

# SQL and XQuery tutorial for IBM DB2, Part 2: Basic queries

## The fundamentals of SQL queries

Skill Level: Introductory

[Pat Moffatt \(pmoffatt@ca.ibm.com\)](mailto:pmoffatt@ca.ibm.com)

Information Management Program Manager, IBM Academic Initiative  
IBM

[Bruce Creighton \(bcreight@ca.ibm.com\)](mailto:bcreight@ca.ibm.com)

Skills Segment Planner  
IBM

[Jessica Cao](#)

Training Tools Developer  
IBM

03 Aug 2006

Through a series of simple examples, this tutorial illustrates how to retrieve data from an IBM® DB2® database with standard SQL SELECT statements. This tutorial describes how to retrieve rows from a relational database table, retrieve specific columns, retrieve specific rows, reform logical operations on retrieved data, and use wildcard characters in search conditions. This tutorial is Part 2 of the [SQL & XQuery tutorial for IBM DB2](#) series.

## Section 1. Before you start

### About this series

This tutorial series teaches basic to advanced SQL and basic XQuery topics and

shows how to express commonly asked business questions as database queries by using SQL queries or XQueries. Developers and database administrators can use this tutorial to enhance their database query skills. Academic Initiative members can use this tutorial series as a part of their database curriculum.

All the examples in this document are based on **Aroma**, a sample database that contains sales data for coffee and tea products sold in stores across the United States. Each example consists of three parts:

- A business question, expressed in everyday language
- One or more example queries, expressed in SQL or XQuery
- A table of results returned from the database

This guide is designed to allow participants to learn the SQL language and XQuery. As with any learning, it is important to supplement it with hands-on exercises. This is facilitated by the table definitions and data.

For students using this as part of an academic class, obtain from your instructor the instructions to connect to the Aroma database and learn about any differences between the guide and your local set up.

This tutorial was written for DB2 Express-C 9 for UNIX®, Linux® and Windows® (formerly known as Viper).

## About this tutorial

Using a series of simple examples, this tutorial illustrates how to retrieve data from an IBM DB2 database with standard SQL SELECT statements.

This tutorial describes how to:

- Retrieve rows from a relational database table
- Retrieve specific columns from a relational database table
- Retrieve specific rows from a relational database table
- Perform logical operations on retrieved data
- Use wildcard characters in search conditions

## Connecting to a database

You need to connect to a database before you can use SQL statements to query or

manipulate data. The `CONNECT` statement associates a database connection with a user name.

Find out from your instructor the database name that you will need to be connected to. For this series, the database name is *aromadb*.

To connect to the *aromadb* database, type the following command in the DB2 command line processor:

```
CONNECT TO aromadb USER userid USING password
```

Replace "userid" and "password" with the user ID and password that you received from your instructor. If no user ID and password are required, simply use the following command:

```
CONNECT TO aromadb
```

The following message tells you that you have made a successful connection:

```
Database Connection Information
Database server      = DB2/NT 9.0.0
SQL authorization ID = USERID
Local database alias = AROMADB
```

Once you are connected, you can start using the database.

---

## Section 2. The six clauses of the `SELECT` statement

There are six clauses that can be used in an SQL statement. These six clauses are `SELECT`, `FROM`, `WHERE`, `GROUP BY`, `HAVING`, and `ORDER BY`. Clauses must be coded in a specific sequence. Let's briefly discuss each here. You will learn more about them as you encounter them later in the tutorial.

- |                          |
|--------------------------|
| 1. <code>SELECT</code>   |
| 2. <code>FROM</code>     |
| 3. <code>WHERE</code>    |
| 4. <code>GROUP BY</code> |
| 5. <code>HAVING</code>   |

## 6. ORDER BY

**Note:** *column name(s)* are more correctly referred to as *elements*, because the SELECT statement displays both columns that exist in the table and columns that may be generated by SQL as a result of a query.

### Example query

```
SELECT perkey, sum(dollars)
FROM aroma.sales
WHERE perkey < 50
GROUP BY perkey
HAVING sum(dollars) > 8000
ORDER BY perkey;
```

### About the query

The SELECT clause is where you list the columns you're interested in. The SELECT displays what you put here. There are other items you can put in a SELECT that will be explained later. In the example, the **perkey** column, as well as the sum of the **dollar** column, are selected.

The FROM clause indicates the table you're getting your information from. You can list more than one table. The number of tables you could list is specific to your operating system. In the example, both columns are selected from the **Sales** table.

SELECT and FROM are required; the rest of these clauses are optional and serve to filter or limit, aggregate or combine, and control the sort.

The WHERE clause is where you indicate a condition. This helps you filter unwanted data from the results. WHERE gives you a subset of the rows in a table. In the example, only rows with a **perkey** value of less than 50 are selected.

GROUP BY allows you to group your data to achieve more meaningful results. Instead of getting a total sum of the dollar sales for all the rows selected, you can break down sales by **perkey** to get the daily sales total. This is done in the example by indicating GROUP BY **perkey**.

HAVING puts a condition on your groups. In the example, only those days that have a total dollar amount greater than 8,000 are returned.

ORDER BY orders your result rows. You can choose to order results by ASC (ascending) or DESC (descending) order. The default is ASC.

The SELECT statement is the most common usage of data manipulation language (DML). Other DML statements (UPDATE, INSERT, and DELETE) and the other two components of SQL (data definition language and control language) are discussed

in Part 6 of this series.

---

## Section 3. Using the SELECT statement to retrieve data

### Question

What regions, districts, and markets are defined in the Aroma database?

### Solution

The first step is to identify the table that contains the rows and columns that can answer this question. In Part 1, there is a diagram of the tables in the AROMADB. Looking at the lower left-hand section of the diagram, there is a table called Market, which contains columns titled mktkey, hq\_city, hq\_state, district, and region. Listing the contents of this table will answer our question and can be done with an SQL SELECT statement.

### Example query

```
SELECT * FROM aroma.market;
```

### Result

Mktkey	Hq_city	Hq_state	District	Region
1	Atlanta	GA	Atlanta	South
2	Miami	FL	Atlanta	South
3	New Orleans	LA	New Orleans	South
4	Houston	TX	New Orleans	South
5	New York	NY	New York	North
6	Philadelphia	PA	New York	North
7	Boston	MA	Boston	North
8	Hartford	CT	Boston	North
9	Chicago	IL	Chicago	Central
10	Detroit	MI	Chicago	Central
11	Minneapolis	MN	Minneapolis	Central
12	Milwaukee	WI	Minneapolis	Central

14	San Jose	CA	San Francisco	West
15	San Francisco	CA	San Francisco	West
16	Oakland	CA	San Francisco	West
17	Los Angeles	CA	Los Angeles	West
19	Phoenix	AZ	Los Angeles	West

## Retrieving data: SELECT statement

You use SELECT statements to retrieve columns and rows of data from database tables; to perform arithmetic operations on the data; and to group, order, or group and order the data. In most cases, a SELECT statement contains a simple query expression that begins with the SELECT keyword and is followed by one or more clauses or subclauses.

The most basic SELECT statement contains two keywords, SELECT and FROM:

```
SELECT column name(s)
FROM table name(s)
```

column name(s)

Column names or SQL expressions are separated by commas. An asterisk (\*) can also be used to list all column names that occur in the list of table name(s)

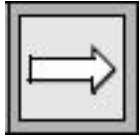
table name(s)

Table names can be a list of tables or a SELECT statement. The table names are separated by commas. Referenced table(s) must contain the column name(s) that are referenced after the SELECT keyword. In this example, the FROM statement refers to AROMA.MARKET. More will be discussed about this naming convention later in Part 3, *Table Names and Schemas*. In this case, the FROM clause refers to the qualified table name. The first part is the schema and the second part is the table name. Schemas are useful to group like tables and other DB2 objects. In the AROMADB database, all the tables were set up with AROMA as the schema.

## Alternate select statement

```
SELECT mktkey, hq_city, hq_state, district, region FROM aroma.market
```

The above query produces the same result as `SELECT * FROM aroma.market;` however, rather than using the asterisk (\*) to list all columns, the column names are identified.



Important: `SELECT` and `FROM` (and all other words shown in uppercase) are reserved SQL keywords. These words must be used exactly as defined by the SQL standard. We use the uppercase format in this document to make the keywords easy to see; SQL is not case sensitive, so keywords can be written in uppercase or lowercase.

## Usage notes

The semicolon (;) at the end of each example in this guide is not a mandatory part of SQL syntax. By convention, the SQL language ignores extra spaces, tabs, and end-of-line indicators. The semicolon, then, is an end-of-statement marker and not necessary if you execute only a single statement. For situations where you create a scripted series of SQL statements, the semicolon provides a definite end to each statement. Depending on the interactive SQL tool you use to enter queries, you may or may not need to specify such a marker. When using the Command Line Processor to execute multiple SQL statements, a semicolon is required to indicated the end of each statement.

---

## Section 4. Using `SELECT` list to retrieve specific columns

### Question

Which districts and regions are defined in the Aroma database?

### Example query

```
SELECT district, region
FROM aroma.market;
```

### Result

District	Region
Atlanta	South
Atlanta	South
New Orleans	South
New Orleans	South
New York	North
New York	North
Boston	North
Boston	North
Chicago	Central
Chicago	Central
Minneapolis	Central
Minneapolis	Central
San Francisco	West
San Francisco	West
San Francisco	West
Los Angeles	West
Los Angeles	West

## Retrieving specific columns

By naming the columns in the SELECT list of a SELECT statement, you can retrieve a specific set of columns from any table. Columns are returned in the order you specify in the SELECT list.

## About the query

The example query requests a list of districts and their corresponding regions from the **Market** table.

### Usage notes

Although column names in the SELECT list must be defined in the tables referenced in the FROM clause, other expressions can also occur in the SELECT list. Several examples of such expressions are discussed later in this series.

When the SELECT list does not include all the columns in a table, a query might return duplicate rows, as in the previous example query. You can eliminate the

duplicates by using the **DISTINCT** keyword. For example, the following query returns only the names of distinct districts and regions in the **Market** table:

```
SELECT DISTINCT district, region
FROM aroma.market;
```

District	Region
Chicago	Central
Minneapolis	Central
Boston	North
New York	North
Atlanta	South
New Orleans	South
Los Angeles	West
San Francisco	West

---

## Section 5. Using the WHERE clause to retrieve specific rows

### Question

What products are sold without packaging?

### Example query

```
SELECT prod_name, pkg_type
FROM aroma.product
WHERE pkg_type = 'No pkg';
```

### Result

Prod_Name	Pkg_Type
Veracruzano	No pkg
Xalapa Lapa	No pkg

Colombiano	No pkg
Espresso XO	No pkg
La Antigua	No pkg
Lotta Latte	No pkg
Cafe Au Lait	No pkg
NA Lite	No pkg
Aroma Roma	No pkg
Demitasse Ms	No pkg
Darjeeling Number 1	No pkg
Darjeeling Special	No pkg
Assam Grade A	No pkg
Assam Gold Blend	No pkg
Earl Grey	No pkg
English Breakfast	No pkg
Irish Breakfast	No pkg
Special Tips	No pkg
Gold Tips	No pkg
Breakfast Blend	No pkg
Ruby's Allspice	No pkg
Coffee Mug	No pkg
Travel Mug	No pkg
Aroma t-shirt	No pkg
Aroma baseball cap	No pkg

## Retrieving specific rows: WHERE clause

By including a set of logical conditions in a query, you can retrieve a specific set of rows from a table. Logical conditions are declared in the WHERE clause. If a row satisfies the conditions, the query returns the row; if not, the row is discarded. Logical conditions are also called search conditions, predicates, constraints, or qualifications.

### The WHERE clause

```
SELECT column name(s) FROM table name(s) [WHERE  
search_condition];
```

```
search_condition
```

```
This condition evaluates to true or false.
```

The square brackets ([ ]) indicate that the WHERE clause is optional.

## About the query

The example query retrieves and displays the names of products that are not prepacked or packaged. IBM DB2 9 evaluates the following condition for each row of the **Product** table and returns only those rows that satisfy the condition:

```
pkg_type = 'No pkg'
```

## Usage notes

A character literal is a character string enclosed within single quotes. To represent a single quote in a character literal, use two single quotes ("). For example:

```
'Scarlet O''Hara'
```

Character literals must be expressed as exactly stored in the database, in either uppercase or lowercase. For example, the following condition:

```
class_type = 'Bulk_beans'
```

is false when the referenced column contains the following string:

```
'BULK_beans'
```

Set functions are not allowed in the WHERE clause. For more information about set functions, refer to Part 3, *Using the ORDER BY clause*.

---

## Section 6. Using AND, NOT, and OR connectives to create complex conditions

### Question

What cities and districts are located in the southern or western regions?

## Example query

```
SELECT hq_city, district, region
FROM aroma.market
WHERE region = 'South' OR region = 'West';
```

## Result

Hq_city	District	Region
Atlanta	Atlanta	South
Miami	Atlanta	South
New Orleans	New Orleans	South
Houston	New Orleans	South
San Jose	San Francisco	West
San Francisco	San Francisco	West
Oakland	San Francisco	West
Los Angeles	Los Angeles	West
Phoenix	Los Angeles	West

## Specifying compound conditions: AND, OR, NOT and parentheses

To refine the selecting of rows, search conditions can be joined and the order of evaluation can be forced. A search condition specifies a condition that is "true," "false," or "unknown" about a given row. The result of a search condition is derived by application of the specified logical operators (AND, OR, NOT) to the result of each specified predicate. If logical operators are not specified, the result of the search condition is the result of the specified predicate. AND and OR are defined in the following table, which P and Q are any predicates:

### Truth tables for AND and OR

P	Q	P AND Q	P OR Q
True	True	True	True
True	False	False	True
True	Unknown	Unknown	True
False	True	False	True

False	False	False	False
False	Unknown	False	Unknown
Unknown	True	Unknown	True
Unknown	False	False	Unknown
Unknown	Unknown	Unknown	Unknown

NOT(true) is false, NOT(false) is true, and NOT(unknown) is unknown.

Search conditions within parentheses are evaluated first. If the order of evaluation is not specified by parentheses, NOT is applied before AND, and AND is applied before OR. The order in which operators at the same precedence level are evaluated is undefined to allow for optimization of search conditions.

## Section 7. Using the AND, OR, and parentheses in a WHERE clause

### Question

Which large or small Aroma Coffee and Tea Company stores are located in Los Angeles or San Jose?

### Example query

```
SELECT store_type, store_name, city
FROM aroma.store
WHERE (store_type = 'Large' OR store_type =
'Small')
      AND (city = 'Los Angeles' OR city = 'San
Jose');
```

### Result

Store_Type	Store_Name	City
Large	San Jose Roasting Company	San Jose
Large	Beaches Brew	Los Angeles
Small	Instant Coffee	San Jose

## Specifying complex search conditions

Search conditions, especially those written for decision-support analysis, can become complex. Though constructed from simple conditions that use the search conditions AND, OR, and NOT, complex conditions might be difficult to understand. Fortunately, SQL is free-form, so the logical structure of these conditions can be shown by using tab characters, blanks, and newline characters to define white space and logical relationships.

### About the query

The example query retrieves and displays the names of Aroma Coffee and Tea Company stores that are both large or small and located in Los Angeles or San Jose.

The parentheses in this query are essential because the AND connective has a higher precedence than the OR connective. If you remove the parentheses, the query returns a different result table.

### Example query (without parentheses)

```
SELECT store_type, store_name, city
FROM aroma.store
WHERE store_type = 'Large' OR store_type = 'Small'
      AND city = 'Los Angeles' OR city = 'San Jose';
```

## Result

Store_Type	Store_Name	City
Large	San Jose Roasting Company	San Jose
Large	Beaches Brew	Los Angeles
Small	Instant Coffee	San Jose
Large	Miami Espresso	Miami
Large	Olympic Coffee Company	Atlanta

### Usage notes

A query retrieves and displays any data that is not explicitly excluded by its search condition, and a query with only a few general conditions can return an enormous number of rows.

Whenever you doubt how the server might evaluate a compound condition, explicitly

group the conditions with parentheses to force the order of evaluation.

---

## Section 8. Using the Greater-Than (>) and Less-Than or Equal-To (<=) operators

### Question

Which cities and districts are identified by **Mktkey** values that are greater than 4 and less than or equal to 12?

### Example query

```
SELECT mktkey, hq_city, hq_state, district
FROM aroma.market
WHERE mktkey > 4
      AND mktkey <= 12;
```

### Result

Mktkey	Hq_City	Hq_State	District
5	New York	NY	New York
6	Philadelphia	PA	New York
7	Boston	MA	Boston
8	Hartford	CT	Boston
9	Chicago	IL	Chicago
10	Detroit	MI	Chicago
11	Minneapolis	MN	Minneapolis
12	Milwaukee	WI	Minneapolis

## Using comparison operators

Conditions evaluate to true or false and can be expressed with comparison operators or comparison predicates. SQL contains the following comparison operators:

Operator	Name
=	equal
<	less than
>	greater than
<>	not equal
>=	greater than or equal
<=	less than or equal

## About the query

The example query retrieves and displays all cities and districts whose Mktkey is greater than 4 but less than or equal to 12.

The **Mktkey** column contains integer values, which are comparable to other numeric values. If you compare an integer to a character, however, the server returns an error message:

```
SELECT mktkey, hq_city, hq_state, district
FROM aroma.market
WHERE mktkey > '4';
```

```
[IBM][CLI Driver][DB2/LINUX] SQL0401N  The data types of the operands
for the operation ">" are not compatible.  SQLSTATE=42818
```

You can obtain help interpreting error messages, and identifying corrective action that can be taken, by using the DB2 Information Center. The Center was introduced in Part 1 of this series.

## Usage notes

Conditions must compare values of comparable data types. If you attempt to compare unlike data types, the server returns either an error message or an incorrect result. Comparison operators can be used to compare one character string with another, as the following legal condition illustrates:

```
(city > 'L')
```

For more information about comparable data types, refer to the [SQL Reference Guide](#).

---

## Section 9. Using the IN comparison predicate

## Question

What cities are in the Chicago, New York, and New Orleans districts?

### Example query

```
SELECT hq_city, hq_state, district
FROM aroma.market
WHERE district IN
('Chicago', 'New York', 'New Orleans');
```

## Result

Hq_City	Hq_State	District
New Orleans	LA	New Orleans
Houston	TX	New Orleans
New York	NY	New York
Philadelphia	PA	New York
Chicago	IL	Chicago
Detroit	MI	Chicago

## Using comparison predicates

A simple condition can be expressed with the following SQL comparison predicates:

Predicate
BETWEEN expression1 AND expression2
LIKE pattern
IN (list)
IS NULL
IS NOT NULL
ALL
SOME or ANY
EXISTS

Examples of the ALL, SOME or ANY, and EXISTS predicates are presented in Part 5 of this series.

For syntax descriptions and examples of all these predicates, as well as detailed definitions of simple and complex expressions, refer to the [SQL Reference Guide](#) .

### About the query

The example query lists all the cities in the Chicago, New York, and New Orleans districts. It could also be written with the *equals* comparison operator (=) and a set of OR conditions:

```
WHERE district = 'Chicago'
       OR district = 'New York'
       OR district = 'New Orleans'
```

### Usage notes

Strive to write logical sets of conditions that are simple, easy to understand, and easy to maintain. Always clarify the logical structure of your compound conditions with ample white space, define logical blocks by indentation, and force evaluation precedence with parentheses.

---

## Section 10. Using the Percent Sign (%) wildcard

### Question

Which cities are in districts that begin with the letters Min?

### Example query

```
SELECT district, hq_city
FROM aroma.market
WHERE district LIKE 'Min%';
```

### Result

District	Hq_City
Minneapolis	Minneapolis
Minneapolis	Milwaukee

## Using wildcard characters

Previous queries have expressed conditions that match complete character strings. With the LIKE predicate and the two wildcard characters, the percent sign (%) and the underscore (\_), you can also express conditions that match a portion of a character string (a substring).

The percent (%) wildcard matches any character string. For example:

- `like 'TOT%'` is true for any string that begins with 'TOT'.
- `like '%ZERO%'` is true for any string that contains the text 'ZERO'.
- `like '%FRESH'` is true for any string that ends with 'FRESH' and does not contain trailing blanks. Trailing blanks in character data are deemed significant when LIKE constraints are applied.

The percent sign (%) can also be used to search for a null character string--zero (0) characters.

The underscore wildcard (\_) matches any one character in a fixed position. For example:

- `like '_EE_'` is true for any four-letter string whose two middle characters are 'EE'.
- `like '%LE_N%'` is true for any string that contains the pattern 'LE\_N'. The strings 'CLEAN', 'KLEEN', and 'VERY KLEEN' all match this pattern.

### About the query

The example query retrieves the names of all districts that begin with the characters 'Min' and lists the cities in these districts. The wildcard percent sign (%) allows for any character combination (including blank spaces) after the 'n' in 'Min', but characters that precede the 'n' must match the character pattern exactly as stored.

### Usage notes

A LIKE condition is true when its pattern matches a substring in a column. If the pattern contains no wildcard characters, the pattern must match the column entry exactly.

For example, the following condition is true only when the column entry contains the character string APRIL and nothing else:

```
month LIKE 'APRIL'
```

In other words, this condition is equivalent to:

```
month = 'APRIL'
```

The LIKE predicate can be used only on columns that contain character strings.

---

## Section 11. Naming columns using the AS clause

### Question

Which cities are in districts that begin with the letters Min? Name the *hq\_city* column to make more sense.

### Example query

```
SELECT district, hq_city AS City
FROM aroma.market
WHERE district LIKE 'Min%';
```

### Result

District	City
Minneapolis	Minneapolis
Minneapolis	Milwaukee

### Naming expressions: AS

The optional AS clause lets you assign a meaningful name to an expression, which makes referring back to the expression easier. When using the AS clause, there are rules on allowable names. In general, valid names must begin with a letter, be no longer than 128 characters, have no blank spaces, and cannot be a SQL keyword. An SQL keyword would be SELECT, FROM, WHERE, and etc. For more detail on SQL keywords, refer to the [SQL Reference Guide](#).

For example, the following AS clause assigns the alias **hq\_city** to the **City** column:

```
hq_city AS City
```

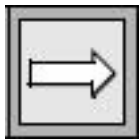
You can assign the alias without using the keyword AS, but it may not be as obvious when you look at your query:

```
hq_city City
```

You will note that regardless of capitalization in the expression, column headers are displayed in all uppercase.

### About the query

The example query returns the same result set as the previous query in this chapter; however, in this case, column aliases are assigned to create headings for the aggregated results.



Important: If the value contained in the column referenced by the column alias is the result of a set function, it cannot occur in the WHERE clause; however, it can occur in the HAVING clause. For more information on the HAVING clause, see Part 4 of this series.

## Section 12. Summary

### The SELECT statement

```
SELECT column name(s)
      FROM table name(s)
      [WHERE search_condition]
      [GROUP BY group_list]
      [HAVING search_condition]
      [ORDER BY order_list];
```

### Search conditions

( )	parentheses (force order of evaluation)
NOT	negation
AND	and
OR	or

## Comparison operators

=	equal
<	less than
>	greater than
<>	not equal
>=	greater than or equal
<=	less than or equal

## Comparison predicates

BETWEEN	expression1 AND expression2
LIKE	pattern
IN	(list)
IS	NULL
IS	NOT NULL

This tutorial discussed how to express many commonly asked business questions as SELECT statements and how to retrieve, group, and order data selected from relational tables.

Most questions discussed in this tutorial are easily expressed as standard SELECT statements and challenge neither the user nor SQL. The remaining tutorials of this series address more difficult questions, questions that require sequential processing, comparisons of aggregated values, more complex join specifications, or lengthy SELECT statements.

Before moving on to the next part, you should go to the DB2 Information Center and read the full online documentation on the SELECT statement. This will help you become familiar with the format of the help documentation and review what you have just learned.

## Downloads

Description	Name	Size	Download method
Aroma Database	Aroma_Data.zip	1MB	<a href="#">HTTP</a>

[Information about download methods](#)

# Resources

## Learn

- View this article series' "[Appendix A](#)" (developerWorks, August 2006).
- Read "[DB2 XML evaluation guide](#)" (developerWorks, June 2006), a step-by-step tutorial introducing the reader to the DB2 Viper data server on the Windows platforms using the XML storage and searching (SQL/XML, XQuery) capabilities available to support next-generation applications.
- Check out this article and "[Get off to a fast start with DB2 Viper](#)" (developerWorks, March 2006).
- Learn how to "[Query DB2 XML data with XQuery](#)" (developerWorks, April 2006).
- Learn how to "[Query DB2 XML data with SQL](#)" (developerWorks, March 2006).
- Read the [IBM Systems Journal](#) and celebrate 10 years of XML.
- Refer to the [SQL Reference, Vol 1](#) for additional information.
- Refer to the [SQL Reference, Vol 2](#) for additional information.
- Refer to the [DB2 information Center](#) for troubleshooting.
- Visit the [DB2 XML technical enablement space](#) for links to more than 25 papers on DB2 XML capabilities.

## Get products and technologies

- Download [DB2 Express-C](#), a no-charge data server for use in development and deployment of applications .

## Discuss

- [Participate in the discussion forum for this content.](#)
- Visit the [DB2 9 On-line Support Forum.](#)

## About the authors

### Pat Moffatt



Pat Moffatt is the Information Management Program Manager for the IBM Academic Initiative. Through the Academic Initiative program, she ensures that appropriate Information Management resources are made available to help faculty integrate Information Management software into their curriculum. To learn more about this program, visit

[www.ibm.com/university/data](http://www.ibm.com/university/data).

---

### Bruce Creighton



Bruce Creighton is a Skills Segment Planner in the Information Management Education Planning and Development department. In this role, he plans investment in educational content and balances the investment between areas where IBM can attain revenue and those where the requirement for skills development are important enough to provide free education.

---

### Jessica Cao



Jessica Cao is an Arts and Science and Computer Science student at McMaster University. She expects to complete her combined honours degree in April 2009. Jessica is working in IBM Toronto lab's DB2 Information Management Skills Channel Planning and Enablement Program to take advantage of her interest in programming, editing, and writing.

## Trademarks

IBM, DB2, Universal Database, OS/2, and pureXML are registered trademarks of IBM Corporation in the United States and other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.