

IBM Information

>>> On Demand

2007



DB2 9 z/OS User Experiences at Univar USA

*Kevin Campbell, Application Architect, Univar USA,
kevin.campbell@univarusa.com*

2082A



Act.Right.Now.

IBM INFORMATION ON DEMAND 2007

October 14 - 19, 2007

Mandalay Bay

Las Vegas, Nevada

Introduction

- This presentation focuses on the testing activities carried out by Univar USA Inc during the DB2 9 ESP (Beta) program and since GA
- Our testing was almost exclusively focused on new functionality
- We performed some migration/fallback and existing workload testing, but that is not the primary topic of this presentation



Univar USA company background

- Largest national distributor of Industrial Chemical and food products and related services, founded in 1924.
- 124 Warehouse locations around the country
- 4,500 Employees, \$4.5 Billion Revenue
- 39.5 Million Gallons Tank Storage
- 10.3 Million Square Feet Warehouse Capacity
- 2,008 vehicle delivery fleet



IT background

- Total IT employees less than 100
- Single primary LPAR supports dev, test, prod
- Core application is CICS ERP system implemented in 1989
 - A number of internally developed Java applications
 - Distributed Financials on LUW
 - MQ Messaging and Broker deployed since 2000
- System currently supports Univar USA, Univar Canada online Q2 2008 in own LPAR
- DB2 introduced to shop with v8 in 2003, installed 3 days after GA.
 - CICS app migrated from VSAM to DB2 in 2003



Motivation for DB2 9 beta participation

- Projects underway/planned that could benefit from new features
 - Evaluate features early in project, validate assumptions that DB2 9 would be applicable.
 - Understand areas where planned projects could re-think approach to leverage features coming in DB2 9.
- Get a head start on DB2 9 migration preparedness
 - Practice migration & begin testing existing apps
 - Understand z/OS pre-reqs and any potential hurdles
- Networking/Education
 - Early access to education
 - Direct interaction with lab



ESP involvement

- Approached during late 2005 by local account team
 - Provided profile of workload to enable ESP team to assess “fit”
 - Variety of legal paperwork required
- Kick off Education Session June 2006
 - Our entire team traveled to SVL (both of us)
 - Other customer representatives from around the world, all of whom are much bigger than us 😊
 - Intensive week of sessions on new features, plus a “who’s who” in the land of DB2 development
- First tapes arrived almost immediately.



z/OS operational model

- Very small test LPAR with z/OS 1.7 for ESP testing
 - Used by SysProgs to test system level changes before production installation, configuration constrained by need to mirror production for maintenance and release levels
 - DB2 9 developed on z/OS 1.8, some required PTFs for 1.7 not RSU during ESP, limited our ability to test certain features (XML, COLLATION KEY)



ESP preparation

- 2 DB2 subsystems established
 - One copied from production and migrated, used for fallback testing and IVPs
 - Other installed clean as 9 and used immediately for new function testing



Testing goals

- Evaluate new functionality
 - Explore new features that appeared desirable for current or upcoming development work.
 - Take advantage of early education opportunities.
- Understand Migration Process
 - Ensure that z/OS pre-reqs are understood and can be planned for.
 - Identify any PTFs that may be required.
 - Typical migration/fallback testing and validation.
- Little focus on regression testing during ESP
 - Extremely limited personnel meant that little effort was made to evaluate existing workloads during ESP.



Installation/Migration

- Installation very similar to v8
- Migration also similar to v8
 - Migrated v8 to 9 subsystem without issue.
- No unusual problems encountered
- Due to issues with z/OS 1.7 fixes (TCP/IP and USS) that were not yet released/RSU, we had no DRDA early on.
- Used production DDL in native 9 subsystem to create databases
 - Populated with production unloads



New function testing

- Native SQL Procedures
- Not logged table spaces
- Index on expression
- Index compression
- Partition by growth
- Clone tables
- Instead of triggers
- SQL Enhancements:
 - Order by/fetch first in subselect
 - Rename column
 - Nested compound statement
 - SOUNDEX()
- Pure XML



Native SQL procedures

- Created a variety of procedures testing control of flow, data access, returned result sets
- Used native SQL procs to create workload for other tests
- Simplifies deployment of SQL procs, no external loadlib created or additional WLM config needed
- Informal notes suggest a meaningful performance improvement over external procs
- No XML parameter support as yet



Native SQL procedures

- ```
CREATE PROCEDURE DCSPT.GET_CUSS
 (IN CUST_NUMBER_IN CHAR(11))
 DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
DECLARE RET_TABLE CURSOR WITH RETURN FOR
 SELECT
 SHIP_TO_NAME
 FROM
 DCSPT.DCSCUSS
 WHERE
 CUST_NUMBER = RTRIM(CUST_NUMBER_IN);
OPEN RET_TABLE;
END~
```
- Absence of FENCED or EXTERNAL keywords causes this to be a *native SQL procedure*



## Native SQL procedures

- No longer have to provide a WMLENV to deploy and execute a native SQL PROC
  - executed in DBM1
- But ...
  - Procedure debugger relies on DB2 generated C program, which in turn requires WLMENV.
  - Can specify default WLMENV in DSNZPARM, if it's NULL and nothing specified in CREATE PROCEDURE you'll get an error when attempting to create proc for debugging.



## Native SQL procs/Data Studio

- Lack of DRDA/DDF support (z/OS 1.7 PTF) early on meant that Data Studio Beta could not be evaluated with DB2 9 initially
  - Once resolved Data Studio proved to be very capable for development/debugging
- To a developer DB2 z/OS little different from DB2 LUW
- Eclipse based development of procs using Data Studio eases proc porting/development in a multi-vendor shop



---

## Not logged table spaces

# NEWSFLASH !

- DB2 logging is very efficient!





## Not logged table spaces

- During Education sessions we were told that existing DB2 logging is very efficient.
  - “No, really, believe us DB2 logging is very efficient”
  - Told to expect limited CPU savings
- Why not logged?
  - Certain types of operation may be easily recoverable from external sources: ERP Installation routines create vast amounts of meta data, ETL processes.
  - Possible to reduce elapsed time if updates overtaking log offloads, or just by reducing I/O waits



## Not logged table spaces

- Changing logging for table space very simple:
  - ALTER TABLESPACE DCSPT.CUSS NOT LOGGED;
  - ALTER TABLESPACE DCSPT.CUSS LOGGED;
- Table space will be in COPY PENDING after enabling logging, as there is no recovery point.



## Not logged table space performance tests

- Sample table with 300,000 rows taken from production
- Native SQL Procedure
  - Updates one column in randomly selected row
  - Repeated 500,000 times

| <i>Reduction with<br/>Not Logged</i> | <b>One Unit of<br/>Work</b> | <b>500,000 units of<br/>work</b> |
|--------------------------------------|-----------------------------|----------------------------------|
| <b>CPU</b>                           | -5.6%                       | -3.3%                            |
| <b>Elapsed Time</b>                  | -26%                        | -21%                             |



## Not logged table space performance tests

- Interesting to note that commit frequency of every row is less efficient than single unit of work (anyone surprised?)
  - 100% greater elapsed time
  - 260% more CPU
- If updates overwhelmed logging, leading to offload waits, elapsed difference likely more dramatic.



## Index on expression

- Allows an index to be created on the result of a function

- UPPER(), SUBSTRING(), TRIM(), PAD() etc

- CREATE INDEX DCSPT.CUSSAX01  
ON DCSPT.DCSCUSS  
UPPER(SHIP\_TO\_SALUTATION)  
USING STOGROUP SYSDEFLT;

- Index will be used when corresponding function appears in a predicate

- SELECT CUST\_NUMBER FROM DCSPT.DCSCUSS  
WHERE  
UPPER(SHIP\_TO\_SALUTATION) = 'BOB' ;



## Index on expression

- Avoids having to maintain an additional column in the table with the result of a function applied
- Works exactly as advertised with native scalar functions
  - It would be nice if scalar UDFs could be used
  - Function result must be same CCSID as table space
- DBA reliant on good monitoring tools or coordination with developers to maximize benefits



## Index compression

- Actually a two for one; compression and support for buffer pools with >4k pages (sort of)
  - Must specify at least 8K page to use compression
  - I/O still done with 4K compressed leaf pages, non leaf pages not compressed
- Uses compression algorithm rather than dictionary, and compresses/decompresses entire page (think PKZip)
- Leaf page is compressed on DASD, not in buffer pool
  - Compression on write, decompression on read.



# Index compression

- CREATE UNIQUE INDEX DCSPT.CUSSPX11  
ON DCSPT.DCSCUSS( ... )  
USING STOGROUP SYSDEFLT  
PRIQTY -1  
SECQTY -1  
BUFFERPOOL BP8K1  
COMPRESS YES;





# Index compression

- We saw 50% compression with 8K page
  - Depending on contents of indexed columns can see somewhat higher with larger buffer pool page sizes
  - Told by lab that 50% with 8K page is what most should expect, won't see better with 8K
- Did you know: unique indexes, compressed or not, typically 8% smaller than non unique?



# Index compression

- Sequential performance test:
  - Open cursor containing only index columns *order by* as per index
  - Fetch next until end
- Random performance tests:
  - Driver table containing index columns and “random” timestamp populated from indexed table
  - Sequential fetch of index columns from driver table *order by* random timestamp
  - Select against indexed table specifying only index columns
- Buffer pools sized very small so minimal caching
  - Workload submitted multiple times on quiet LPAR



## Index compression

- Sequential access shows least overhead
  - CPU increased by 1% to 3%
- Random reads cost more with larger pages in BP
  - Entire page must be uncompressed for just one key row
  - 4% to 100% CPU increase depending on index and page size
- With small VPSIZE there is little elapsed difference either way
  - Increasing VPSIZE gave slight elapsed time edge to uncompressed index



# Index compression

- Our random workload rather artificial
  - Extremely small VPSIZE meant no caching
  - As soon as VPSIZE increased to more normal value for size of index the overhead dropped
- Consider nature of workload before implementing compression
  - Huge DW indexes used largely for range scans probably good candidate
  - Tune BP page size according to index design and data patterns
  - 16K or 32K pages may give up to 75% compression, but more uncompressing required



## Partition by growth

- Good when no obvious partition key, or when table expected to be >64GB and no suitable key
  - Content Manager
  - ERP systems with impenetrable data models
- Creates new partition at size boundaries only as needed
- ```
CREATE TABLESPACE CUST
  IN          DCSPT
  USING      STOGROUP ...
  MAXPARTITIONS 10
  DSSIZE      2G
```
- Up to 10, 2GB partitions in this example



Partition by growth

- Our tests show “it does what it says on the packet”
- A few considerations, mostly obvious
 - Must be Universal Table Space
 - Key based partition features not supported: LOAD PART, REBALANCE part of REORG, partition elimination during table space scan, ALTER to add or rotate
 - can ALTER to increase MAXPARTITIONS



Clone tables

- Two tables share the same definition and in same universal table space
 - Identical in catalog; same index definitions, before triggers
- Datasets used for each table can be exchanged on the fly, meaning that the contents of the two tables appear to swap
- Applicability for Data Warehouse applications for tables that are rebuilt rather than updated



Clone tables

- We created partition by growth table space
 - CLONE requires universal table space, no other tables in it
 - Must be either partition by range or growth
- Create and populate table with 300,000 rows
- ALTER TABLE to add clone, then populate it
 - ALTER TABLE DCSPT.DCSCUST ADD CLONE DCSCUSTC ;



Clone tables

- Load clone with different data from base table, same key values
- Test program queries base table, reporting when a row changes
 - EXCHANGE DATA BETWEEN TABLE DCSPT.DCSCUST AND DCSPT.DCSCUSTC ;
 - Must be a COMMIT between successive EXCHANGE statements
- As with any such alter a heavy workload may -904
- Functionality works as advertised



INSTEAD OF ... Triggers

- Application design may create complex views in order to simplify application coding
 - Might be good for select, but still requires specialized coding to handle INSERT or UPDATE
- We have several views used to de-normalize table design
 - Most extreme example runs to > 6,000 lines of CREATE VIEW DDL
- Created INSTEAD OF INSERT and INSTEAD OF UPDATE triggers on complex parent/child view



INSTEAD OF ... Triggers

- View has columns PID,Y,Z,A1,B1,C1,A2,B2,C2
- ```
CREATE TRIGGER DCSPT.TI_VUSS
 INSTEAD OF INSERT ON DCSPT.DCSVUSS
 REFERENCING NEW AS N
 FOR EACH ROW MODE DB2SQL
BEGIN ATOMIC
 INSERT INTO DCSPT.DCSCUSS(PID,Y,Z)
 VALUES (N.PID,N.Y,N.Z);
 INSERT INTO DCSPT.DCSCUSS_1(PID,SEQ,A,B,C)
 VALUES (N.PID,1,N.A1,N.B1,N.C1);
 INSERT INTO DCSPT.DCSCUSS_1(PID,SEQ,A,B,C)
 VALUES (N.PID,2,N.A2,N.B2,N.C2);
END~
```
- This example populates parent row and two child rows from single row in view DCSVUSS



# SQL Enhancements

- Order By / Fetch First in sub-select adds considerable flexibility to application developers

- SELECT  
    C.CUST\_NAME ,  
    C.CUST\_NUMBER  
FROM CUSTOMER C  
WHERE  
    C.SALES\_REP IN  
    (SELECT SALES\_REP  
    FROM  
        SALESPERSON  
    ORDER BY TOTAL\_COMMISSION DESC  
    FETCH FIRST 100 ROWS ONLY);
- Shows customers serviced by top 100 commission earning reps.



# SQL Enhancements

- Order By/Fetch First in sub-select continued
  - Common design pattern is to limit number of rows returned in result set to client application.
  - Problem arises with ORDER BY that entire result set may need to be created and ordered before finding FIRST x ROWS.
  - ```
SELECT
    CUSTOMER_NAME ,
    CUSTOMER_NUMBER
FROM
    (SELECT CUSTOMER_NAME , CUSTOMER_NUMBER FROM CUSTOMER
     WHERE CUSTOMER_STATE='WA' FETCH FIRST 300 ROWS ONLY)
ORDER BY CUSTOMER_NAME ;
```
 - Results may seem arbitrary as it's finding 300, then sorting as opposed to sorting then returning first 300.



SQL Enhancements

- Rename Column

- As it suggests ALTER TABLE ... RENAME COLUMN x TO y will rename a column
- But, not if:
 - Column referenced in a view, index on expression, check constraint or field procedure
 - Table has any triggers
 - Table is an MQT or referenced by MQTs
 - Table has valid proc or edit proc defined
 - Table is a catalog table.
- We see some applicability and suspect this targeted at heavy dynamic SQL apps, such as “household name” ERP suites.



SQL Enhancements

- SOUNDEX()

- New to DB2 but not a new concept
- Function returns 4 byte value based on English phonetic spelling of word
- Useful for approximations, and might be a good candidate for index on expression theoretically like this:
 - ```
CREATE INDEX DCSPT.CUSSAX01
ON DCSPT.DCSCUSS(SOUNDEX(SHIP_TO_SALUTATION));
```
- But, expression used must return same CCSID as table space, and (why?) SOUNDEX returns SBCS CCSID! So:
  - ```
CREATE INDEX DCSPT.CUSSAX01
ON DCSPT.DCSCUSS(CAST(SOUNDEX(SHIP_TO_SALUTATION) AS CHAR(4)
CCSID EBCDIC));
```
- Unless your table space is using SBCS ☺



SQL Enhancements

- **SOUNDEX()**

- Having appropriate index now supports queries such as:

- ```
SELECT CUSTOMER_NUMBER, CUSTOMER_NAME
FROM CUSTOMER
WHERE
CAST(SOUNDEX(SHIP_TO_SALUTATION) AS CHAR(4) CCSID
EBCDIC) = CAST(SOUNDEX("SMITH") AS CHAR(4) CCSID
EBCDIC);
```

- Did I mention that having SOUNDEX() return CCSID SBCS was a pain?

- **See also DIFFERENCE()**

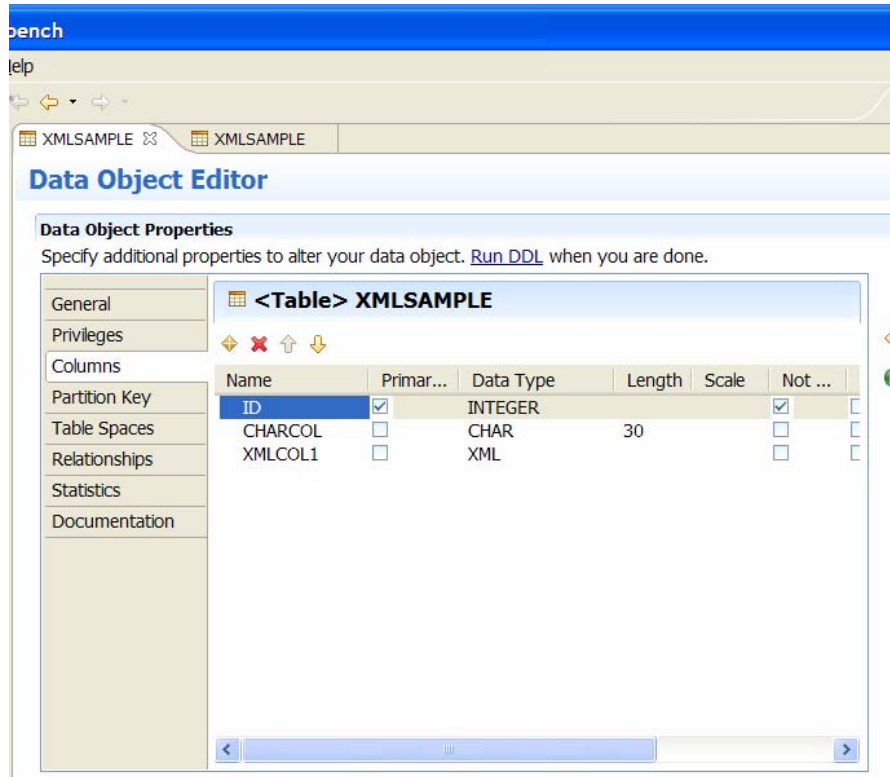
- Returns a value from 0 to 4 indicating how similar two words sound, 4 being “most similar”





# Pure XML

- Encountered very late in testing as z/OS 1.7 prerequisites were not in place
- Create table with XML column type:



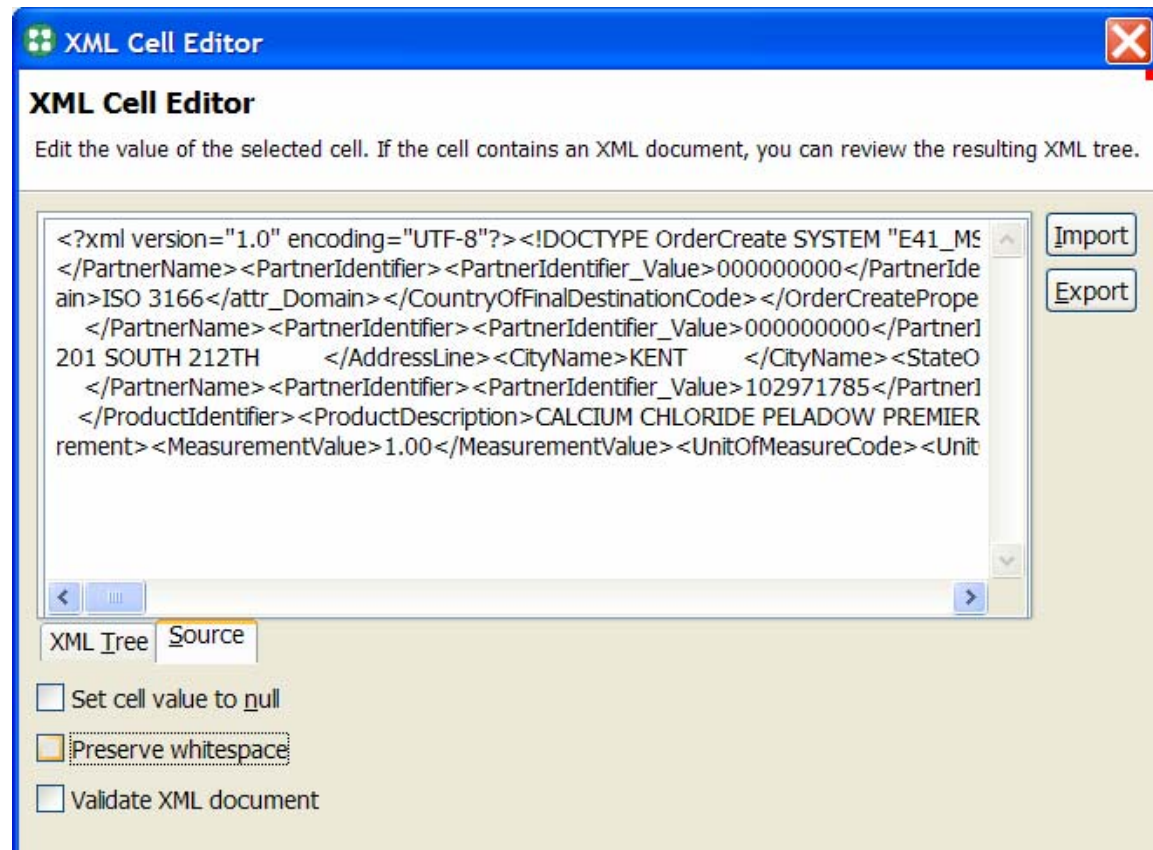
The screenshot shows the Data Object Editor interface for a table named XMLSAMPLE. The table has three columns: ID (INTEGER, primary key), CHARCOL (CHAR, length 30), and XMLCOL1 (XML). The interface includes a left-hand navigation pane with options like General, Privileges, Columns, Partition Key, Table Spaces, Relationships, Statistics, and Documentation. The main area displays the table structure with columns for Name, Primary Key, Data Type, Length, Scale, and Not Null.

| Name    | Primar...                           | Data Type | Length | Scale | Not ...                             |
|---------|-------------------------------------|-----------|--------|-------|-------------------------------------|
| ID      | <input checked="" type="checkbox"/> | INTEGER   |        |       | <input checked="" type="checkbox"/> |
| CHARCOL | <input type="checkbox"/>            | CHAR      | 30     |       | <input type="checkbox"/>            |
| XMLCOL1 | <input type="checkbox"/>            | XML       |        |       | <input type="checkbox"/>            |



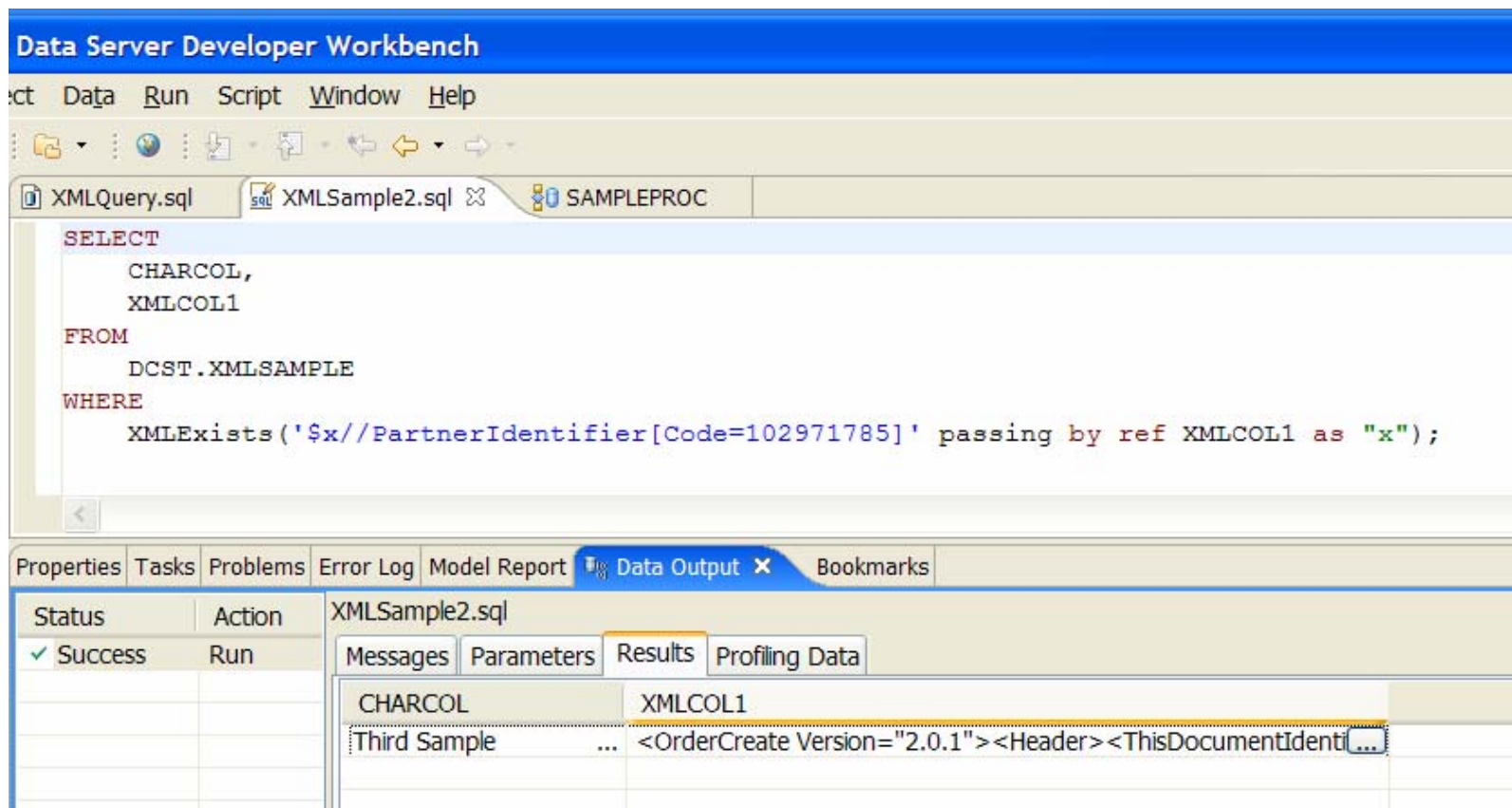
# Pure XML

- Use Developer Workbench to import XML files to table:



# Pure XML

- Then use XMLEXISTS with XPath expression in search:



The screenshot shows the Data Server Developer Workbench interface. The main window displays an SQL query in a text editor:

```
SELECT
 CHARCOL,
 XMLCOL1
FROM
 DCST.XMLSAMPLE
WHERE
 XMLEXISTS('$x//PartnerIdentifier[Code=102971785]' passing by ref XMLCOL1 as "x");
```

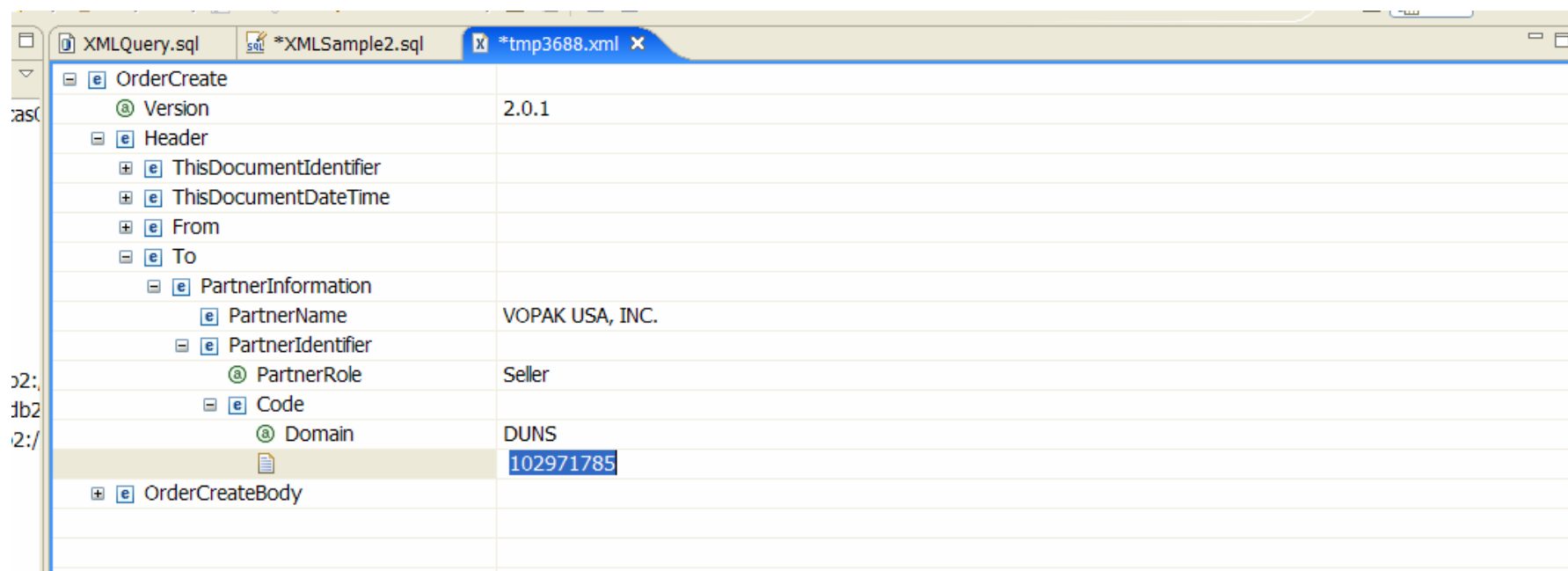
Below the editor, the 'Data Output' tab is active, showing the results of the query. The results are displayed in a table with two columns: CHARCOL and XMLCOL1.

| CHARCOL      | XMLCOL1                                                       |
|--------------|---------------------------------------------------------------|
| Third Sample | <OrderCreate Version="2.0.1"><Header><ThisDocumentIdentifi... |



# Pure XML

- Developer Workbench automatically opens XML column data in XML Editor:



The screenshot shows the XML Editor window in SQL Developer. The window title is '\*tmp3688.xml'. The XML document is expanded to show the following structure:

|                        |                 |
|------------------------|-----------------|
| OrderCreate            |                 |
| Version                | 2.0.1           |
| Header                 |                 |
| ThisDocumentIdentifier |                 |
| ThisDocumentDateTime   |                 |
| From                   |                 |
| To                     |                 |
| PartnerInformation     |                 |
| PartnerName            | VOPAK USA, INC. |
| PartnerIdentifier      |                 |
| PartnerRole            | Seller          |
| Code                   |                 |
| Domain                 | DUNS            |
| Value                  | 102971785       |
| OrderCreateBody        |                 |



## General ESP Observations

- DB2 9 for z/OS was already pretty polished when we first saw it mid 2006
- Our two sub-systems were very stable during testing
- Significant new functionality for application developers in this release
  - Pure XML and native SQL procedures likely to be of interest to many application developers
- Worthwhile improvements in a number of utilities from either wall clock or CPU perspective



---

# Questions?

