IBM Information >>> On Demand 2007

IBM

New Cool SQL: DB2 9 for z/OS

Willie Favero, Senior Certified IT Software Specialist

IBM S&D, Americas, West Region

wfavero@ibm.us.com

Session #1094

Data Servers - System z - DB2 and Tools

Act.Right.Now.

IBM INFORMATION ON DEMAND 2007
October 14 – 19, 2007
Mandalay Bay
Las Vegas, Nevada

Disclaimer

The information contained in this presentation has not been submitted to any formal IBM review and is distributed on an "As Is" basis without any warranty either expressed or implied. The use of this information is a customer responsibility.

The materials in this presentation are also subject to

- enhancements at some future date,
- a new release of DB2, or
- a Programming Temporary Fix (PTF)

IBM MAY HAVE PATENTS OR PENDING PATENT APPLICATIONS COVERING SUBJECT MATTER IN THIS DOCUMENT. THE FURNISHING OF THIS DOCUMENT DOES NOT IMPLY GIVING LICENSE TO THESE PATENTS.

TRADEMARKS: THE FOLLOWING TERMS ARE TRADEMARKS OR ® REGISTERED TRADEMARKS OF THE IBM CORPORATION IN THE UNITED STATES AND/OR OTHER COUNTRIES: AIX, AS/400, DATABASE 2, DB2, e-business logo, Enterprise Storage Server, ESCON, FICON, OS/390, OS/400, ES/9000, MVS/ESA, Netfinity, RISC, RISC SYSTEM/6000, iSeries, pSeries, xSeries, SYSTEM/390, IBM, Lotus, NOTES, WebSphere, z/Architecture, z/OS, zSeries, System z.

THE FOLLOWING TERMS ARE TRADEMARKS OR REGISTERED TRADEMARKS OF THE MICROSOFT CORPORATION IN THE UNITED STATES AND/OR OTHER COUNTRIES: MICROSOFT, WINDOWS, WINDOWS NT, ODBC and WINDOWS 95.

For additional information visit the URL

http://www.ibm.com/legal/copytrade.phtml ***for*** **"Copyright and trademark information"**

SQL: Productivity, DB2 family & porting

- XML
- MERGE & TRUNCATE
- SELECT FROM UPDATE, DELETE, MERGE
- INSTEAD OF TRIGGER
- BIGINT, VARBINARY, BINARY, DECIMAL FLOAT
- Native SQL Procedure Language
- Nested compound
- Optimistic locking
- LOB File reference variable & FETCH CONTINUE
- FETCH FIRST & ORDER BY in subselect and fullselect
- INTERSECT & EXCEPT
- Many new built-in functions, caseless comparisons
- Index on expression
- Improved DDL consistency
- CURRENT SCHEMA

DB2 SQL

z z/OS V7
Common
LUE Linux, Unix & Windows V8.2

z

Common

Inner and Outer Joins, Table Expressions, Subqueries, GROUP BY, Complex Correlation, Global Temporary Tables, CASE, 100+ Built-in Functions, Limited Fetch, Insensitive Scroll Cursors, UNION Everywhere, MIN/MAX Single Index Support, Self Referencing Updates with Subqueries, Sort Avoidance for ORDER BY, and Row Expressions, Call from trigger, statement isolation

LUW

Updateable UNION in Views, ORDER BY/FETCH FIRST in subselects & table expressions, GROUPING SETS, ROLLUP, CUBE, INSTEAD OF TRIGGER, EXCEPT, INTERSECT, 16 Built-in Functions, MERGE, Native SQL Procedure Language, SET CURRENT ISOLATION, BIGINT data type, file reference variables, SELECT FROM UPDATE, DELETE & MERGE, multi-site join, 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, FOR READ ONLY KEEP UPDATE LOCKS, SET CURRENT SCHEMA, Client special registers, long SQL object names, SELECT from INSERT

DB2 SQL

z z/OS V8

Common
LUW (Linux, Unix & Windows) V8.2

z

Multi-row INSERT, FETCH & multi-row cursor UPDATE, Dynamic Scrollable Cursors, GET DIAGNOSTICS, Enhanced UNICODE for SQL, join across encoding schemes, IS NOT DISTINCT FROM, Session variables, range partitioning

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 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 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

LUW

Updateable UNION in Views, ORDER BY/FETCH FIRST in subselects & table expressions, GROUPING SETS, ROLLUP, CUBE, INSTEAD OF TRIGGER, EXCEPT, INTERSECT, 16 Built-in Functions, MERGE, Native SQL Procedure Language, SET CURRENT ISOLATION, BIGINT data type, file reference variables, SELECT FROM UPDATE or DELETE, multi-site join, MDC

DB2 SQL

z z/OS V9

Common

LUW (Linux, Unix & Windows) V9

z

Multi-row INSERT, FETCH & multi-row cursor UPDATE, Dynamic Scrollable Cursors, GET DIAGNOSTICS, Enhanced UNICODE for SQL, join across encoding schemes, IS NOT DISTINCT FROM, Session variables, **TRUNCATE, DECIMAL FLOAT, VARBINARY, optimistic locking, FETCH CONTINUE, ROLE, MERGE, SELECT from MERGE**

Common

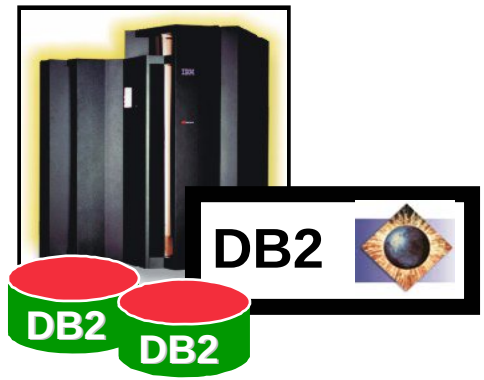
Inner and Outer Joins, Table Expressions, Subqueries, GROUP BY, Complex Correlation, Global Temporary Tables, CASE, 100+ Built-in Functions including SQL/XML, Limited Fetch, Insensitive Scroll Cursors, UNION Everywhere, MIN/MAX Single Index Support, Self Referencing Updates with Subqueries, Sort Avoidance 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 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 and fullselect, caseless comparisons, INTERSECT, EXCEPT, not logged tables**, **range partitioning, compression**

LUW

Updateable UNION in Views, GROUPING SETS, ROLLUP, CUBE, 16 Built-in Functions, SET CURRENT ISOLATION, multi-site join, MERGE, MDC, **XQuery**

List of topics

- INTERSECT, EXCEPT
- INSTEAD OF triggers
- MERGE
- SELECT FROM MERGE / UPDATE / DELETE
- TRUNCATE
- ORDER BY and FETCH FIRST in subselect
- New data types
  - BIGINT
  - BINARY / VARBINARY
  - DECFLOAT

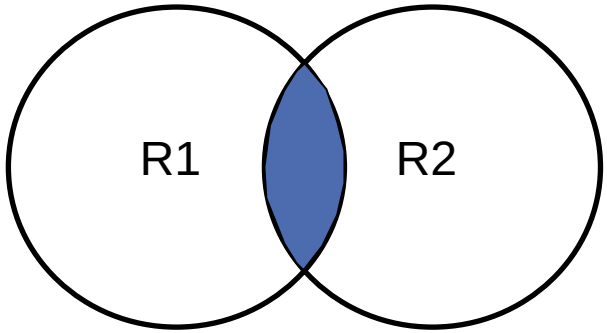


All functions discussed are marked if available in V9 CM or NFM!

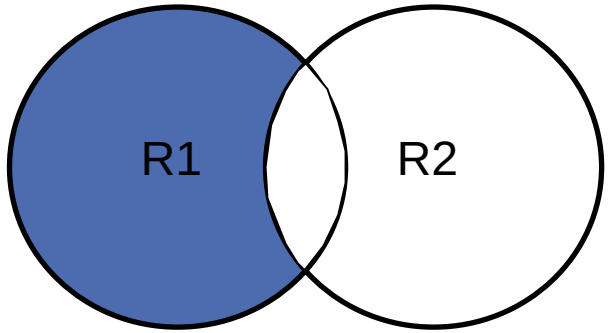
INTERSECT and EXCEPT

NFM

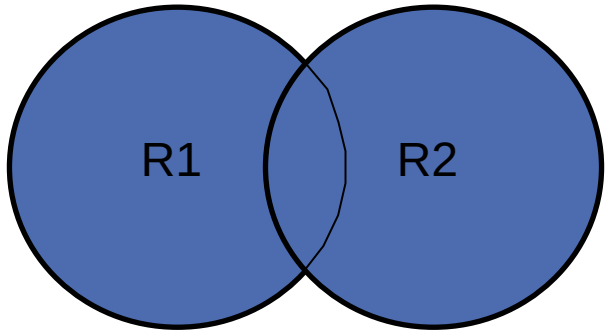
Options for joining sets



INTERSECT



EXCEPT



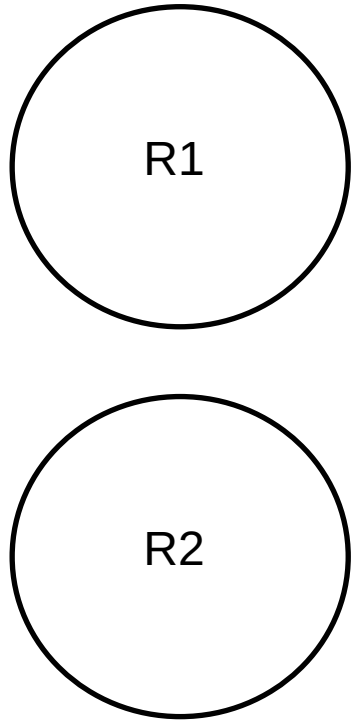
UNION

EXCEPT and INTERSECT Columns Participation

- Same as UNION today . . .
- R1 and R2 must have the same number of columns
  - data type for the n-th column of R1 must be compatible with the n-th column of R2
  - data type must not be CLOB, BLOB, DBCLOB, XML, or distinct type based on these types
- If the n-th column of R1 and the n-th column of R2 have the same name, then the n-th column of the result table has the same name; else unnamed
- Qualified column names cannot be used in the ORDER BY clause when the set operators are specified

Sample query for following examples

```
SELECT LAST_NAME, FIRST_NAME, . . .
        FROM first_table
        WHERE . . .
UNION | INTERSECT | EXCEPT
SELECT LAST_NAME, FIRST_NAME, . . .
        FROM second_table
        WHERE . . .
```



Result set R1

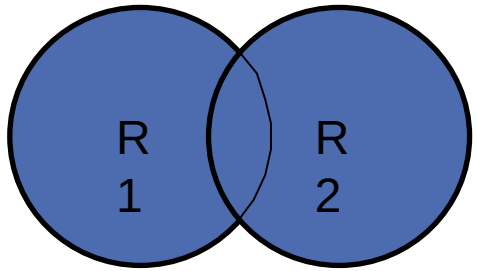
LAST_NAME	FIRST_NAME		. . .
Castellini	Massimiliano	row 1	. . .
Castellini	Massimiliano	1	. . .
Castellini	Massimiliano	1	. . .
Cook	Ian	2	. . .
Cook	Ian	2	. . .
Cook	Ian	2	. . .
Crocker	Tom	3	. . .
Jones	Gareth	4	. . .
Jones	Gareth	4	. . .
Wilson	Mark	5	. . .

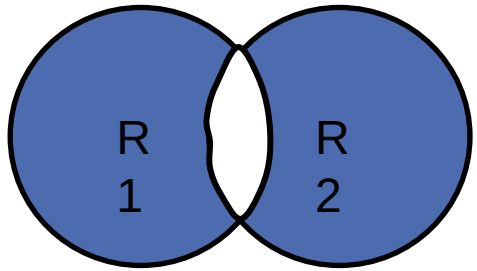
Let's Start with Tables R1 and R2

R1	R2
1	1
1	1
1	3
2	3
2	3
2	3
3	4
4	
4	
5	

Result of operations -- *r1* UNION *r2*

R1	R2	UNION ALL	UNION
1	1	1	1
1	1	1	2
1	3	1	3
2	3	1	4
2	3	1	5
2	3	2	
3	4	2	
4		2	
4		3	
5		3	
		3	
		3	
		3	
		4	
		4	
		4	
		5	



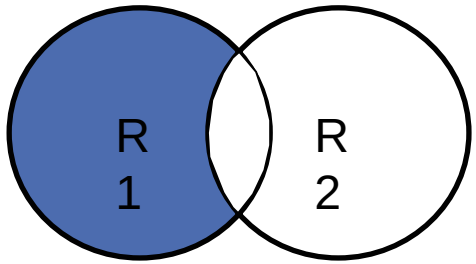


Show me all of the rows from the result table of each SELECT statement.

Result of operations -- *r1* EXCEPT *r2*

R1	R2
1	1
1	1
1	3
2	3
2	3
2	3
3	4
4	
4	
5	

EXCEPT ALL	EXCEPT
1	2
2	5
2	
2	
4	
5	

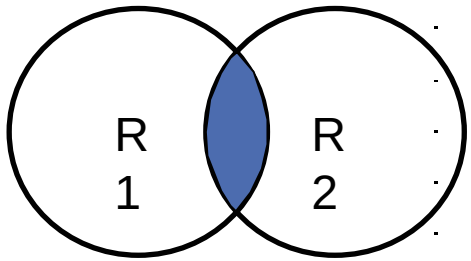


Show me the rows in R1 which do not have a corresponding row in R2.

Result of operations -- *r1* INTERSECT *r2*

R1	R2
1	1
1	1
1	3
2	3
2	3
2	3
3	4
4	
4	
5	

INTER-SECT ALL	INTER-SECT
1	1
1	3
3	4
4	



Show me the rows which appear in both R1 *and* R2.

Result of operations -- Examples of All

R1	R2	UNION ALL	UNION	EXCEPT ALL	EXCEPT	INTER-SECT ALL	INTER-SECT
1	1	1	1	1	2	1	1
1	1	1	2	2	5	1	3
1	3	1	3	2		3	4
2	3	1	4	2		4	
2	3	1	5	4			
2	3	2		5			
3	4	2					
4		2					
4		3					
5		3					
		3					
		3					
		3					
		4					
		4					
		4					
		5					

INSTEAD OF triggers

NFM

INSTEAD OF triggers:

current problem and goal

- Customers use views for read access control
- Many views are not updatable, so customers have to access base tables for data changes. Triggers can be used to help control updates.
- No INSERT / UPDATE / DELETE for read-only views
- **Goal:** to provide a mechanism to unify the target for all read / write access by an application (i.e., through views)

INSTEAD OF triggers

- A new type of trigger (~ BEFORE, AFTER triggers)
- Processed instead of the UPDATE, DELETE or INSERT statement that activated the trigger
- Can only be defined on views
  - provides an extension to the updatability of views
  - requested update operation against the view gets replaced by the trigger logic
  - application still believes all operations are performed against the view
  - applicable even for updatable views

Example

```
CREATE TABLE WEATHER (CITY VARCHAR(25), TEMPF DECIMAL(5,2))

CREATE VIEW CELCIUS_WEATHER (CITY, TEMPC) AS
      SELECT CITY, (TEMPF-32)/1.8 FROM WEATHER

CREATE TRIGGER CW_INSERT INSTEAD OF INSERT
        ON CELCIUS_WEATHER
    REFERENCING NEW AS NEWCW
     FOR EACH ROW MODE DB2SQL
         INSERT INTO WEATHER
               VALUES (NEWCW.CITY, 1.8*NEWCW.TEMPC+32)

CREATE TRIGGER CW_UPDATE INSTEAD OF UPDATE
        ON CELCIUS_WEATHER
   REFERENCING NEW AS NEWCW OLD AS OLDCW
     FOR EACH ROW MODE DB2SQL
         UPDATE WEATHER W
               SET W.CITY = NEWCW.CITY,
                     W.TEMPF = 1.8*NEWCW.TEMPC+32
            WHERE W.CITY = OLDCW.CITY
```


Restrictions

- Only 1 INSTEAD OF INSERT, UPDATE, DELETE per view
- View cannot be symmetric
- Only has row granularity
- No WHEN clause
- Cannot specify UPDATE OF column list
- Cannot change transition variables
- Does not work with position UPDATE / DELETE
- No LOB, XML

- SELECT FROM UPDATE/DELETE/INSERT not supported
- MERGE into a view with INSTEAD OF trigger is not supported

MERGE

NFM

Customer requirements for MERGE

- A customer requested a way of issuing a searched UPDATE from a multiple-row source data. The source data is used to update the database if there is a match found. Otherwise the source data is used to insert a new row in the database.
- Such support would also be useful in client/server oriented applications, which request a set of rows from the database, allow the user to modify the data through a GUI and then store the set of data back to the database. A combined UPDATE and INSERT operation would allow the user to update and add new rows on, say, a GUI window and then would allow the underlying application to easily reflect these changes in the database.

MERGE

- Combine UPDATE and INSERT operation to a target table or view, from a input source of host-variable-arrays modeled as a source table
  - When source rows match to target, update target rows from source
  - When source rows do not match to target, insert source rows into target

Example of MERGE

```
MERGE  INTO account AS T
USING (VALUES (:hv_id, :hv_amt) FOR 5 ROWS)  AS S (id,amt)
ON T.id = S.id
WHEN MATCHED THEN
   UPDATE SET balance = T.balance + S.amt
WHEN NOT MATCHED THEN
   INSERT (id, balance) VALUES (S.id, S.amt)
NOT ATOMIC CONTINUE ON SQLEXCEPTION
```

Target

Account - before

T.id	balance
1	1000
10	500
200	600
300	300
315	100
500	4000
...	

```
MERGE  INTO account AS T
USING (VALUES (:hv_id, :hv_amt) FOR 5 ROWS)  AS S (id,amt)
ON T.id = S.id
WHEN MATCHED THEN
   UPDATE SET balance = T.balance + S.amt
WHEN NOT MATCHED THEN
   INSERT (id, balance) VALUES (S.id, S.amt)
NOT ATOMIC CONTINUE ON SQLEXCEPTION
```

Source

:hv_id

S.id
1
5
10
5
1

:hv_amt

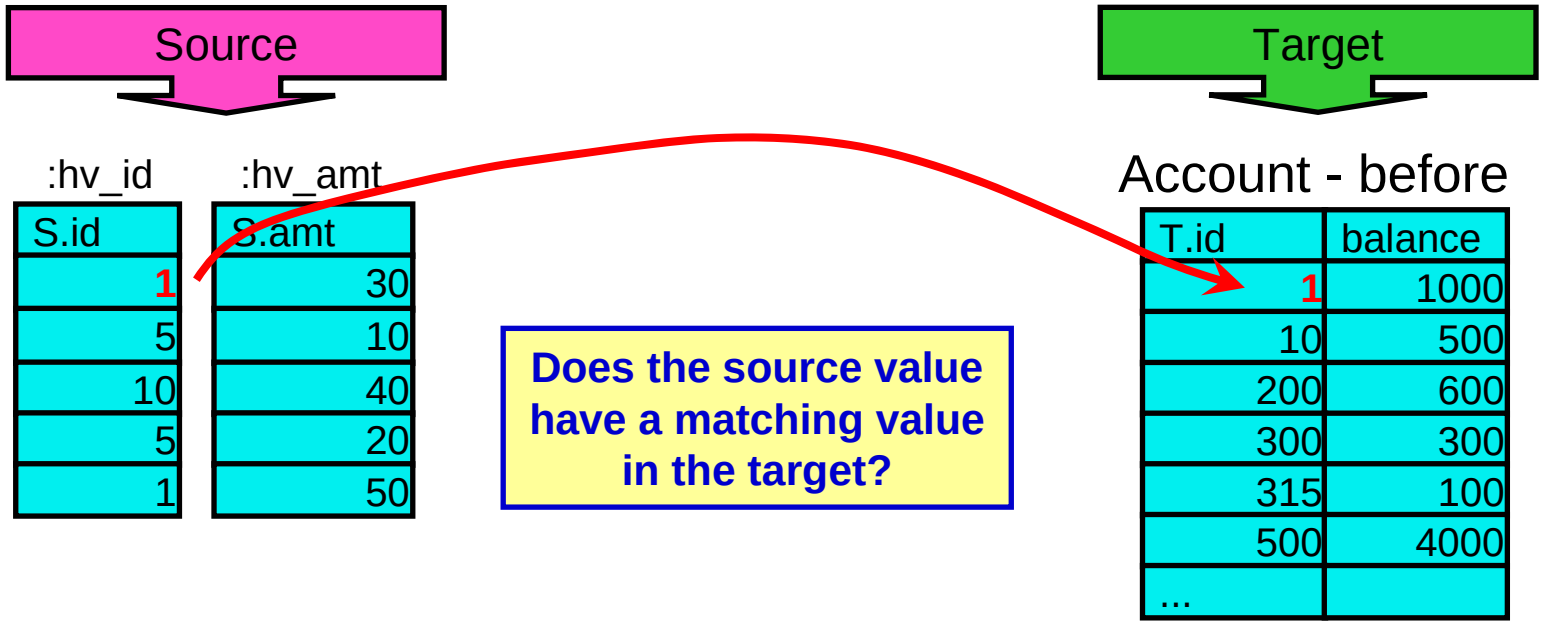
S.amt
30
10
40
20
50

Target

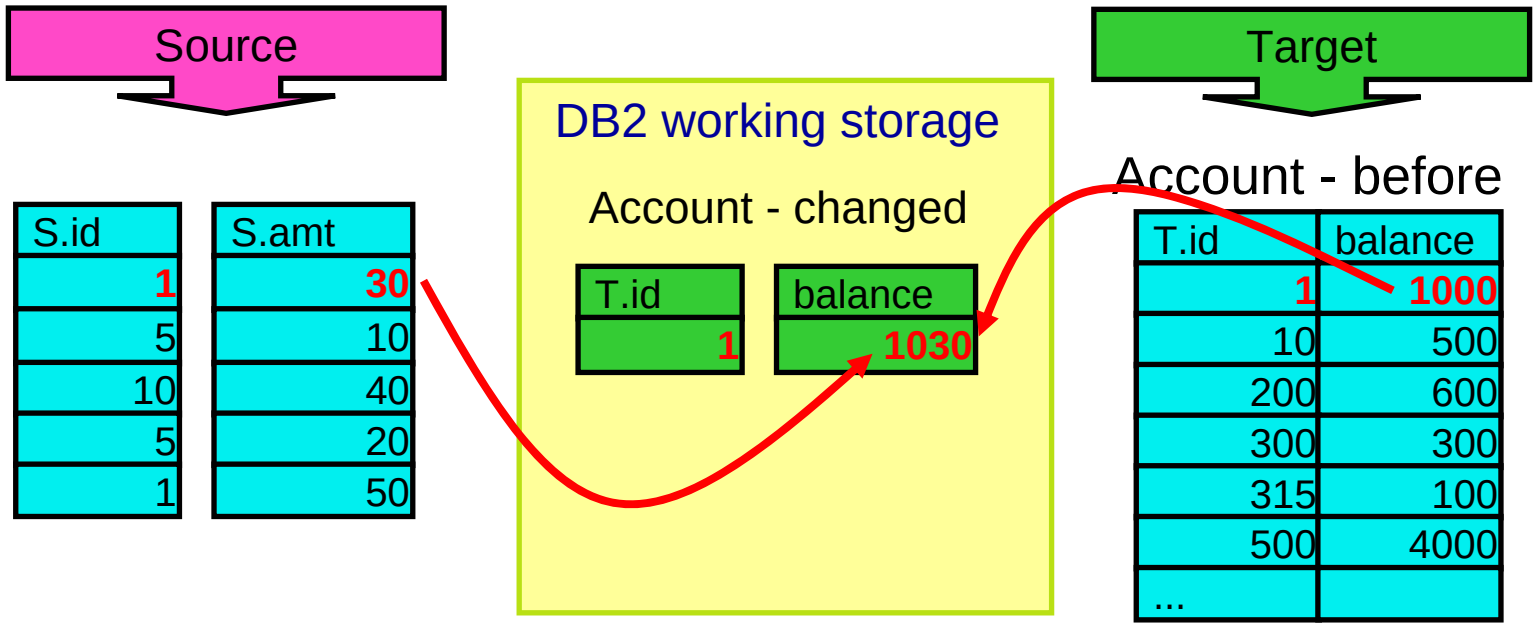
Account - before

T.id	balance
1	1000
10	500
200	600
300	300
315	100
500	4000
...	

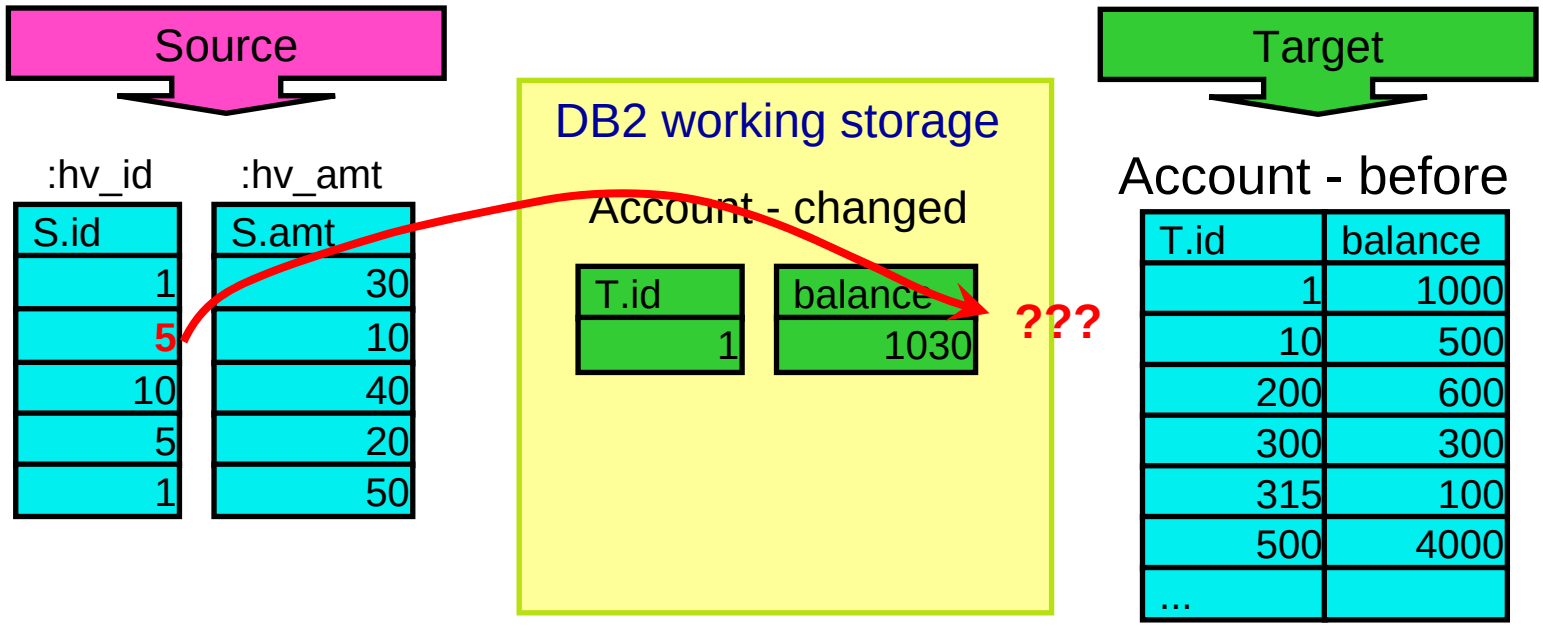
```
MERGE  INTO account AS T
USING (VALUES (:hv_id, :hv_amt) FOR 5 ROWS)  AS S (id,amt)
ON T.id = S.id
WHEN MATCHED THEN
   UPDATE SET balance = T.balance + S.amt
WHEN NOT MATCHED THEN
   INSERT (id, balance) VALUES (S.id, S.amt)
NOT ATOMIC CONTINUE ON SQLEXCEPTION
```



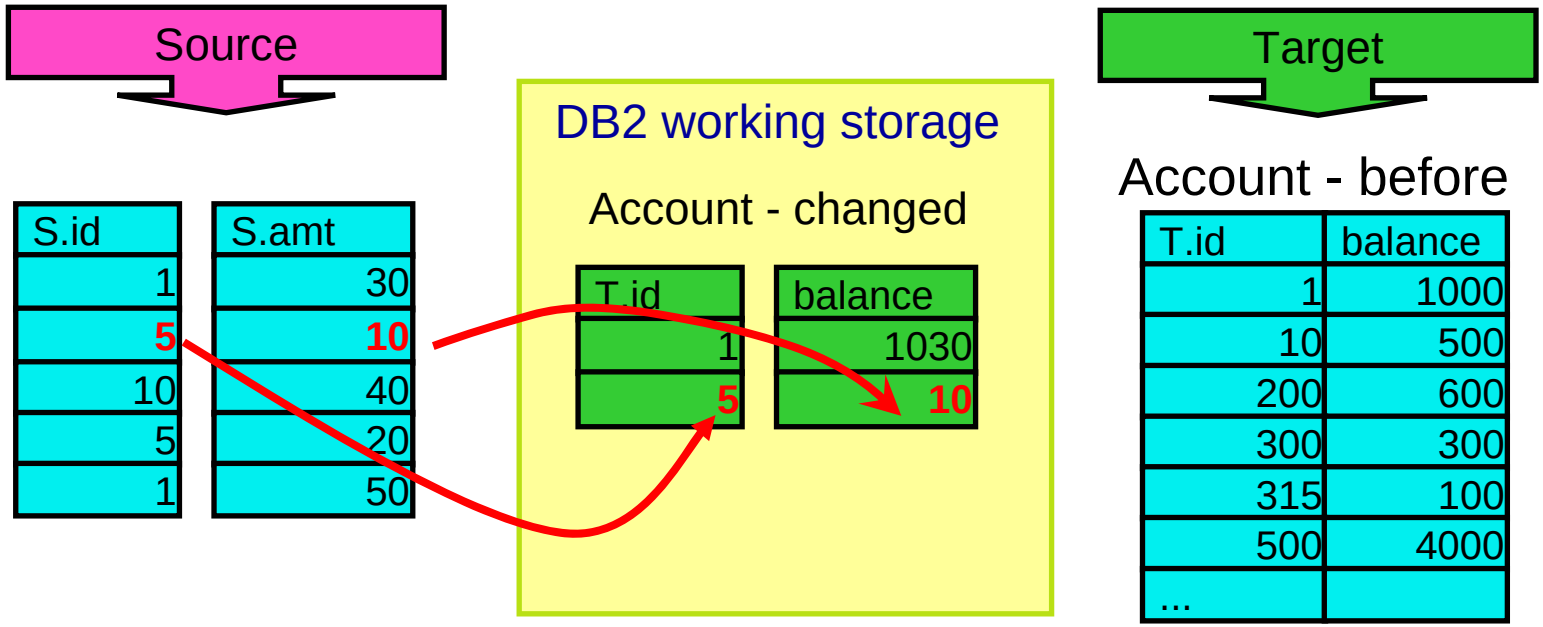

```
MERGE  INTO account AS T
USING (VALUES (:hv_id, :hv_amt) FOR 5 ROWS)  AS S (id,amt)
ON T.id = S.id
WHEN MATCHED THEN
   UPDATE SET balance = T.balance + S.amt
WHEN NOT MATCHED THEN
   INSERT (id, balance) VALUES (S.id, S.amt)
NOT ATOMIC CONTINUE ON SQLEXCEPTION
```



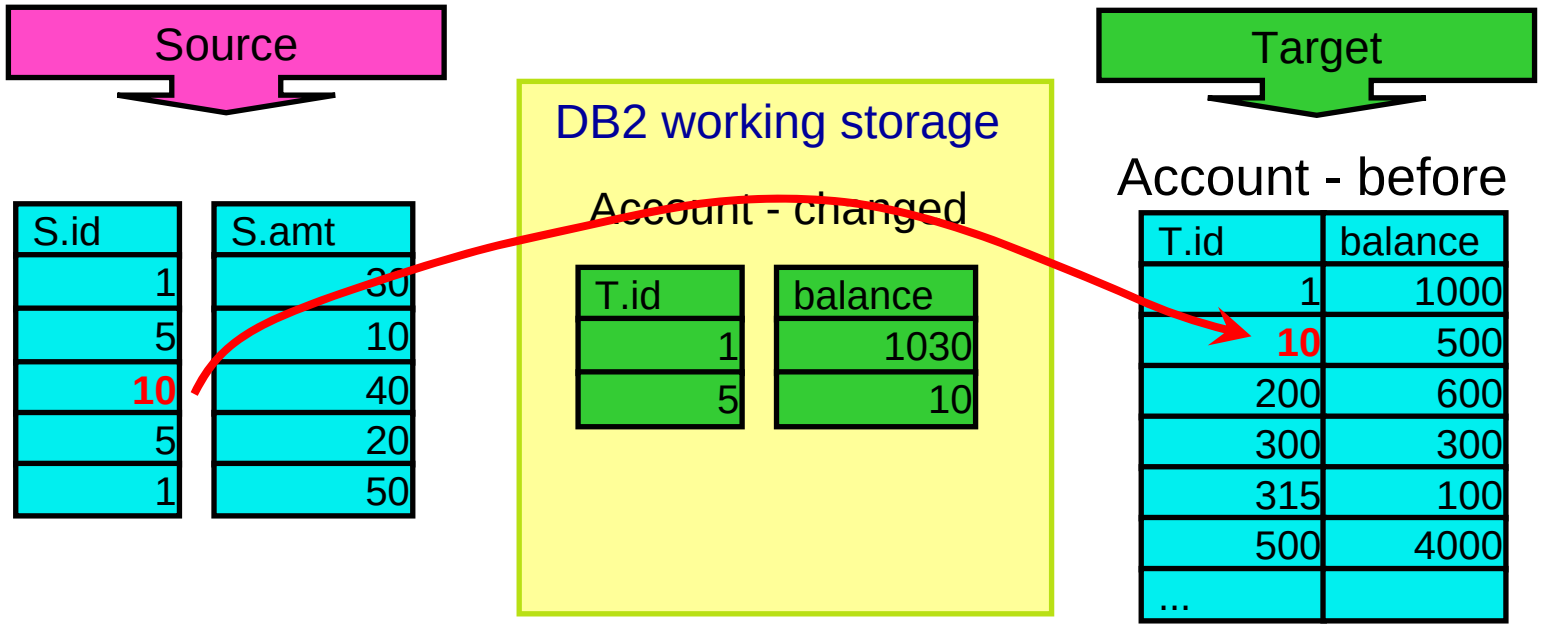

```
MERGE  INTO account AS T
USING (VALUES (:hv_id, :hv_amt) FOR 5 ROWS)  AS S (id,amt)
ON T.id = S.id
WHEN MATCHED THEN
   UPDATE SET balance = T.balance + S.amt
WHEN NOT MATCHED THEN
   INSERT (id, balance) VALUES (S.id, S.amt)
NOT ATOMIC CONTINUE ON SQLEXCEPTION
```



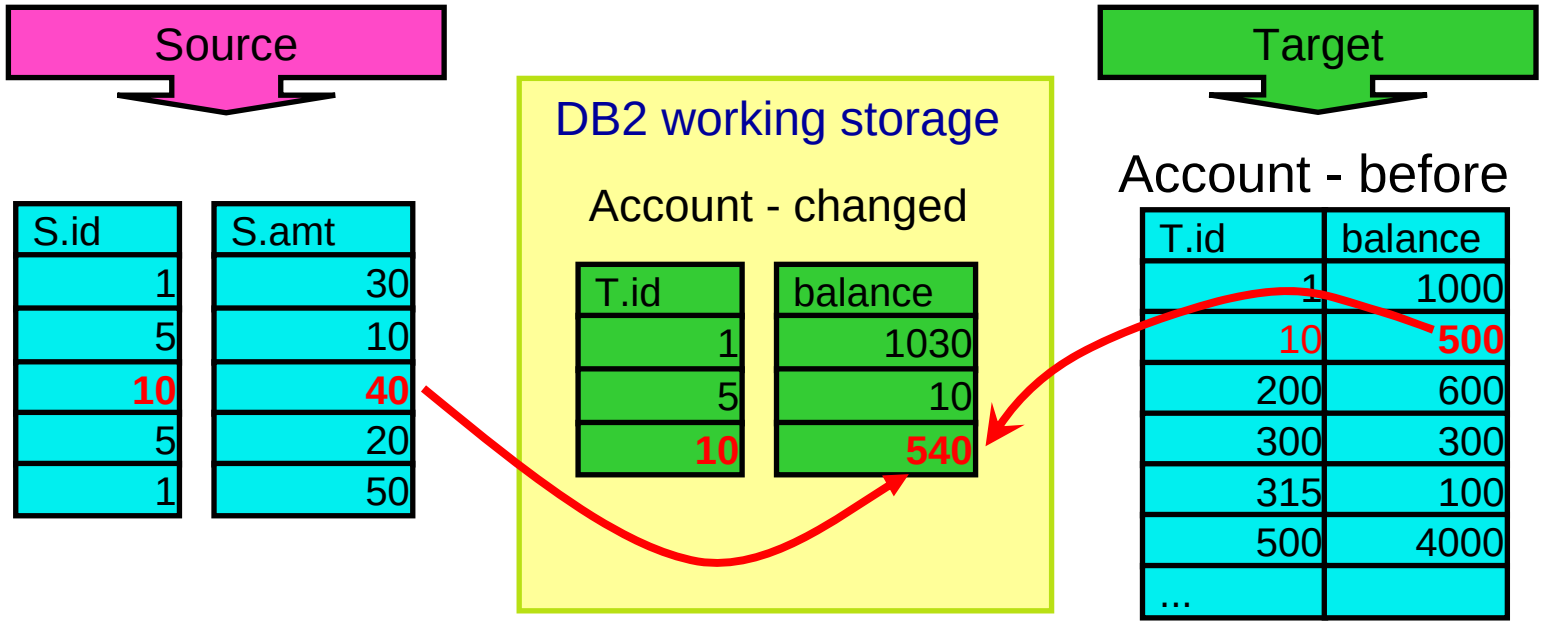

```
MERGE  INTO account AS T
USING (VALUES (:hv_id, :hv_amt) FOR 5 ROWS)  AS S (id,amt)
ON T.id = S.id
WHEN MATCHED THEN
   UPDATE SET balance = T.balance + S.amt
WHEN NOT MATCHED THEN
   INSERT (id, balance) VALUES (S.id, S.amt)
NOT ATOMIC CONTINUE ON SQLEXCEPTION
```



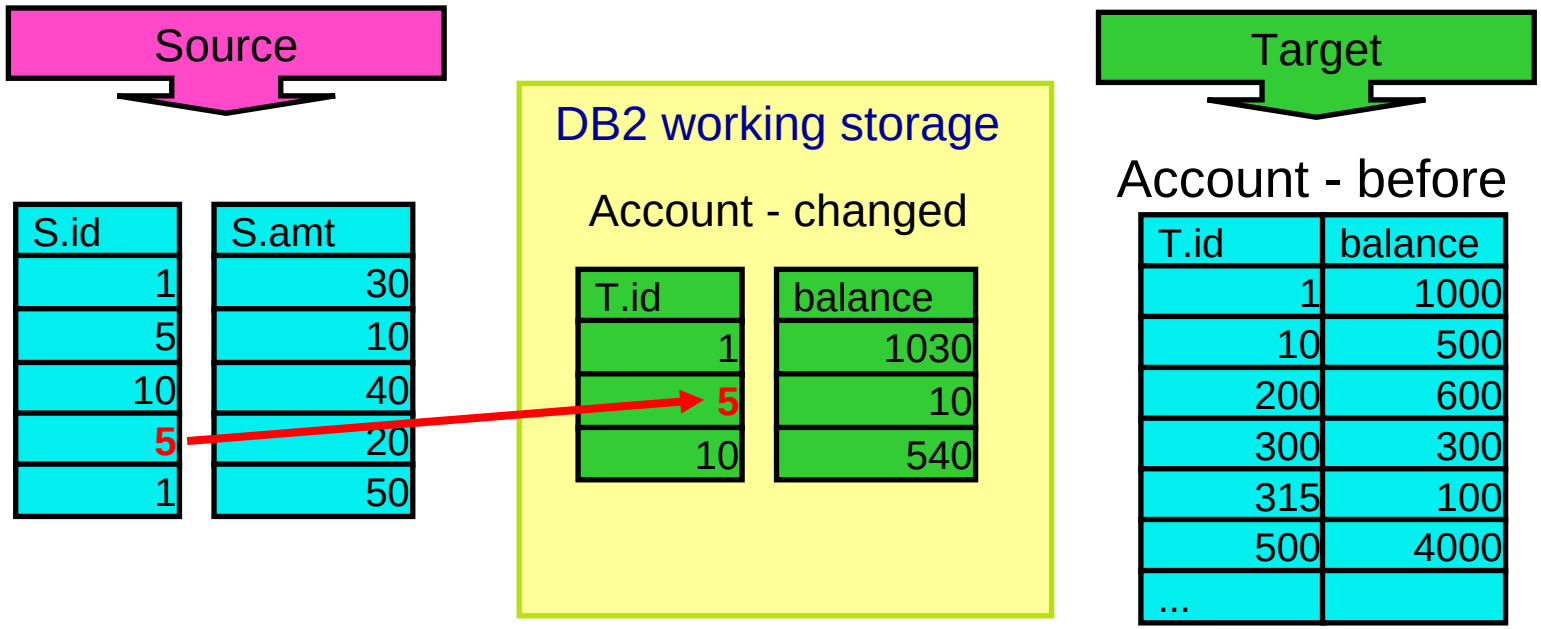

```
MERGE  INTO account AS T
USING (VALUES (:hv_id, :hv_amt) FOR 5 ROWS)  AS S (id,amt)
ON T.id = S.id
WHEN MATCHED THEN
   UPDATE SET balance = T.balance + S.amt
WHEN NOT MATCHED THEN
   INSERT (id, balance) VALUES (S.id, S.amt)
NOT ATOMIC CONTINUE ON SQLEXCEPTION
```



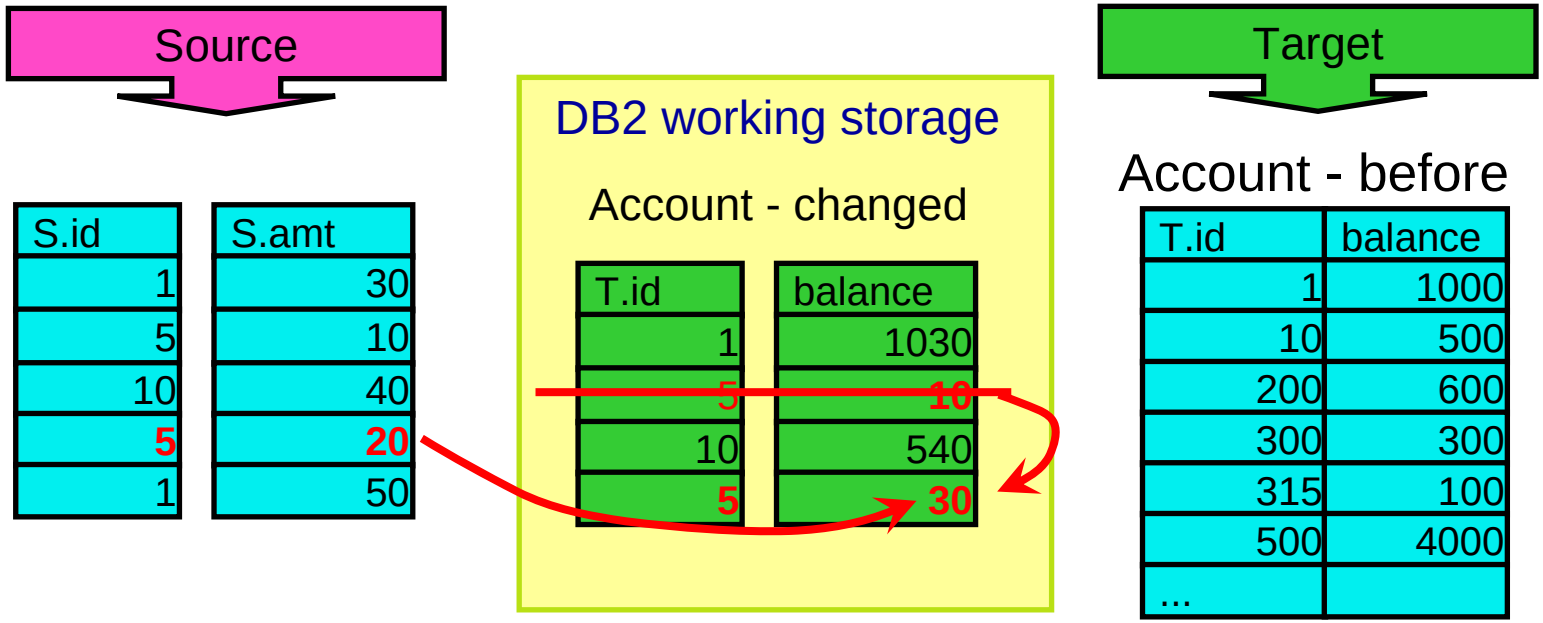

```
MERGE  INTO account AS T
USING (VALUES (:hv_id, :hv_amt) FOR 5 ROWS)  AS S (id,amt)
ON T.id = S.id
WHEN MATCHED THEN
   UPDATE SET balance = T.balance + S.amt
WHEN NOT MATCHED THEN
   INSERT (id, balance) VALUES (S.id, S.amt)
NOT ATOMIC CONTINUE ON SQLEXCEPTION
```



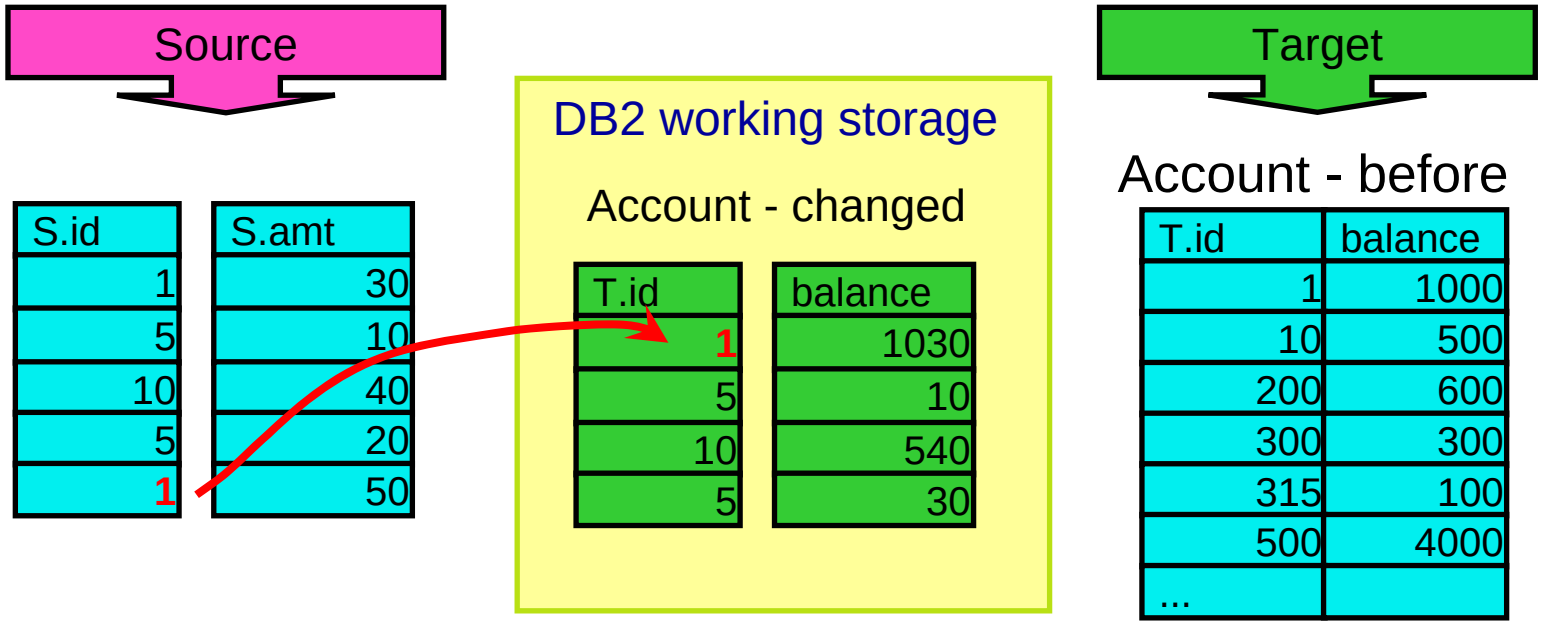

```
MERGE  INTO account AS T
USING (VALUES (:hv_id, :hv_amt) FOR 5 ROWS)  AS S (id,amt)
ON T.id = S.id
WHEN MATCHED THEN
   UPDATE SET balance = T.balance + S.amt
WHEN NOT MATCHED THEN
   INSERT (id, balance) VALUES (S.id, S.amt)
NOT ATOMIC CONTINUE ON SQLEXCEPTION
```



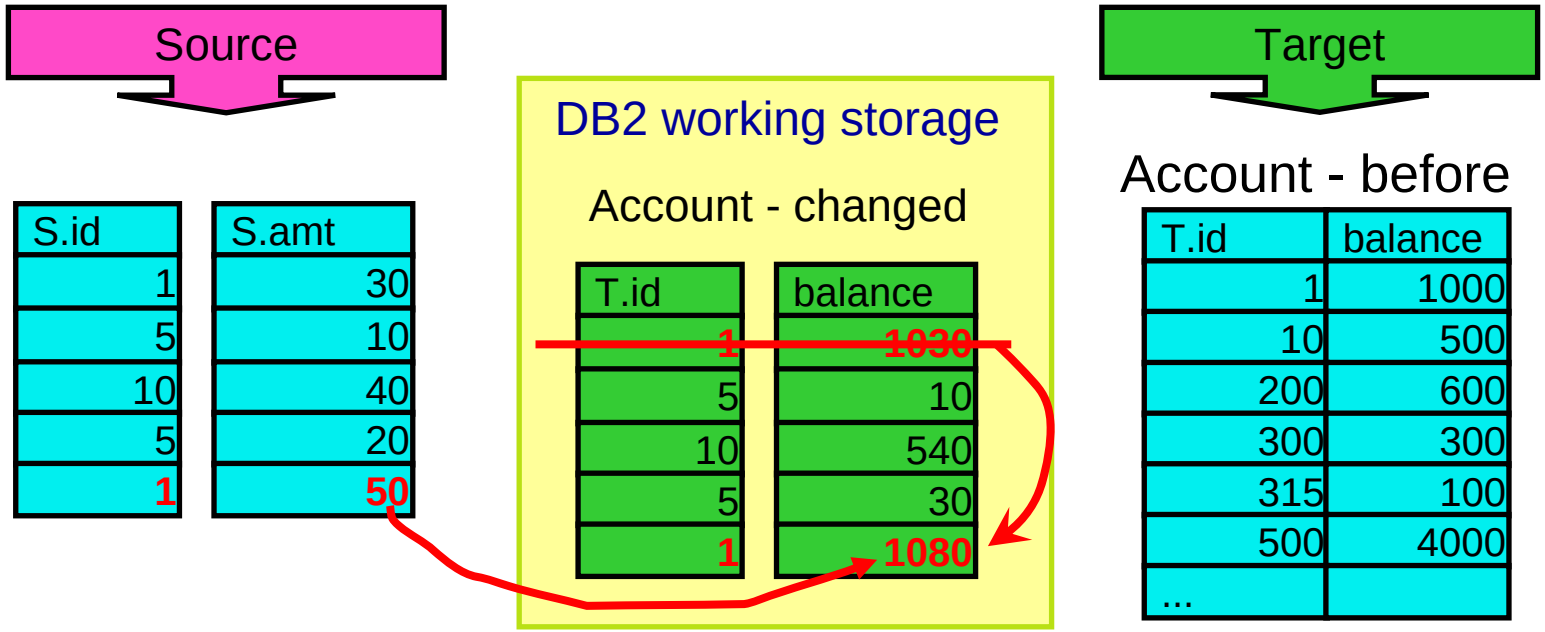

```
MERGE  INTO account AS T
USING (VALUES (:hv_id, :hv_amt) FOR 5 ROWS)  AS S (id,amt)
ON T.id = S.id
WHEN MATCHED THEN
   UPDATE SET balance = T.balance + S.amt
WHEN NOT MATCHED THEN
   INSERT (id, balance) VALUES (S.id, S.amt)
NOT ATOMIC CONTINUE ON SQLEXCEPTION
```




```
MERGE  INTO account AS T
USING (VALUES (:hv_id, :hv_amt) FOR 5 ROWS)  AS S (id,amt)
ON T.id = S.id
WHEN MATCHED THEN
   UPDATE SET balance = T.balance + S.amt
WHEN NOT MATCHED THEN
   INSERT (id, balance) VALUES (S.id, S.amt)
NOT ATOMIC CONTINUE ON SQLEXCEPTION
```




```
MERGE  INTO account AS T
USING (VALUES (:hv_id, :hv_amt) FOR 5 ROWS)  AS S (id,amt)
ON T.id = S.id
WHEN MATCHED THEN
   UPDATE SET balance = T.balance + S.amt
WHEN NOT MATCHED THEN
   INSERT (id, balance) VALUES (S.id, S.amt)
NOT ATOMIC CONTINUE ON SQLEXCEPTION
```




```
MERGE  INTO account AS T
USING (VALUES (:hv_id, :hv_amt) FOR 5 ROWS)  AS S (id,amt)
ON T.id = S.id
WHEN MATCHED THEN
   UPDATE SET balance = T.balance + S.amt
WHEN NOT MATCHED THEN
   INSERT (id, balance) VALUES (S.id, S.amt)
NOT ATOMIC CONTINUE ON SQLEXCEPTION
```

Source

S.id	S.amt
1	30
5	10
10	40
5	20
1	50

Target

Account - before

T.id	balance
1	1000
10	500
200	600
300	300
315	100
500	4000
...	

```
MERGE  INTO account AS T
USING (VALUES (:hv_id, :hv_amt) FOR 5 ROWS)  AS S (id,amt)
ON T.id = S.id
WHEN MATCHED THEN
   UPDATE SET balance = T.balance + S.amt
WHEN NOT MATCHED THEN
   INSERT (id, balance) VALUES (S.id, S.amt)
NOT ATOMIC CONTINUE ON SQLEXCEPTION
```

Account - after

T.id	balance
1	1080
5	30
10	540
200	600
300	300
315	100
500	4000
...	

SELECT FROM MERGE, UPDATE, DELETE

NFM

SELECT FROM UPDATE / DELETE / MERGE

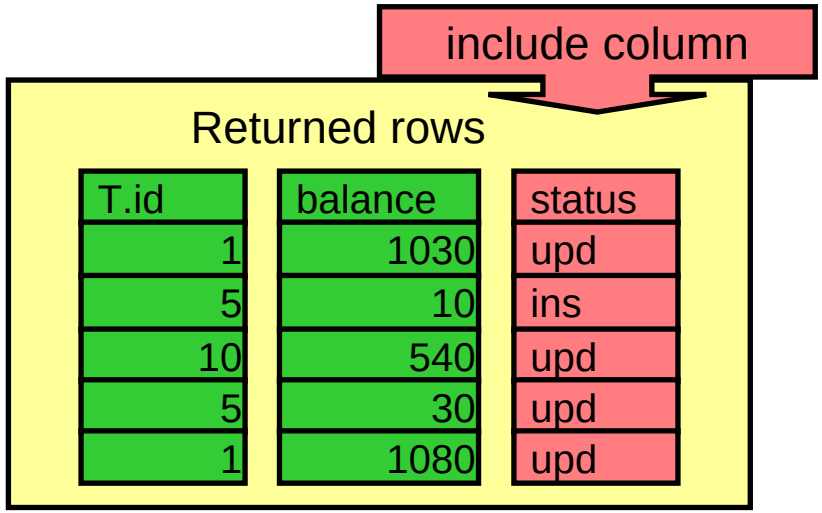
- SELECT from UPDATE or DELETE will be implemented by allowing a searched UPDATE or searched DELETE statement in the FROM clause of a select-statement that is a subselect or in the SELECT INTO statement. By allowing a searched UPDATE or searched DELETE to appear in a select-statement or SELECT INTO statement, the database will allow the user to know which values were updated in a table and which rows were deleted from a table via a single SQL statement.
- SELECT FROM MERGE will return all the updated rows and inserted rows, including column values which are generated by DB2.
- An INCLUDE column specification is being introduced to allow the user to identify a new column for the select-list and as a method for sorting the data (also added to SELECT from INSERT).

Example

- A user would like to know the sum of salaries of employees who are at level 'OPERATOR' and received a salary increase. In this scenario we can use FINAL TABLE with a searched UPDATE:

```
SELECT sum(salary) INTO :salary
FROM FINAL TABLE
  (UPDATE emp
  SET salary = salary * 1.05
  WHERE level = 'OPERATOR');
```

INCLUDE column



Source

S.id	S.amt
1	30
5	10
10	40
5	20
1	50

T.id	balance	status
1	1030	upd
5	10	ins
10	540	upd
5	30	upd
1	1080	upd

Account – target table

T.id	balance
1	1000
10	500
200	600
300	300
315	100
500	4000
...	

```
SELECT id, balance, status
FROM FINAL TABLE (
MERGE  INTO account AS T   INCLUDE (status  char(3) )
USING ( VALUES (:hv_id, :hv_amt) FOR 5 ROWS  AS S (id,amt)
ON T.id = S.id
WHEN MATCHED THEN
   UPDATE SET balance = T.balance + S.amt, status = 'upd'
WHEN NOT MATCHED THEN
   INSERT (id, balance, status) VALUES (S.id, S.amt, 'ins' )
NOT ATOMIC CONTINUE ON SQLEXCEPTION     )
```

Account - after

T.id	balance
1	1080
5	30
10	540
200	600
300	300
315	100
500	4000
...	

TRUNCATE

NFM

TRUNCATE TABLE

Customer Requirements

- Delete rows from a table without firing DELETE triggers
  - triggers could be dropped and recreated . . . changes the table definition

- Have an option to LOAD REPLACE that works on a table level in a segmented table space with multiple tables

What TRUNCATE does

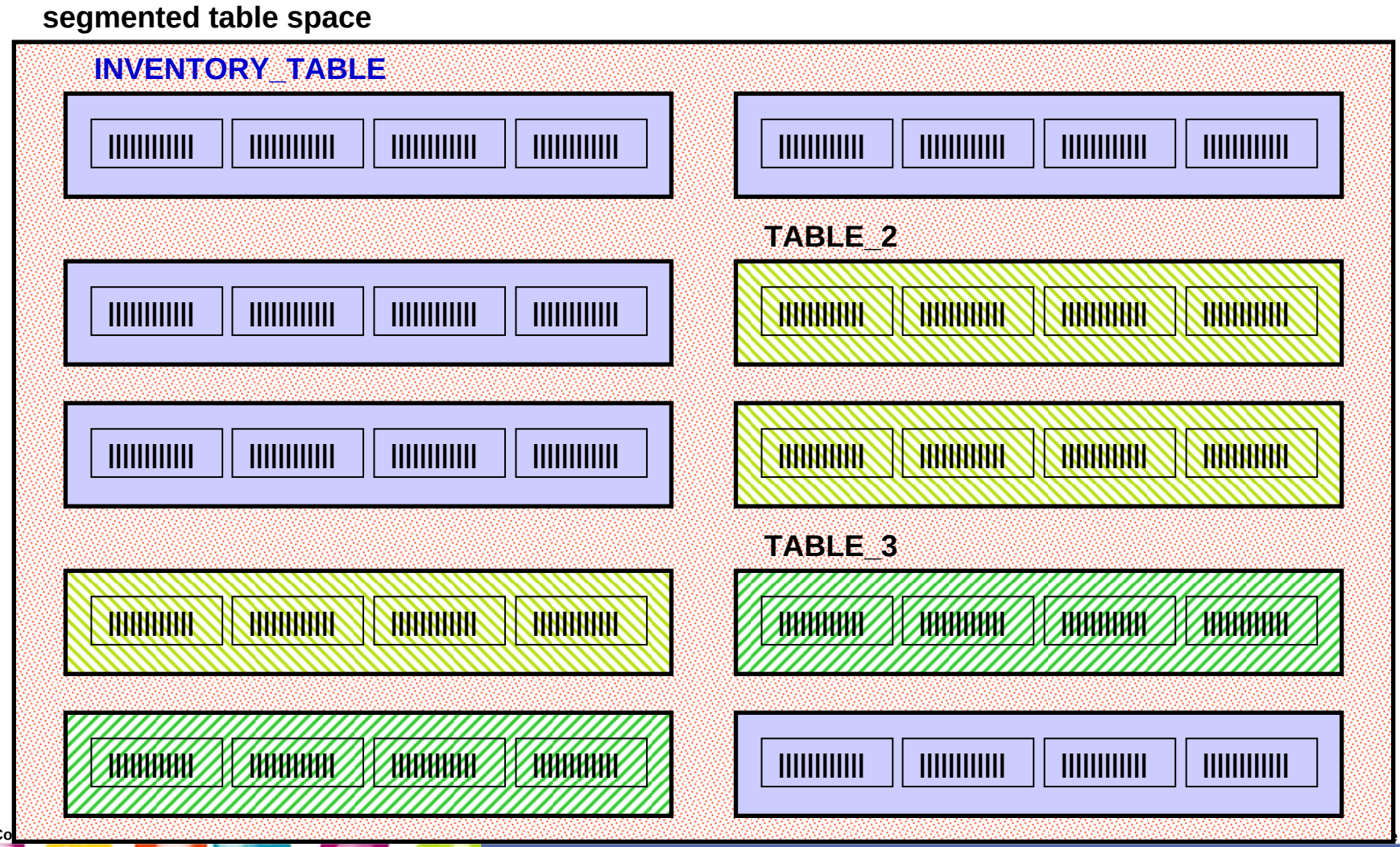
- Gives users an alternative way of emptying a table, with more flexibility over the current DELETE statement with no WHERE clause (i.e., a mass delete operation):
  - Delete all data rows in a designated DB2 table without activating DELETE triggers
    - » IGNORE DELETE TRIGGERS (default)
    - » RESTRICT WHEN DELETE TRIGGERS
  - DB2 catalog definition of the table (i.e., dropping and recreating of the delete triggers) is not needed for faster processing
  - Provides an option to allow the users to empty the designated DB2 table permanently without going through the current commit phase
  - Provides an option to reuse deallocated storage

Processing modes for TRUNCATE

- *Normal way* - truncate operation must process each data page to physically delete data records from the page
  - table in a simple table space
  - table in a partitioned table space
  - any table with table attributes
    - CDC-enabled (Change Data Capture)
    - MLS-enabled (Multiple Level Security)
    - VALIDPROC-enabled

- *Fast way* - truncate operation deletes data records without physically processing each data page
  - table in a segmented table space or a universal table space without the above table attributes

Storage --
-- 3 tables in segmented table space



TRUNCATE . . . IMMEDIATE

- **Specifies that the truncate operation is processed immediately and cannot be undone.**
- When IMMEDIATE option is specified, the table must not contain any uncommitted updates.
  - For a DGTT table object, the IMMEDIATE option does not apply to it. The truncate operation will fail since the table space contains a DGTT will be always in the update mode
  - No uncommitted DDL is allowed on the table prior to the TRUNCATE
- The truncated table is immediately available for use in the same unit of work.
- Although a ROLLBACK statement is allowed after the TRUNCATE statement, the truncate operation is not undone, and the table remains truncated. Other data changes following the TRUNCATE are rolled back.

ORDER BY and FETCH FIRST in subselect

CM

Customer requirement

One customer has a huge table of which they want just the first 2000 rows sorted in a particular order. Unfortunately, the sort is done first, and the fetch first after. This would cause a huge sort for no reason. They had to code this using a temp table which is a lot more work than a simple select. The solution to this to allow FETCH FIRST n ROWS in subquery:

```
SELECT A, B, C FROM
   (SELECT A,B,C FROM TABLEA
    WHERE...
    FETCH FIRST 2000 ROWS ONLY ) AS TABLEB
ORDER BY C,B
```

In V9

- Allow all semantically relevant clauses of the select statement to be pushed into subqueries. The original query can be taken as is and wrapped by more SQL, such as shown in the example above
- Provides more function by being able to select, e.g., the top n rows in a leg of a join, a leg of union, or a subquery.

```
(SELECT * FROM T1
   ORDER BY C1 FIRST 3  ROW ONLY)
UNION
SELECT * FROM T2
```

Restrictions

- A subselect that contains an ORDER BY or FETCH FIRST clause cannot be specified:
  - In the outermost fullselect of a view.
  - In a materialized query table
  - Unless the subselect is enclosed in parenthesis

Example:

```
CREATE VIEW V1 AS
  (SELECT * FROM T1 ORDER BY C1);
```

SQLCODE -20211

```
SELECT * FROM T1
  ORDER BY C1
UNION
SELECT * FROM T2
  ORDER BY C2
```

SQLCODE -104

Data types

New data types

- BIGINT — NFM
- BINARY and VARBINARY — NFM
- DECFLOAT — NFM

BIGINT

- An exact numeric capable of representing 63-bit integers
  - 8 bytes of storage
  - Range:
    - -9223372036854775808 *to*
    - 9223372036854775807
- Compatible with all numeric types
- Introduced for compatibility with Java, C, C++, and SQL standards

BIGINT extensions to existing functions

- CHAR
- DIGITS
- LENGTH
- MOD
- MULTIPLY_ALT
- POWER
- VARCHAR

VARBINARY and BINARY

- BINARY - fixed-length binary string
  - 1 to 255 bytes
- VARBINARY - variable-length binary string
  - 1 to 32704 bytes; maximum length determined by the maximum record size associated with the table
- Both are compatible with BLOBs
- Neither are compatible with character string data types
  - Similar to FOR BIT DATA character strings
  - Can use CAST specification to change FOR BIT DATA character string into binary string
  - **There is a difference in padding characters:**
    - **[VAR]CHAR padded with spaces (X'40' for EBCDIC, X'20' for ASCII and Unicode)**
    - **BINARY padded with hex zeros (X'00')**
    - **VARBINARY not padded, even during comparisons**

Comparison of binary strings

- Two binary strings are equal only if the lengths are identical
- If two strings are equal up to the length of the shorter string length
  - the shorter string is considered less than the longer string
  - even when the remaining bytes in the longer string are hex zeros

Hex value of operand 1	relationship	Hex value of operand 2
X’4100’	<	X’410000’
X’4100’	<	X’42’
X’4100’	=	X’4100’
X’4100’	>	X’41’
X’4100’	>	X’400000’

Functions extended for VARBINARY and BINARY

- INSERT
- LEFT
- LTRIM
- POSSTR; POSITION does not support binary types
- REPEAT
- REPLACE
- RIGHT
- RTRIM
- STRIP
- SUBSTR

Online schema and binary columns

- **A column data type could be altered only to a compatible data type.**
- However, to ease the migration of existing applications, altering CHAR FOR BIT DATA or VARCHAR FOR BIT DATA column data types to BINARY or VARBINARY data types will be allowed (even though they are not considered to be compatible).
- When a CHAR FOR BIT DATA, or VARCHAR FOR BIT DATA column is altered to a BINARY or VARBINARY data type, and there is an index defined on that column, the index will be put in RBDP.
- Altering BINARY or VARBINARY data types to CHAR FOR BIT DATA or VARCHAR FOR BIT DATA will not be allowed.

Caution!!

- Caution should be taken when a CHAR FOR BIT DATA column is altered to a BINARY data type due to differences in padding.
- When a CHAR FOR BIT DATA column is altered to BINARY, the existing space characters in the table will not be changed to hexadecimal zeros (X'00). In addition, if the new length attribute is greater than current length attribute of the column, the values in the table are padded with hexadecimal zeros (X'00).

```
CREATE TABLE T1 ( C1 CHAR(5) FOR BIT DATA) CCSID EBCDIC;
INSERT INTO T1 VALUES(X'C1C2C3');
INSERT INTO T1 VALUES(X'C1C2C3C4C5');
COMMIT;

SELECT HEX( C1 ) FROM T1;
        returns:        C1C2C34040
                        C1C2C3C4C5
```

DECFLOAT

- Decimal floating point (DECFLOAT) is similar to both
  - Packed decimal (or binary coded decimal), and
  - Floating point (IEEE or hex)
- The main advantages that decimal floating point has over packed decimal or binary floating point (IEEE):
  - it can contain a larger number
    - in terms of digits of significance
    - in terms of exponent
- The rules for manipulation of DECFLOAT more closely follow the rules for manipulation of packed decimal:
  - DECFLOAT processing deals with exact numbers
  - IEEE floating point (binary) deals with numerical approximations

DECFLOAT details

- All numbers have a sign, a precision, and a scale
- The precision is the total number of decimal digits excluding the sign (either 16 or 34 for DECFLOAT)
- The scale is the total number of decimal digits to the right of the decimal point
- An exponent range of respectively 10^{-383} to 10^{+384} or 10^{-6143} to 10^{+6144}
- DECFLOAT(16):
  - Negative values: $-9.999999999999999 \times 10^{+384}$ to $-1.000000000000000 \times 10^{-383}$
  - Positive values: $1.000000000000000 \times 10^{-383}$ to $9.999999999999999 \times 10^{+384}$
- DECFLOAT(34):
  - Negative values: $-9.999999999999999999999999999999999 \times 10^{+6144}$ to $-1.000000000000000000000000000000000 \times 10^{-6143}$
  - Positive values: $1.000000000000000000000000000000000 \times 10^{-6143}$ to $9.999999999999999999999999999999999 \times 10^{+6144}$

Equality for DECFLOAT

- The DECFLOAT data type allows for multiple bit representations of the same number. Additionally, numbers with the same coefficient can have different exponents, and therefore different bit representations.
- For example 2.00 and 2.0 are two numbers with the same coefficient, but different exponent values.
- Thus, 2.00 <> 2.0 at a binary level, however the = (equal) predicate will return true for a comparison of 2.0 = 2.00. Given that 2.0 = 2.00 (the comparison is true), 2.0 < 2.00 is false. The behavior that is described here holds true whenever DB2 compares DECFLOAT data (such as for UNION, SELECT DISTINCT DECFLOAT_column, COUNT(DISTINCT DECFLOAT_column), basic predicates, IN predicates, etc)
- Example:

```
SELECT 2.0 FROM SYSIBM.SYSDUMMY1
UNION [DISTINCT]
SELECT 2.00 FROM SYSIBM.SYSDUMMY1          yields 1 row
```

Language support for DECFLOAT

- The following languages are currently supported:
  - Java
  - Assembler
  - REXX
- See the Reference Material, DB2 manuals for details

Shameless Self promotion

http://blogs.ittoolbox.com/database/db2zos

Other resources

http://www.ibm.com/redbooks

Redbooks

SG24-7330 - DB2 9 for z/OS Technical Overview
SG24-7473 - DB2 9 for z/OS Performance Topics
SG24-7421 - DB2 9 for z/OS New Tools for Query Optimization

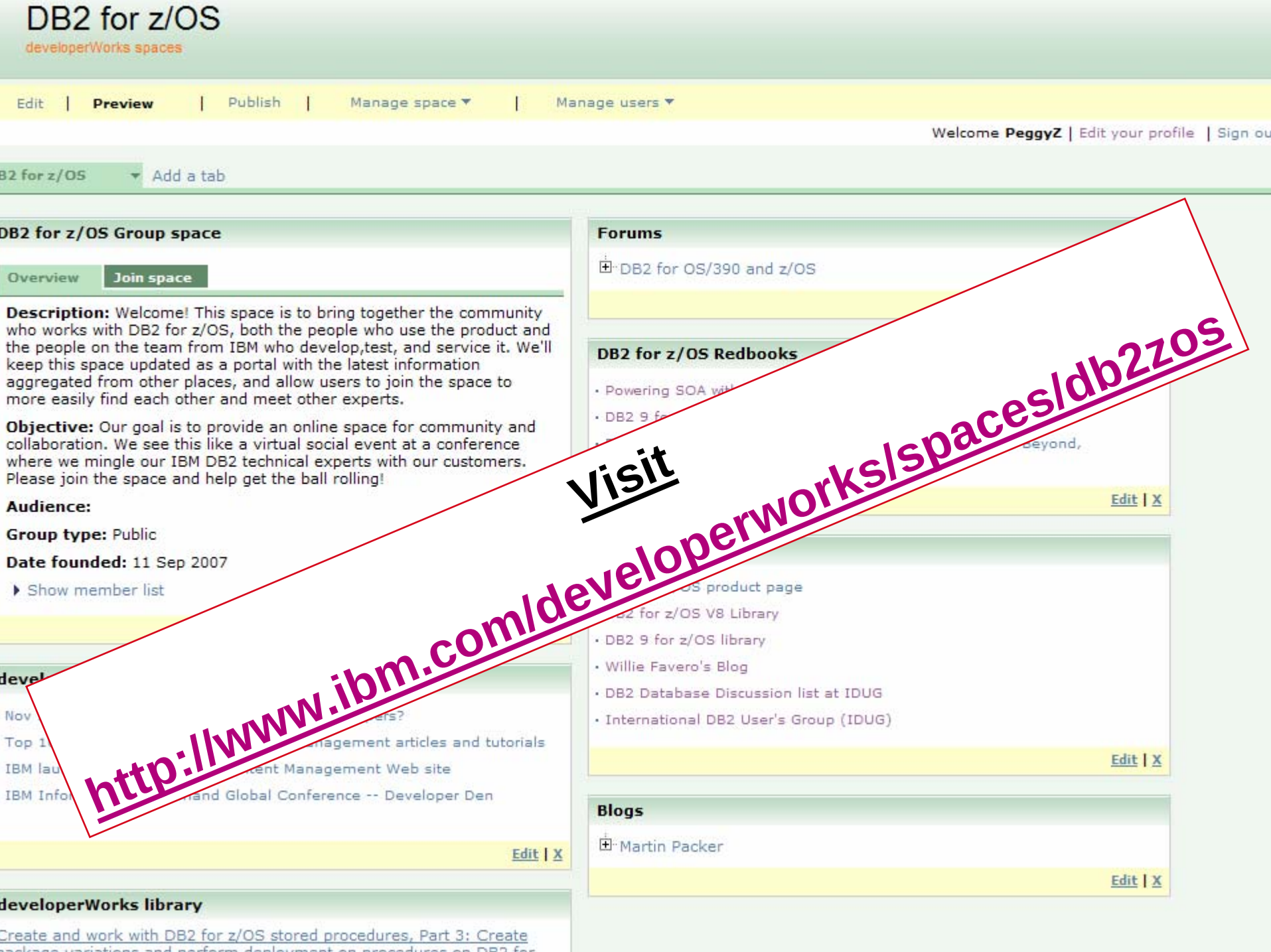
Redpaper

REDP-4345 - Index Compression with DB2 9 for z/OS

Summary -- it's all exciting new SQL!!!

- INTERSECT, EXCEPT
- INSTEAD OF triggers
- MERGE
- SELECT FROM MERGE / UPDATE / DELETE
- TRUNCATE
- ORDER BY and FETCH FIRST in subselect
- New data types
  - BIGINT
  - BINARY / VARBINARY
  - DECFLOAT

All functions discussed are marked if available in V9 CM or NFM!


DB2 for z/OS
developerWorks spaces
Edit | Preview | Publish | Manage space | Manage users
Welcome PeggyZ | Edit your profile | Sign ou
B2 for z/OS
Add a tab
DB2 for z/OS Group space
Overview
Join space
Description: Welcome! This space is to bring together the community who works with DB2 for z/OS, both the people who use the product and the people on the team from IBM who develop,test, and service it. We'll keep this space updated as a portal with the latest information aggregated from other places, and allow users to join the space to more easily find each other and meet other experts.
Objective: Our goal is to provide an online space for community and collaboration. We see this like a virtual social event at a conference where we mingle our IBM DB2 technical experts with our customers. Please join the space and help get the ball rolling!
Audience:
Group type: Public
Date founded: 11 Sep 2007
Show member list
Forums
DB2 for OS/390 and z/OS
DB2 for z/OS Redbooks
Edit | X
DB2 for z/OS V8 Library
DB2 9 for z/OS library
Willie Favero's Blog
DB2 Database Discussion list at IDUG
International DB2 User's Group (IDUG)
Edit | X
Blogs
Martin Packer
Edit | X
developerWorks library
Edit | X
Visit
http://www.ibm.com/developerworks/spaces/db2zos

Session #1094

New Cool SQL: DB2 9 for z/OS

Willie Favero

Senior Certified IT Software Specialist

DB2 for z/OS Specialty SSR

IBM Certified Database Administrator - DB2 Universal Database V8.1 for z/OS

IBM Certified Database Administrator – DB2 9 for z/OS

IBM zChampion

wfavero@attglobal.net

and/or

wfavero@us.ibm.com