DB2 XML evaluation guide

A step-by-step introduction to the XML storage and query capabilities of DB2 Viper

Skill Level: Intermediate

Gerald Leung (gktleung@ca.ibm.com)
Database Technology Advocate
IBM

Grant Hutchison (ghutchis@ca.ibm.com)
Product Marketing Manager
IBM

01 Jun 2006

The DB2 XML evaluation guide is a step-by-step tutorial to introduce the reader to the DB2® Viper data server on Windows® platforms using the XML storage and searching (SQL/XML, XQuery) capabilities available to support next-generation applications.

Section 1. Before you start

The DB2 Viper data server provides many new capabilities including: flexible security options, data storage optimization, and improved support for storing, managing, and querying XML data. DB2 is designed to optimize access to XML and relational data and these capabilities are available to C++, .NET®, COBOL, Java™, and PHP application developers. DB2 Viper is now available (for evaluation purposes) and this tutorial can be used to aid in understanding how to use XML and SQL together to build the basis of a flexible scheduling application.

About this tutorial

The *DB2 XML Evaluation Guide* is a step-by-step tutorial to introduce the reader to the DB2 Viper data server on Windows platforms using the XML storage and searching (SQL/XML, XQuery) capabilities available to support next-generation applications.

Introduction to DB2 Viper

DB2 Viper is the industry's first hybrid data server for managing data from both relational and pure XML formats. DB2 has been providing high-performance data storage and access for relational data based on SQL standards and data storage optimizations such as data partitioning and advanced indexing and query optimization techniques. Now, DB2 has introduced an optimized data storage engine for XML data alongside the existing relational engine.

Application developers can now store XML data directly inside a DB2 server and reap the benefits of transactions, advanced data resiliency, secure access, and, of course, the ability to search large amounts of XML data using XQuery.

XML data is often used for inter-application data exchange and document management purposes. The flexible and self-describing nature of XML data makes it ideal for many application scenarios.

The ability to query XML data has been quite limited. Recently, a new standards-based XML query language has been published. This query language is known as XQuery Version 1.0.

XQuery Version 1.0 was released in November 2005 and it is an extension of the existing XPath Version 2.0 standard. See Resources for Web sites to read more about XQuery and XPath.

Objectives

This tutorial will help familiarize you with DB2's new support for XML data. You will perform the following tasks:

1. Create a DB2 Viper database (to support relational and XML data objects).
2. Create a DB2 table to store XML data.
3. Add data to the table using SQL INSERT statements.
4. Query a table using SQL and XQuery.
5. Adding additional data into DB2 tables using the DB2 IMPORT utility.
6. Issue various queries using XQuery.
7. Querying using XQuery on a XML column of an indexed table.
8. Create and test a view for simplified data access.

Prerequisites

This tutorial is written for database administrators who have limited experience with DB2 Viper and have some experience with SQL and some knowledge of XML.

System requirements

To run the examples in this tutorial, you need a Windows 2000 or higher machine with an account having administrator privileges.

Section 2. Getting started with DB2 Viper

Ensure you have downloaded and properly installed the DB2 Viper test drive based on the upcoming DB2 Viper data server (see Resources for download links).

To install DB2 Viper, follow these steps (default options should be used unless otherwise specified):

Installation of DB2 Viper

1. Identify the Windows drive assigned to the CD drive with the DB2 Viper test drive CD. (This tutorial assumes the CD-ROM is mapped to the Z: drive).
2. Double-click on `setup.exe` from `Z:\Run\Install\Windows_x86\db2ese_viper_WIN_x86` and the **DB2 Setup Launchpad will appear**.
3. Within the **DB2 Setup Launchpad**, click on **Install a Product** from the panel on the left.
4. Click on **Install New** under **DB2 Enterprise Server Edition**.
5. In the installer, click **Next** to view the **License Agreement**. You will need to **Accept** the agreement to continue and click **Next**.
6. Choose a **Typical** installation and click **Next**.
7. Confirm that you plan to **Install DB2 Enterprise Server Edition on this computer and save my settings in a response file**. Click **Next**.
8. Verify the installation directory, and click **Next**.

9. In the **Configure DB2 instances** screen, click **Next**.
10. In the user information screen, set the Domain to **"None - use local user account"** using the pull-down menu, leave User name default (**db2admin**), and set a password. (Optionally, you can select **LocalSystem** account, but there are limitations introduced if this option is used; click the **Help** button to learn more.)
11. Make sure the checkbox **Use the same user name and password for the remaining DB2 services** is checked and click **Next**.
12. Click **Next** to skip the **Prepare the DB2 tools catalog** screen.
13. To simplify installation, uncheck **Set up your DB2 server to send notifications** and click **Next**.
14. Verify the **Enable operating system security** checkbox is checked and leave **DB2 administrators group** as the default value **DB2ADMNS** and leave **DB2 users group** as the default value **DB2USERS**, and click Next.
15. Click **Finish**, to start the DB2 product installation, when you reach the **Start copying and files and create response file** screen.
16. When the installation is complete, a window with the message **Setup is complete** will appear. Click **Finish** to complete the installation procedure.
17. A window entitled **DB2 First Steps** will now appear. Click **Create profile**. Your Web browser will open a page called **DB2 First Steps**. This tutorial contains many useful links to information on DB2, but it will not be used during this tutorial. Close your Web browser and continue with this tutorial

Section 3. Step 1. Creating a DB2 Viper database

To get started, create a new DB2 database. DB2 Viper supports pure XML data storage and query for databases enabled for the UNICODE character set. In the following steps, you will create a UNICODE DB2 Viper database.

DB2 databases can be created using graphical tools.

DB2 Control Center - CREATE DATABASE

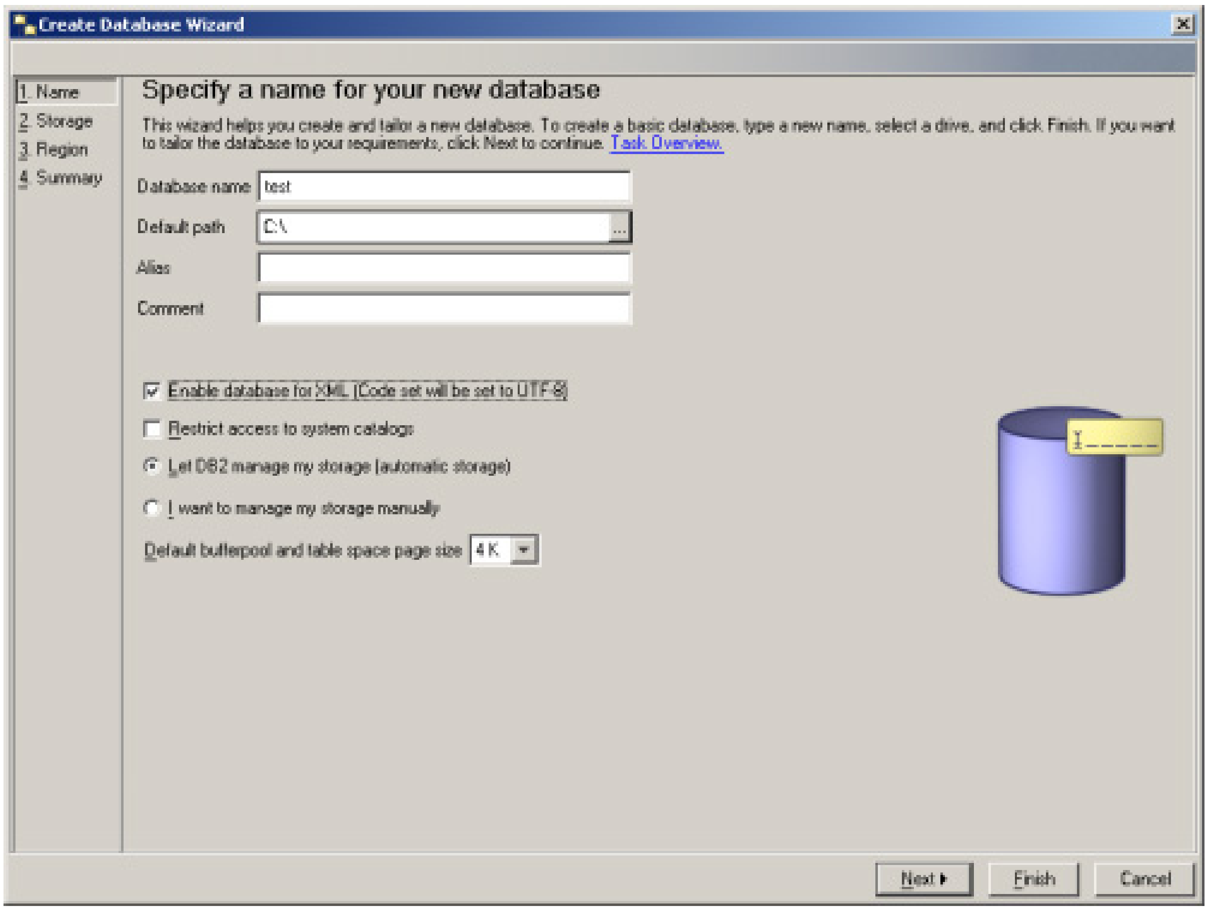
Creating a 'TEST' DB2 database (DB2 Control Center)

Start -> IBM DB2 -> DB2 (default) -> General Administration Tools -> Control

Center

Within the DB2 Control Center application, choose the **Basic view**. Right-click on **All Databases** and choose **Create Database** -> **Standard**. You will be presented with the **Create Database Wizard**. Fill in the name as **test** and select **Enable database for XML**. Your screen should appear like Figure 1.

Figure 1. Create database wizard



Specify a name for your new database

This wizard helps you create and tailor a new database. To create a basic database, type a new name, select a drive, and click Finish. If you want to tailor the database to your requirements, click Next to continue. Task Overview.

Database name: test
Default path: C:\
Alias:
Comment:

[✓] Enable database for XML (Code set will be set to UTF-8)
[ ] Restrict access to system catalogs
(●) Let DB2 manage my storage (automatic storage)
( ) I want to manage my storage manually
Default bufferpool and table space page size: 4 K

Click **Next** twice to get to the **Region** screen. Set the **Country/Region** to the value **United States** and leave **Code set** as the default value **UTF-8**. Click **Finish** to complete.

The **TEST** database is now created and ready to store XML and relational data.

NOTE: To create the same DB2 Viper database using DB2 CLP:

```
 create db test using codeset utf-8 territory us
```

Section 4. Step 2. Creating a DB2 table

DB2 columns can be defined using various data types, including XML. To store XML data in DB2, simply specify the XML data type for one or more columns. DB2 tables may contain a mixture of traditional and XML data types. DB2 uses optimal storage techniques to store and maintain XML data, in addition to providing powerful query capabilities.

(Option 1) DB2 Command Line Processor - CREATE TABLE

In this exercise, you will create a single table called **CONTACT_SCHEDULES**, and it is used to track basic contact information for a large number of people. The column called **SCHEDULE** is used to store each person's schedule of activities as a single XML document with one or more scheduled activities.

The data architect and application developer for this project decided to use XML to enable a flexible infrastructure. Activities are often integrated with other applications (PDA devices and web collaboration tools) and using a pure XML format for each person's activities is ideal.

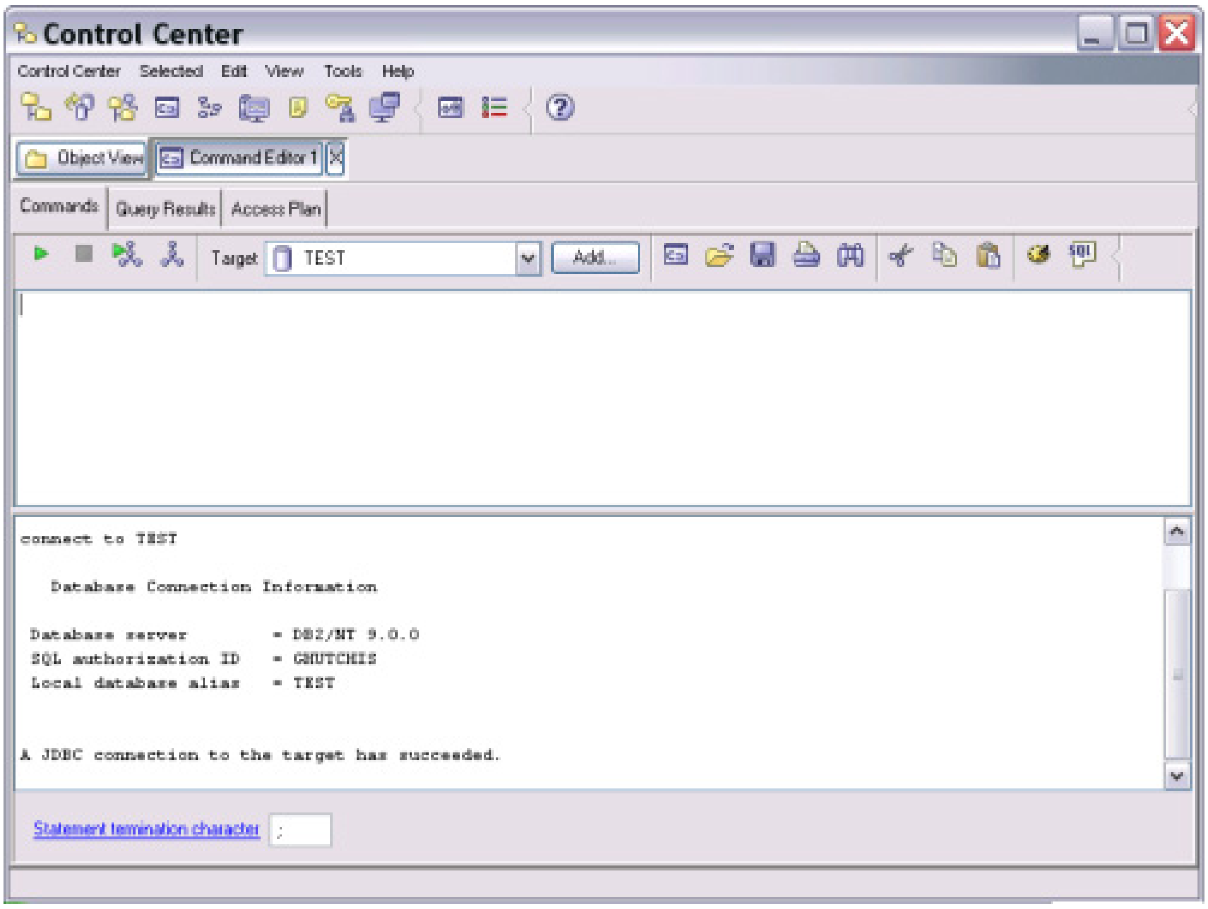
Listing 1. Creating the CONTACT_SCHEDULES table

```
CREATE TABLE user1.contact_schedules
(
        id INT PRIMARY KEY GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1),
        fname VARCHAR(20) NOT NULL,
        lname VARCHAR(20) NOT NULL,
        title VARCHAR(30) NOT NULL,
        workphone CHAR(10) NOT NULL,
        homephone CHAR(10) NOT NULL,
        mobilephone CHAR(10) NOT NULL,
        schedule XML
);
```

As you can see in Listing 1, the `CONTACT_SCHEDULES` table has a combination of standard relational (INTEGER, CHARACTER) data types and a single XML column. The ID column is also defined to be an auto-generated column value and it has also been defined as a primary key for the table.

To use the **DB2 Command Editor**, launch the DB2 Control Center as described in step 1. Right -click on the **TEST** database from the navigation tree and select **Query** from the pop-up menu. The DB2 Command Editor should now be open and will establish a connection to the **TEST** database.

Figure 2. DB2 Command Editor



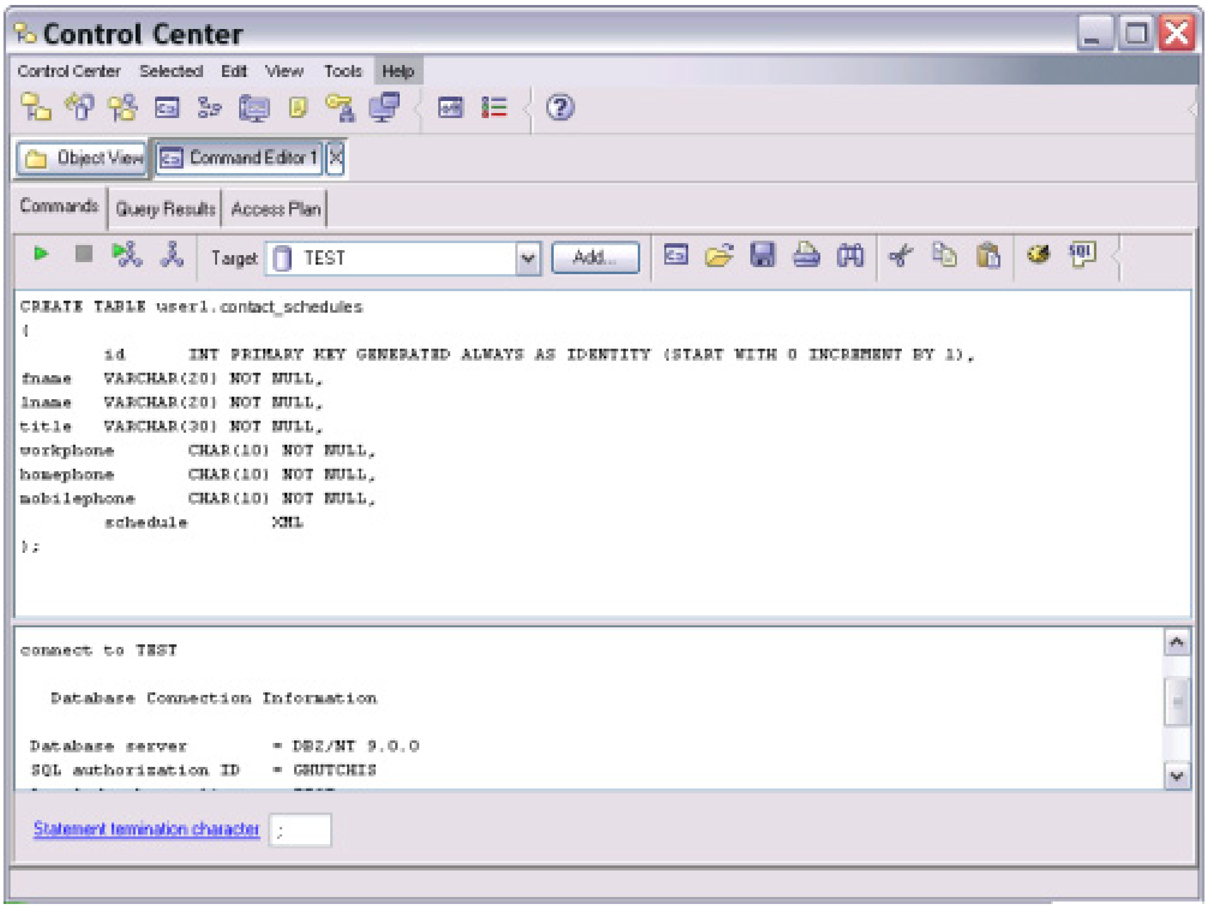
```
connect to TEST

   Database Connection Information

 Database server        = DB2/NT 9.0.0
 SQL authorization ID   = GHUTCHIS
 Local database alias   = TEST


A JDBC connection to the target has succeeded.
```

Figure 3. Creating the CONTACT_SCHEDULES table using the DB2 Command Editor


Control Center
Control Center Selected Edit View Tools Help
Object View Command Editor 1
Commands Query Results Access Plan
Target TEST Add...
Statement termination character ;

```
CREATE TABLE user1.contact_schedules
(
        id      INT PRIMARY KEY GENERATED ALWAYS AS IDENTITY (START WITH 0 INCREMENT BY 1),
fname   VARCHAR(20) NOT NULL,
lname   VARCHAR(20) NOT NULL,
title   VARCHAR(30) NOT NULL,
workphone       CHAR(10) NOT NULL,
homephone       CHAR(10) NOT NULL,
mobilephone     CHAR(10) NOT NULL,
        schedule        XML
);
```

```
connect to TEST

   Database Connection Information

 Database server        = DB2/NT 9.0.0
 SQL authorization ID   = GHUTCHIS
```

The CREATE TABLE statement can be typed into the Command Editor (as shown in Listing 1) or imported into the Command Editor by clicking on the "open folder" icon

The CREATE TABLE statement is provided in the following location:
`c:\temp\ViperIntro\listings\listing1.txt`

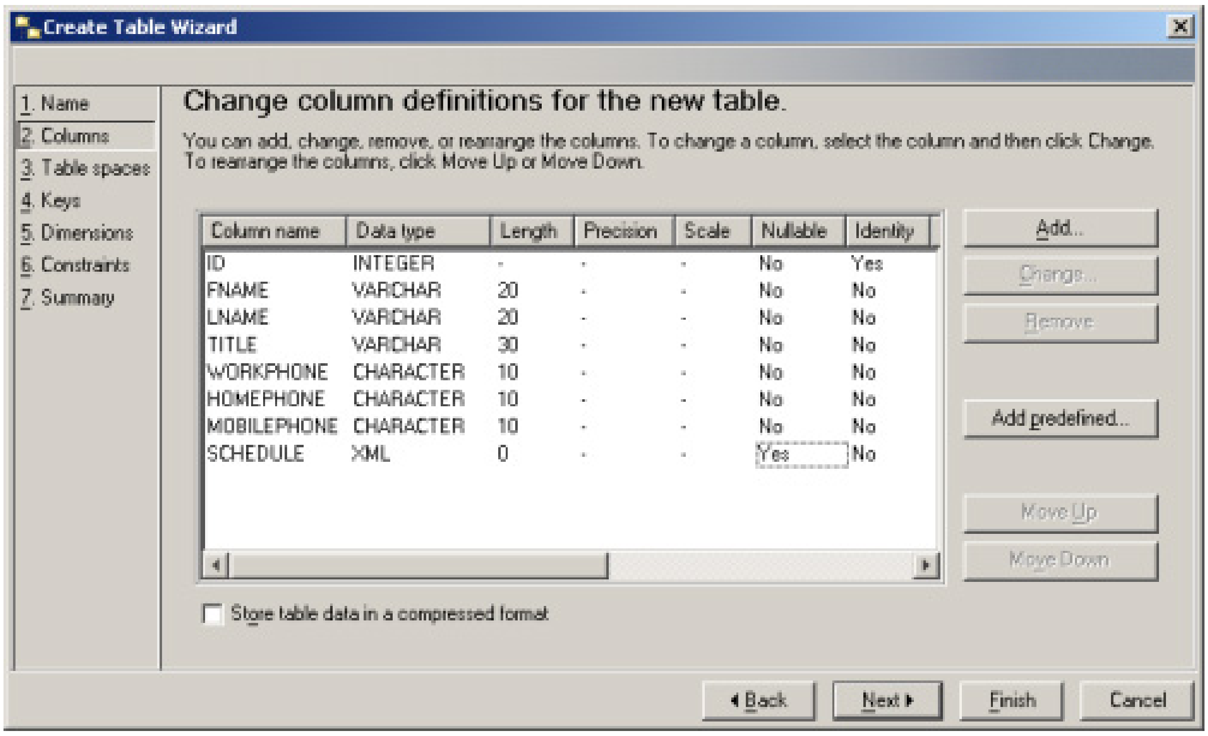
To execute the **CREATE TABLE** statement, simply click on the **green arrow** on the graphical toolbar menu or **Selected -> Execute** on the menu bar.

(Option 2) DB2 Control Center - CREATE TABLE

Within the **DB2 Control Center**, expand **All Databases** -> **Test**. Right-click on **Tables** under **Test** and choose **Create**. Set **Table Schema** to **USER1** and **Table name** to **CONTACT_SCHEDULES**. Click **Next** to get to the **Columns** section.

Add columns using the same structure as Listing 1 and you should end up with Figure 3. You can ignore the primary key for now, and be sure to check **Identity** in the **Value generation** section when creating the **ID** column. Also ensure that the Nullable attribute is not checked for the ID column, as primary key columns are not allowed to contain **NULL** values.

Figure 4. Create table wizard



Change column definitions for the new table.

You can add, change, remove, or rearrange the columns. To change a column, select the column and then click Change. To rearrange the columns, click Move Up or Move Down.

Column name	Data type	Length	Precision	Scale	Nullable	Identity
ID	INTEGER	-	-	-	No	Yes
FNAME	VARCHAR	20	-	-	No	No
LNAME	VARCHAR	20	-	-	No	No
TITLE	VARCHAR	30	-	-	No	No
WORKPHONE	CHARACTER	10	-	-	No	No
HOMEPHONE	CHARACTER	10	-	-	No	No
MOBILEPHONE	CHARACTER	10	-	-	No	No
SCHEDULE	XML	0	-	-	Yes	No

On step 4 (Keys) within the wizard, define the primary key for **ID** by clicking **Add Primary**. You can leave the primary key name as the default or enter a name of your choice. Click **Finish**.

Section 5. Step 3. Storing XML data using DB2 INSERT

The `INSERT` statement can be used to store relational and XML data in a DB2 Viper database. The XML data can be provided from any supported application programming language or it can be provided directly within a DB2 CLP script. This tutorial uses the **DB2 Command Editor** to store some initial schedules in the **TEST** database.

Now insert a record into the **CONTACT_SCHEDULES** table for *Grant Hutchison* and *Gerald Leung*.

For convenience, the INSERT statements have been provided in the following file: `c:\temp\ViperIntro\listings\listing2.txt`

Listing 2. Inserting XML data interactively

```
INSERT INTO user1.contact_schedules
(fname, lname, title, workphone, homephone, mobilephone, schedule)

VALUES ('Grant', 'Hutchison', 'DB2 Marketing Manager','0123456789', '1234567890',
'2345678901',

XMLPARSE
( DOCUMENT
'<schedule>
```

```
        <activity>
        <activityDate>
                <start>2006-05-07</start>
                <end>2006-05-11</end>
        </activityDate>
        <name>IDUG</name>
        <preferredContact>mobilephone</preferredContact>
        <address>
                <city>Tampa</city>
                <country>USA</country>
        </address>
        </activity>
</schedule>' STRIP WHITESPACE)
),
('Gerald', 'Leung', 'Developer','0123456789', '1234567890', '2345678901',
XMLPARSE
( DOCUMENT
'<schedule>
        <activity>
        <activityDate>
                <start>2006-05-07</start>
                <end>2006-05-11</end>
        </activityDate>
        <name>IDUG</name>
        <preferredContact></preferredContact>
        <address>
                <city>Tampa</city>
                <country>USA</country>
        </address>
        </activity>
</schedule>' STRIP WHITESPACE)
);
```

Note that supplying the XML data inline (as shown in Listing 2) requires you to invoke the **XMLPARSE** function to convert the document from a character data type to XML. In this case, the input document was quite simple. If the document was large or complex, it would be impractical to type the XML data into the INSERT statement as shown. In most cases, you'd write an application to insert the data using a host variable or a parameter marker. However, since this is an introductory tutorial, we won't be discussing application development topics in detail. Instead, we'll discuss another option for populating DB2 XML columns with data -- using the DB2 IMPORT facility.

Section 6. Step 4. Querying XML data using SQL and XQUERY

XQuery is a new language used for querying XML data. XQuery and SQL can be used in the same query to a database, as seen in the next example. In the next query, you will use a SQL/XML statement. The part of the query outside of **xmlexists()** is not case sensitive because that part is SQL. The part of the query inside of **xmlexists()** is case sensitive because it is XQuery, so this statement has XQuery embedded in an SQL statement.

Listing 3. The following query retrieves all of the attendees of the "IDUG" event:

```
SELECT *
FROM user1.contact_schedules
WHERE xmlexists('$s[schedule/activity/name="IDUG"]' PASSING schedule AS "s");
```

The query from Listing 3 should retrieve the two records you just inserted. The **xmlexists** function is using an XPath expression to determine which records should be returned.

The next statement has SQL embedded in an XQuery statement. Any XQuery statement must start with the keyword "**XQUERY**". As usual, the part inside sqlquery() is an SQL statement, so that part is not case sensitive, but the part outside of sqlquery() is XQuery, so that part is case sensitive. The following query tells us which activity Grant Hutchison is attending on 2006-05-08:

Listing 4. Searching for the schedule of "Grant Hutchison"

```
XQUERY db2-fn:sqlquery(
"SELECT schedule
FROM user1.contact_schedules
WHERE fname='Grant' AND lname='Hutchison'"
)/schedule/activity[activityDate/start <= "2006-05-8"][activityDate/end >= "2006-05-08"];
```

The query in Listing 4 returns all the activities for Grant Hutchison on May 8th, 2006. This query uses a combination of relational predicates and XQuery to limit the result to a specific person and date.

Relational condition (predicate)
`WHERE fname = 'Grant' AND lname = 'Hutchison'`

XQuery condition
`schedule/activity[activityDate/start <= "2006-05-08"][activityDate/end >= "2006-05-08"]`

You will examine many more XQuery examples in Step 6.

Section 7. Step 5. Using IMPORT to populate DB2 tables with XML

If you already have your XML data in files, the DB2 IMPORT command provides a convenient method of populating your DB2 tables with relational and XML data. The IMPORT command can use various import control file formats. In this example, you will use a delimited text file. The default delimiter for DB2 IMPORT is the comma (','), but this can be changed. In this example, you will use comma-separated values and a special reference technique for the XML data for the **schedule** column.

You can create the delimited ASCII file using the text editor of your choice. Each line in your file represents a row of data to be imported into your table. If your line contains an **XML Data Specification (XDS)**, IMPORT will read the XML data from the referenced file. For example, the first line in Listing 5 contains information for

Raymond Collins, including his name, title, and phone numbers.

Listing 5. Sample delimited ASCII file for input to DB2 IMPORT

```
Raymond,Collins,Project Manager,0041775644,5661833101,2502307836,
<XDS FIL='schedule3.xml' />
Mark,Peterson,CEO,3580218673,2053503260,4881565066,
<XDS FIL='schedule4.xml' />
. . .
```

With your XML files and delimited ASCII files available, you're now ready to use the DB2 IMPORT utility. The DB2 IMPORT commands can be used to add more data to your existing **CONTACT_SCHEDULES** table.

Listing 6. Importing data into the "clients" table

```
IMPORT FROM 'c:\temp\ViperIntro\schedules_import.del' OF DEL
XML FROM 'C:\temp\ViperIntro\schedules'
MODIFIED BY IDENTITYMISSING
INSERT INTO user1.contact_schedules;
```

Note: There are 1,000 XML documents in: `c:\temp\ViperIntro\schedules`

Now IMPORT all of these documents with a single DB2 command (as shown in Listing 6.)

Execute the command in Listing 6 from the DB2 Command Editor.

The schedules_import.del file contains references to the 1,000 data records, including XML files.

Section 8. Step 6. Additional XQuery examples

Let's take a quick look at some basic XQuery capabilities of DB2 Viper.

Listing 7. XQuery - Retrieve all schedules

```
xquery db2-fn:xmlcolumn('USER1.CONTACT_SCHEDULES.SCHEDULE');
```

The XQuery in Listing 7 can be used to retrieve all of the schedules stored in the **CONTACT_SCHEDULES** table within the **SCHEDULE** column. The **xmlcolumn()** function requires that the referenced table and column names is uppercase. The output of the query in Listing 7 is equivalent to the following SQL query:

`SELECT schedule from USER1.CONTACT_SCHEDULES`

You will now reduce the returned data set to only include the names of the cities of scheduled activities for each person.

Listing 8. XQuery - Retrieve all cities for each person

```
xquery
for $y in db2-fn:xmlcolumn('USER1.CONTACT_SCHEDULES.SCHEDULE')
        /schedule/activity/address/city
return $y;
```

You will now limit the results to only activities in the city of '**Toronto**'. There are 73 such scheduled entries in the table at this time.

Listing 9. XQuery - Retrieve all addresses for each person with an activity in Toronto

```
xquery
for $y in db2-fn:xmlcolumn('USER1.CONTACT_SCHEDULES.SCHEDULE')
        /schedule/activity/address[city='Toronto']
return $y;
```

Now let's determine what the events are occurring in Toronto and return the name of the event.

Listing 10. XQuery - Retrieve all activities for each person that takes place in Toronto

```
xquery
for $y in db2-fn:xmlcolumn('USER1.CONTACT_SCHEDULES.SCHEDULE')
        /schedule/activity
where $y/address[city='Toronto']
return $y/name;
```

Section 9. Step 7. Indexing an XML column of a table

Query performance on XML data can be improved using XML indexes. Let's compare the performance of an XQuery over the 1,000 records without an index and with an index.

Listing 11 is a query to search for activity details for people attending the 'IDUG' event. Enter Listing 11 into the Command Editor and click the **Execute and Access plan** button . This will execute the query and provide a graphical representation of the query's access plan.

Listing 11. Query on an XML node

```
xquery
for $y in db2-fn:xmlcolumn('USER1.CONTACT_SCHEDULES.SCHEDULE')
        /schedule/activity
where $y[name='IDUG']
return $y
```

Click the **Access Plan** tab within the Command Editor to view the DB2 access plan. Note that a table scan (TBSCAN) is performed on the `USER1.CONTACT_SCHEDULES` table.

You will create an index on the **name** node of the XML data to improve query performance. In the Control Center, expand the TEST database and right-click on **Indexes** and choose **Create** -> **Index**.

Fill in the fields with following information:

```
Table schema: USER1
Table name: CONTACT_SCHEDULES

XML column options: Yes (create the new index on an XML column)
```

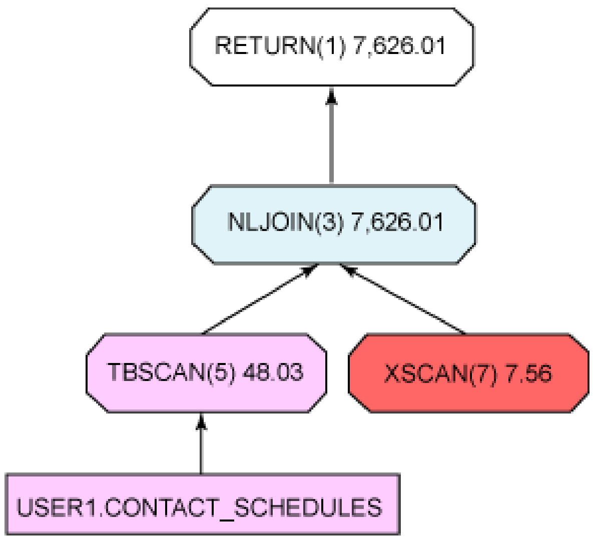
Click **Next** to proceed.

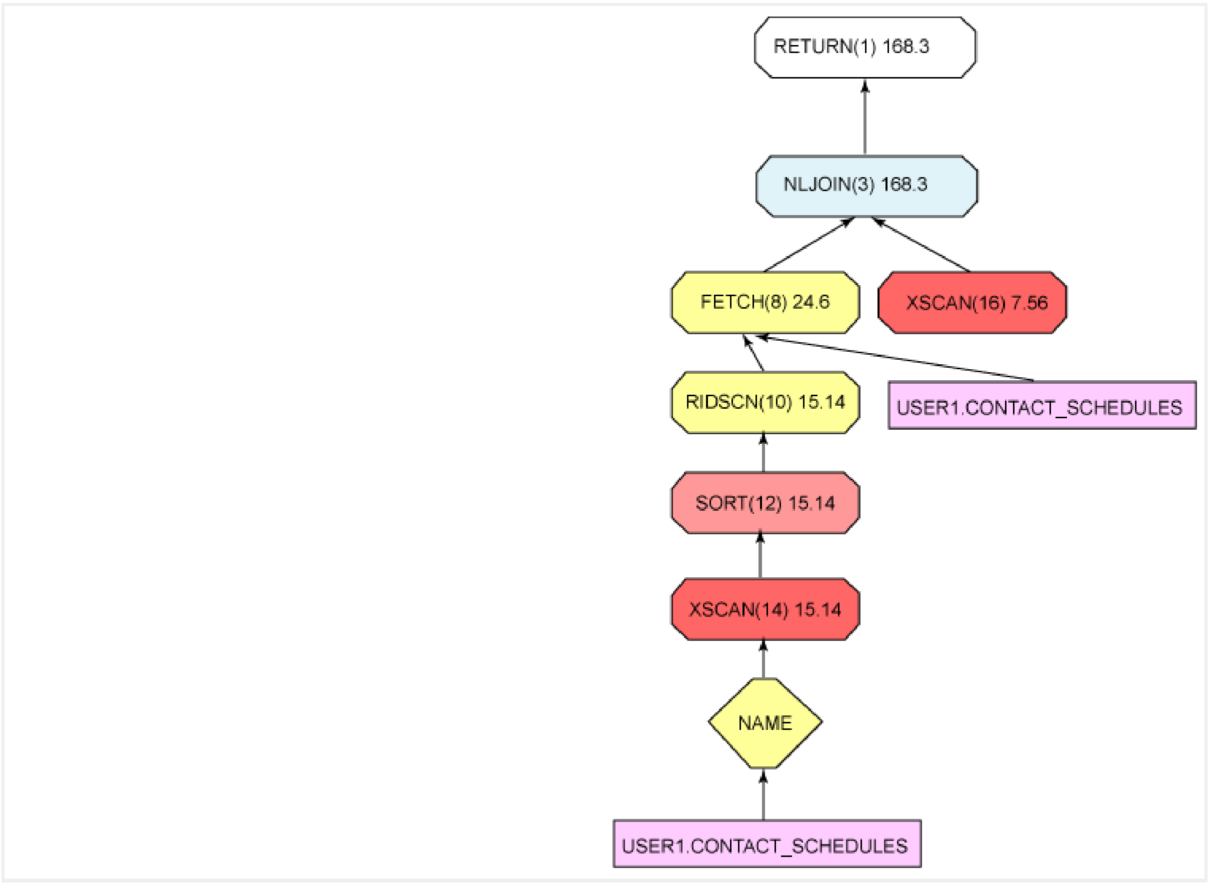
Highlight the SCHEDULE column on the left side and click on the right arrow to move it to the right side. Click **Next** to proceed.

Expand the **schedule** node in the bottom area if not already done. Right-click on the **name** node from /schedule/activity and choose **Add index** -> **This element only**. Click **Finish** to complete the creation of the index for the XML data.

We will execute the **RUNSTATS** operation on the USER1.CONTACT_SCHEDULES table to ensure that the new XML index will be considered by DB2. From the Control Center, click on **Tools** -> **Customize Control Center**, choose **Advanced**, and click **OK**. From the Control Center, select the USER1.CONTACT_SCHEDULES table and right-click and initiate the **RUNSTATS** operation on the object. Reissue the query from Listing 11 and click the **Execute and Access plan** button. Notice that there is an index scan (XISCAN) performed on the **name** index of USER1.CONTACT_SCHEDULES Also notice that the **Total cost** has been significantly lowered.

The next two figures show the Access plan before and after the index has been created.

Query in Listing 11 before indexes	Query in Listing 11 after indexes
 RETURN(1) 7,626.01 NLJOIN(3) 7,626.01 TBSCAN(5) 48.03 XSCAN(7) 7.56 USER1.CONTACT_SCHEDULES	



Section 10. Step 8. Creating and querying a DB2 view

You can create views over tables containing XML data, just as you can create views over tables containing traditional SQL data types. The example in Listing 12 creates a view of all the Web Developers without a scheduled activity TODAY.

Listing 12. Creating a view that contains XML data

```
CREATE VIEW user1.webDevToday(numAvailable) AS
SELECT COUNT(id)
FROM user1.contact_schedules
WHERE title='Web Developer' AND
        NOT XMLEXISTS('$c/schedule/activity/activityDate
        [start < current-date()][end > current-date()]' PASSING schedule AS "c");
```

Listing 12 is using an XQuery expression embedded in an SQL statement. Views over XML data can simplify data access for all developers as the XQuery details can be stored within the DB2 server and not necessarily exposed to each user or developer.

Listing 13. How many Web Developers are available today?

```
SELECT numAvailable FROM user1.webDevToday;
```

Congratulations! Your DB2 Viper test drive tour is now complete.

Hopefully, you now have a sense of the hybrid nature of data storage and access provided with the new DB2 Viper data server. If you would like to learn more, join the growing DB2 Viper community on the Web.

Acknowledgments

Thanks to Antonio Cangiano and Melody Ng for reviewing this tutorial.

Resources

Learn

- Read more about XQuery.
- Read more about XPath.
- "DB2 Express-C, the developer-friendly alternative" (developerWorks, February 2006) introduces you to intermediate DB2 Express-C administrative tools, tasks, and an overview of developer options.
- "Develop Java applications for DB2 XML data" (developerWorks, May 2006) takes you through the development of a Java application that uses DB2 Viper's XML capabilities.
- "Use DB2 native XML with PHP" compares PHP applications developed using DB2 Viper versus using another database management server that does not have XML capabilities.
- Learn about DB2 Express-C, the no-charge version of DB2 Express Edition for the community.

Get products and technologies

- Download a free trial version of DB2 Viper data server.
- Build your next development project withIBM trial software, available for download directly from developerWorks.
- Now you can use DB2 for free. Download DB2 Express-C, a no-charge version of DB2 Express Edition for the community that offers the same core data features as DB2 Express Edtion and provides a solid base to build and deploy applications.

Discuss

- Participate in the discussion forum for this content.
- Go to IBM DB2 Express Forum to ask questions to IBM's DB2 Community Team and to other users of DB2 Express-C.

About the authors

Gerald Leung
Gerald Leung is a computer science co-op student at the University of Toronto. He expects to complete his B. Sc. in August 2007. Gerald is working at IBM Toronto lab's Developer Initiatives department to take advantage of his interest in experimenting, configuring, and testing client/server technologies.

Grant Hutchison
Grant Hutchison is a Senior Product Manager responsible for supporting the application development community for IBM database servers, including DB2 UDB and Cloudscape/Apache Derby. Grant has a graduate degreee in Software Engineering from the University of Waterloo and a B.Sc.from Wilfrid Laurier University. He co-authored the first DB2 UDB Certification Guide (1996) and has held various technical and management roles within the DB2 database team in the IBM Toronto lab over the past 14 years.

Trademarks