November 17, 2011

Northeast Ohio Database Users Group

Key Application Enablement Features of DB2 for z/OS Versions 9 and 10

Robert Catterall
IBM
rfcatter@us.ibm.com

Before we begin…

- I describe in this presentation MOST of the SQL-related enhancements delivered in DB2 9 and DB2 10 for z/OS
- That's "most," as in "not all"
  - Time doesn't permit exhaustive coverage of the topic
  - For more information, see these red books (find them by searching on document number at www.ibm.com/redbooks):
    - DB2 9 for z/OS Technical Overview (SG24-7330)
    - DB2 10 for z/OS Technical Overview (SG24-7892)
- My focus here is on SQL enhancements that affect application development

Agenda (1)

- DB2 9 for z/OS
  - MERGE
  - SELECT FROM UPDATE/DELETE/MERGE
  - INTERSECT and EXCEPT
  - New data types
  - New built-in functions
  - OLAP functions
  - Index on expression
  - Global query optimization
  - pureXML

Agenda (2)

- DB2 10 for z/OS
  - Enhanced SQL user-defined functions
  - Implicit casting of character string and numeric values
  - Timestamp extensions
  - OLAP moving aggregates
  - Temporal tables
  - DSNULI, the “almost universal” language interface module
  - XML enhancements
  - LOB enhancements
  - RETURN TO CLIENT cursors

DB2 9 for z/OS SQL Enhancements

MERGE

- Also known as the “UPSERT” statement, MERGE takes a set of input records and compares them – using user-specified criteria – to existing rows in a table
  - If an input record is found to be a match for an existing table row, that row will be updated with information from the input record
  - If an input row does not match any of the existing rows in the table, it will be inserted into the table
  - A row newly inserted by MERGE could be updated with information in another record in the input set
- Simplifies coding: 1 statement takes the place of an INSERT and an UPDATE statement (and maybe 1 or more SELECTs)
  - MERGE can significantly improve performance, too, versus the old coding techniques used before MERGE was available

MERGE example

Correlation name for the "source table"

The target table

Input values in array variables – one per column (i.e., one per input record field)

```
MERGE INTO ACCOUNT AS A
USING (VALUES (:hv_id, :hv_amount)
FOR 3 ROWS)
AS T (ID, AMOUNT)
ON (A.ID = T.ID)
WHEN MATCHED
 THEN UPDATE SET BALANCE = A.BALANCE + T.AMOUNT
WHEN NOT MATCHED THEN INSERT (ID, BALANCE)
VALUES (T.ID, T.AMOUNT)
NOT ATOMIC CONTINUE ON SQLEXCEPTION;
```

The number of rows to merge

What constitutes a row match

What to do when there is a row match

Input rows are processed individually

What to do when an input row does not have a match in the target table

SELECT FROM UPDATE/DELETE/MERGE

Extends the SELECT FROM INSERT functionality introduced with DB2 V8

If you want to get row-by-row changes for an update or a delete, declare a cursor for the SELECT statement

If each designer got a 5% raise, what would be the total salary expense related to designers?

```
SELECT SUM(SALARY) INTO :salary FROM FINAL TABLE
(UPDATE EMP SET SALARY = SALARY * 1.05
WHERE JOB = 'DESIGNER');
```

```
SELECT SUM(SALARY) INTO :salary FROM OLD TABLE
(DELETE FROM EMP
WHERE JOB = 'OPERATOR');
```

"Old table" gives you the values from deleted rows

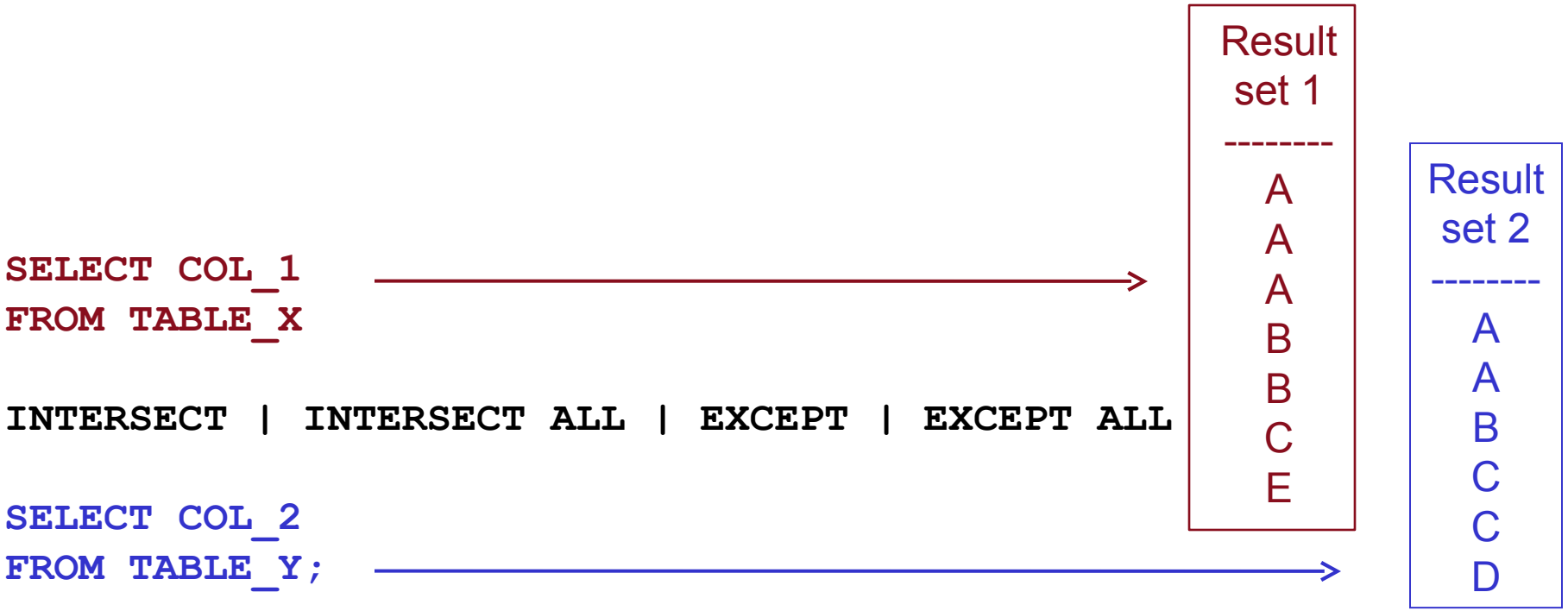
If the company outsourced the operator function, what would be the total decrease in salary expense?

More on SELECT FROM U/D/M…

- As with SELECT FROM INSERT, “job one” for a SELECT FROM UPDATE/DELETE/MERGE is an UPDATE, DELETE, or MERGE operation
  - The SELECT part piggy-backs on the data-changing statement
- What does it do for you?
  - Simplifies coding (versus driving a separate SELECT after the data-changing statement
  - Reduces the number of SQL statements issued by a program, and that translates into CPU savings

INTERSECT and EXCEPT

- New set operators make it easier to generate a result set based on comparison of two other result sets



INTERSECT DISTINCT vs. ALL

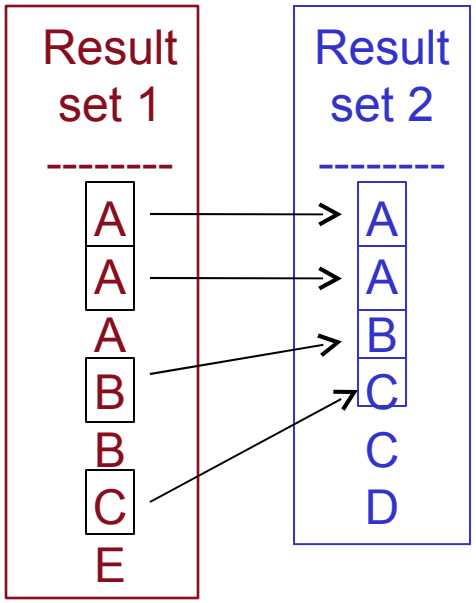
- As with UNION, default is DISTINCT
- DISTINCT compares overall result sets, ALL is value-to-value compare

```
set 1
INTERSECT
set 2
------
A
B
C
```

Which values appear in both sets?

```
set 1
INTERSECT ALL
set 2
------
A
A
B
C
```

What are the "matched equal-value pairs"?



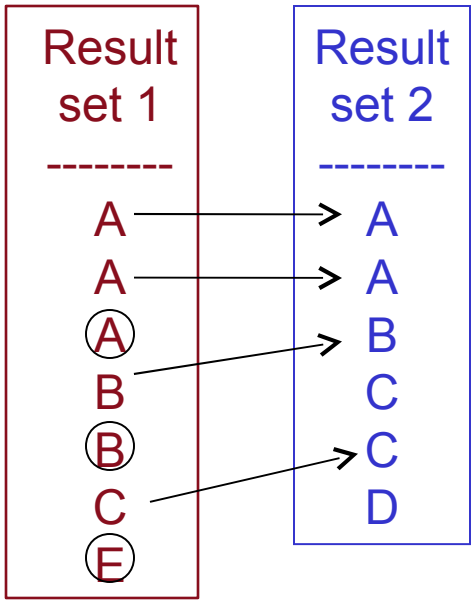
EXCEPT DISTINCT vs. ALL

```
set 1
EXCEPT
set 2
------
E
```

Which values appear in set 1 but not in set 2?

```
set 1
EXCEPT ALL
set 2
------
A
B
E
```

What individual values in set 1 have no match in set 2?



Note: set 1 INTERSECT set 2 is identical to set 2 INTERSECT set 1, but with EXCEPT, order matters!

New data types: BIGINT and BINARY

- BIGINT – occupies 8 bytes of storage, can represent 63-bit integer values
  - Range: -9223372036854775808 to 9223372036854775807 (that's about 9 quintillion, positive or negative)
- BINARY and VARBINARY
  - No code page association, no data value translation when inserted from non-mainframe client systems (sometimes, you don't want data translation)
  - A better way to store strings of bits versus using CHAR and VARCHAR columns with the FOR BIT DATA specification

New data types: DECFLOAT

- Allows for a huge range of values
  - DECFLOAT(16) – sixteen digits of precision
    - 10^{-383} to 10^{+384}
  - DECFLOAT(34) – thirty-four digits of precision
    - 10^{-6143} to 10^{+6144}
- Can represent some interesting values:
  - Infinity
  - “Quiet” not-a-number (no warning or exception issued if used in a numerical operation)
  - “Signaling” not-a-number (warning or exception issued)

Some new functions

- Aggregate
  - CORRELATION – returns the coefficient of correlation of a set of number pairs
  - COVARIANCE – returns the covariance of a set of number pairs
- Scalar
  - LPAD and RPAD – pads a string to the left or the right to bring it to a specified length
  - OVERLAY – overlays part of a string with another string
  - SOUNDEX – returns a 4-character code representing sound of a word
  - Difference – returns a value from 0 to 4 that represents the difference between the sounds of two strings
  - TOTALORDER – returns -1, 0, or 1, indicating the comparison order of two DECFLOAT arguments

OLAP functions

- The DB2 9 OLAP functions enable you to assign a number to each row in a result set, based on a specified ordering sequence
  - Refers to ordering “in the middle of” SQL statement processing – “final” ORDER BY sequence can be different
- Three OLAP functions:
  - RANK
  - DENSE_RANK
  - ROW_NUMBER

OLAP functions – example

```
SELECT UNIT_SALES, RANK() OVER(ORDER BY UNIT_SALES DESC) AS RANK
FROM PRODUCT_SALES WHERE PROD_ID = 'A12345'
AND MONTH IN ('JAN','FEB','MAR','APR','MAY','JUN')
ORDER BY RANK;
```

UNIT_SALES	RANK	(if I had specified DENSE_RANK)	(if I had specified ROW_NUMBER)
1000	1	1	1
900	2	2	2
800	3	3	3
800	3	3	4
800	3	3	5
700	6	4	6

Index on expression (1)

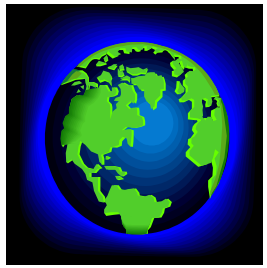
- Consider predicates like these:
  - WHERE UPPER(CUST_NAME) = 'JOHNSON'
  - WHERE SALARY + BONUS < 100000.00
- Stage 2, not indexable (at least, not by traditional index)
- Maybe live with poor query performance
- Maybe add a column to the table to hold (for example) upper-case names or the sum of SALARY and BONUS
  - Not a very attractive solution
    - How will the new column be maintained?
    - Are there programs with SELECT * FROM table-name? Will these break when a new column is added to the table?

Index on expression (2)

```
CREATE INDEX IXONEXP
ON TABLE_ABC
(UPPER(CUST_NAME))
USING…
```

- WHERE UPPER(CUST_NAME) = ‘JOHNSON’ now indexable, stage 1
- Note that overhead impact vs. “regular” index is higher for:
  - Inserts, some deletes, updates of column(s) in index key expression
  - LOAD, REORG TABLESPACE, CHECK INDEX, REBUILD INDEX
- Still, index on expression will often be a big win, as positive impact on query run times can be dramatic

Global query optimization (1)



- What it means:
  - In essence: “improved subquery processing” (A. Pannu, IBM)
  - Applies to queries with subquery predicates, for example:
    - WHERE T1.C1 IN (SELECT C2 FROM T2 WHERE…)
    - WHERE EXISTS (SELECT 1 FROM T2 WHERE T2.C2 = T1.C1…)
  - Prior to V9:
    - DB2 would optimize each SELECT in a query (outer SELECT and subselects) in isolation
    - In executing query, DB2 would process subqueries according to their position within the overall query
    - Result: access path for individual subqueries might be efficient, but access path for overall query might be quite inefficient

Global query optimization (2)

- What changed with DB2 9:
  - In optimizing subqueries, DB2 will consider how access path of one query block will affect access efficiency for overall query
  - DB2 will treat result set of subquery as a “virtual table,” and might:
    - Change order of table access from order indicated by position of subquery within overall query
    - Transform a subquery from correlated to non-correlated, or vice versa
    - Transform a subquery to a join
  - Potentially huge improvement in query performance
- In a DB2 9 environment:
  - Automatic benefit for dynamic SQL
  - Rebind to get benefit for static SQL

Global query optimization (3)

- What you might see in DB2 9 EXPLAIN output:

```
SELECT * FROM TABLE_1
WHERE EXISTS
(SELECT 1 FROM TABLE_2
WHERE TABLE_1.COL1 = TABLE_2.COL1
AND TABLE_2.COL2 = 1234
AND…)
```

DB2 converts the subquery from correlated to non-correlated, materializes (and sorts) that result set in a workfile, and joins that to TABLE_1

QB	M	TNAME	AT	SCU	SCO	QBTYPE
1	0	DSNWFQB(02)	R	N	N	SELECT
1	1	TABLE_1	I	N	N	SELECT
2	0	TABLE_2	I	N	N	NCOSUB
2	3			Y	Y	NCOSUB

pureXML

- A huge leap forward with respect to managing XML data in a relational database
- Before, there were two unpalatable options for storing an XML document in a DB2 table:
  - You could store it as a CLOB (character large object) or a long character string (if less than 32,000 bytes)
    - Poor performance if you wanted to retrieve data based on a particular value in a particular part of a document
  - You could “shred” the document, placing data from the different nodes in different columns of one or more tables
    - Could deliver good data retrieval performance, but made it hard to accommodate changes in the XML document’s structure (required database changes)

More on pureXML

- New data type: XML
- DB2 9 actually understands and preserves the hierarchical structure of a document stored in an XML column
- XQuery expressions can be used in defining indexes on XML columns, so that particular values in particular nodes can be located very quickly
- XQuery expressions can also be included in SQL statements, so that only a portion of the data in an XML document can be retrieved
- DB2 9 has a schema repository, in which information can be stored and used to validate structure (and maybe content) or an XML document before it's stored in the database

Information Management

IBM

DB2 10 for z/OS SQL Enhancements

Enhanced SQL user-defined functions

- DB2 9 and earlier: “logic” in a SQL scalar UDF Is what you can code in the RETURN statement, and that’s pretty limited
  - RETURN can’t contain a SELECT statement
  - RETURN can’t include a column name
- Basically limited to receiving a value (or values) as input, transforming that value (or values) arithmetically and/or with scalar functions, and returning result of that transformation
  - Example:

```
CREATE FUNCTION KM_MILES(X DECIMAL(7,2))
RETURNS DECIMAL(7,2)
LANGUAGE SQL
…
RETURN X*0.62;
```

Enhanced SQL UDFs (continued)

- DB2 10: SQL scalar UDFs can do MUCH more than before
  - RETURN can contain a scalar fullselect

```
RETURN(SELECT WORKDEPT FROM EMP WHERE EMPNO = P1);
```

  - RETURN can be at the end of a compound SQL statement, in which variables can be declared and SET, and SQL control statements used to generate the value to be returned

```
BEGIN
  DECLARE VAR1, VAR2 CHAR(10);
  SET VAR1 = …;
  IF P1 = …;
  RETURN VAR2;
END@
```

- Support for SQL table UDFs, too
  - Can code fullselect in RETURN statement

Implicit casting of character and numeric values

- Consider this statement:

```
SELECT 1 CONCAT '+' CONCAT 1 CONCAT '=' CONCAT 2
FROM SYSIBM.SYSDUMMY1;
```

- In a DB2 9 environment, you get this result:

```
SQLCODE = -171, ERROR: THE DATA TYPE, LENGTH,
OR VALUE OF ARGUMENT 1 OF CONCAT IS INVALID
```

- In a DB2 10 (NFM) system, you get this:

```
1+1=2
```

- Implicit casting for assignment (SET) statements, too (but not for special registers)
- Numeric values cast to VARCHAR, character values cast to DECFLOAT(34)

Timestamp extensions

- New: timestamp values down to the picosecond (that's a trillionth of a second)
  - Important for some users, since mainframe engines are so fast now that "mere" microsecond timestamp values can be duplicates
- New: variable-precision timestamps
  - From 0 (no fractions of a second) to 12 (picoseconds), with 6 being the default
  - Syntax: TIMESTAMP(n)
- New: TIMESTAMP WITH TIME ZONE
  - New data type
  - Sample value: '2011-03-03 10:15:00.123456-05:00'

OLAP moving aggregates (1)

- Complete coverage would require a lot of slides and time
- Suffice it to say that this is a new SQL syntax that allows:
  - Partitioning of a result set (e.g., by name)
  - Ordering of rows within a partition (e.g., by date)
  - Generation of aggregate values based on the “moving” current position within a set of rows (e.g., sum of sales for the current row plus the two preceding rows)
  - Example:

```
SELECT NAME, DATE, BOXES_SOLD,
SUM(BOXES_SOLD) OVER(PARTITION BY NAME
                ORDER BY DATE
                ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) SUM
FROM GIRL_SCOUT_COOKIE_SALES;
```

OLAP moving aggregates (2)

NAME	DATE	BOXES_SOLD	SUM
Mary	2011-01-10	7	7
Mary	2011-01-11	8	15
Mary	2011-01-12	5	20
Mary	2011-01-13	6	19
Sally	2011-01-10	4	4
Sally	2011-01-11	9	13
Sally	2011-01-12	8	21
Sally	2011-01-13	5	22

Sum of this row's BOXES_SOLD (5) plus the BOXES_SOLD values of the preceding two rows in the set (7 and 8)

Temporal tables

- Generating a lot of buzz in the DB2 community
- Basically, allows you to give a time dimension to data
- Two flavors:
  - System time: DB2 maintains a history table associated with the base table, inserting into the history table the “prior version” of a row for every update or delete operation
    - DB2 also maintains “from” and “to” times in the base and history tables, showing when a row was current
  - Business time: a dimension that shows when data in a row is valid from the user’s perspective (i.e., a price that will go into effect next month)
    - You maintain business time values, but DB2 can help by (for example) not allowing FROM and TO time period “overlaps” (enforced by an index)
  - You can combine system and business time in one table (“bi-temporal”)

More on temporal tables

- SELECT syntax has been extended to include the time dimension of a table:

Could specify BUSINESS_TIME if table has that dimension

Alternatively, could specify FROM and TO, or BETWEEN two timestamp values

```
SELECT COL1, COL2,,,
FROM POLICY
FOR SYSTEM_TIME AS OF TIMESTAMP '2010-02-24 00.00.00'
WHERE POLICY_NUM = '127348';
```

Why people are psyched about temporal

- System time makes it pretty easy to provide an audit history of data changes in a DB2 table
- Business time provides interesting “forward looking” data analysis possibilities
  - Real-world example: forecasting future profit margins using prices that will go into effect at a later time
- DB2-provided temporal capabilities (DDL, DML) GREATLY increase programmer productivity versus rolling your own temporal data functionality
- DB2-implemented temporal table functionality also delivers better performance than the roll-your-own alternative

The new DSNULI language interface module

- Local DB2-accessing programs have to be link-edited with the appropriate language interface module (e.g., DSNCLI for CICS, DSNELI for the TSO attach facility, etc.)
- This can be an irritant when you have one DB2-accessing program that you want to execute in more than one environment (e.g., CICS and TSO)
  - You'd rather not have to deal with two different load modules
- With DB2 10, DSNULI can be used in place of DSNCLI, DSNELI, DSNALI, and DSNRLI (not IMS – DFSLI00)
  - Detects runtime environment, dynamically loads appropriate language interface module
  - Trade-off for greater flexibility: some additional CPU and elapsed time

XML enhancements

- Lots there – here are a few:
  - New XML type modifier allows specification in table DDL of the schema to be used to validate data in an XML column
  - XML schema validation “in the engine,” versus by way of a DB2-supplied UDF that runs in a WLM-managed address space
    - Better performance, and zIIP redirect
  - CHECK DATA utility can check on structural validity of XML documents in an XML tablespace – not just consistency between base table and XML tablespaces
  - Ability to update part of an XML document (versus replacing the whole thing) via new XMLMODIFY built-in scalar function
    - Can insert a node into a document, replace a node, delete a node, or replace values of a node

LOB enhancements (1)

- DB2 9 provided an important enhancement, called progressive LOB streaming, that enabled a remote client to retrieve a LOB value in reasonably-sized “chunks”
  - DB2 10 provides progressive LOB streaming for local apps
- DB2 10 also delivers in-line LOBs
  - Refers to the ability to specify in a table’s definition the amount of space in a row that a LOB value will occupy – any amount over that limit goes into the appropriate LOB tablespace
  - Great for tables for which most LOB values are small (not uncommon)
  - Can significantly improve the performance of LOB-reading and LOB-inserting programs
  - Also enables compression of LOBs in base table
  - Also allows creation of index on expression on LOB column (SUBSTR)

Lob enhancements (2)

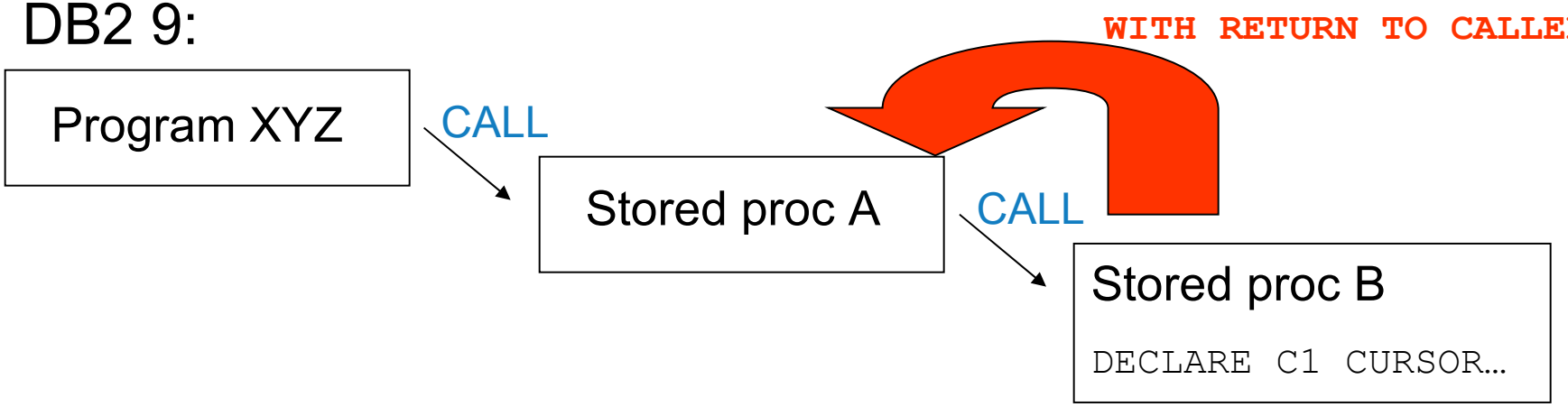
- Variable-block spanned (VBS) record format now supported for data sets used for table LOAD and UNLOAD
  - What this means for LOB users: FINALLY, you can unload a table with a LOB column (or columns) so that ALL of the data – LOB and non-LOB – goes into one sequential data set (e.g., a tape data set)
    - Before, had to unload LOB values to members of a PDS, or to individual files in the hierarchical file system (HFS)
- The LOB enhancements provided by DB2 9 (including online REORG of LOB tablespaces) and DB2 10 have me thinking that LOBs' time has come!
  - If you haven't used DB2 for LOB data before, now could be a great time to start

RETURN TO CLIENT cursors

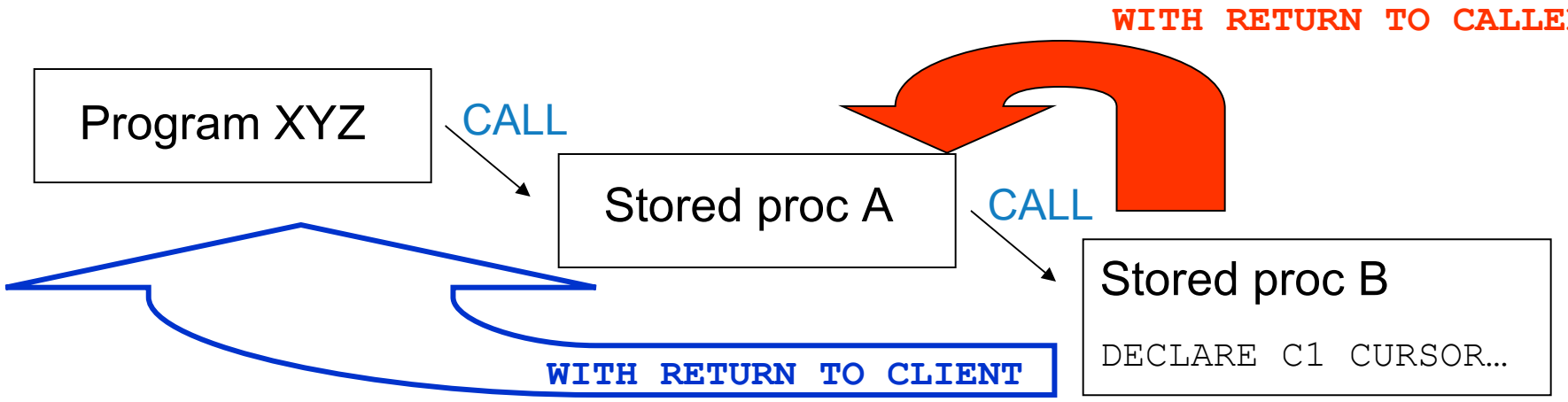
- DB2 9: stored procedure can return result set one level up in chain of nested procedures (WITH RETURN TO CALLER)
  - For example, if program PROG_A calls stored procedure PROC_B, and PROC_B calls PROC_C, PROC_B can fetch from a cursor declared and opened in PROC_C, but PROG_A cannot
    - If PROG_A needs that result set, PROC_C can put it in a temporary table, and PROG_A can get the rows from that temp table, OR
    - PROC_B can declare and open a cursor referencing the temp table, and PROG_A can fetch the result set rows through that cursor)
- A DB2 10 stored procedure can declare a cursor WITH RETURN TO CLIENT (just like DB2 for LUW)
  - “Top-level” program (one that calls first stored procedure, which then calls another stored procedure) can fetch rows, but the cursor’s result set is invisible to stored procedures between it and top-level program

Previous slide's point, in a picture…

- DB2 9:



- DB2 10:



Thanks for your time!

Robert Catterall
rfcatter@us.ibm.com