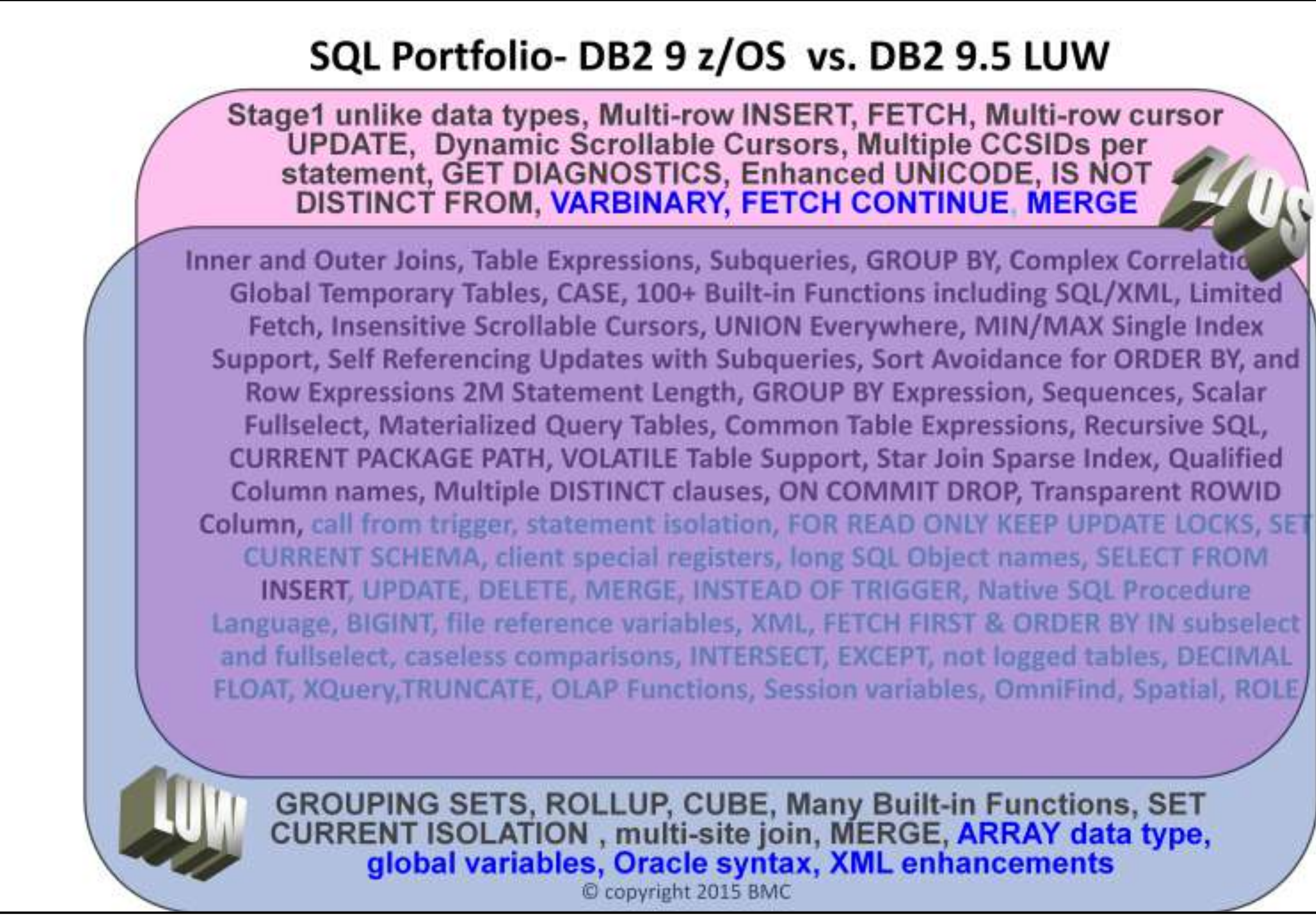
Advanced and Complex SQL

DB2 9, 10 & 11

Newer DB2 Features (DB2 9,10,11)

Call from trigger, statement isolation, FOR READ ONLY KEEP UPDATE LOCKS, SET CURRENT SCHEMA, Client special registers, long SQL object names, SELECT from INSERT, UPDATE or DELETE, INSTEAD OF TRIGGER, SQL PL in routines, BIGINT, file reference variables, XML, FETCH FIRST & ORDER BY in subselect & fullselect, caseless comparisons, INTERSECT, EXCEPT, MERGE not logged tables, OmniFind, spatial, range partitions, data compression, DECFLOAT, optimistic locking, ROLE, TRUNCATE, index & XML compression, created temps, inline LOB, administrative privileges, implicit cast, increased timestamp precision, currently committed, moving sum & average, index include columns, row and column access controls, time travel query, GROUPING SETS, ROLLUP, CUBE, global variables, Text Search functions, accelerated tables, DROP COLUMN, array data type, XML enhancements



SQL Portfolio- DB2 9 z/OS vs. DB2 9.5 LUW

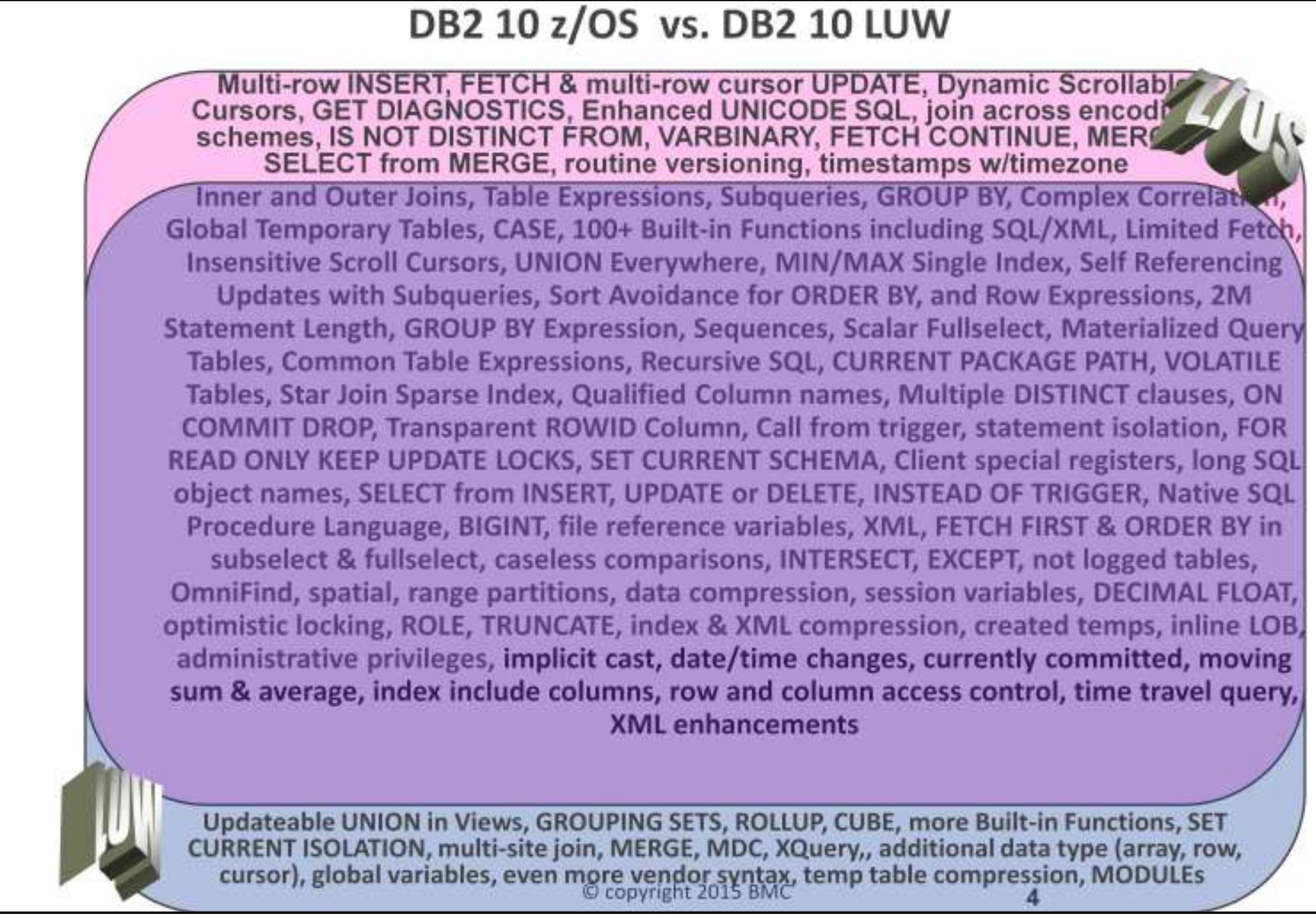
Stage1 unlike data types, Multi-row INSERT, FETCH, Multi-row cursor UPDATE, Dynamic Scrollable Cursors, Multiple CCSIDs per statement, GET DIAGNOSTICS, Enhanced UNICODE, IS NOT DISTINCT FROM, VARBINARY, FETCH CONTINUE, MERGE

Z/OS

Inner and Outer Joins, Table Expressions, Subqueries, GROUP BY, Complex Correlatio Global Temporary Tables, CASE, 100+ Built-in Functions including SQL/XML, Limited Fetch, Insensitive Scrollable Cursors, UNION Everywhere, MIN/MAX Single Index Support, Self Referencing Updates with Subqueries, Sort Avoidance for ORDER BY, and Row Expressions 2M Statement Length, GROUP BY Expression, Sequences, Scalar Fullselect, Materialized Query Tables, Common Table Expressions, Recursive SQL, CURRENT PACKAGE PATH, VOLATILE Table Support, Star Join Sparse Index, Qualified Column names, Multiple DISTINCT clauses, ON COMMIT DROP, Transparent ROWID Column, call from trigger, statement isolation, FOR READ ONLY KEEP UPDATE LOCKS, SET CURRENT SCHEMA, client special registers, long SQL Object names, SELECT FROM INSERT, UPDATE, DELETE, MERGE, INSTEAD OF TRIGGER, Native SQL Procedure Language, BIGINT, file reference variables, XML, FETCH FIRST & ORDER BY IN subselect and fullselect, caseless comparisons, INTERSECT, EXCEPT, not logged tables, DECIMAL FLOAT, XQuery,TRUNCATE, OLAP Functions, Session variables, OmniFind, Spatial, ROLE

LUW

GROUPING SETS, ROLLUP, CUBE, Many Built-in Functions, SET CURRENT ISOLATION , multi-site join, MERGE, ARRAY data type, global variables, Oracle syntax, XML enhancements

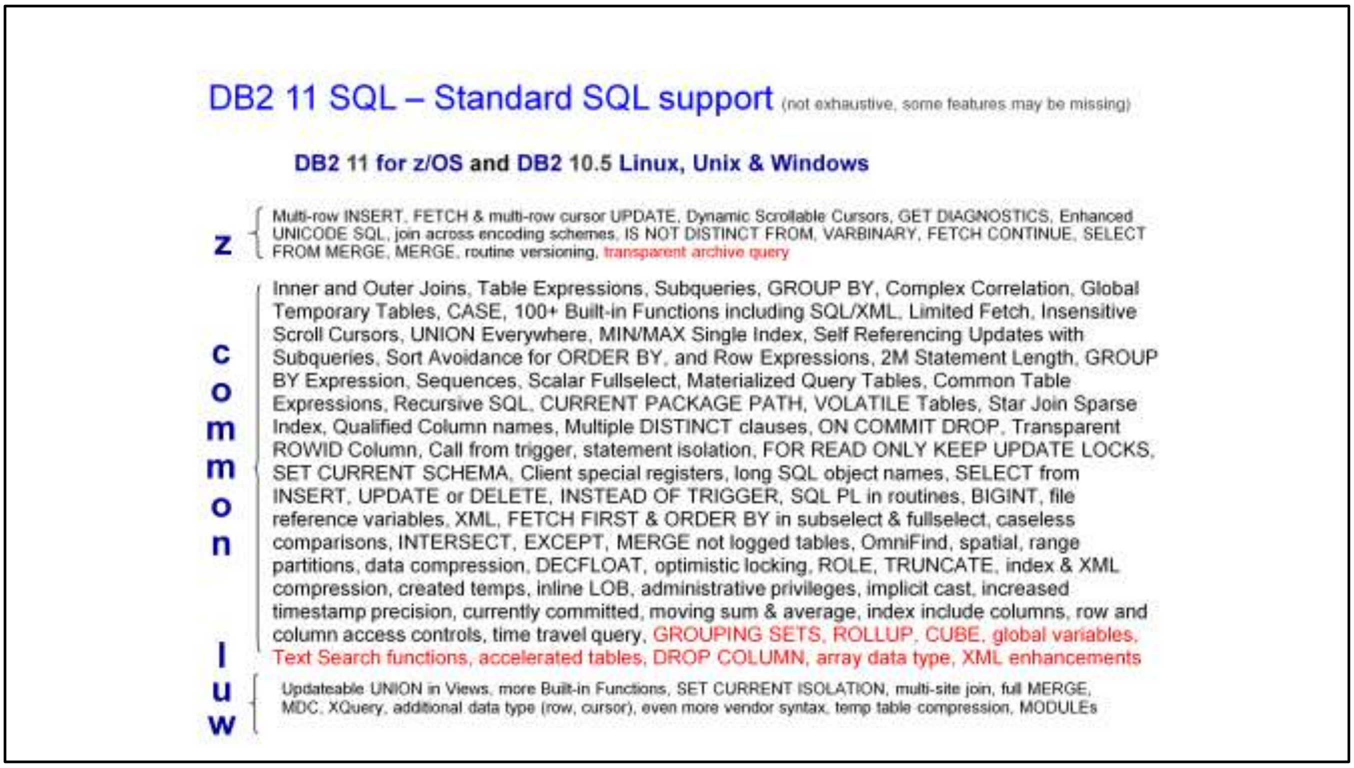


DB2 10 z/OS vs. DB2 10 LUW

Multi-row INSERT, FETCH & multi-row cursor UPDATE, Dynamic Scrollabl Cursors, GET DIAGNOSTICS, Enhanced UNICODE SQL, join across encodi schemes, IS NOT DISTINCT FROM, VARBINARY, FETCH CONTINUE, MERG SELECT from MERGE, routine versioning, timestamps w/timezone

Inner and Outer Joins, Table Expressions, Subqueries, GROUP BY, Complex Correlat n, Global Temporary Tables, CASE, 100+ Built-in Functions including SQL/XML, Limited Fetch, Insensitive Scroll Cursors, UNION Everywhere, MIN/MAX Single Index, Self Referencing Updates with Subqueries, Sort Avoidance for ORDER BY, and Row Expressions, 2M Statement Length, GROUP BY Expression, Sequences, Scalar Fullselect, Materialized Query Tables, Common Table Expressions, Recursive SQL, CURRENT PACKAGE PATH, VOLATILE Tables, Star Join Sparse Index, Qualified Column names, Multiple DISTINCT clauses, ON COMMIT DROP, Transparent ROWID Column, Call from trigger, statement isolation, FOR READ ONLY KEEP UPDATE LOCKS, SET CURRENT SCHEMA, Client special registers, long SQL object names, SELECT from INSERT, UPDATE or DELETE, INSTEAD OF TRIGGER, Native SQL Procedure Language, BIGINT, file reference variables, XML, FETCH FIRST & ORDER BY in subselect & fullselect, caseless comparisons, INTERSECT, EXCEPT, not logged tables, OmniFind, spatial, range partitions, data compression, session variables, DECIMAL FLOAT, optimistic locking, ROLE, TRUNCATE, index & XML compression, created temps, inline LOB, administrative privileges, **implicit cast, date/time changes, currently committed, moving sum & average, index include columns, row and column access control, time travel query, XML enhancements**

Updateable UNION in Views, GROUPING SETS, ROLLUP, CUBE, more Built-in Functions, SET CURRENT ISOLATION, multi-site join, MERGE, MDC, XQuery,, additional data type (array, row, cursor), global variables, even more vendor syntax, temp table compression, MODULEs



DB2 11 SQL – Standard SQL support (not exhaustive, some features may be missing)

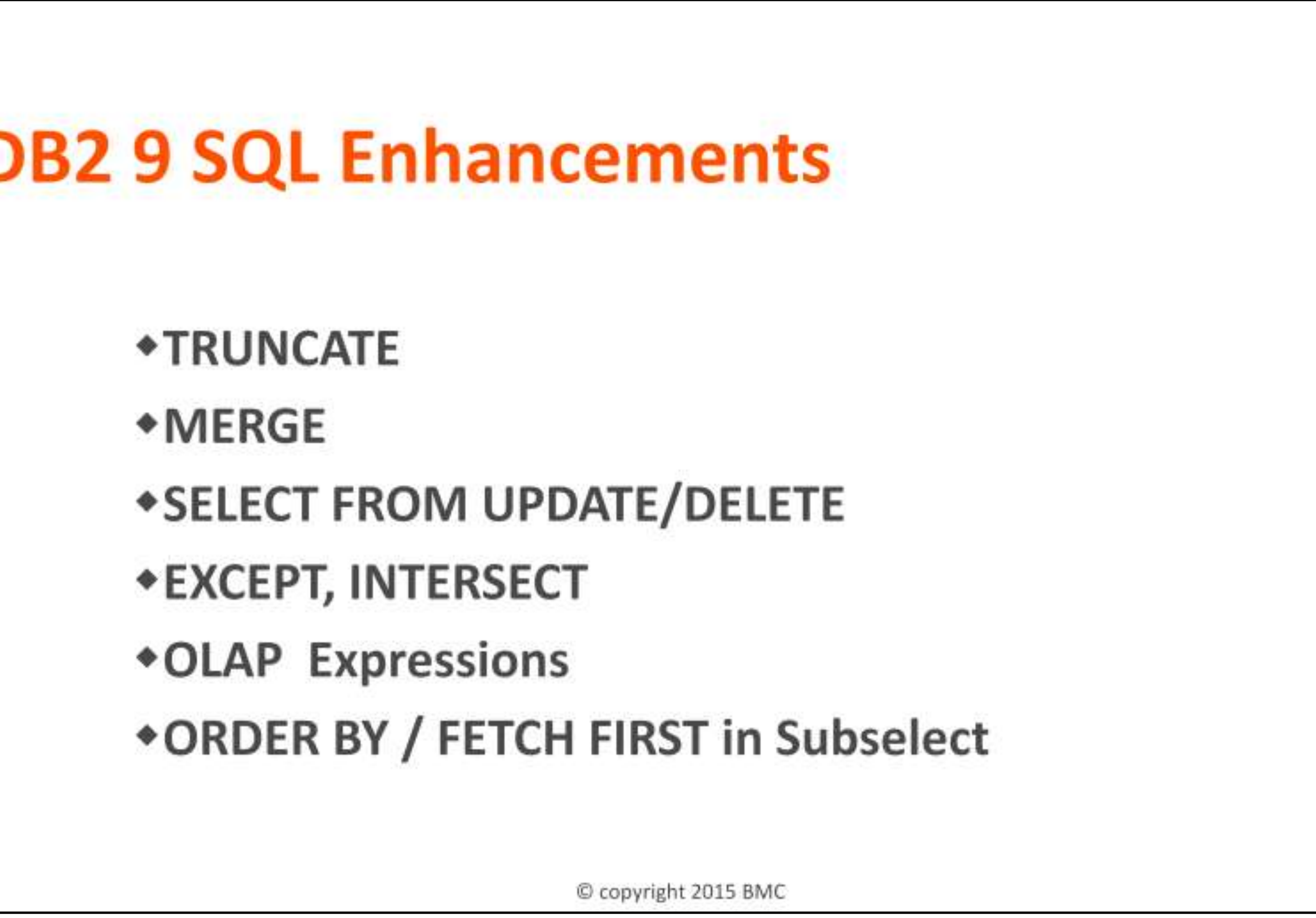
DB2 11 for z/OS and DB2 10.5 Linux, Unix & Windows

z: Multi-row INSERT, FETCH & multi-row cursor UPDATE, Dynamic Scrollable Cursors, GET DIAGNOSTICS, Enhanced UNICODE SQL, join across encoding schemes, IS NOT DISTINCT FROM, VARBINARY, FETCH CONTINUE, SELECT FROM MERGE, MERGE, routine versioning, transparent archive query

common: Inner and Outer Joins, Table Expressions, Subqueries, GROUP BY, Complex Correlation, Global Temporary Tables, CASE, 100+ Built-in Functions including SQL/XML, Limited Fetch, Insensitive Scroll Cursors, UNION Everywhere, MIN/MAX Single Index, Self Referencing Updates with Subqueries, Sort Avoidance for ORDER BY, and Row Expressions, 2M Statement Length, GROUP BY Expression, Sequences, Scalar Fullselect, Materialized Query Tables, Common Table Expressions, Recursive SQL, CURRENT PACKAGE PATH, VOLATILE Tables, Star Join Sparse Index, Qualified Column names, Multiple DISTINCT clauses, ON COMMIT DROP, Transparent ROWID Column, Call from trigger, statement isolation, FOR READ ONLY KEEP UPDATE LOCKS, SET CURRENT SCHEMA, Client special registers, long SQL object names, SELECT from INSERT, UPDATE or DELETE, INSTEAD OF TRIGGER, SQL PL in routines, BIGINT, file reference variables, XML, FETCH FIRST & ORDER BY in subselect & fullselect, caseless comparisons, INTERSECT, EXCEPT, MERGE not logged tables, OmniFind, spatial, range partitions, data compression, DECFLOAT, optimistic locking, ROLE, TRUNCATE, index & XML compression, created temps, inline LOB, administrative privileges, implicit cast, increased timestamp precision, currently committed, moving sum & average, index include columns, row and column access controls, time travel query, GROUPING SETS, ROLLUP, CUBE, global variables, Text Search functions, accelerated tables, DROP COLUMN, array data type, XML enhancements

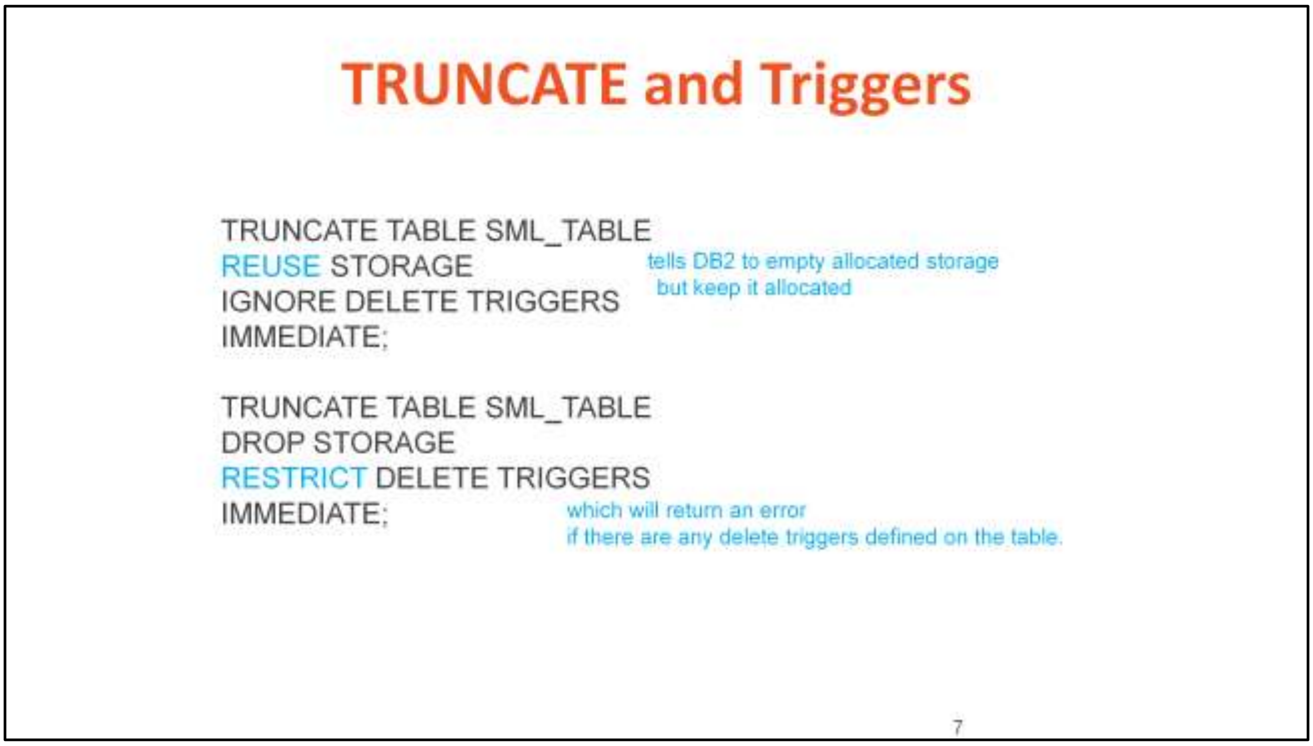
luw: Updateable UNION in Views, more Built-in Functions, SET CURRENT ISOLATION, multi-site join, full MERGE, MDC, XQuery, additional data type (row, cursor), even more vendor syntax, temp table compression, MODULEs

This chart shows the relationship of DB2 for Linux, Unix & Windows with D
There are three sets of SQL noted above, with some that is unique to DB2
The Cross-Platform SQL Reference Version 4.1 documents the prior comb
Cross-Platform Development Version 4.1, http://www.ibm.com/developerworks/db2/library/techarticle/



B2 9 SQL Enhancements

- TRUNCATE
- MERGE
- SELECT FROM UPDATE/DELETE
- EXCEPT, INTERSECT
- OLAP Expressions
- ORDER BY / FETCH FIRST in Subselect



TRUNCATE and Triggers

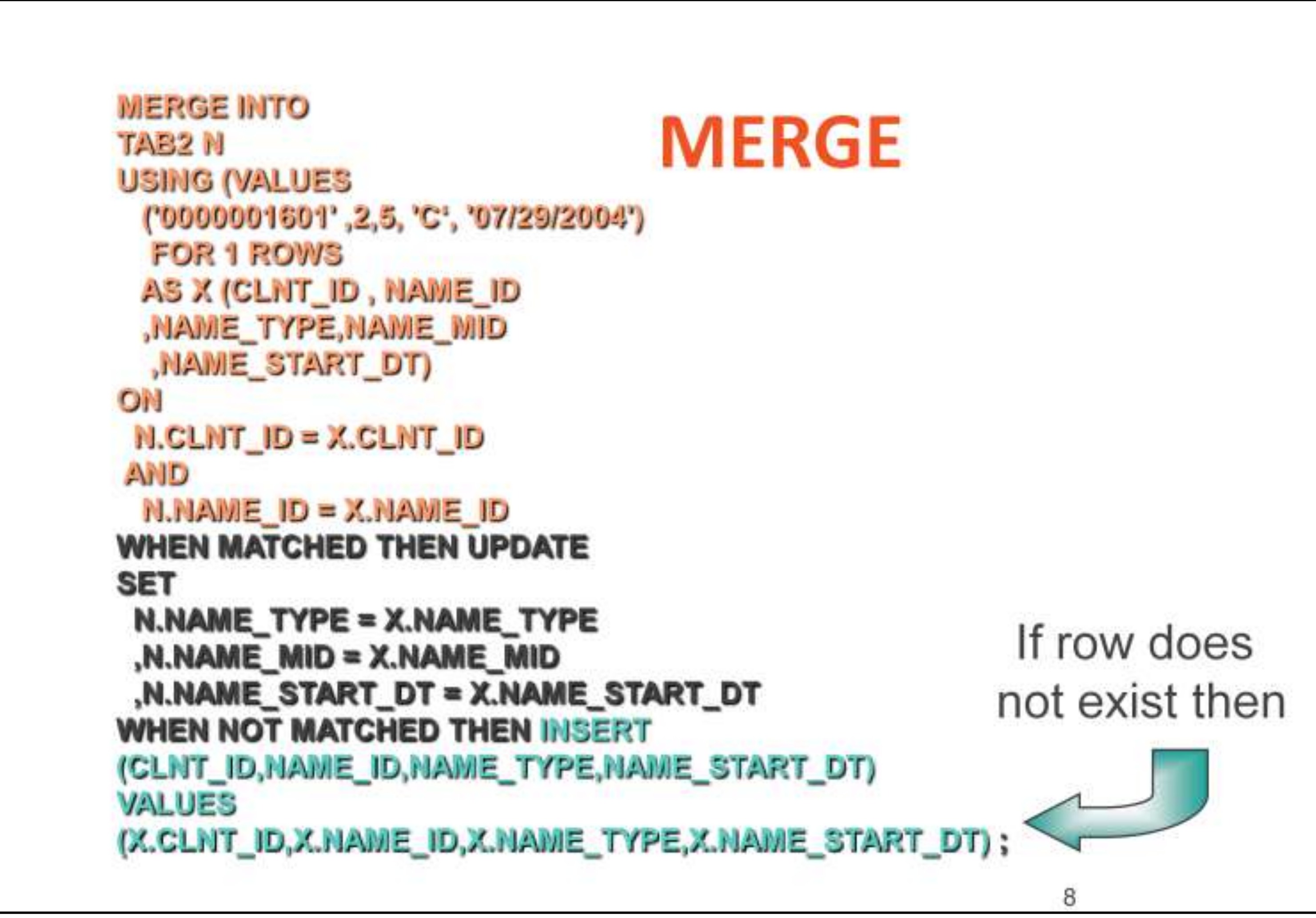
```
TRUNCATE TABLE SML_TABLE
REUSE STORAGE
IGNORE DELETE TRIGGERS
IMMEDIATE;
```

tells DB2 to empty allocated storage but keep it allocated

```
TRUNCATE TABLE SML_TABLE
DROP STORAGE
RESTRICT DELETE TRIGGERS
IMMEDIATE;
```

which will return an error if there are any delete triggers defined on the table.

7

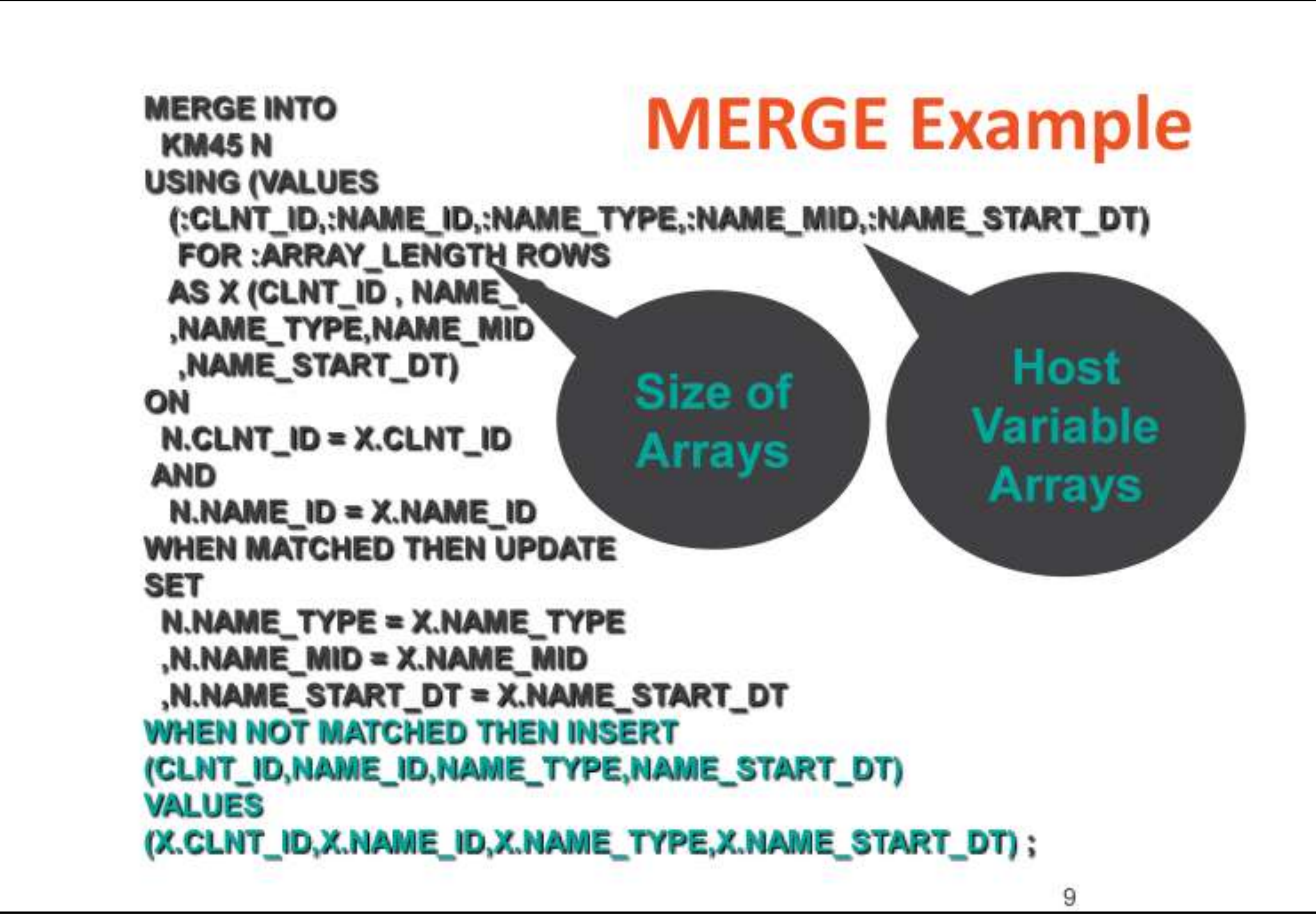


MERGE

```
MERGE INTO
TAB2 N
USING (VALUES
   ('0000001601' ,2,5, 'C', '07/29/2004')
    FOR 1 ROWS
   AS X (CLNT_ID , NAME_ID
   ,NAME_TYPE,NAME_MID
    ,NAME_START_DT)
ON
  N.CLNT_ID = X.CLNT_ID
 AND
   N.NAME_ID = X.NAME_ID
WHEN MATCHED THEN UPDATE
SET
  N.NAME_TYPE = X.NAME_TYPE
  ,N.NAME_MID = X.NAME_MID
  ,N.NAME_START_DT = X.NAME_START_DT
WHEN NOT MATCHED THEN INSERT
(CLNT_ID,NAME_ID,NAME_TYPE,NAME_START_DT)
VALUES
(X.CLNT_ID,X.NAME_ID,X.NAME_TYPE,X.NAME_START_DT) ;
```

If row does not exist then

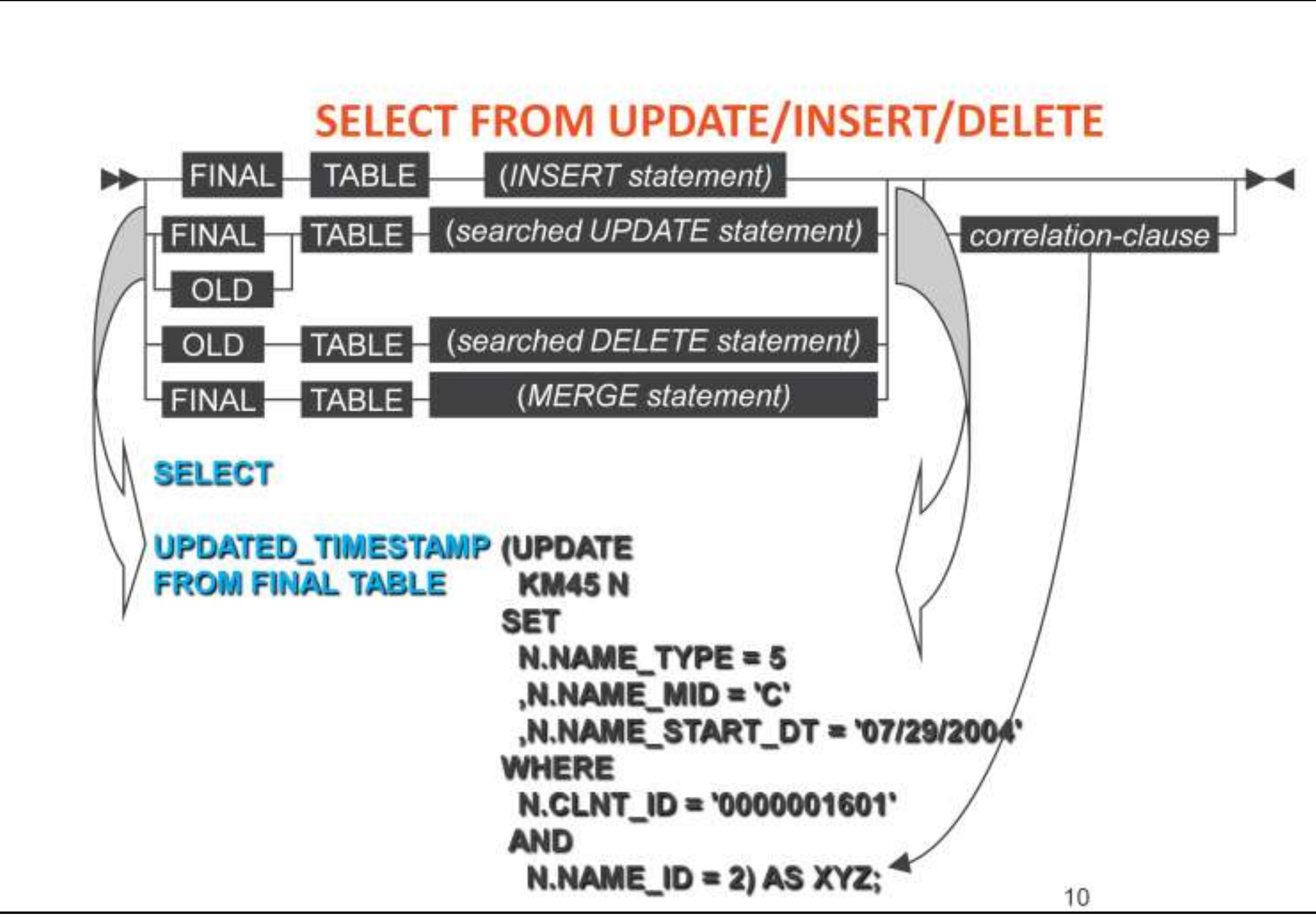
8


Size of Arrays
Host Variable Arrays

MERGE Example

```
MERGE INTO
  KM45 N
USING (VALUES
   (:CLNT_ID,:NAME_ID,:NAME_TYPE,:NAME_MID,:NAME_START_DT)
    FOR :ARRAY_LENGTH ROWS
   AS X (CLNT_ID , NAME_
   ,NAME_TYPE,NAME_MID
    ,NAME_START_DT)
ON
  N.CLNT_ID = X.CLNT_ID
 AND
   N.NAME_ID = X.NAME_ID
WHEN MATCHED THEN UPDATE
SET
  N.NAME_TYPE = X.NAME_TYPE
  ,N.NAME_MID = X.NAME_MID
  ,N.NAME_START_DT = X.NAME_START_DT
WHEN NOT MATCHED THEN INSERT
(CLNT_ID,NAME_ID,NAME_TYPE,NAME_START_DT)
VALUES
(X.CLNT_ID,X.NAME_ID,X.NAME_TYPE,X.NAME_START_DT) ;
```

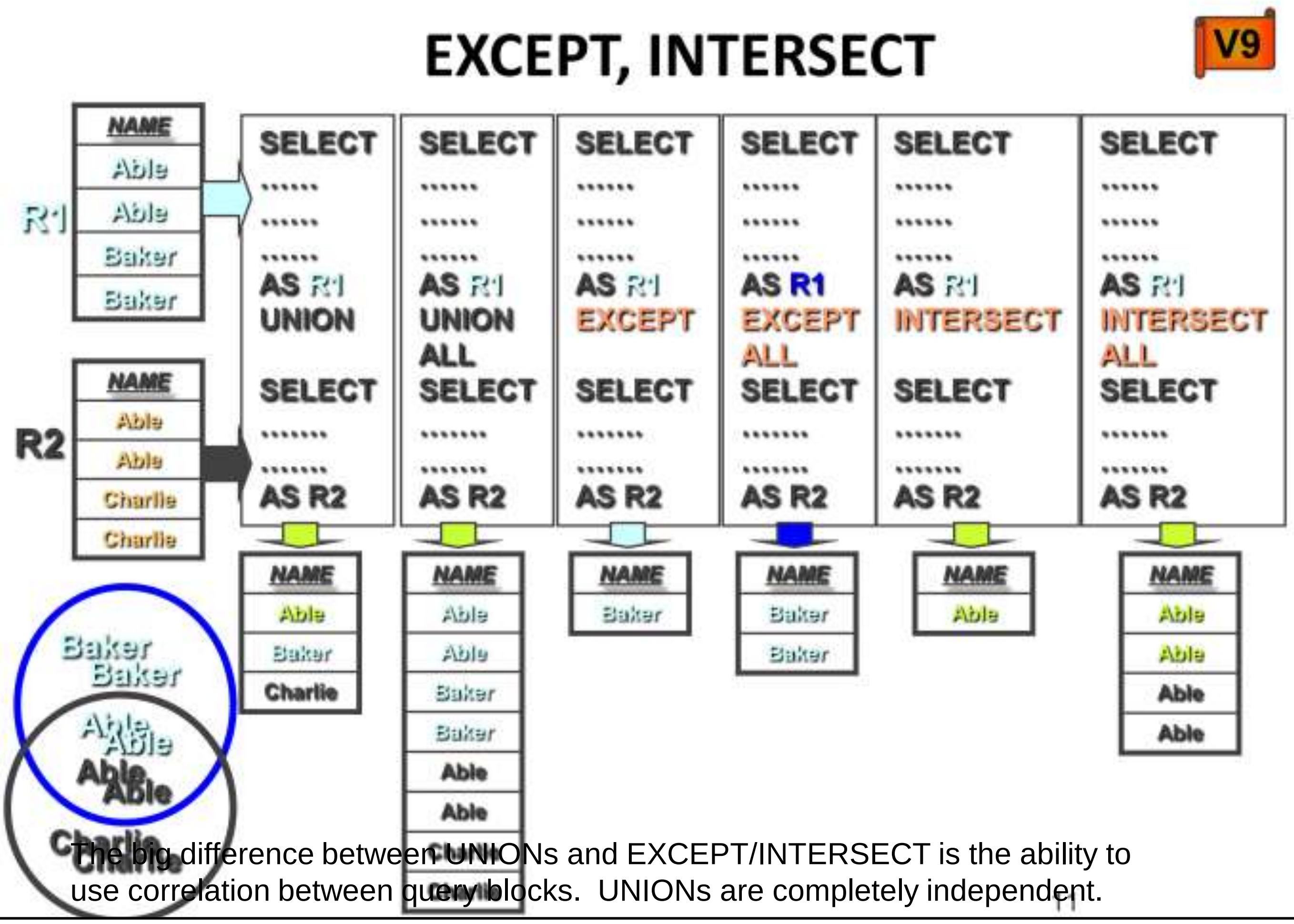
9


FINAL
TABLE
(INSERT statement)
FINAL
OLD
TABLE
(searched UPDATE statement)
OLD
TABLE
(searched DELETE statement)
FINAL
TABLE
(MERGE statement)
correlation-clause

SELECT FROM UPDATE/INSERT/DELETE

```
SELECT

UPDATED_TIMESTAMP (UPDATE
FROM FINAL TABLE    KM45 N
                   SET
                    N.NAME_TYPE = 5
                    ,N.NAME_MID = 'C'
                    ,N.NAME_START_DT = '07/29/2004'
                   WHERE
                    N.CLNT_ID = '0000001601'
                    AND
                     N.NAME_ID = 2) AS XYZ;
```

10



EXCEPT, INTERSECT

V9

R1

NAME
Able
Able
Baker
Baker

R2

NAME
Able
Able
Charlie
Charlie

```
SELECT
......
......
......
AS R1
UNION

SELECT
.......
.......
AS R2
```

```
SELECT
......
......
......
AS R1
UNION
ALL
SELECT
.......
.......
AS R2
```

```
SELECT
......
......
......
AS R1
EXCEPT

SELECT
.......
.......
AS R2
```

```
SELECT
......
......
......
AS R1
EXCEPT
ALL
SELECT
.......
.......
AS R2
```

```
SELECT
......
......
......
AS R1
INTERSECT

SELECT
.......
.......
AS R2
```

```
SELECT
......
......
......
AS R1
INTERSECT
ALL
SELECT
.......
.......
AS R2
```

NAME
Able
Baker
Charlie

NAME
Able
Able
Baker
Baker
Able
Able
Charlie

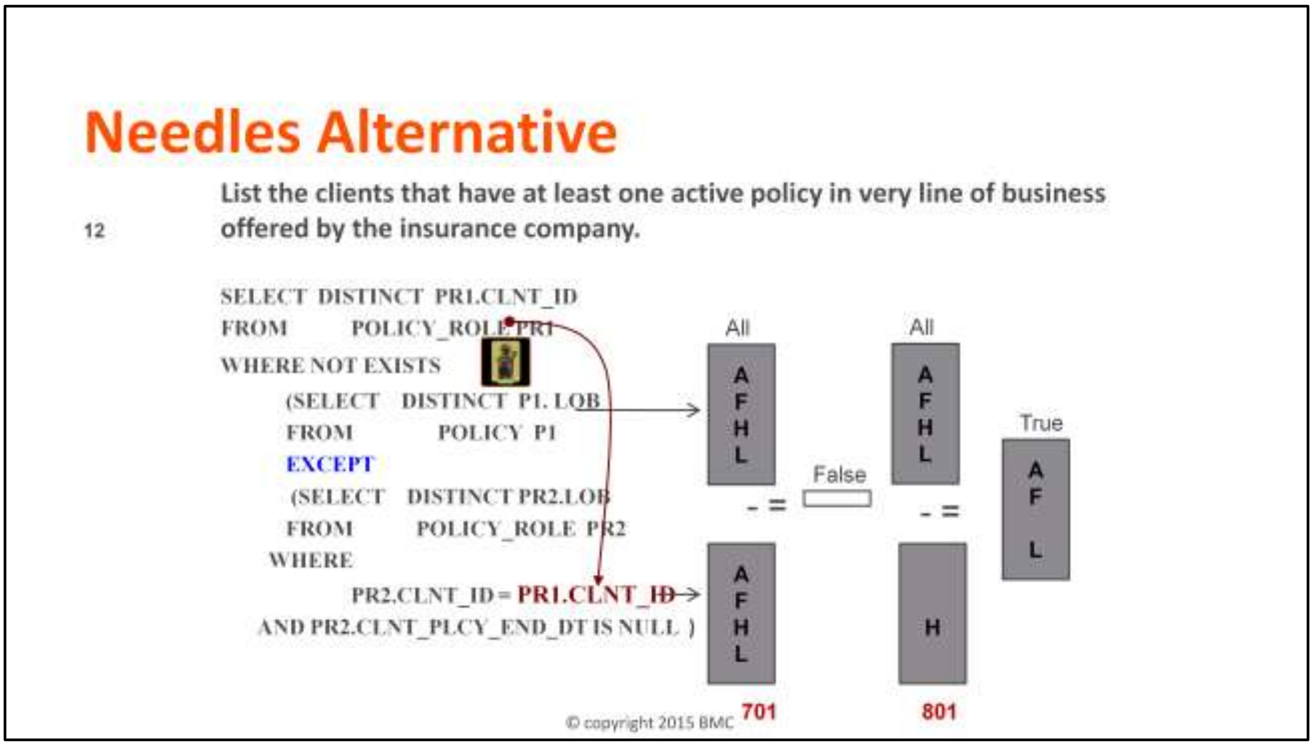
NAME
Baker

NAME
Baker
Baker

NAME
Able

NAME
Able
Able
Able
Able

The big difference between UNIONs and EXCEPT/INTERSECT is the ability to use correlation between query blocks. UNIONs are completely independent.



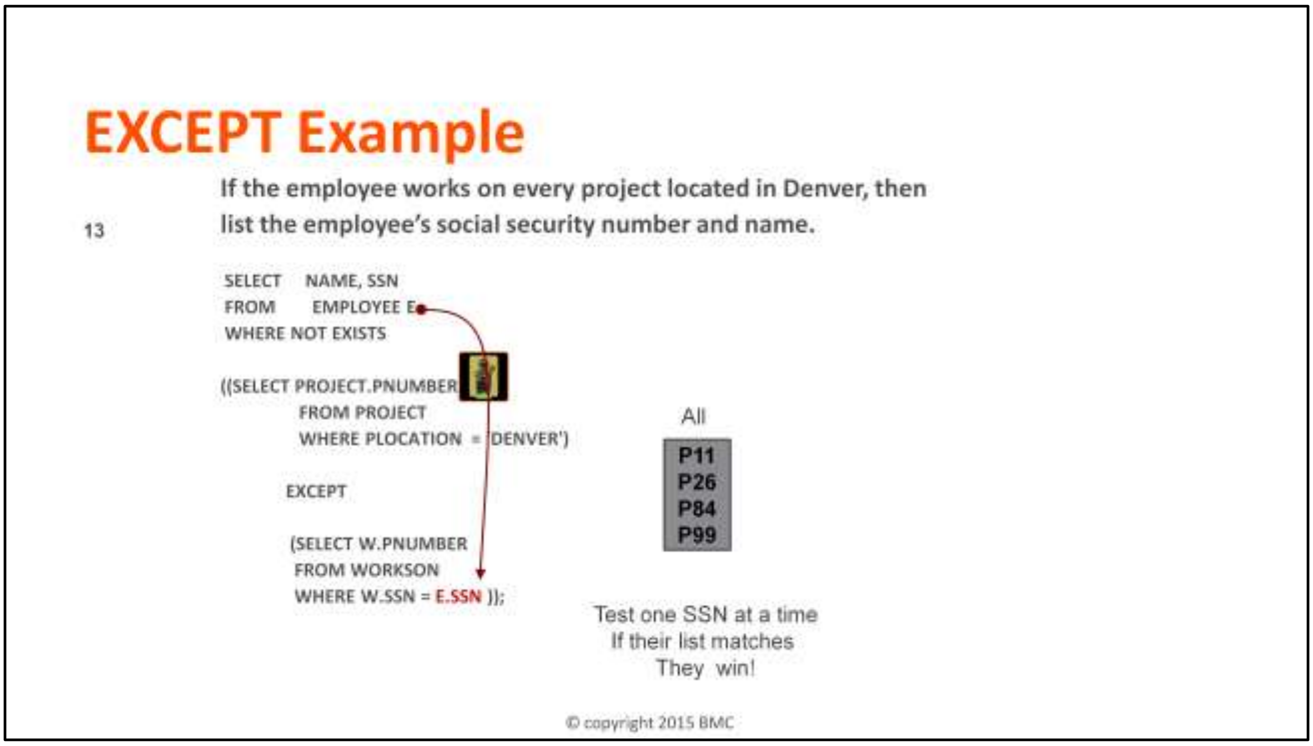
Needles Alternative

12

List the clients that have at least one active policy in very line of business offered by the insurance company.

```
SELECT DISTINCT PR1.CLNT_ID
FROM       POLICY_ROLE PR1
WHERE NOT EXISTS
      (SELECT   DISTINCT  P1. LOB
      FROM          POLICY  P1
      EXCEPT
       (SELECT   DISTINCT PR2.LOB
      FROM          POLICY_ROLE  PR2
     WHERE
               PR2.CLNT_ID = PR1.CLNT_ID
    AND PR2.CLNT_PLCY_END_DT IS NULL )
```

This technique works when the lists compared and the EXCEPT operation are easily arrived.



EXCEPT Example

13

If the employee works on every project located in Denver, then list the employee's social security number and name.

```
SELECT    NAME, SSN
FROM      EMPLOYEE E
WHERE NOT EXISTS

((SELECT PROJECT.PNUMBER
        FROM PROJECT
        WHERE PLOCATION = 'DENVER')

      EXCEPT

       (SELECT W.PNUMBER
        FROM WORKSON
        WHERE W.SSN = E.SSN ));
```

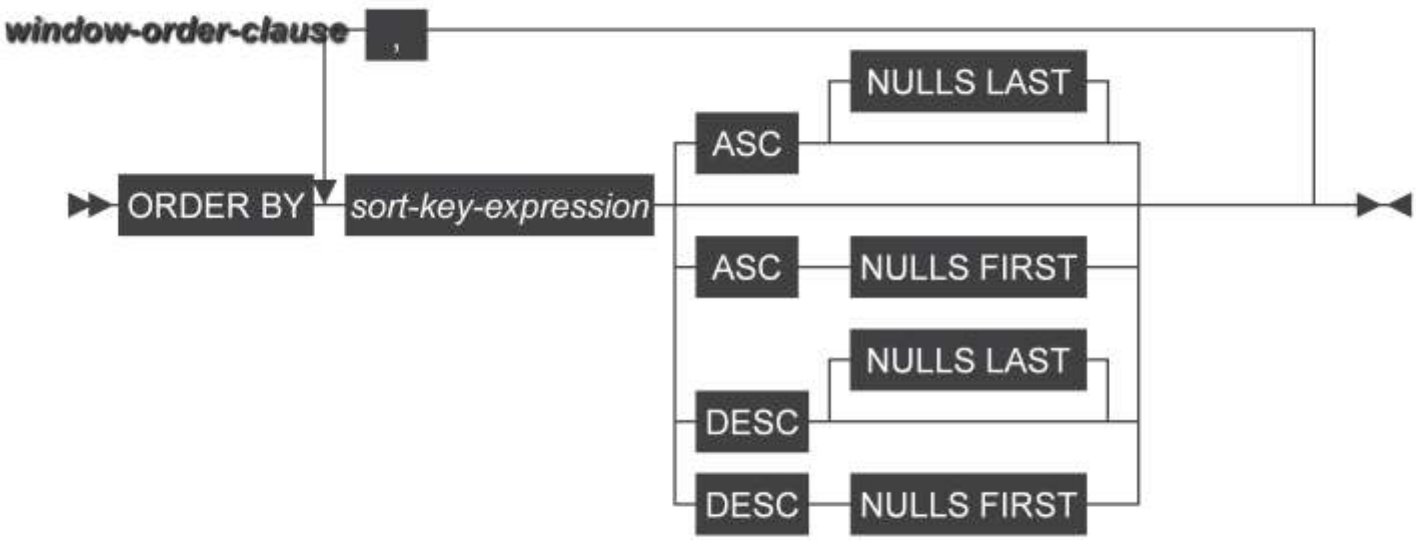
All

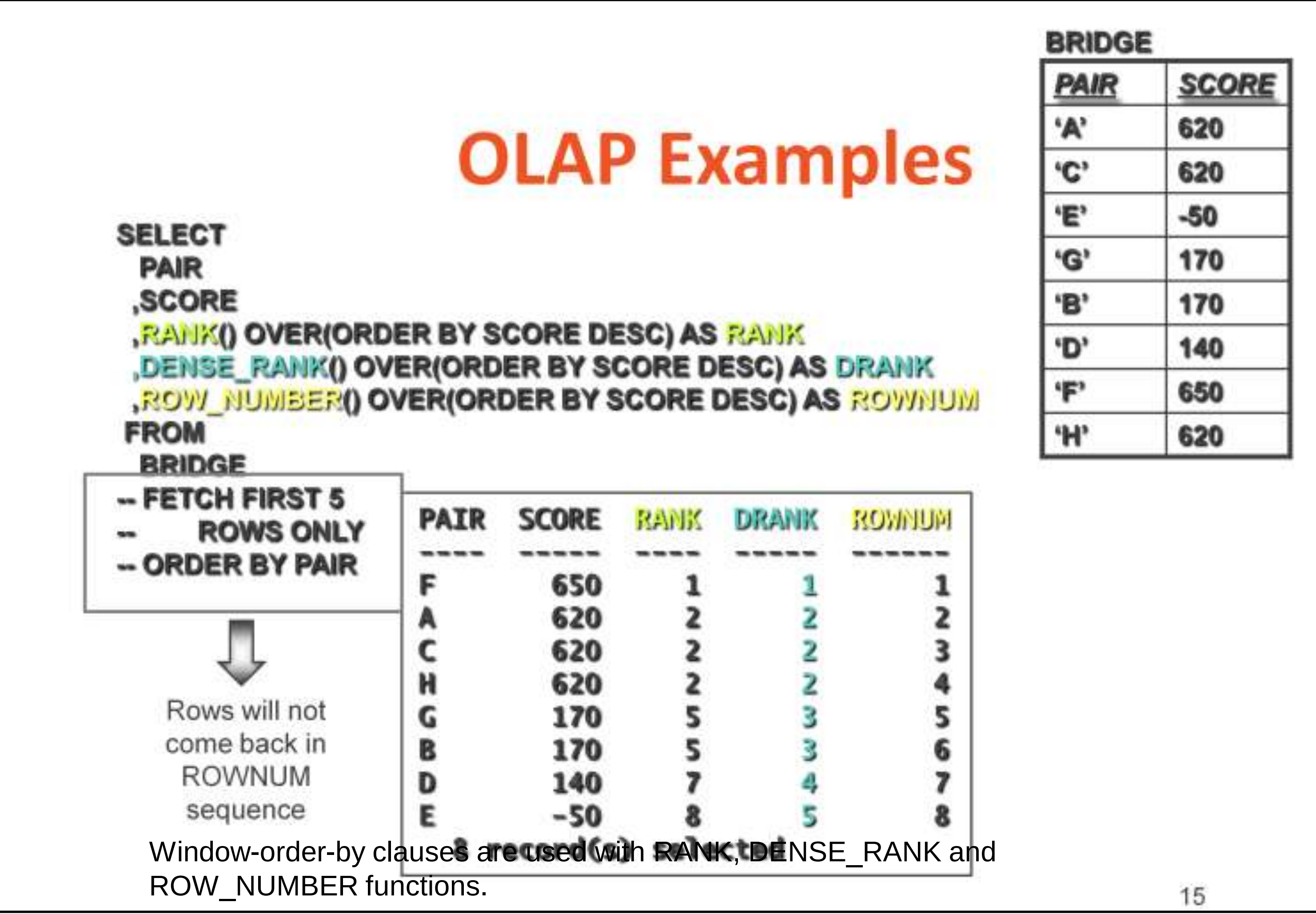
P11
P26
P84
P99

Test one SSN at a time
If their list matches
They win!

One more sample.

OLAP Expressions





OLAP Examples

```
SELECT
  PAIR
 ,SCORE
 ,RANK() OVER(ORDER BY SCORE DESC) AS RANK
 ,DENSE_RANK() OVER(ORDER BY SCORE DESC) AS DRANK
 ,ROW_NUMBER() OVER(ORDER BY SCORE DESC) AS ROWNUM
FROM
  BRIDGE
-- FETCH FIRST 5
--      ROWS ONLY
-- ORDER BY PAIR
```

BRIDGE

PAIR	SCORE
'A'	620
'C'	620
'E'	-50
'G'	170
'B'	170
'D'	140
'F'	650
'H'	620

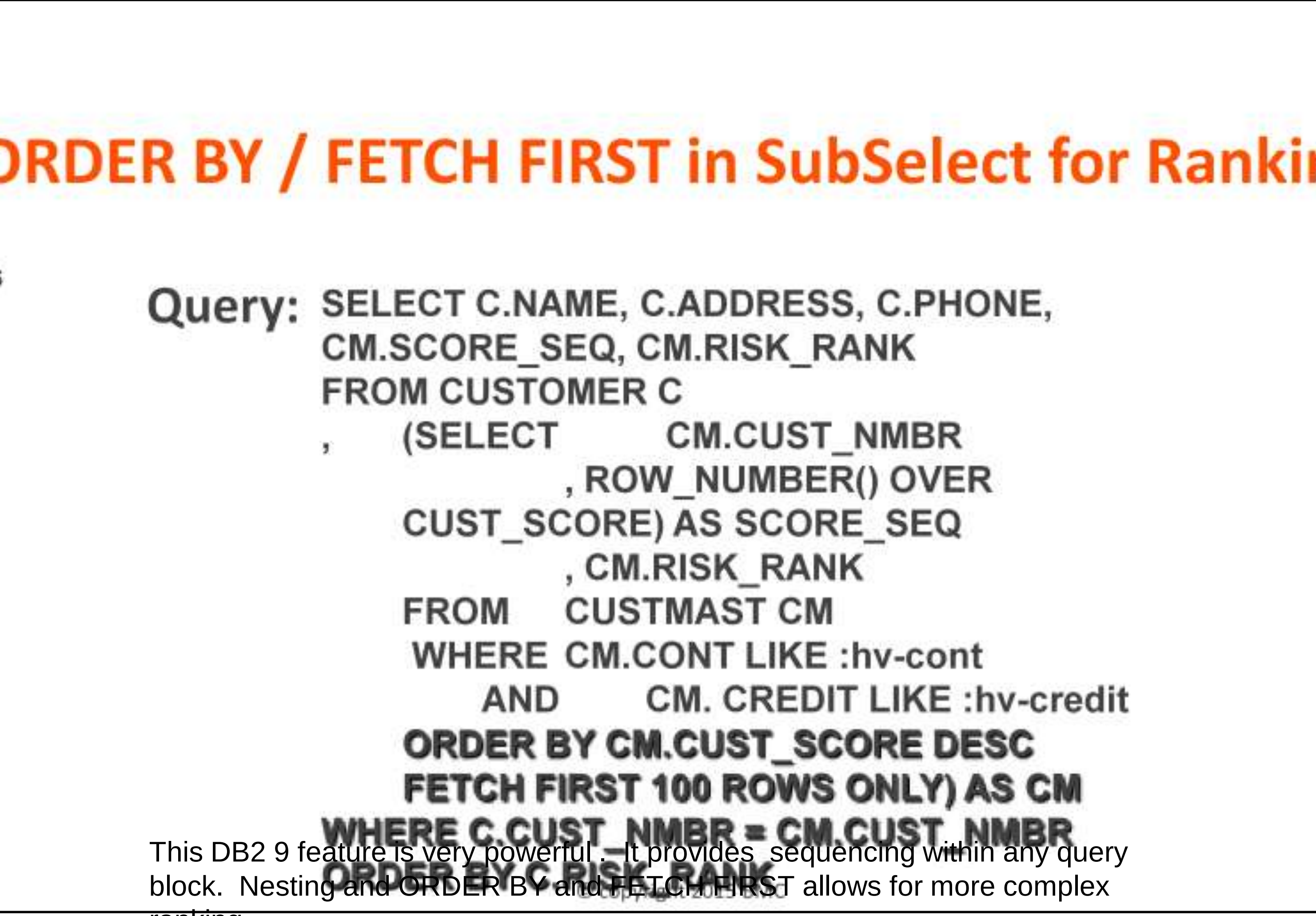
```
PAIR  SCORE  RANK  DRANK  ROWNUM
----  -----  ----  -----  ------
F       650     1      1       1
A       620     2      2       2
C       620     2      2       3
H       620     2      2       4
G       170     5      3       5
B       170     5      3       6
D       140     7      4       7
E       -50     8      5       8
  8 record(s) selected
```

Rows will not come back in ROWNUM sequence

15

Window-order-by clauses are used with RANK, DENSE_RANK and ROW_NUMBER functions.

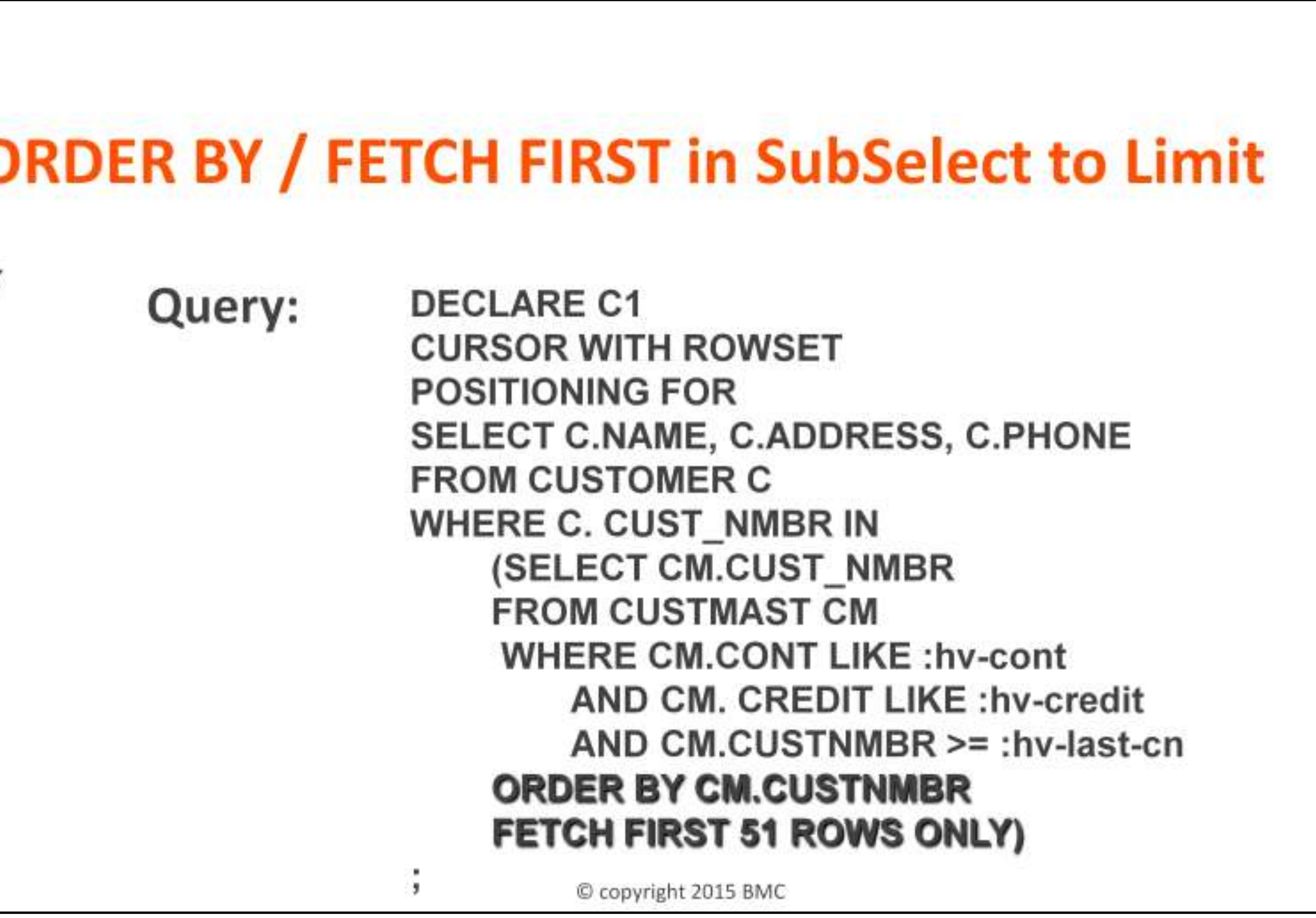
RANK is a mixture of DENSE_RANK and ROW_NUMBER.



RDER BY / FETCH FIRST in SubSelect for Ranki

Query:
```
SELECT C.NAME, C.ADDRESS, C.PHONE,
CM.SCORE_SEQ, CM.RISK_RANK
FROM CUSTOMER C
,       (SELECT          CM.CUST_NMBR
                    , ROW_NUMBER() OVER
        CUST_SCORE) AS SCORE_SEQ
                    , CM.RISK_RANK
        FROM      CUSTMAST CM
         WHERE  CM.CONT LIKE :hv-cont
             AND        CM. CREDIT LIKE :hv-credit
        ORDER BY CM.CUST_SCORE DESC
        FETCH FIRST 100 ROWS ONLY) AS CM
WHERE C.CUST_NMBR = CM.CUST_NMBR
ORDER BY C.RISK_RANK
```

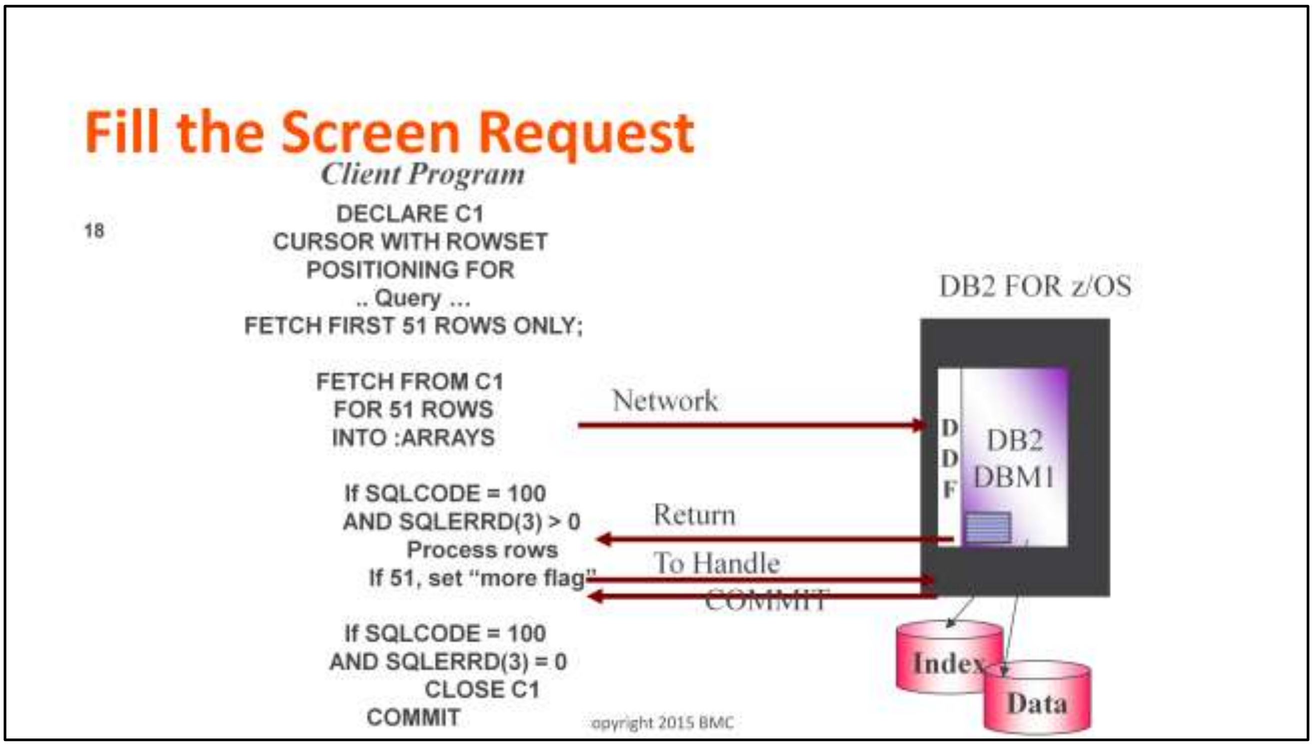
This DB2 9 feature is very powerful . It provides sequencing within any query block. Nesting and ORDER BY and FETCH FIRST allows for more complex ranking.



RDER BY / FETCH FIRST in SubSelect to Limit

Query:

```
DECLARE C1
CURSOR WITH ROWSET
POSITIONING FOR
SELECT C.NAME, C.ADDRESS, C.PHONE
FROM CUSTOMER C
WHERE C. CUST_NMBR IN
        (SELECT CM.CUST_NMBR
        FROM CUSTMAST CM
         WHERE CM.CONT LIKE :hv-cont
                AND CM. CREDIT LIKE :hv-credit
                AND CM.CUSTNMBR >= :hv-last-cn
        ORDER BY CM.CUSTNMBR
        FETCH FIRST 51 ROWS ONLY)
;
```



Fill the Screen Request

Client Program

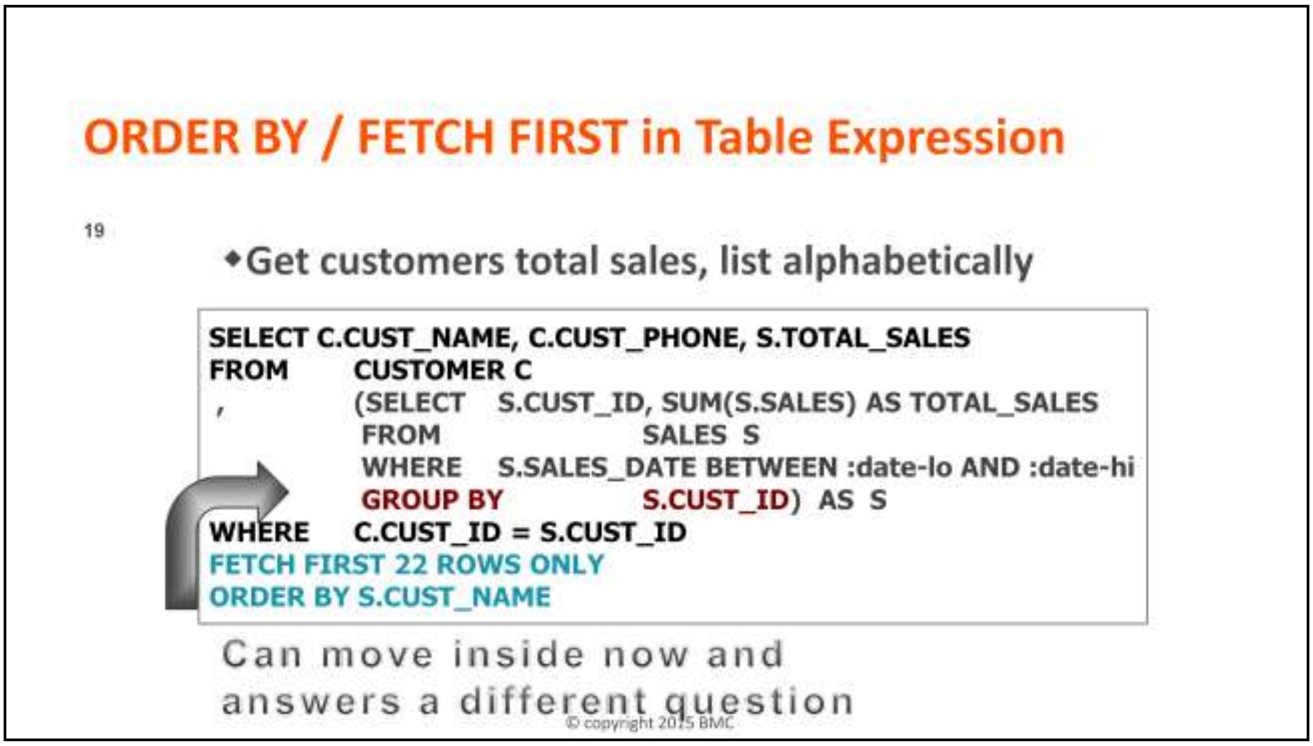
18

DECLARE C1
CURSOR WITH ROWSET
POSITIONING FOR
.. Query ...
FETCH FIRST 51 ROWS ONLY;

FETCH FROM C1
FOR 51 ROWS
INTO :ARRAYS

If SQLCODE = 100
AND SQLERRD(3) > 0
Process rows
If 51, set "more flag"

If SQLCODE = 100
AND SQLERRD(3) = 0
CLOSE C1
COMMIT



ORDER BY / FETCH FIRST in Table Expression

19

- Get customers total sales, list alphabetically

```
SELECT C.CUST_NAME, C.CUST_PHONE, S.TOTAL_SALES
FROM       CUSTOMER C
 ,         (SELECT   S.CUST_ID, SUM(S.SALES) AS TOTAL_SALES
            FROM                SALES  S
            WHERE    S.SALES_DATE BETWEEN :date-lo AND :date-hi
            GROUP BY            S.CUST_ID)  AS  S
WHERE      C.CUST_ID = S.CUST_ID
FETCH FIRST 22 ROWS ONLY
ORDER BY S.CUST_NAME
```

Can move inside now and answers a different question

Typical summary reports also require detail data from another table. GROUP BY prohibits getting the summary data and detail together Table Expressions can be used to separate the GROUP BY work which is usually done against one table.

The first benefit is the number of rows participating in the join are now reduced from the GROUP BY processing.

The second benefit occurs when an index is available on the GROUP BY column. This enables the DB2 to process the groups one at a time and pass them into the join without needing a sort.

DB2 9 allows FETCH FIRST and ORDER BY inside table expressions and other query blocks. This adds powerful ranking capabilities.

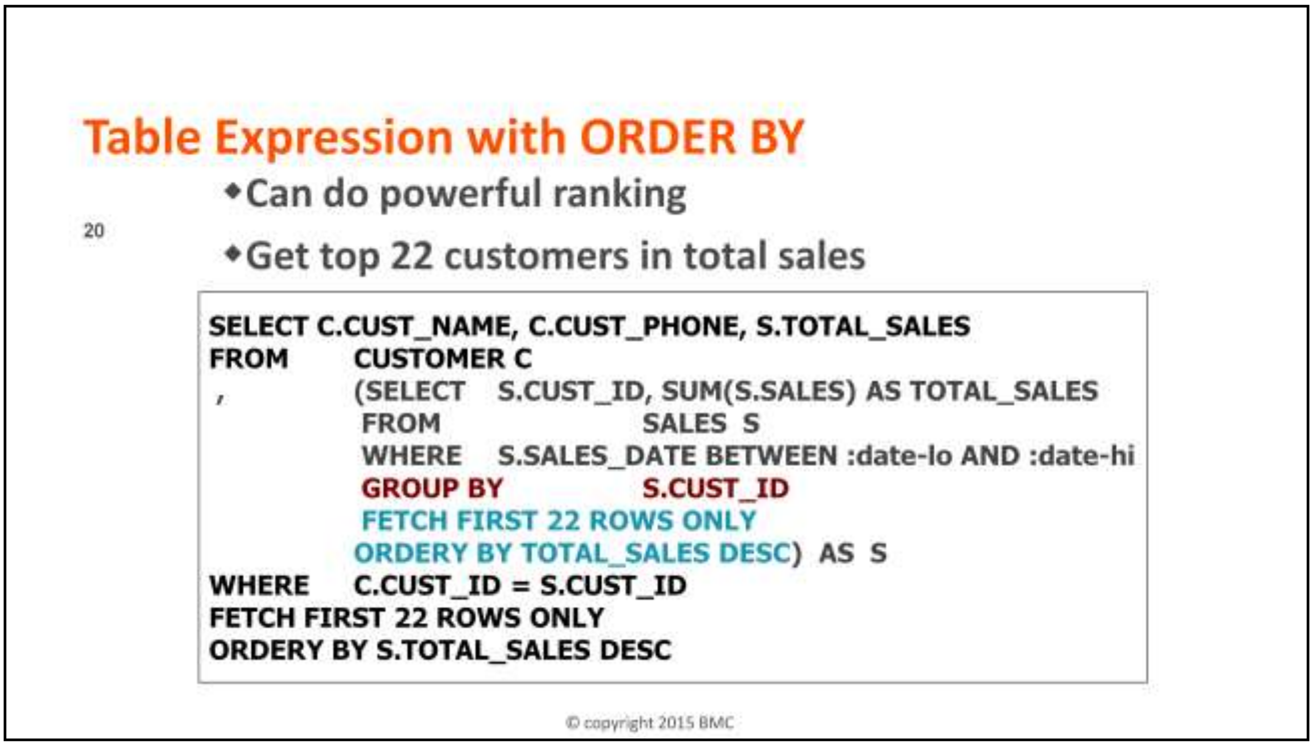


Table Expression with ORDER BY

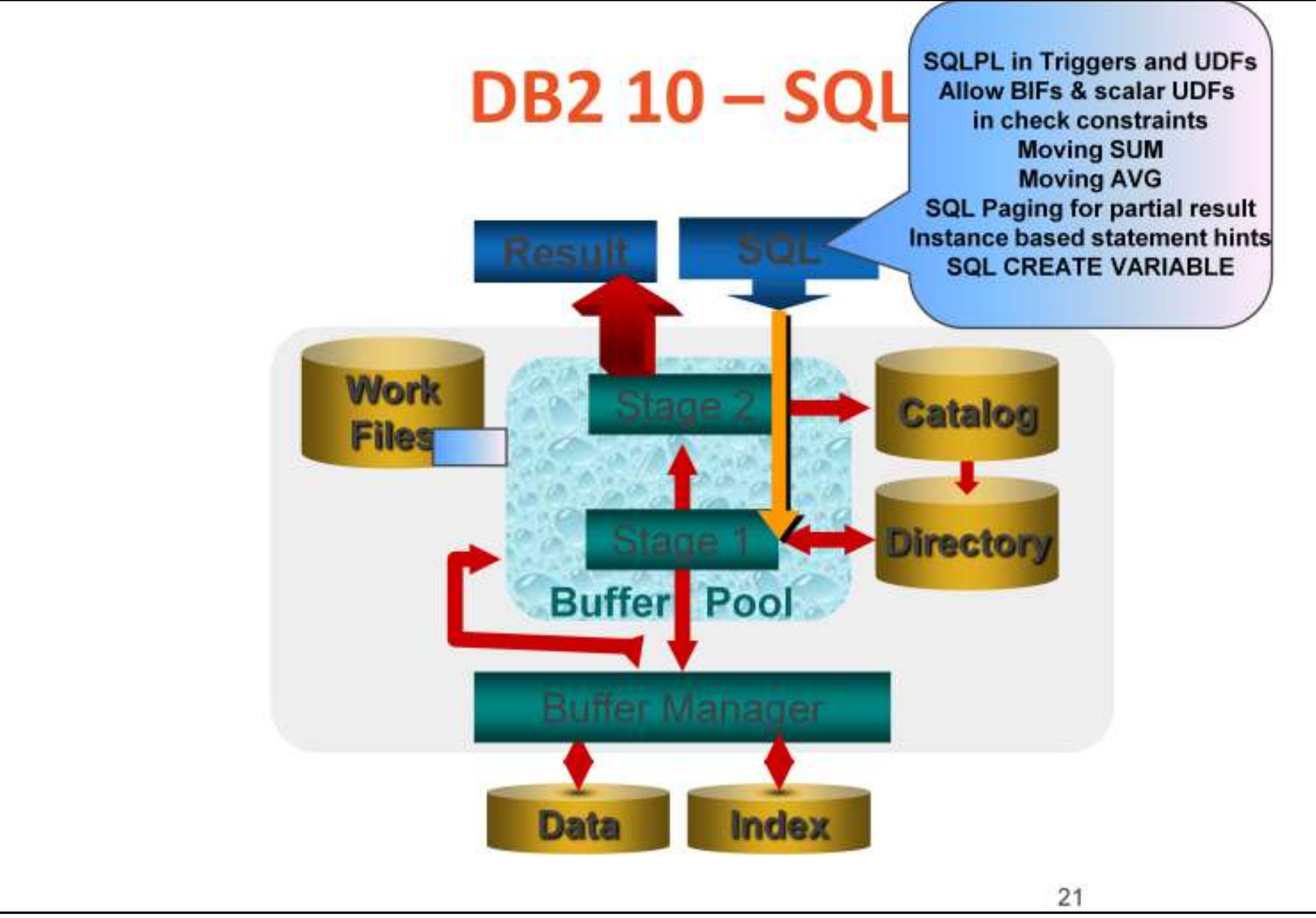
20

- Can do powerful ranking
- Get top 22 customers in total sales

```
SELECT C.CUST_NAME, C.CUST_PHONE, S.TOTAL_SALES
FROM     CUSTOMER C
 ,       (SELECT   S.CUST_ID, SUM(S.SALES) AS TOTAL_SALES
          FROM              SALES  S
          WHERE    S.SALES_DATE BETWEEN :date-lo AND :date-hi
          GROUP BY          S.CUST_ID
          FETCH FIRST 22 ROWS ONLY
         ORDERY BY TOTAL_SALES DESC)  AS  S
WHERE    C.CUST_ID = S.CUST_ID
FETCH FIRST 22 ROWS ONLY
ORDERY BY S.TOTAL_SALES DESC
```

DB2 9 allows FETCH FIRST and ORDER BY inside table expressions and other query blocks. This adds powerful ranking capabilities.

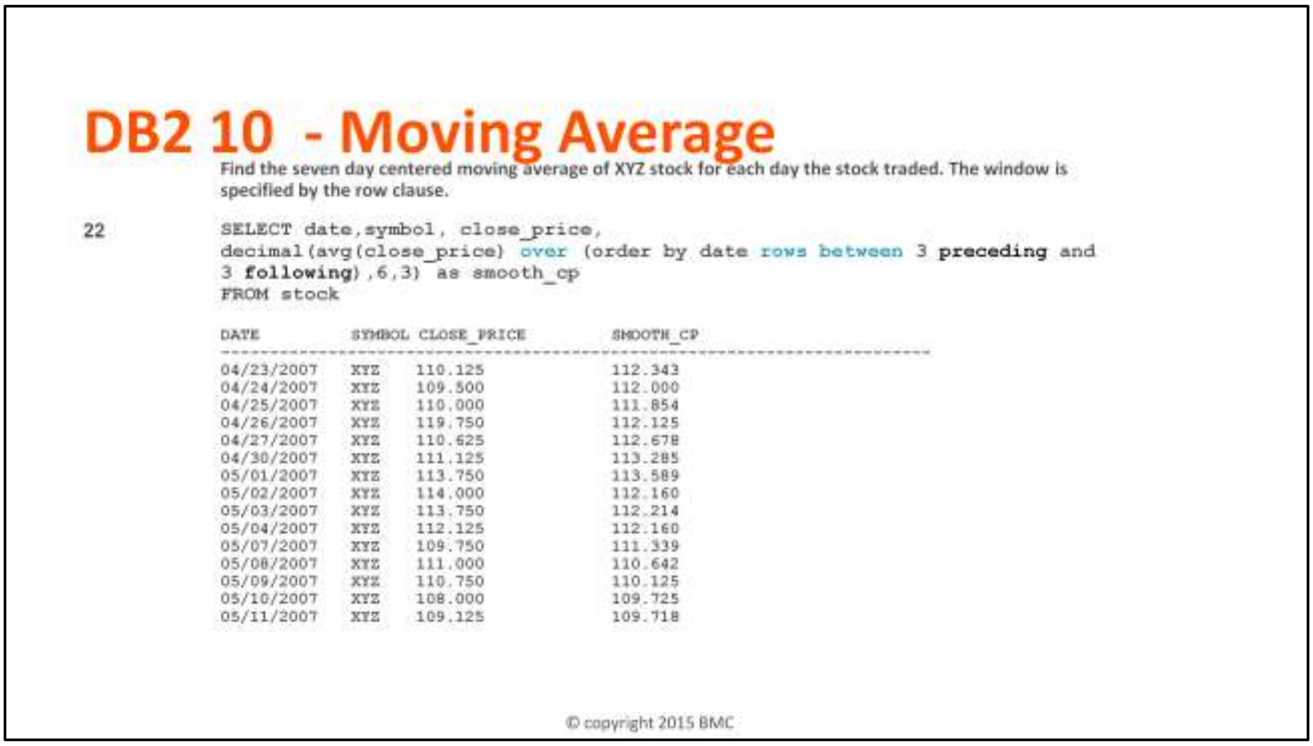
Each queryblock can independently sequence data prior to final ORDER BY.



DB2 10 – SQL

SQLPL in Triggers and UDFs
Allow BIFs & scalar UDFs
in check constraints
Moving SUM
Moving AVG
SQL Paging for partial result
Instance based statement hints
SQL CREATE VARIABLE

21



DB2 10 - Moving Average

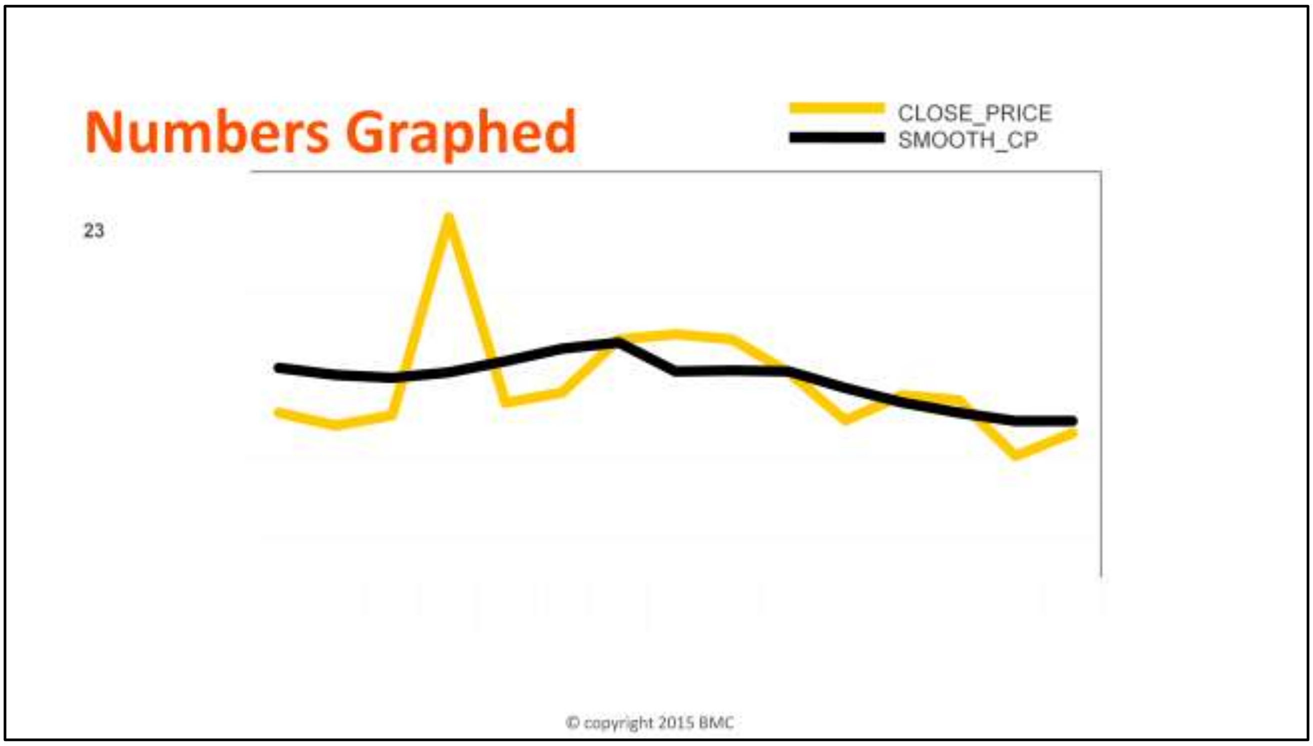
Find the seven day centered moving average of XYZ stock for each day the stock traded. The window is specified by the row clause.

22

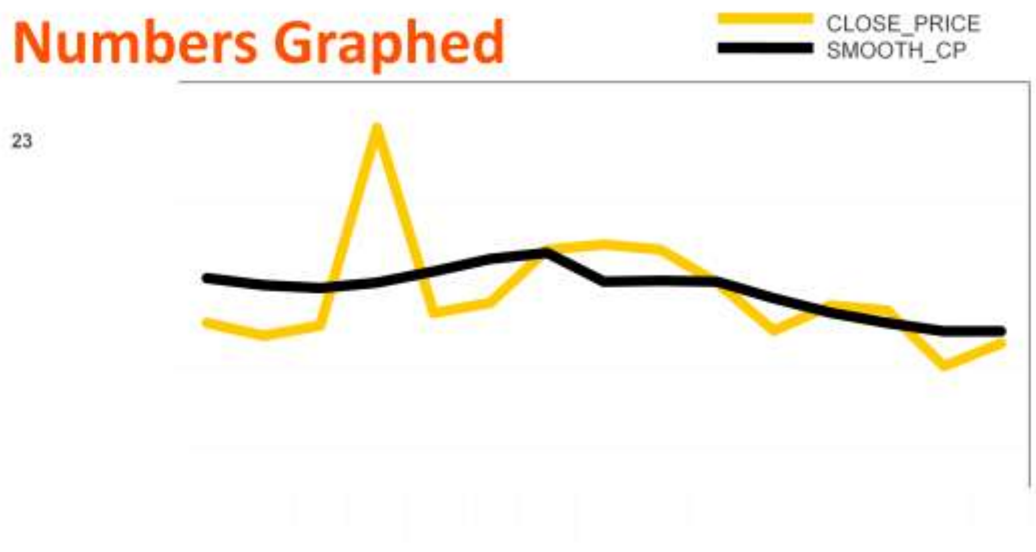
```
SELECT date,symbol, close_price,
decimal(avg(close_price) over (order by date rows between 3 preceding and
3 following),6,3) as smooth_cp
FROM stock

DATE          SYMBOL CLOSE_PRICE         SMOOTH_CP
-------------------------------------------------------------------------------
04/23/2007    XYZ    110.125             112.343
04/24/2007    XYZ    109.500             112.000
04/25/2007    XYZ    110.000             111.854
04/26/2007    XYZ    119.750             112.125
04/27/2007    XYZ    110.625             112.678
04/30/2007    XYZ    111.125             113.285
05/01/2007    XYZ    113.750             113.589
05/02/2007    XYZ    114.000             112.160
05/03/2007    XYZ    113.750             112.214
05/04/2007    XYZ    112.125             112.160
05/07/2007    XYZ    109.750             111.339
05/08/2007    XYZ    111.000             110.642
05/09/2007    XYZ    110.750             110.125
05/10/2007    XYZ    108.000             109.725
05/11/2007    XYZ    109.125             109.718
```

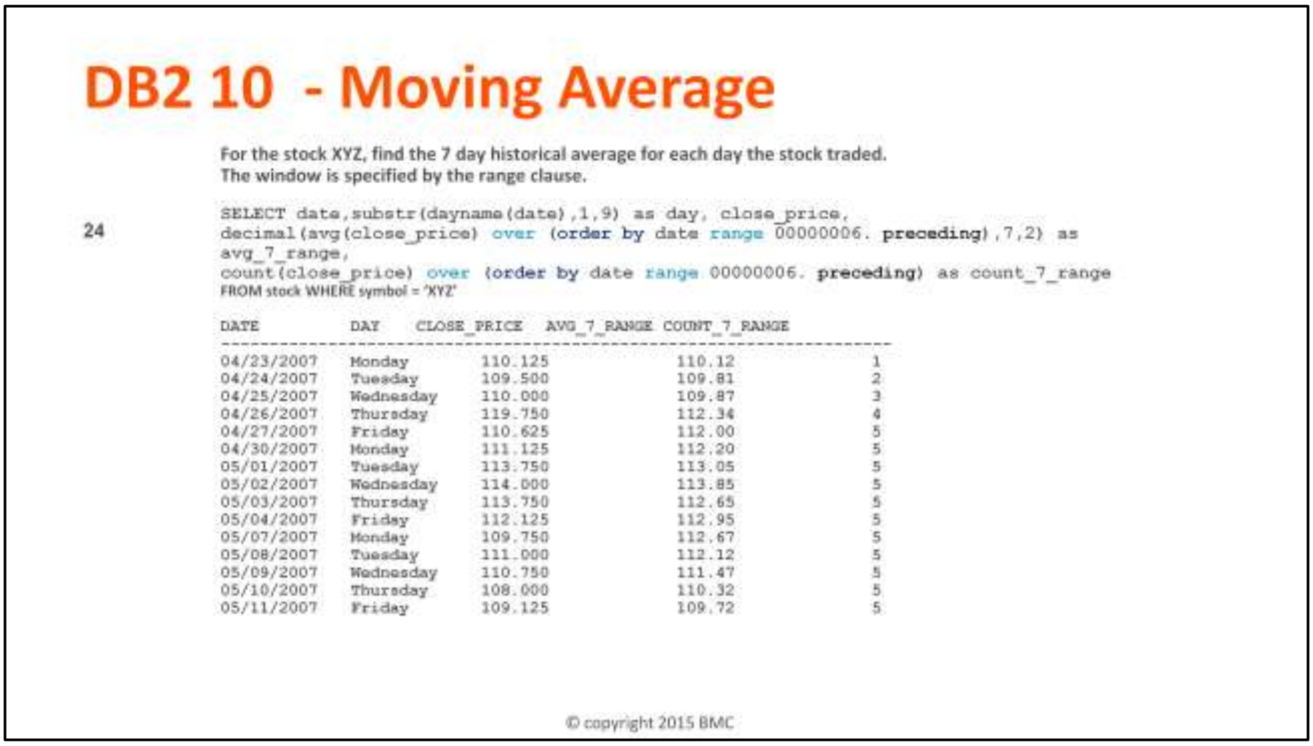
This is IBM’s proposed SQL text from DB2X and is subject to change.

Moving averages are calculated over a defined range using PRECEDING and FOLLOWING key words.



Numbers Graphed





DB2 10 - Moving Average

24

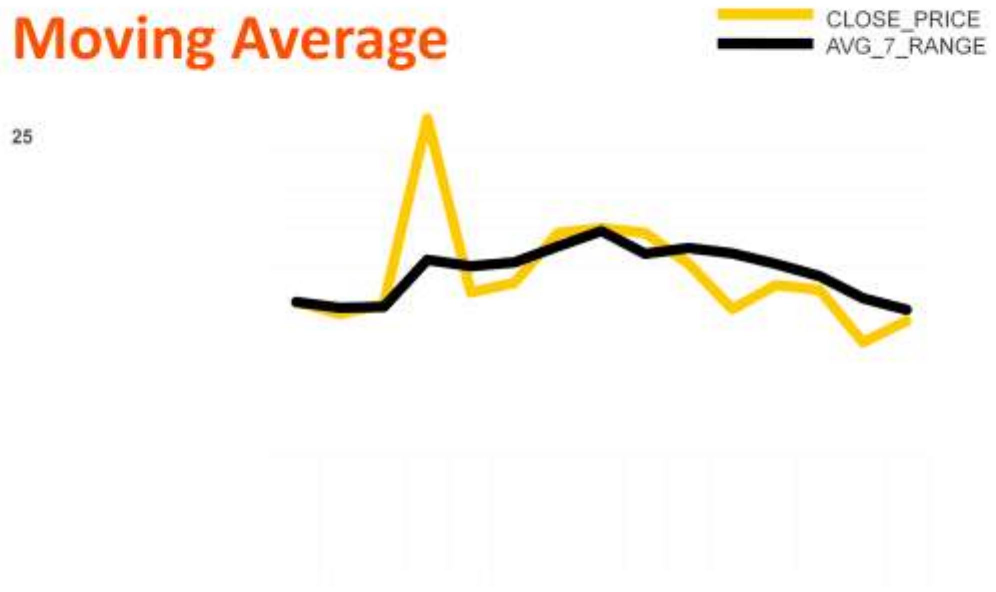
For the stock XYZ, find the 7 day historical average for each day the stock traded.
The window is specified by the range clause.

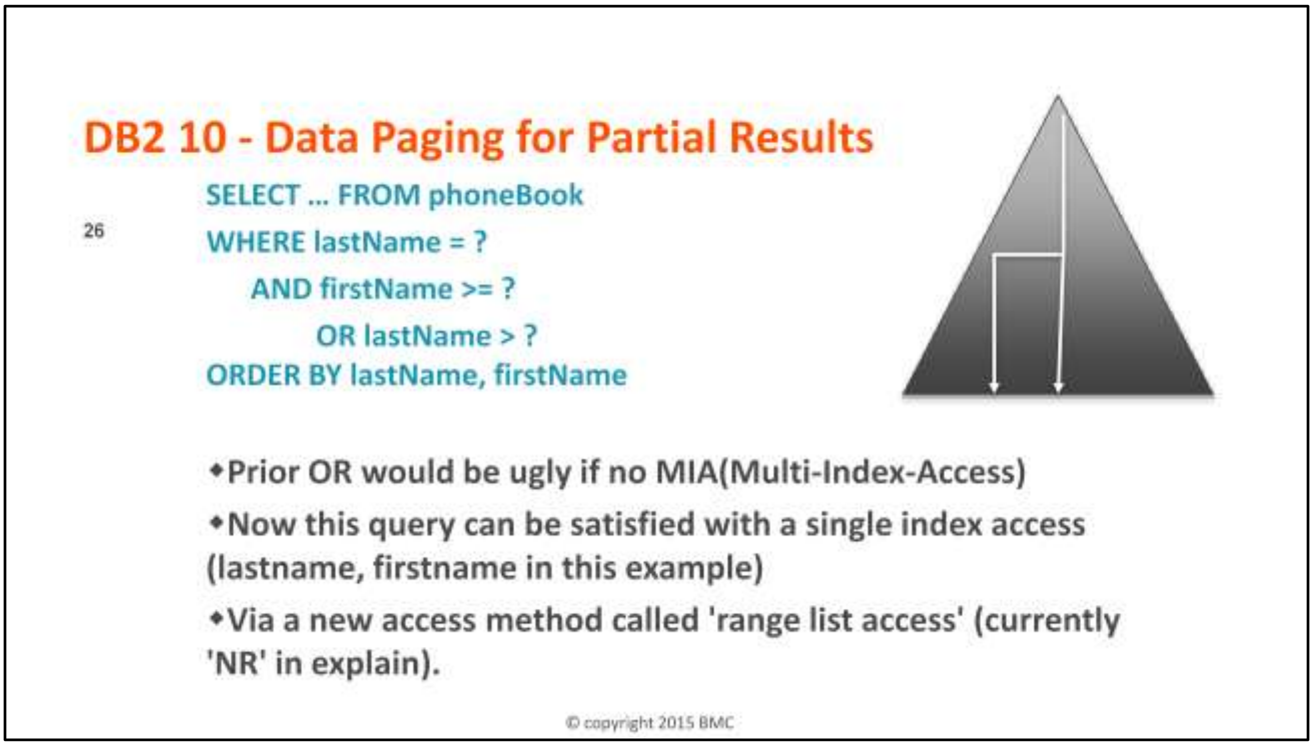
```
SELECT date,substr(dayname(date),1,9) as day, close_price,
decimal(avg(close_price) over (order by date range 00000006. preceding),7,2) as
avg_7_range,
count(close_price) over (order by date range 00000006. preceding) as count_7_range
FROM stock WHERE symbol = 'XYZ'
```

```
DATE          DAY      CLOSE_PRICE   AVG_7_RANGE COUNT_7_RANGE
---------------------------------------------------------------------
04/23/2007    Monday         110.125           110.12             1
04/24/2007    Tuesday        109.500           109.81             2
04/25/2007    Wednesday      110.000           109.87             3
04/26/2007    Thursday       119.750           112.34             4
04/27/2007    Friday         110.625           112.00             5
04/30/2007    Monday         111.125           112.20             5
05/01/2007    Tuesday        113.750           113.05             5
05/02/2007    Wednesday      114.000           113.85             5
05/03/2007    Thursday       113.750           112.65             5
05/04/2007    Friday         112.125           112.95             5
05/07/2007    Monday         109.750           112.67             5
05/08/2007    Tuesday        111.000           112.12             5
05/09/2007    Wednesday      110.750           111.47             5
05/10/2007    Thursday       108.000           110.32             5
05/11/2007    Friday         109.125           109.72             5
```

More information on this syntax can be found in the Database Fundamentals :
http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp

Moving Average

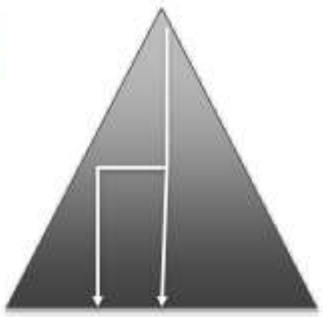




DB2 10 - Data Paging for Partial Results

26

```
SELECT … FROM phoneBook
WHERE lastName = ?
   AND firstName >= ?
       OR lastName > ?
ORDER BY lastName, firstName
```



- Prior OR would be ugly if no MIA(Multi-Index-Access)
- Now this query can be satisfied with a single index access (lastname, firstname in this example)
- Via a new access method called 'range list access' (currently 'NR' in explain).

MIA = Multiple index access

NR = New Range

DB2 11 Analytics

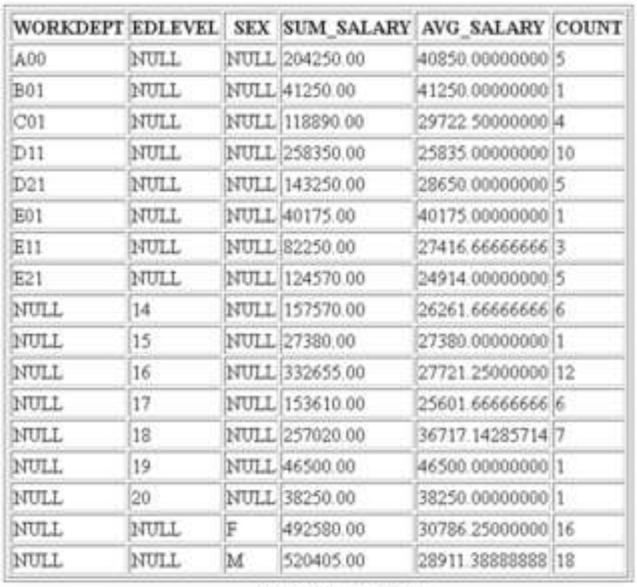
27

Example 6-16 Sample SQL statement utilizing GROUP BY GROUPING SETS

```
SELECT WORKDEPT, EDLEVEL, SEX, SUM(SALARY) as
SUM_SALARY, AVG(SALARY) as AVG_SALARY, COUNT(*) as
COUNT
FROM DSN81110.EMP WHERE SALARY > 20000
GROUP BY GROUPING SETS (WORKDEPT, EDLEVEL, SEX)
```

http://www.redbooks.ibm.com/redbooks/pdfs/sg248180.pdf

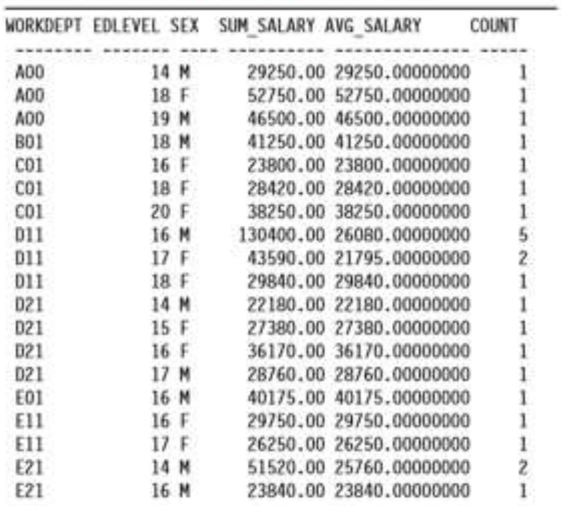
GROUPING SETS (WORKDEPT, EDLEVEL, SEX)



WORKDEPT	EDLEVEL	SEX	SUM_SALARY	AVG_SALARY	COUNT
A00	NULL	NULL	204250.00	40850.00000000	5
B01	NULL	NULL	41250.00	41250.00000000	1
C01	NULL	NULL	118890.00	29722.50000000	4
D11	NULL	NULL	258350.00	25835.00000000	10
D21	NULL	NULL	143250.00	28650.00000000	5
E01	NULL	NULL	40175.00	40175.00000000	1
E11	NULL	NULL	82250.00	27416.66666666	3
E21	NULL	NULL	124570.00	24914.00000000	5
NULL	14	NULL	157570.00	26261.66666666	6
NULL	15	NULL	27380.00	27380.00000000	1
NULL	16	NULL	332655.00	27721.25000000	12
NULL	17	NULL	153610.00	25601.66666666	6
NULL	18	NULL	257020.00	36717.14285714	7
NULL	19	NULL	46500.00	46500.00000000	1
NULL	20	NULL	38250.00	38250.00000000	1
NULL	NULL	F	492580.00	30786.25000000	16
NULL	NULL	M	520405.00	28911.38888888	18

GROUP BY ROLLUP (WORKDEPT, EDLEVEL, SEX)

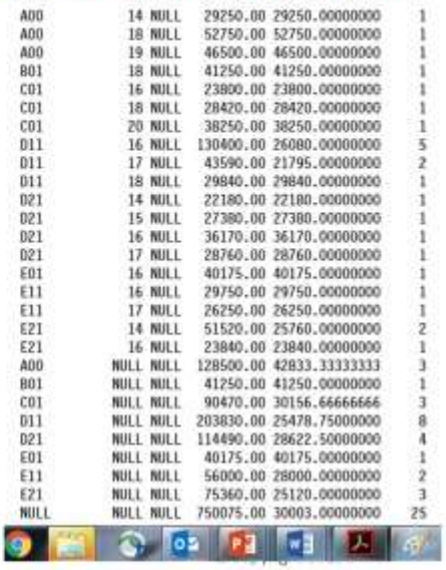
29



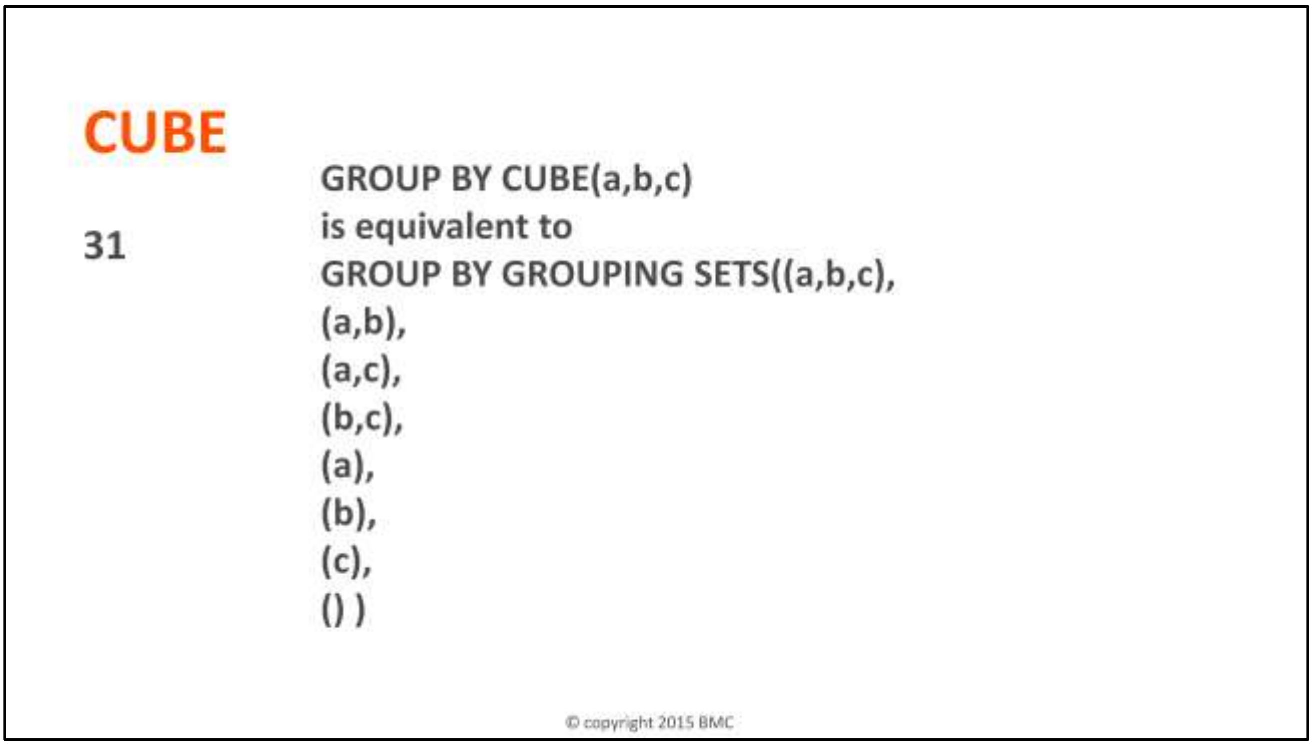
```
WORKDEPT EDLEVEL SEX  SUM_SALARY AVG_SALARY      COUNT
 -------- ------- ---- ---------- -------------- -----
 A00           14 M     29250.00 29250.00000000     1
 A00           18 F     52750.00 52750.00000000     1
 A00           19 M     46500.00 46500.00000000     1
 B01           18 M     41250.00 41250.00000000     1
 C01           16 F     23800.00 23800.00000000     1
 C01           18 F     28420.00 28420.00000000     1
 C01           20 F     38250.00 38250.00000000     1
 D11           16 M    130400.00 26080.00000000     5
 D11           17 F     43590.00 21795.00000000     2
 D11           18 F     29840.00 29840.00000000     1
 D21           14 M     22180.00 22180.00000000     1
 D21           15 F     27380.00 27380.00000000     1
 D21           16 F     36170.00 36170.00000000     1
 D21           17 M     28760.00 28760.00000000     1
 E01           16 M     40175.00 40175.00000000     1
 E11           16 F     29750.00 29750.00000000     1
 E11           17 F     26250.00 26250.00000000     1
 E21           14 M     51520.00 25760.00000000     2
 E21           16 M     23840.00 23840.00000000     1
```

GROUP BY ROLLUP (WORKDEPT, EDLEVEL, SEX)

30



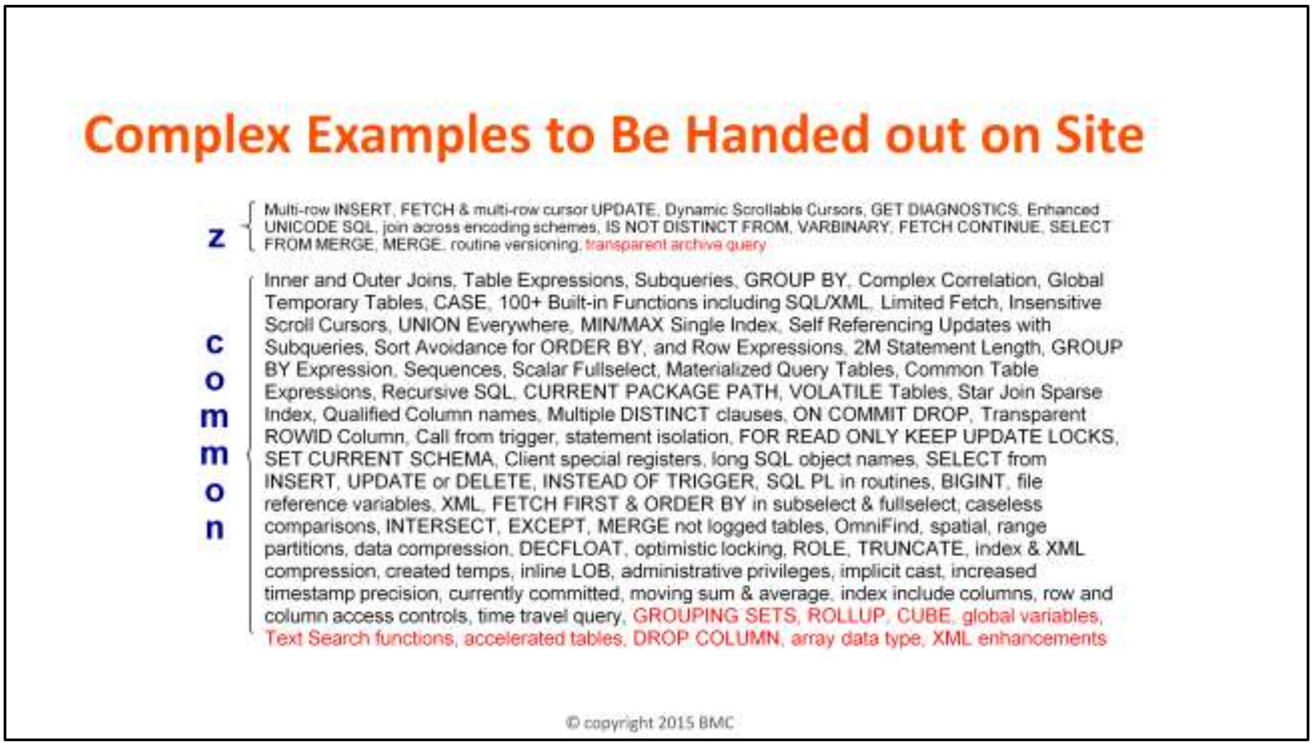
```
A00           14 NULL   29250.00 29250.00000000    1
A00           18 NULL   52750.00 52750.00000000    1
A00           19 NULL   46500.00 46500.00000000    1
B01           18 NULL   41250.00 41250.00000000    1
C01           16 NULL   23800.00 23800.00000000    1
C01           18 NULL   28420.00 28420.00000000    1
C01           20 NULL   38250.00 38250.00000000    1
D11           16 NULL  130400.00 26080.00000000    5
D11           17 NULL   43590.00 21795.00000000    2
D11           18 NULL   29840.00 29840.00000000    1
D21           14 NULL   22180.00 22180.00000000    1
D21           15 NULL   27380.00 27380.00000000    1
D21           16 NULL   36170.00 36170.00000000    1
D21           17 NULL   28760.00 28760.00000000    1
E01           16 NULL   40175.00 40175.00000000    1
E11           16 NULL   29750.00 29750.00000000    1
E11           17 NULL   26250.00 26250.00000000    1
E21           14 NULL   51520.00 25760.00000000    2
E21           16 NULL   23840.00 23840.00000000    1
A00         NULL NULL  128500.00 42833.33333333    3
B01         NULL NULL   41250.00 41250.00000000    1
C01         NULL NULL   90470.00 30156.66666666    3
D11         NULL NULL  203830.00 25478.75000000    8
D21         NULL NULL  114490.00 28622.50000000    4
E01         NULL NULL   40175.00 40175.00000000    1
E11         NULL NULL   56000.00 28000.00000000    2
E21         NULL NULL   75360.00 25120.00000000    3
NULL        NULL NULL  750075.00 30003.00000000   25
```



CUBE

31

GROUP BY CUBE(a,b,c)
is equivalent to
GROUP BY GROUPING SETS((a,b,c),
(a,b),
(a,c),
(b,c),
(a),
(b),
(c),
())



Complex Examples to Be Handed out on Site

z — Multi-row INSERT, FETCH & multi-row cursor UPDATE, Dynamic Scrollable Cursors, GET DIAGNOSTICS, Enhanced UNICODE SQL, join across encoding schemes, IS NOT DISTINCT FROM, VARBINARY, FETCH CONTINUE, SELECT FROM MERGE, MERGE, routine versioning, transparent archive query

common — Inner and Outer Joins, Table Expressions, Subqueries, GROUP BY, Complex Correlation, Global Temporary Tables, CASE, 100+ Built-in Functions including SQL/XML, Limited Fetch, Insensitive Scroll Cursors, UNION Everywhere, MIN/MAX Single Index, Self Referencing Updates with Subqueries, Sort Avoidance for ORDER BY, and Row Expressions, 2M Statement Length, GROUP BY Expression, Sequences, Scalar Fullselect, Materialized Query Tables, Common Table Expressions, Recursive SQL, CURRENT PACKAGE PATH, VOLATILE Tables, Star Join Sparse Index, Qualified Column names, Multiple DISTINCT clauses, ON COMMIT DROP, Transparent ROWID Column, Call from trigger, statement isolation, FOR READ ONLY KEEP UPDATE LOCKS, SET CURRENT SCHEMA, Client special registers, long SQL object names, SELECT from INSERT, UPDATE or DELETE, INSTEAD OF TRIGGER, SQL PL in routines, BIGINT, file reference variables, XML, FETCH FIRST & ORDER BY in subselect & fullselect, caseless comparisons, INTERSECT, EXCEPT, MERGE not logged tables, OmniFind, spatial, range partitions, data compression, DECFLOAT, optimistic locking, ROLE, TRUNCATE, index & XML compression, created temps, inline LOB, administrative privileges, implicit cast, increased timestamp precision, currently committed, moving sum & average, index include columns, row and column access controls, time travel query, GROUPING SETS, ROLLUP, CUBE, global variables, Text Search functions, accelerated tables, DROP COLUMN, array data type, XML enhancements

Sheryl M. Larsen, Sr. DB2 Product Specialist
Sheryl_Larsen@bmc.com
(224) 343-5427