物件. 在物件導向程式設計中，由資料以及與該資料相關的作業組成的抽象物。

直欄資料. 儲存在 DB2 直欄的資料。此資料類型可以是 DB2 支援的任何資料類型。

表格空間 (table space). 指儲存資料庫物件的配置區集合之摘要。表格空間提供資料庫與儲存在資料庫內的表格，兩者之間的間接層次。表格空間：

- 在已指定給它的媒體儲存裝置上有空間。
- 裡面已建立表格。這些表格將使用屬於表格空間的配置區中的空間。表格的資料、索引、長欄位及 LOB 部份，都可儲存在相同表格空間中，也可以個別分在不同的表格空間中。

九劃

查詢. 根據特定條件來要求資料庫資訊；例如，一個查詢可能是要求列出客戶表格中餘額超過 1000 的全部客戶。

述詞. 搜尋條件的一個元素，該元素明示或暗示比較作業。

十劃

特殊類型. 請參閱*使用者定義類型*。

索引. 根據一個鍵值以邏輯方式排序的一組指標。索引提供對資料的快速存取權，並且能夠對表格中的橫列實施唯一性。

純量函數. 一種 SQL 作業，它從另一個值產生單一個值，而且該值顯示成函數名稱，它的後面是用括弧括住的一列引數。

十一劃

區段搜尋. 在一個區段內提供文字搜尋，應用程式可定義該區段。若要支援結構式文字搜尋，您可以根據 Xpath 的縮寫位置路徑定義區段。

強制轉型函數. 一種函數，它用來將 (來源) 資料類型案例轉換成不同 (目標) 資料類型案例。一般而言，強制轉型函數有目標資料類型的名稱。它有一個單一引數，該引數的類型是來源資料；它的傳回類型是目標資料類型。

條件. 選取 XML 資料的準則或合併 XML 集合表的方法之指定。

頂層 element_node. 在 DAD 中，XML 文件的 Root 元素表示。

十二劃

程序. 請參閱*儲存程序*。

結果表格. 一種表格，它含有一些橫列作爲 SQL 查詢或執行儲存程序的結果。

結果集. 儲存程序傳回的一組橫列。

結構式文字索引. 使用 DB2 Text Extender，根據 XML 文件的樹狀結構來編製字串索引。

絕對位置路徑. 物件的完整路徑名稱。從最高層或 ″Root″ 元素開始絕對路徑名稱，以正斜線 (/) 或反斜線 (\) 字元識別該名稱。

超載函數. 有多重函數案例的函數名稱。

十三劃

節點. 在資料庫分割中，與資料庫分割區同義。

資料交換. 共用應用程式之間的資料。XML 支援資料交換，但不必從專屬格式首先轉換資料。

資料來源. 本端或遠端關聯式或非關聯式資料管理程式，該管理程式能夠透過支援 ODBC API 的 ODBC 驅動程式支援資料存取。

資料鏈結. 一個 DB2 資料類型，它可以從資料庫啓用邏輯參照，至儲存在資料庫外部的檔案。

資料類型 (data type). 直欄與文字的屬性。

路徑表示式. 請參閱*位置路徑*。

電子資料交換 (EDI). 企業消費型商務 (B2B) 應用程式的電子資料交換標準。

預設強制轉型函數. 將 SQL 基本類型強制轉型成 UDT。

預設概略表. 一種資料呈現，在該呈現中合併 XML 表與所有它的相關輔助表格。

十四劃

對映方法. 在關聯式資料庫中如何顯示 XML 資料的定義。在 DAD 中指定對映方法。 XML Extender 提供兩種對映方法： *SQL 對映*及*關聯式資料庫節點 (RDB_node) 對映*。

管理支援表. DB2 擴充元用來處理使用者向 XML 物件提出要求的表格。有些管理支援表識別爲擴充元啓用的使用者表格及直欄。有些管理支援表含有關於啓用的直欄中物件之屬性資訊。與中間資料表同義。

綱目. 一種資料庫物件集合，如表格、概略表、索引或觸發函式。它提供資料庫物件的邏輯分類。

輔助表格. XML Extender 建立的一些附加表格，在 XML 直欄搜尋元素或屬性時可使用這些表格增進效能。

十六劃

靜態 SQL. 內含在一個程式的一些 SQL 陳述式，而且在準備該程式期間備妥這些 SQL 陳述式之後才執行該程式。備妥這些靜態 SQL 陳述式之後，有一個靜態 SQL 陳述式不變，不過該陳述式指定的主變數值可能變更。

十七劃

儲存程序. 一個程式結構及內含的 SQL 陳述式區塊，它儲存在資料庫而且可依名稱呼叫它。儲存程序容許應用程式分成兩個部份執行。在從屬站執行一個部份，在伺服器執行另一個部份。這樣做可讓一個呼叫產生資料庫的數個存取。

應用程式設計介面 (API).

(1) 作業系統或可單獨購買的授權程式所提供的功能介面。API 可讓使用高階語言撰寫的應用程式，使用作業系統或授權程式的特定資料或函數。

(2) 在 DB2 中，位於介面內的函數；例如，取得錯誤訊息 API。

(3) 在 DB2 中，此介面內的一個函數。例如，取得錯誤訊息 API。

十八劃

瀏覽器. 請參閱 *Web 瀏覽器*。

簡單位置路徑. 以單一斜線 (/) 連接的一連串元素類型名稱。

雙位元組大型物件 (DBCLOB). 一個雙位元組字元字串，或單一位元組及雙位元組字元組合，其中字串可多達 2 GB。 DBCLOB 有一個相關的字碼頁。含有雙位元組字元的文字物件以 DBCLOB 形式儲存在 DB2 資料庫。

十九劃

關聯式資料庫節點 (RDB_node). 一種節點，它含有表格、選用性直欄及選用性條件的一個或多個元素定義。表格及直欄是用來定義 XML 資料儲存在資料庫的方法。此條件指定選取 XML 資料的準則或合併 XML 集合表的方法。

二十一劃

屬性. 請參閱 *XML 屬性*。

二十三劃

驗證. 使用 DTD 以確定 XML 文件有效，而且允許結構化搜尋 XML 資料的處理。 DTD 儲存於 DTD 儲存庫中。

A

API. 請參閱*應用程式設計介面*。

attribute_node. 元素屬性的表示。

B

B-tree 索引. DB2 引擎提供的原來索引方法。它在 B 樹狀結構中建置索引登錄。支援 DB2 基本資料類型。

C

CLOB. 字元大型物件。

D

DAD. 請參閱*文件存取定義*。

DBCLOB. 雙位元組大型物件。

DTD. (1) . (2) 請參閱*文件類型定義*。

DTD 參照表 (DTD_REF 表). 一個含有 DTD 的表格，它是用來驗證 XML 文件以及協助應用程式定義 DAD。使用者可將自己的 DTD 插入 DTD_REF 表。針對 XML 啓用資料庫時建立本表格。

DTD 儲存庫. 一個名稱是 DTD_REF 的 DB2 表格，該表格的每一列代表一個具有其它 meta 資料資訊的 DTD。

DTD_REF 表. DTD 參照表。

E

EDI. 電子資料交換。

element_node. 元素的表示。element_node 可以是 Root 元素或子項元素。

J

Java Database Connectivity (JDBC). 一個應用程式設計介面 (API)，該介面與 Open Database Connectivity (ODBC) 具有相同特性，但 JDBC 是專門設計給 Java 資料庫應用程式使用。另外，就那些沒有 JDBC 驅動程式的資料庫而言， JDBC 包含一個 JDBC 到 ODBC 橋接器，這個橋接器是將 JDBC 轉換到 ODBC 的機制； JDBC 對 Java 資料庫應用程式提供 JDBC API 而且將它轉換成 ODBC。 Sun Microsystems, Inc. 及協力廠商以及供應商共同開發 JDBC。

JDBC. Java Database Connectivity.

L

LOB. 大型物件。

O

ODBC. 在關聯式及非關聯式資料庫管理系統中，存取資料的標準應用程式設計介面 (API)。使用這個 API 時，資料庫應用程式可存取儲存在各種電腦上的資料庫管理系統的資料，即使各個資料庫管理系統使用不同資料儲存格式及程式設計介面也一樣。ODBC 是根據 X/Open SQL Access Group 的呼叫層次介面 (CLI) 規格而定，而且是由 Digital Equipment Corporation (DEC)、Lotus、Microsoft 及 Sybase 聯合開發。與 *Java Database Connectivity* 相對照。

ODBC. ODBC

R

RDB_node. 關聯式資料庫節點。

RDB_node 對映. XML 元素內容的位置或 XML 屬性值，皆由 RDB_node 定義。 XML Extender 使用這個對映決定儲存或擷取 XML 資料的位置。

root ID. 一個唯一識別字，它使全部輔助表格與應用程式表格相關。

Root 元素. XML 文件的第一個元素。

S

SQL 對映. XML 元素內容或 XML 屬性值與關聯式資料的關係定義，該定義使用一個或多個 SQL 陳述式及 XSLT 資料模型。 XML Extender 使用此定義決定儲存或擷取 XML 資料的位置。在 DAD 中使用 SQL_stmt 元素定義 SQL 對映。

T

text_node. 元素的 CDATA 文字表示。

U

UDF. 請參閱*使用者定義的函數*。

UDT. 請參閱*使用者定義類型*。

UNION. 一種 SQL 作業，它合併兩個 select 陳述式結果。 UNION 經常用來合併得自數個表格的值列示。

URL. 通用資源定位器。

W

Web 瀏覽器. 一種從屬站程式，它對 Web 伺服器起始要求，並顯示該伺服器傳回的資訊。

X

XML. 可延伸的標記語言。

XML UDF. XML Extender 提供的一種 DB2 使用者定義的函數。

XML UDT. XML Extender 提供的一種 DB2 使用者定義的類型。

XML 元素. 在 XML DTD 中指定的 XML 標示或 ELEMENT。XML Extender 使用位置路徑識別元素。

XML 物件. 等於 XML 文件。

XML 直欄. 應用程式表格的一個直欄，已經針對 XML Extender UDT 啓用該直欄。

XML 表格. 一種應用程式表格，它含有一個或多個 XML Extender 直欄。

XML 集合. 一個關係表格集合，這些表格提供資料編製 XML 文件，或提供要從 XML 文件分解的資料。

XML 路徑語言. 定址一部份 XML 文件的語言。「XML 路徑語言」專供 XSLT 使用。您可使用爲 XPath 定義的語法表達位置路徑。

XML 標示. 任何有效的 XML 標記語言標示，主要是 XML 元素。可輪流使用術語標示及元素。

XML 屬性. 在 DTD 中的 XML 元素之下由 ATTLIST 指定的任何屬性。 XML Extender 使用位置路徑識別屬性。

XPath. 定址一部份 XML 文件的語言。

XPath 資料模型. 您可以使用樹狀結構來建立模型及使用節點來導覽 XML 文件。

XSL. XML 樣式表語言。

XSLT. XML 樣式表語言轉換。

索引

索引順序以中文字，英文字，及特殊符號之次序排列。

〔二劃〕

〔四劃〕

〔五劃〕

〔六劃〕

〔七劃〕

〔八劃〕

〔九劃〕

〔十劃〕

〔十一劃〕

〔十二劃〕

〔十三劃〕

〔十四劃〕

〔十五劃〕

〔十七劃〕

〔十八劃〕

〔十九劃〕

〔二十三劃〕

A

B

F

J

M

O

R

S

T

U

W

X

連絡 IBM

當您有技術上的問題時，請在洽詢「DB2 客戶支援中心」之前，仔細閱讀並執行*疑難排解指南*所建議的動作。該指南會告訴您必須預先準備的資訊，協助「DB2 客戶支援中心」提供更完善的服務。

若要取得 DB2 Universal Database 產品的相關資訊，或是訂購該系列產品，請洽詢當地 IBM 分公司的業務代表，或是 IBM 授權的軟體經銷商。

美國地區居民請撥下列電話：

- 1-800-237-5511，客戶支援中心
- 1-888-426-4343，取得可用服務選項的資訊

產品資訊

美國地區居民請撥下列電話：

- 1-800-IBM-CALL (1-800-426-2255) 或 1-800-3IBM-OS2 (1-800-342-6672)，訂購產品或取得一般資訊。
- 1-800-879-2755，訂購出版品。

http://www.ibm.com/software/data/

DB2 全球資訊網 (WWW) 網頁提供最新的 DB2 資訊，包括新聞、產品說明、教育課程等。

http://www.ibm.com/software/data/db2/library/

DB2 Product and Service Technical Library 提供常見問題、修正程式、書籍及最新的 DB2 技術資訊。

註: 本資訊僅提供英文版。

http://www.elink.ibmlink.ibm.com/pbl/pbl/

International Publications 訂購網站會提供書籍的訂購資訊。

http://www.ibm.com/education/certify/

IBM 網站中的 Professional Certification Program 會提供包括 DB2 在內之各種 IBM 產品的認證測試資訊。

ftp.software.ibm.com

以匿名方式登入。您可以在目錄 `/ps/products/db2` 中找到 DB2 及其它產品的相關示範程式、修訂程式、資訊及工具。

comp.databases.ibm-db2, bit.listserv.db2-l

使用者可以利用這些 Internet 新聞群組討論 DB2 產品的使用經驗。

在 Compuserve 上：GO IBMDB2

輸入此項指令，即可存 IBM DB2 Family 論壇。這些論壇支援所有 DB2 產品。

關於美國以外地區如何連絡 IBM 的資訊，請參考 *IBM Software Support Handbook* 的附錄 A。 若要存取本文件，請造訪下列網頁：http://www.ibm.com/support/，然後選取接近網頁底端的 IBM Software Support Handbook 鏈結。

註: 在某些國家，IBM 授權經銷商應連絡其經銷商支援組織，而不是 IBM 支援中心。

IBM DB2 Universal Database
XML Extender 管理與程式設計
版本 7

讀者意見表

爲使本書盡善盡美，本公司極需您寶貴的意見；懇請您使用過後，撥冗填寫下表，惠予指教。

請於下表適當空格內，填入記號（∨）；我們會在下一版中，作適當修訂，謝謝您的合作!

評估項目	評　估　意　見		備　註
正確性	內容說明與實際程序是否符合	□是□否	
	參考書目是否正確	□是□否	
一致性	文句用語及風格，前後是否一致	□是□否	
	實際畫面訊息與本書所提之畫面訊息是否一致	□是□否	
完整性	是否遺漏您想知道的項目	□是□否	
	字句、章節是否有遺漏	□是□否	
術語使用	術語之使用是否恰當	□是□否	
	術語之使用，前後是否一致	□是□否	
可讀性	文句用語是否通順	□是□否	
	有否不知所云之處	□是□否	
內容說明	內容說明是否詳盡	□是□否	
	例題說明是否詳盡	□是□否	
排版方式	本書的形狀大小，版面安排是否方便使用	□是□否	
	字體大小，顏色編排，是否有助於閱讀	□是□否	
目錄索引	目錄內容之編排，是否便於查考	□是□否	
	索引語錄之排定，是否便於查考	□是□否	
	※評估意見爲"否"者，請於備註欄說明。		

其他：（篇幅不夠時，請另紙說明。）

上述改正意見，一經採用，本公司有合法之使用及發佈權利，特此聲明。

Printed in Singapore