



Db2 SQL and SQL PL

-A Journey Through Db2 12 and 13 Functions Levels

Chris Crone

Broadcom Distinguished Engineer – Database Technologies

cjc@broadcom.com

Central Canada Db2 User Group 2023
September 2023



Agenda

- Db2 12 M500 SQL
- Db2 12 M500 SQLPL
- Db2 12 M501 and Above
- Db2 13 and M500 and Above
- Summary



Db2 12 M500 SQL



Copyright © 2022 Broadcom. All Rights Reserved. The term "Broadcom" refers to Broadcom Inc. and/or its subsidiaries.

Pagination

- Data-Dependent Pagination

Db2 12 now supports $<$, $<=$, $>$, $>=$ (= and $<>$ were supported in V10) as new basic predicate operators in **row-value-expressions**

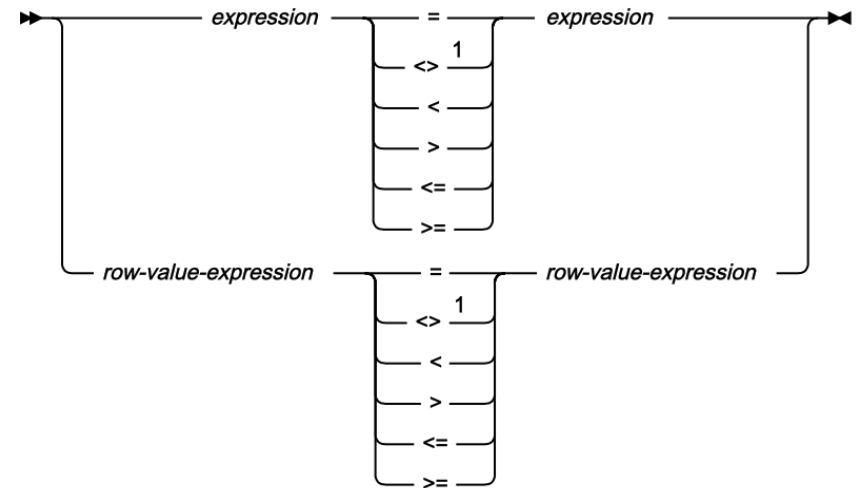
- Prior to V12 – you would code:

```
WHERE      (LASTNAME = 'Trump' AND FIRSTNAME > 'Donald')  
           OR (LASTNAME > 'Trump')
```

- In Db2 12 – you can code:

```
WHERE (LASTNAME, FIRSTNAME) > ('Trump', 'Donald')
```

- Db2 will convert to original 'OR' syntax and use Db2 10 "NI" Access Path



Numeric Pagination - OFFSET

Accessing Data within a result set

Used when applications are using pagination and want to “skip” multiple pages in a result set

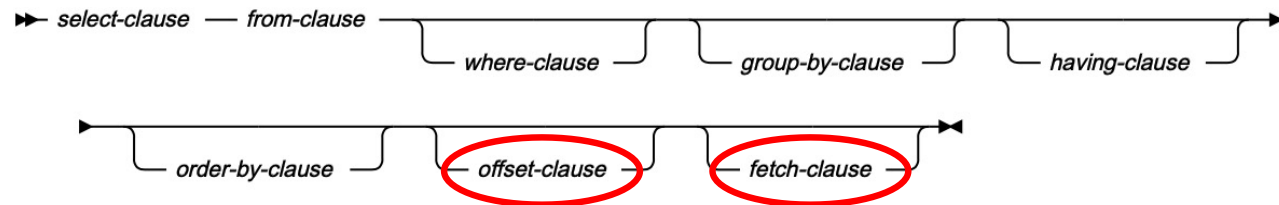
Prior to Db2 12

Scrollable Cursor

ROWSET Positioning

Discarding Rows

Db2 12 – OFFSET “m”



FETCH “m” ROWS OFFSET “m” ROWS can be used together

OFFSET 0 ROWS FETCH FIRST 10 ROWS ONLY

OFFSET 10 ROWS FETCH FIRST 10 ROWS ONLY

FETCH FIRST (LIMIT) Example

Newer Applications are written to provide data to multiple screen sizes

iPhone, iPad, Desktop

Prior to Db2 12

FETCH FIRST n ROWS ONLY clause only supports literal

```
SELECT LASTNAME FROM EMP FETCH FIRST 5 ROWS ONLY;
```

Requires one statement in statement cache for each screen size you want to support

With Db2 12

FETCH FIRST "M" ROWS ONLY

Support specification of a Host Variable or Parameter Marker that is castable to BIGINT

```
SELECT LASTNAME FROM EMP FETCH FIRST ? ROWS ONLY;
```

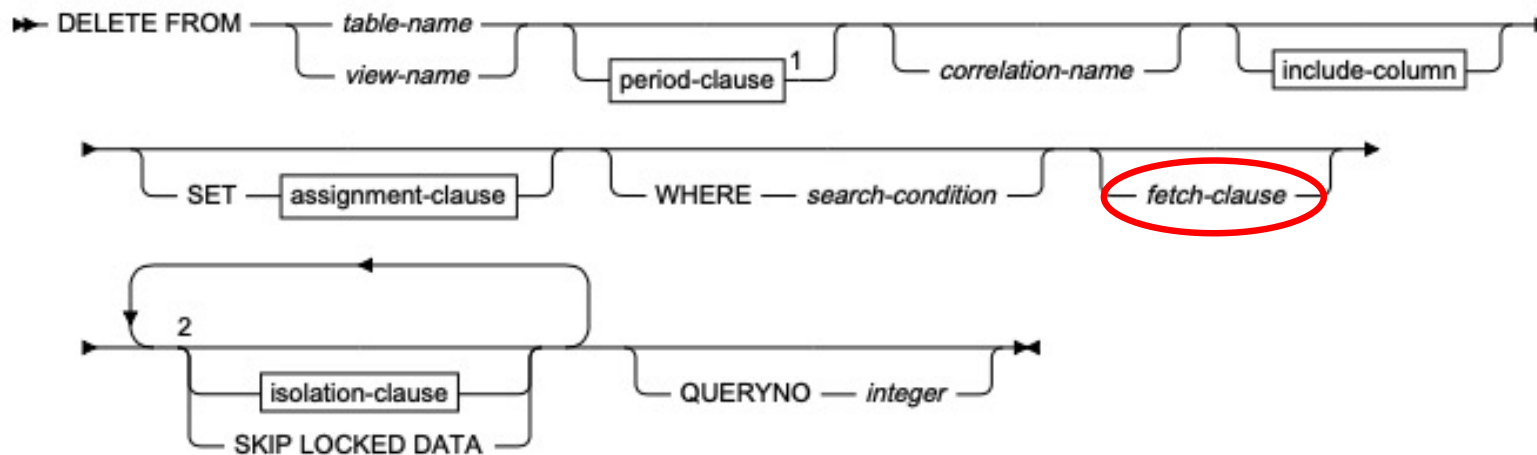
LIMIT Synonyms

- Syntax – Alternative Syntax Supported for compatibility with other DBMS

Alternative syntax	Equivalent syntax
LIMIT x	FETCH FIRST x ROWS ONLY
LIMIT x OFFSET y	OFFSET y ROWS FETCH NEXT x ROWS ONLY
LIMIT y, x	OFFSET y ROWS FETCH NEXT x ROWS ONLY

DELETE FETCH FIRST 'm' ROWS on Searched DELETE

searched delete:



Notes:

¹ If the *period-clause* is specified, the *fetch-clause* must not be specified.

² The same clause must not be specified more than one time.

DELETE FETCH FIRST 'm' ROWS

- Customers need to DELETE data as part of normal data maintenance
 - Don't want to have to code a Positioned DELETE
 - Searched Delete is easy to use, however:
 - Locking Impact (Including Lock Escalation)
 - Logging Impact (including concerns of rollback)
- DELETE WITH FETCH FIRST 'm' ROWS ONLY
 - Simple processing DELETE...; COMMIT; DELETE...; COMMIT; ...
 - Can be used in a loop with a commit
 - Can be used in an autonomous SQL PL Procedure
<https://www.idug.org/p/bl/et/blogid=278&blogaid=648>
- Restrictions
 - A view that is defined with an instead of trigger if the fetch-clause is specified.
 - A created global temporary table if the fetch-clause is specified.
 - An accelerator-only table if the fetch-clause is specified.

Piecewise DELETE Example

```
CREATE PROCEDURE FFNR_PURGE_ROWS_SP(IN P_SEL_STMT VARCHAR(1000), IN P_TB_NM VARCHAR(128),
IN P_NUM_OF_LPS INTEGER, IN P_COMMIT INTEGER) LANGUAGE SQL AUTONOMOUS
P1: BEGIN

    DECLARE v_DELETE VARCHAR(1000);
    DECLARE v_counter integer DEFAULT 0;

    SET v_DELETE = 'DELETE FROM ' || P_TB_NM || ' WHERE ' || P_SEL_STMT || ' FETCH FIRST ' ||
        P_COMMIT || ' ROWS ONLY';

    delete_loop: LOOP

        EXECUTE IMMEDIATE v_DELETE;
        COMMIT;

        SET v_counter = v_counter + P_COMMIT;
        IF v_counter >= P_NUM_OF_LPS THEN
            LEAVE delete_loop;
        END IF;

    END LOOP delete_loop;
END P1#
```

Pagination, LIMIT, OFFSET - Performance Considerations

- LIMIT and OFFSET used together to “Window” through the data can improve performance when compared to an application fetching all the rows back and windowing through the data itself
 - **OFFSET 0 ROWS FETCH FIRST 20 ROWS ONLY**
 - **OFFSET 20 ROWS FETCH FIRST 20 ROWS ONLY**
- If OPTIMIZE FOR n ROWS is omitted
 - FETCH: 25
 - DELETE: 10000
- Db2 will replicate FETCH FIRST n ROWS ONLY into inner query block if possible
 - Must satisfy ORDER BY and FETCH FIRST push-down rules
 - Reduce the number of rows being returned from underneath subselect or fullselect

Original Query:

```
SELECT C1 FROM T1
UNION ALL
SELECT C1 FROM T2
ORDER BY 1 FETCH FIRST 10 ROWS ONLY;
```

After Rewrite by Optimizer:

```
(SELECT C1 FROM T1 ORDER BY 1
  FETCH FIRST 10 ROWS ONLY
UNION ALL
(SELECT C1 FROM T2 ORDER BY 1
  FETCH FIRST 10 ROWS ONLY)
ORDER BY 1 FETCH FIRST 10
```

Extended MERGE

Original MERGE – aka UPSERT

Extended MERGE – ANSI SQL Standard based MERGE

- table-reference as an additional way of specifying source data for the MERGE statement

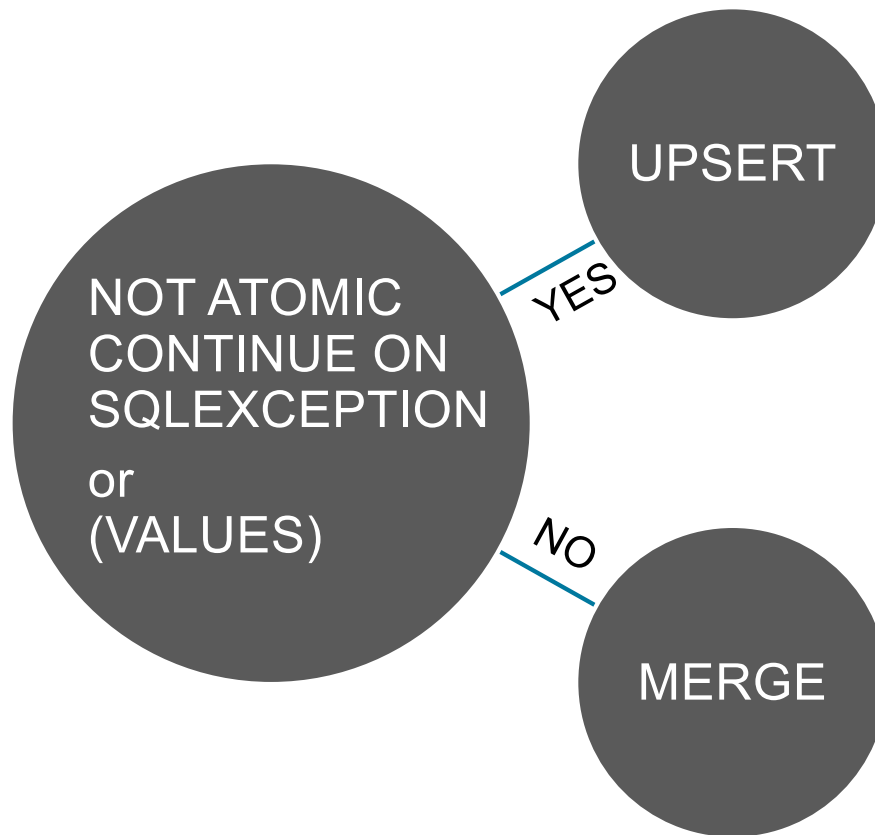
- Multiple MATCHED clauses

- Enhanced predicates with MATCHED or NOT MATCHED

- DELETE operations

- IGNORE and the SIGNAL statement as actions

Original MERGE (aka UPSERT) and Enhanced MERGE



Example Db2 9 MERGE – aka UPSERT

- MERGE Data from an Application into a Table

MERGE INTO ACCOUNT AS TRG

USING VALUES (:acct_hv, :amount_hv) FOR :hv ROWS AS SRC (ACCT, AMT)

ON TRG.ACCT = SRC.ACCT

WHEN MATCHED THEN

UPDATE SET TRG.AMT = TRG.AMT + SRC.AMT

WHEN NOT MATCHED THEN

INSERT (ACCT, AMT) (SRC.ACCT, SRC.AMT)

NOT ATOMIC CONTINUE ON SQLEXCEPTION;

SRC.ACCT	SRC.AMT
13579	1000
24680	1500
24680	2500
95136	835
94578	5996



TRG.ACCT	TRG.AMT
10203	3000
24680	2000
95223	2000
96136	3000
94578	10000

MERGE



TRG.ACCT	TRG.AMT
10203	3000
13579	1000
24680	6000
95223	2000
95136	3835
94578	15996

Example MERGE with FULLSELECT

- MERGE Data from a Staging Table into another Table

MERGE INTO TRG AS TRG

USING (SELECT ACCT, AMT FROM SRC) AS SRC ON TRG.ACCT = SRC.ACCT
WHEN MATCHED THEN UPDATE

SET TRG.AMT = TRG.AMT + SRC.AMT

WHEN NOT MATCHED THEN

INSERT (ACCT, AMT) VALUES (SRC.ACCT, SRC.AMT) ;

SRC.ACCT	SRC.AMT
13579	1000
24680	1500
95223	2500
95136	835
94578	5996



TRG.ACCT	TRG.AMT
10203	3000
24680	2000
95223	2000
96136	3000
94578	10000

MERGE



TRG.ACCT	TRG.AMT
10203	3000
13579	1000
24680	3500
95223	4500
95136	3835
94578	15996

Example MERGE with FULLSELECT

MERGE Data from a Staging Table into another Table with additional logic

```
MERGE INTO TRG AS TRG
  USING ( SELECT ACCT, AMT FROM SRC) AS SRC ON TRG.ACCT = SRC.ACCT
  WHEN MATCHED AND GV = ' ' THEN
    UPDATE SET TRG.AMT = TRG.AMT + SRC.AMT
  WHEN MATCHED AND GV = 'REPLACE' THEN
    UPDATE SET TRG.AMT = SRC.AMT
  WHEN MATCHED AND GV = 'DELETE' THEN
    DELETE
  WHEN MATCHED AND GV = 'SS' THEN
    SIGNAL SQLSTATE '75000' SET MESSAGE_TEXT = 'SS Failure'
  WHEN NOT MATCHED THEN
    INSERT (ACCT, AMT) VALUES (SRC.ACCT, SRC.AMT)
  ELSE IGNORE;
```

Db2 12 New Built-In Aggregate Functions

MEDIAN

```
SELECT MAX(COLCOUNT) AS MAX,  
       MIN(COLCOUNT) AS MIN,  
       AVG(COLCOUNT) AS AVG,  
       MEDIAN(COLCOUNT) AS MEDIAN  
FROM "SYSIBM".SYSTABLES;
```

MAX	MIN	AVG	MEDIAN
750	0	8	3

PERCENTILE_CONT and PERCENTILE_DISC

(PERCENTILE_DISC is always a value that appeared in the input set.)

```
SELECT MEDIAN(SALARY) AS MEDIAN,  
       PERCENTILE_CONT(.50) WITHIN GROUP  
       (ORDER BY SALARY) AS PCT_CONT,  
       PERCENTILE_DISC(.50) WITHIN GROUP  
       (ORDER BY SALARY) AS PCT_DISC  
FROM EMP ;
```

MEDIAN	PCT_CONT	PCT_DISC
49,545	49,545	49,250

Db2 12 New Built-In Scalar Functions

- GENERATE_UNIQUE_BINARY
 - 16 Byte STCKE built to Scale Beyond 256 Processors
- VARCHAR_BIT_FORMAT
 - Takes a string representation of a hex value and returns a bit string representation of that value
- HASH Functions
 - HASH_CRC32
 - HASH_MD5
 - HASH_SHA1
 - HASH_SHA256

Array Enhancements

- Arrays are extended to support
 - Global Variable Arrays
 - Arrays of LOBs (including Global Variable Arrays of LOBs)
- ARRAY_AGG function improvements:
 - ARRAY_AGG can be invoked without an ORDER BY clause.
 - ARRAY_AGG can be used with associative arrays.

Associative Array Example (V12)

```
-- Manipulate an Associative Array
CREATE OR REPLACE PROCEDURE MY_PROC (OUT SHOVELS ASSOC_ARRAY_VC)
BEGIN
    -- Build up Array
    SELECT ARRAY_AGG(NAME, PRICE) INTO SHOVELS FROM PRODUCT;

    -- Remove Ice Scraper
    SET SHOVELS = ARRAY_DELETE(SHOVELS, 'Ice Scraper, Windshield 4 inch');

END#
```

PRODUCT TABLE

NAME	PRICE
Snow Shovel, Basic 22 inch	9.99
Snow Shovel, Deluxe 24 inch	19.99
Snow Shovel, Super Deluxe 26 inch	49.99
Ice Scraper, Windshield 4 inch	3.99

INITIAL SHOVELS ARRAY

NAME	PRICE
Snow Shovel, Basic 22 inch	9.99
Snow Shovel, Deluxe 24 inch	19.99
Snow Shovel, Super Deluxe 26 inch	49.99
Ice Scraper, Windshield 4 inch	3.99

FINAL SHOVELS ARRAY

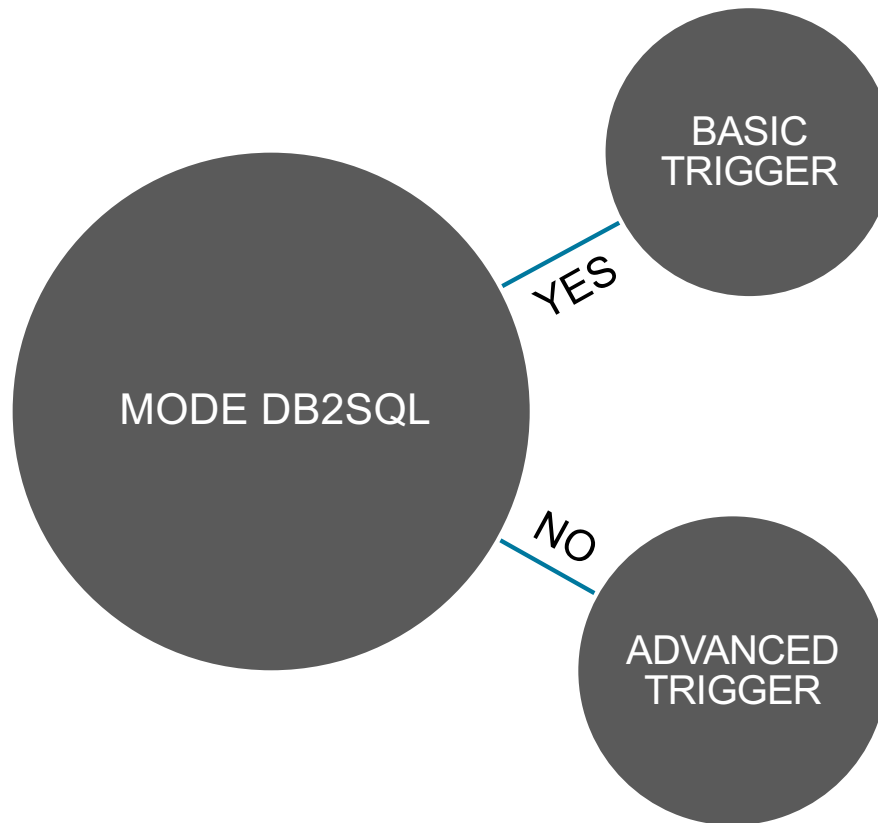
NAME	PRICE
Snow Shovel, Basic 22 inch	9.99
Snow Shovel, Deluxe 24 inch	19.99
Snow Shovel, Super Deluxe 26 inch	49.99



Db2 12 M500 SQLPL



Basic and Advanced Triggers



Trigger Body Can Now Contain Logic – Advanced Trigger

```
CREATE OR REPLACE TRIGGER TRIG_1
BEFORE INSERT ON T1
FOR EACH ROW

TR1: BEGIN
  DECLARE MAX_VAL INTEGER CONSTANT 100;
  DECLARE C1 INTEGER;

  IF (C1 = 1) THEN
    SET C1 = MAX_VAL;
  END IF;
END TR1#
```

TRIGGER Order Example

```
CREATE TABLE T1 (C1 INT) ;
```

```
CREATE SEQUENCE SEQ1 AS INT START WITH 1 INCREMENT BY 1 ;
```

```
CREATE TRIGGER TR1  
VERSION ONE  
BEFORE INSERT ON T1  
REFERENCING NEW AS NT1  
FOR EACH ROW  
TRG1: BEGIN  
    SET NT1.C1 = NEXTVAL FOR SEQ1 ;  
END TRG1
```

```
INSERT INTO T1 VALUES (100) ;
```

```
INSERT INTO T1 VALUES (100) ;
```

```
INSERT INTO T1 VALUES (100) ;
```

```
SELECT * FROM T1 ;
```

C1
2
4
6

TRIGGER Ordering Example (cont)

```
ALTER SEQUENCE SEQ1 RESTART WITH 1;
```

```
DROP TRIGGER TR1;
```

```
CREATE OR REPLACE TRIGGER TR1
```

```
VERSION ONE
```

```
BEFORE INSERT ON T1
```

```
REFERENCING NEW AS NT1
```

```
FOR EACH ROW
```

```
TRG1: BEGIN
```

```
    SET NT1.C1 = SEQ1.NEXTVAL;
```

```
END TRG1
```

```
INSERT INTO T1 VALUES (100);
```

```
INSERT INTO T1 VALUES (100);
```

```
INSERT INTO T1 VALUES (100);
```

```
SELECT * FROM T1;
```

C1
1
2
3

TRIGGER Ordering Example – CREATE OR REPLACE

```
ALTER SEQUENCE SEQ1 RESTART WITH 1;
```

```
CREATE OR REPLACE TRIGGER TR1
```

```
VERSION ONE
```

```
BEFORE INSERT ON T1
```

```
REFERENCING NEW AS NT1
```

```
FOR EACH ROW
```

```
TRG1: BEGIN
```

```
    SET NT1.C1 = SEQ1.NEXTVAL;
```

```
END TRG1
```

```
INSERT INTO T1 VALUES (100);
```

```
INSERT INTO T1 VALUES (100);
```

```
INSERT INTO T1 VALUES (100);
```

```
SELECT * FROM T1;
```

C1
2
4
6

TRIGGER Versioning

- TRIGGER Versioning - Enabling a TRIGGER to be replaced in the same execution order

```
CREATE OR REPLACE TRIGGER TR1
VERSION ONE
BEFORE INSERT ON T1
REFERENCING NEW AS NT1
FOR EACH ROW
TRG1: BEGIN
    SET NT1.C1 = NEXTVAL FOR SEQ1;
END TRG1
```


SQLPL Functions – Dynamic SQL Support

```
CREATE FUNCTION DYN_QUERY(SCH VARCHAR(128), TB VARCHAR(128)) RETURNS BIGINT
DETERMINISTIC NO EXTERNAL ACTION PARAMETER CCSID UNICODE READS SQL DATA
BEGIN

  DECLARE CNT BIGINT;
  DECLARE STMT_STR VARCHAR(256);
  DECLARE S1 STATEMENT;
  DECLARE C1 CURSOR FOR S1;

  -- Set up statement that will query the table that we want to Count
  SET STMT_STR = 'SELECT COUNT(*) FROM' || SCH || '.' || TB;
  PREPARE S1 FROM STMT_STR;

  OPEN C1;
  FETCH C1 INTO CNT;
  CLOSE C1;

  RETURN CNT;

END
```

SQL PL Obfuscation Support

```
SELECT WRAP('CREATE OR REPLACE TRIGGER trig1
BEFORE INSERT ON emp
REFERENCING NEW AS n FOR EACH ROW
WHEN (n.bonus IS NULL)
SET n.bonus = n.salary * .04')
FROM SYSIBM.SYSDUMMY1Copy
```

The result is similar to the following form:

```
CREATE TRIGGER trig1 WRAPPED DSN12015
```

```
abIGWmdiWmtyTmdUTmJqTmtaUmtGUnteUmZKWmtqWidaWmdaWmdaXmdyWncaGica
GK6ot_81NzyodncdrRIJFp_tBjpJelwg_dTKNHcdtHPSaNCpmqBKH2pMwExkRTJW
Zr:dJd0_gSbehW:4Xx1UGPGnDxvmJfa5ZAGOr_1sfFiyaPrkOXzt5UMTmsASfyJR
ksbPfM2dIATbq:0RW
```

Obfuscated Text



WRAP Function

➤ WRAP — (— *object-definition-string* —) ➤

- *object-definition-string*

A string of any built-in character type that contains any of the following data definition statements: CREATE FUNCTION (compiled SQL scalar)

- CREATE FUNCTION (inlined SQL scalar)
- CREATE PROCEDURE (SQL - native)
- CREATE PROCEDURE (SQL table)
- CREATE TRIGGER (basic)
- CREATE TRIGGER (advanced)

object-definition-string must not be bit data, and it cannot contain the VERSION keyword.

WRAP Example – Logic is Hidden, Statements are Visible

```
SELECT WRAP
(
CREATE OR REPLACE TRIGGER TR3
BEFORE INSERT ON T1
REFERENCING NEW AS NT1
FOR EACH ROW TRG3:
BEGIN
    SET NT1.C1 = C1*2;
END TRG3
') FROM SYSIBM.SYSDUMMYU;
```

Obfuscated Text

```
CREATE OR REPLACE TRIGGER TR3
WRAPPED DSN12100
ablGWmdiWmJaTmdmTmJmTmtGUmdeUm
ZqUnJG2nJa3idaWmdaWmdaWotCZcIa
GicaGhn2EhzI ve67reC12qiE :Lwl
:gjTnPv:gufTITcLRvb8gvzknffWhZ
Gt4Bzn_gzhsPUzMu5Be9AYT8moQDsw
h3UhfjERim6Ulxc3:BZ:5B07YNYT0s
_n:ik_HCSxPjtN6YCYqaa
```

Statement text in SYSIBM.SYSPACKSTMT is not obfuscated.

```
STATEMENT = UPDATE NT1 SET NT1.C1 = :H:H*2
```



Db2 12 M501 and Above



LISTAGG

– APPLCOMPAT(V12R1M501)

```
SELECT workdept, LISTAGG(ALL RTRIM(job),', ')  
WITHIN GROUP(ORDER BY RTRIM(job)) AS POSITION  
FROM emp  
GROUP BY workdept;
```



WORKDEPT	POSITION
A00	CLERK, CLERK, PRES, SALESREP, SALESREP
B01	MANAGER
C01	ANALYST, ANALYST, ANALYST, MANAGER
D11	DESIGNER, DESIGNER, DESIGNER, DESIGNER, DESIGNER, DESIGNER, DESIGNER, DESIGNER, DESIGNER, DESIGNER, MANAGER
D21	CLERK, CLERK, CLERK, CLERK, CLERK, CLERK, MANAGER
E01	MANAGER
E11	MANAGER, OPERATOR, OPERATOR, OPERATOR, OPERATOR, OPERATOR, OPERATOR
E21	FIELDREP, FIELDREP, FIELDREP, FIELDREP, FIELDREP, MANAGER

LISTAGG (cont)

– APPLCOMPAT(V12R1M501)

```
SELECT workdept, LISTAGG(DISTINCT RTRIM(job),'; ' )  
  WITHIN GROUP(ORDER BY RTRIM(job)) AS POSITION  
FROM emp  
GROUP BY workdept;
```

WORKDEPT	POSITION
A00	CLERK; PRES; SALESREP
B01	MANAGER
C01	ANALYST; MANAGER
D11	DESIGNER; MANAGER
D21	CLERK; MANAGER
E01	MANAGER
E11	MANAGER; OPERATOR
E21	FIELDREP; MANAGER



Explicit Casting to GRAPHIC/VARGRAPHIC

– APPLCOMPAT(V12R1M502)

Support Casting of Numeric to Unicode UTF-16 (GRAPHIC/VARGRAPHIC)

```
SELECT FIRSTNME, LASTNAME,  
       GRAPHIC(SALARY + BONUS + COMM) AS TOTAL_COMP  
FROM U_EMP WHERE LASTNAME = 'HAAS';
```

FIRSTNME	LASTNAME	TOTAL_COMP
CHRISTINE	HAAS	157970.00



Temporal Auditing

- APPLCOMPAT(V12R1M503)

- **Temporal Auditing Changed Behavior**
- The temporal query result change for system-period temporal tables defined with the ON DELETE ADD EXTRA ROW attribute that also contain a DATA CHANGE OPERATION column.
- Rows that contain null values in the history table column that corresponds to the DATA CHANGE OPERATION column will now be considered part of the intermediate result set for a system-period temporal query. However, rows that were added to a history table for the ON DELETE ADD EXTRA ROW attribute will still be excluded.
- Application behavior is tied to APPLCOMPAT(V12R1M503)
- **REPLICATION_OVERRIDE**
- New Global Variable to enable replication of system-period temporal tables.
- Replication SW must be bound APPLCOMPAT(V12R1M503)



IDAA BIF Passthrough

- APPLCOMPAT(V12R1M504)

•Function level 504 introduces support for the following passthrough-only built-in functions, which are passed through from Db2 for z/OS to IBM Db2 Analytics Accelerator:

- [CUME_DIST](#)
- [CUME_DIST \(aggregate\)](#)
- [FIRST_VALUE](#)
- [LAG](#)
- [LAST_VALUE](#)
- [LEAD](#)
- [NTH_VALUE](#)
- [NTILE](#)
- [PERCENT_RANK](#)
- [PERCENT_RANK \(aggregate\)](#)
- [RATIO_TO_REPORT](#)
- [REGEXP_COUNT](#)
- [REGEXP_INSTR](#)
- [REGEXP_LIKE](#)
- [REGEXP_REPLACE](#)
- [REGEXP_SUBSTR](#)



New Syntax Alternative – APPLCOMPAT(V12R1M504)



Existing Syntax	New Syntax Alternatives
CURRENT CLIENT_ACCTNG	CLIENT ACCTNG
CURRENT CLIENT_APPLNAME	CLIENT APPLNAME
CURRENT CLIENT_USERID	CLIENT USERID
CURRENT CLIENT_WRKSTNNAME	CLIENT WRKSTNNAME
CURRENT SERVER	CURRENT_SERVER
CURRENT TIME_ZONE CURRENT TIMEZONE	CURRENT_TIMEZONE
IS NULL	ISNULL
IS NOT NULL	NOTNULL

ENCRYPT_DATAKEY, DECRYPT_DATAKEY – APPLCOMPAT(V12R1M505)

```
CREATE TABLE T1  
(NAME CHAR(20), PHONE_NUMBER VARBINARY(95));
```

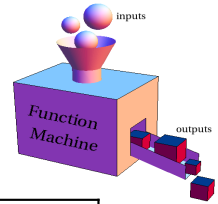
```
INSERT INTO T1 VALUES  
('Chris', ENCRYPT_DATAKEY('408-867-5309', 'MYKEYLABEL', AES256D));
```

```
SELECT  
NAME, DECRYPT_DATAKEY_VARCHAR(PHONE_NUMBER)  
FROM SAMPLE_TABLE;
```


News from the Lab Blog on ENCRYPT_DATAKEY -
<https://ibm.co/39M55UW>



New Function Syntax Alternatives – APPLCOMPAT(V12R1M506)



Newly supported alternative name	Existing equivalent function
CHAR_LENGTH	CHARACTER_LENGTH, which returns the length of its argument in the number of string units that are specified
COVAR_POP	COVARIANCE or COVAR, which return the population covariance of a set of number pairs
HASH	HASH_MD5, HASH_SHA1, or HASH_SHA256, which return the result of applying a hash algorithm to an input argument, depending on the value specified for the second argument for the HASH function:
	0 (default)
	HASH_MD5
	HASH_SHA1
	HASH_SHA256
HASH	The HASH function returns a varying length (VARBINARY) value, unlike the existing functions, which return fixed length (BINARY) values.
POW	POWER, which returns the value of one argument raised to the power of a second argument
RANDOM	RANDOM, which returns a double precision floating-point random number
STRLEFT	LEFT, which returns a string that consists of the specified number of leftmost bytes or the specified string units
STRPOS	POSSTR, which returns the position of the first occurrence of an argument within another argument
STRRIGHT	RIGHT, which returns a string that consists of the specified number of rightmost bytes or specified string units
TO_CLOB	CLOB, which returns a CLOB representation of the first argument
TO_TIMESTAMP	TIMESTAMP_FORMAT, which returns a timestamp for a character string expression, using a specified format to interpret the string



NUMLKUS and NUMLKTS Global Variables – APPLCOMPAT(V12R1M507)

New built-in global variables support granularity for locking limits:

SYSIBMADM.MAX_LOCKS_PER_TABLESPACE

An integer value for the maximum number of page, row, or LOB locks that the application can hold simultaneously in a table space. If the application exceeds the maximum number of locks in a single table space, lock escalation occurs.

Corresponds to the existing NUMLKTS subsystem parameter.

SYSIBMADM.MAX_LOCKS_PER_USER

An integer value integer value that specifies the maximum number of page, row, or LOB locks that a single application can concurrently hold for all table spaces.

The limit applies to all table spaces that are defined with the LOCKSIZE PAGE, LOCKSIZE ROW, or LOCKSIZE ANY options. MAX_LOCKS_PER_USER.

Corresponds to the existing NUMLKUS subsystem parameter.

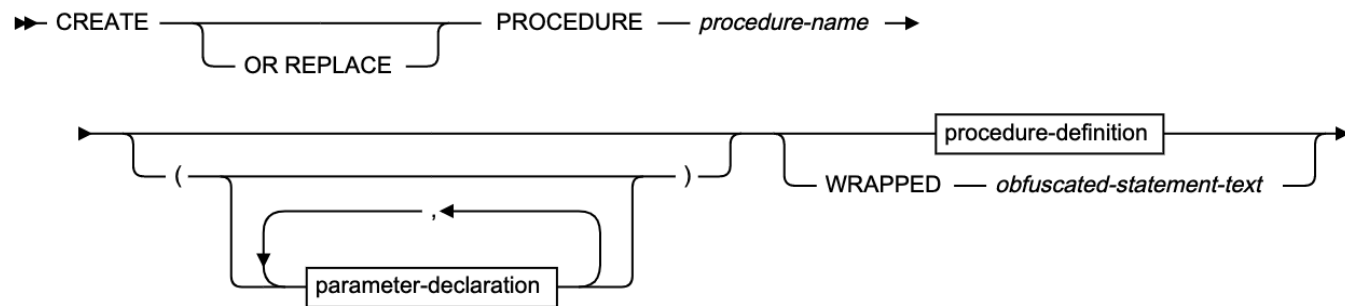
IDAA BIF Passthrough – APPLCOMPAT(V12R1M507)

- The following built-in functions are now supported as passthrough-only expressions, which are passed through from Db2 for z/OS® to IBM Db2 Analytics Accelerator:
- [ADD DAYS](#)
- [BTRIM](#)
- [DAYS BETWEEN](#)
- [NEXT MONTH](#)
- [Regression Functions](#)
(REGR_AVGX, REGR_AVGY, REGR_COUNT, ...)
- [ROUND TIMESTAMP](#) (if invoked with a date expression)

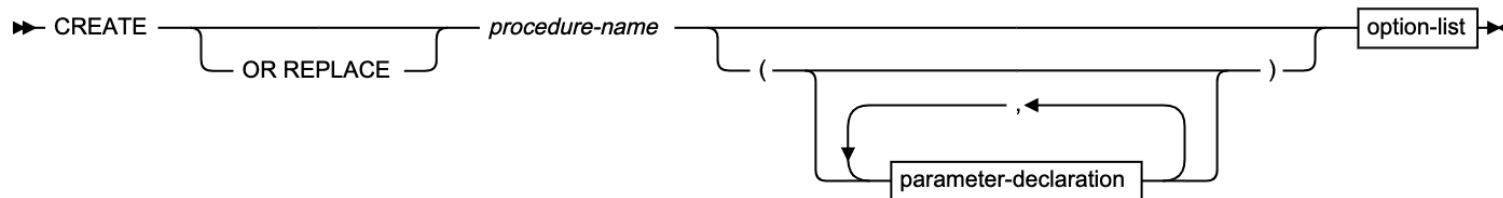
CREATE OR REPLACE for procedures

- APPLCOMPAT(V12R1M507)

Syntax - Native SQL Procedures



Syntax - External Procedures



Temporal RI allows UPDATE or DELETE on the parent table – APPLCOMPAT(V12R1M509)

When an UPDATE statement with a FOR PORTION OF clause attempts to update the parent table in a temporal RI relationship, the update is allowed as long as the rules of temporal RI are not violated.

When a DELETE statement with a FOR PORTION OF clause attempts to delete from the parent table in a temporal RI relationship, the deletion is allowed, as long as the rules of temporal RI are not violated.

At any lower application compatibility level, such UPDATE or DELETE statements for a parent table in an RI relationship are restricted with SQLCODE -4736.





**THIS
CHANGES
EVERY
NOTHING**

FL 510

Activation of FL 510 activates all previous FLs and prepares the system for migration to Db2 13



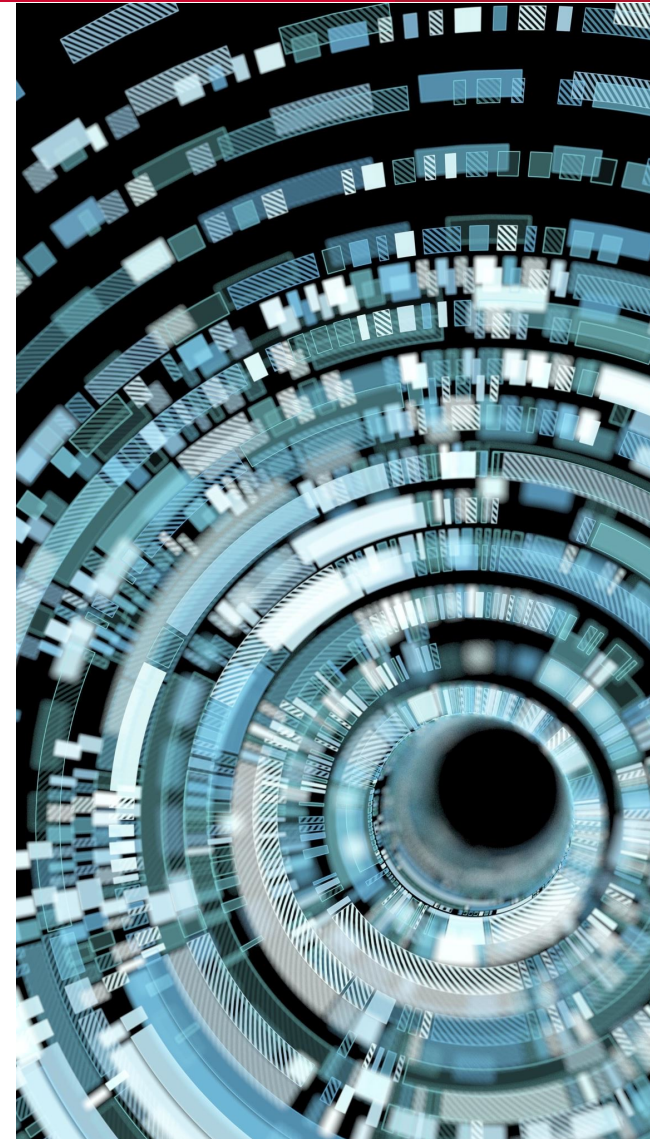
Db2 13 M500 and Above



SQL Data Insights

```
SELECT
  AI_SIMILARITY(X.customerID,'3668-QPYBK') AS SimilarityScore, X.*
FROM DSNAIDB.CHURN X
WHERE X.customerID <> '3668-QPYBK'
ORDER BY SimilarityScore DESC
FETCH FIRST 10 ROWS ONLY
```

Db2 built-in AI semantic function	Semantic query type
AI_SIMILARITY	Similarity query
AI_SIMILARITY	Dissimilarity query
AI_SEMANTIC_CLUSTER	Inductive reasoning semantic clustering query
AI_ANALOGY	Inductive reasoning analogy query

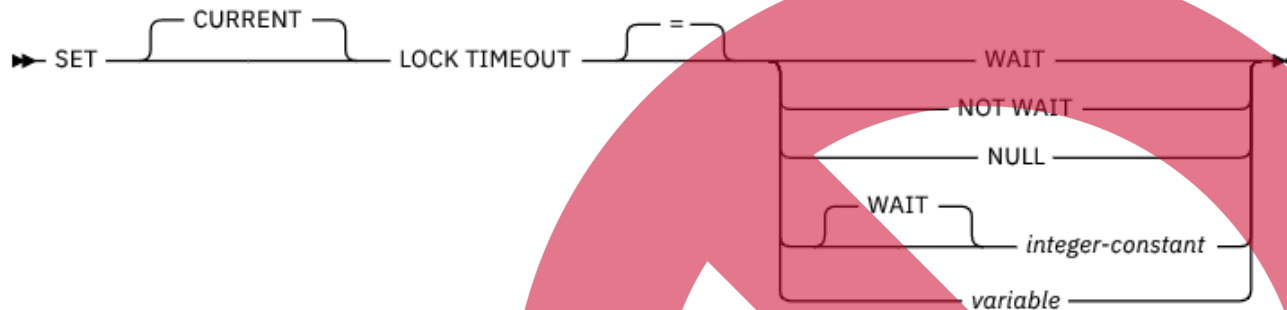


Column names longer than 30 bytes



- You can define a column with a name longer than 30 EBCDIC bytes up to 128 bytes.
 - When TABLE_COL_NAME_EXPANSION = ON. This can occur in the following situations:
 - When objects such as tables, views, or indexes define new columns or rename existing columns.
 - INSERT, UPDATE, DELETE or MERGE statements define **include columns**.
 - Common table expressions that explicitly specify column names.
 - An AS clause in a SELECT or FROM clause that defines an alternate name for a column.
- Limitations for column names longer than 30 bytes
 - Column names with a length greater than 30 bytes of EBCDIC might be truncated on a character boundary when they are returned in an SQLDA.
 - APIs that do not use the SQLDA to obtain a column name may return complete column names.
 - Example interfaces, which use an SQLDA to get column names, will return at most 30 bytes of a column name:
 - Direct use of an SQLDA
 - SPUFI, QMF, DSNTDP2, DSNTDP4, DSNTIAD, DSNTIAUL, and DCLGEN
 - ODBC APIs such as SQLDescribeCol and SQLColAttributes (or SQLColAttribute)
 - A truncated column name may unintentionally be interpreted by Db2 as a reference to another column that happens to have that truncated name, a warning may be returned with the truncated name, or an error may be returned rather than truncating the name.

CURRENT LOCK TIMEOUT APPLCOMPAT V13R1M500



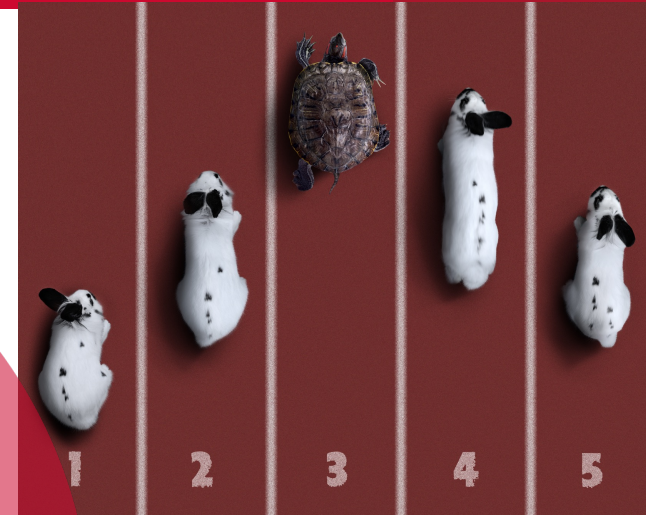
- Use the new SPREG_LOCK_TIMEOUT zPARM parameter to control the maximum value that can be specified for the SET CURRENT LOCK TIMEOUT statement.
 - The default value for SPREG_LOCK_TIMEOUT_MAX is -1, which allows all valid values to be set for the CURRENT LOCK TIMEOUT.
- Monitor IFCID 196 and/or new text in the DSNT376:

“HOLDER USING TIMEOUT VALUE tout-interval2 FROM timeout-source2.”

 - timeout-source2 can be: IRLMRWT, Special Register or IRLM
- Unlike Global Variables – Special registers have no Authorization – use the zPARM defensively

DEADLOCK_RESOLUTION_PRIORITY APPLCOMPAT V13R1M501

- The type is SMALLINT.
- The schema is SYSIBMADM.
- The scope of this global variable is session.
- The default value is NULL or DEFAULT.
- A user with the **WRITE** privilege on the global variable can “SET” the GV
- The acceptable range of values is 0 - 255. The higher the value, the less likely that lock requests acquired by an application will deadlock when the application is involved in a deadlock situation.



APPLCOMPAT V13R1M503

- Support for OPTIMIZE for N ROWS in SELECT INTO
 - SELECT CURRENT TIMESTAMP INTO :HV
FROM SYSIBM.SYSDUMMY1 FETCH FIRST ONE ROW ONLY OPTIMIZE FOR 2 ROWS;
FETCH FIRST 1 ROW ONLY causes Db2 to use a sort avoidance preference that is associated with OPTIMIZE FOR 1 ROWS during access path (AP) selection.
However, sometimes avoiding sort can result in a more expensive (in total cost) access path.
If this occurs, OPTIMIZE FOR 2 ROWS enables Db2 to consider an AP that uses a sort.
- Support for IN list predicates with more than 32K elements when the query is offloaded to IDAA
- Default value change for ROW CHANGE TIMESTAMPE
 - ALTER TABLE Timestamp placed in SYSCOLUMNS DEFAULTVALUE column



Summary



Summary

- Db2 12 has delivered
 - Increased Db2 Family Compatibility
 - MERGE
 - ARRAY
 - LIMIT/OFFSET
 - SQLPL
 - Syntax Alternatives
 - IDAA Passthrough
 - Increased support for OLAP and Analytical SQL
 - Improved APS
 - IDAA Offload and Passthrough
 - Improved Native OLAP support with functions such as MEDIAN
 - Improved SQL PL Capability
 - Triggers
 - Constants
 - Obfuscation
- Db2 13 has delivered
 - AI capability with SQL Data Insights
 - Improved Db2 capability for applications that need Longer Column names, Lock Timeout Flexibility, and Deadlock Detection Flexibility – all three of these features are use at your own risk.



Questions?





Chris Crone
cjc@broadcom.com