



IBM Software Group

A Sneak Peak at DB2 9 for z/OS

DB2 Information Management Software

William Favero
IBM S&D, West Region
Senior Certified IT Software Specialist
wfavero@attglobal.net



Copyright © 2007 IBM Corporation

IBM Software Group | DB2 Information Management Software



Shameless Self promotion

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

DB2 9 for z/OS Innovation: Insight

DB2 9 for z/OS Innovation: Insight

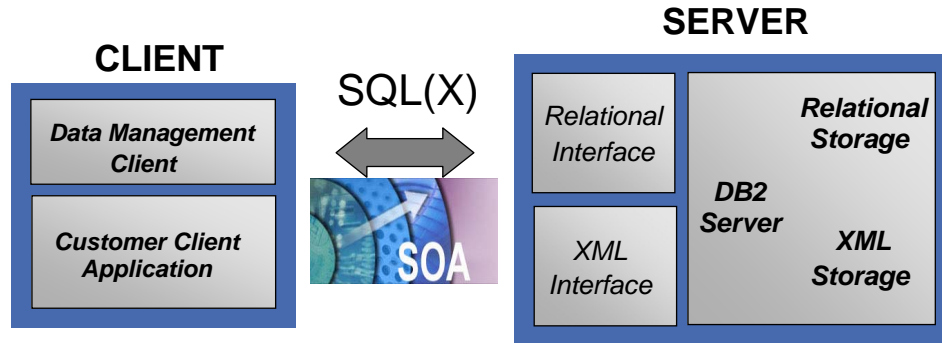
- Business Insight
 - Hybrid data server: XML and relational
 - Enhanced SQL
 - Business Partner improvements
 - QMF cross-platform work station & web
- Cost Savings through Optimization
- Business Resiliency





Capabilities Inside the Engine

Performance, Performance, Performance



Native storage Schema Index Functions Utilities

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
- ROLE & trusted context
- Many new built-in functions, caseless comparisons
- Index on expression
- Improved DDL consistency
- CURRENT SCHEMA

SOA and Information on demand

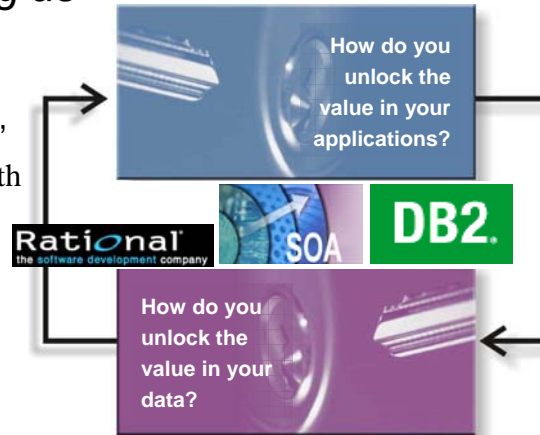
What the market is telling us

Huge investment in traditional programs & data

- “200 Billion lines of COBOL ...”
- “more than a trillion dollars’ worth of legacy mainframe code.”

The key to unlocking value:

- Unstructured and structured
- Easy to find & use, tailored, cost-effective & integrated



It's time to unlock the value of these assets

Business Partners: Many enhancements

- SHRLEVEL(REFERENCE) for REORG of LOB table spaces
- Online RENAME COLUMN
- Online RENAME INDEX
- Online CHECK DATA and CHECK LOB
- Faster REORG by intra-REORG parallelism
- More online REORG by eliminating BUILD2 phase
- LOB Lock reduction
- Skipping locked rows option
- Online REBUILD INDEX
- Change SCHEMA & VCAT
- Tape support for BACKUP and RESTORE SYSTEM utilities
- Recovery of individual tablespaces and indexes from volume-level backups
- Enhanced STOGROUP definition
- Utility TEMPLATE switching
- Conditional restart: automatic search for appropriate checkpoint
- CLONE Table: fast replacement of one table with another
- Buffer management by WLM
- Global query optimization
- Generalizing sparse index and in-memory data caching method
- Optimization Service Center
- Autonomic reoptimization
- Logging enhancements
- LOBs Network Flow Optimization
- Faster operations for variable-length rows
- NOT LOGGED table spaces
- Index on expressions
- Universal Table spaces
- Partition-by-growth table spaces
- APPEND option at insert
- Autonomic index page split
- Index page sizes 8K, 16K, 32K
- Support for optimistic locking
- Faster and more automatic DB2 restart
- MODIFY RECOVERY enhancements
- RLF improvements for remote application servers such as SAP
- Preserving consistency when recovering individual objects to a prior point in time
- DECIMAL FLOAT, BIGINT
- VARBINARY, BINARY
- TRUNCATE TABLE statement
- MERGE statement
- FETCH CONTINUE
- ORDER BY and FETCH FIRST n ROWS in sub-select and full-select
- ORDER OF extension to ORDER BY
- Various scalar functions
- XML support in DB2 engine
- Native SQL Stored Procedures, able to use zIIP
- SELECT FROM UPDATE/DELETE/MERGE
- Enhanced CURRENT SCHEMA
- IPv6 support
- Unified Debugger
- Network Trusted Context
- Database ROLES
- Automatic creation of database objects
- Modify early code without requiring an IPL
- Utilities CPU reduction
- Temporary space consolidation
- . . .

TCO Reduction

Continuous Availability

Performance

Scalability

SQL

Portability

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

c

o

m

m

o

n

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

i

u

w

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 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, TRUNCATE, DECIMAL FLOAT, VARBINARY, optimistic locking, FETCH CONTINUE, ROLE, MERGE, SELECT from MERGE

c

o

m

m

o

n

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

i

u

w

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

QMF

- Drag and drop querying, reporting and analytics
- Executive dashboards and data visualization
- Enhanced OLAP with DB2 Cube Views
- Reengineered cross-platform development environment
- New security model for access control & personalization
- Enabled for WebSphere Application Server



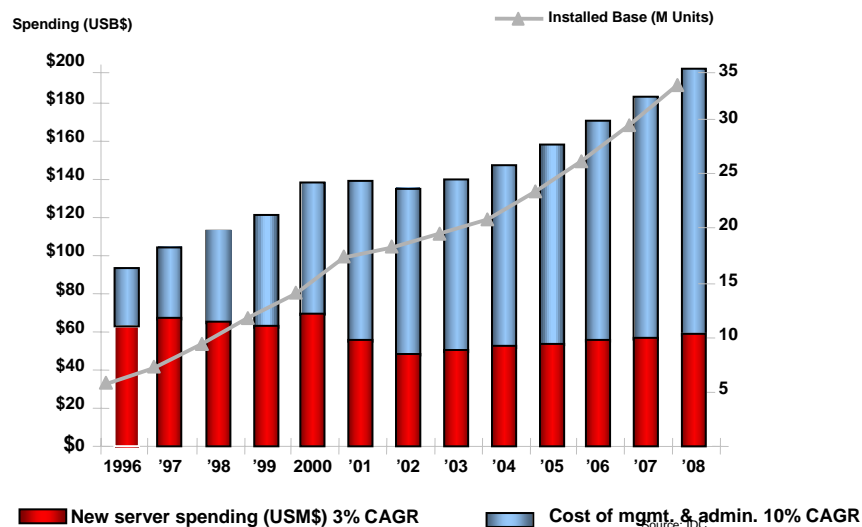
DB2 9 for z/OS Innovation: Cost

DB2 9 for z/OS Innovation: Cost

- Business Insight
- Cost Savings through Optimization
 - Security and Regulatory Compliance
 - Performance improvements
 - Synergy with System z
 - Query enhancements
- Business Resiliency

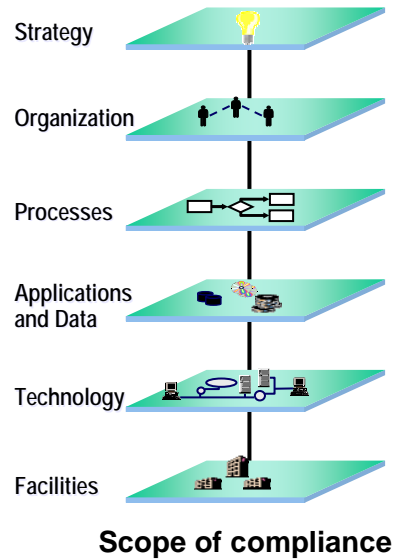


IDC: Since 2000, Labor Costs Have Exceeded the Cost of All Servers ... and are *Still* Growing



Regulatory Compliance Characteristics

- **Mandatory:** Agencies may require qualification and/or adherence to specific standards
- **Non-directive:** Usually do not specify exact steps needed to comply
- **Non-certifiable:** Agencies generally do not approve, recommend, or validate solutions
- **Continuously changing:** Regulations, their interpretation, and their codification into corporate policies change frequently
- **Increasing IT impact:** While few apply directly to IT, regulations increasingly affect IT systems
- **Intrinsically related to risk:** Agencies now recommend a risk-based management approach



Regulatory Compliance in DB2 9 for z/OS

- Key implementations
- Network Trusted Contexts
- Roles
- Improved auditing
- Secure Socket Layer
- Data Encryption



DB2 9 for z/OS Performance Improvements

- CPU reductions in utilities: LOAD, REORG, ...
 - Online REORG with no BUILD2 phase
- LOB performance, function, scalability
- SQL and optimization improvements, inserts
- Use zIIP in remote native SQL Procedure Language
- Improved varying length performance
- Sequential disk access



System z Synergy & DB2 9

- System z9 Integrated Information Processor (zIIP) Enterprise Class & Business Class
- Enhanced Cryptography
- Channels (4 Gb & MIDAW)
- Faster Processors
- Up to 54 Processors EC
- More memory, better value; 64 bit virtual storage
- z/Architecture new instructions
- Parallel Sysplex
- IPv6
- SSL
- Java
- Decimal float
- Backup & restore
- Security
- Unicode collation
- Compression
- System z Application Assist Processor (zAAP)
- WLM enhanced ...



DB2 for z/OS Version 8 News

now

- Cross loader with LOBs
- Built in functions ASCII, TIMESTAMPDIFF
- DSN1COPY with 1000 OBIDs
- QMF with multi-row fetch
- Online Check Index
- z/OS 1.7 up to 7257 extents
- LOAD, UNLOAD with LOBs
- IBM System z9 Integrated Information Processor (zIIP)
- New and updated books: Library refresh Feb. 2006
- Messages, Codes separate books
- Redbooks: Data Sharing in a Nutshell, Data Integrity, MIDAW performance, Disk & DB2, Design Guidelines for High Performance & Availability, Business Value, Performance Topics, WebSphere, MLS, Disaster Recovery, others updated ...



Query Enhancements

- SQL enhancements: INTERSECT, EXCEPT, cultural sort, caseless comparisons, FETCH FIRST in fullselect, OLAP specifications: RANK, ROW_NUMBER, ...
- pureXML integration and text improvements
- Index improvements
 - Index on expression Larger index pages
 - Index compression Improved page split
- Improved Optimization statistics: Histogram
- Optimization techniques
 - Cross query block optimization
 - Generalize sparse index & in-memory data cache method
 - Dynamic Index ANDing for Star Schema
- Analysis: instrumentation & Optimization Support Center

Usability

- Application Programming



- Backup and restore objects, use tapes

- Optimization Support Center



- INDEX page split

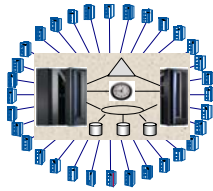
- Buffer pool management by WLM



DB2 9 for z/OS Innovation: Resiliency

DB2 9 for z/OS Innovation: Resiliency

- **Business Insight**
- **Cost Savings through Optimization**
- **Business Resiliency**
 - **Availability: Database on Demand**
 - **Scalability**
 - **Usability**



Online Schema Evolution →

Database Definition On Demand

- Online reorganization with no BUILD2 phase
- Fast replacement of one table with another
- Table space that can add partitions, for growth
- Improve ability to rebuild an index online
- Rename column and index
- Modify early code without requiring an IPL
- Alter table space and index logging
- Create & alter STOGROUP SMS constructs
- Alter column set default



Scalability

- Insert performance APPEND INDEX LOG
INDEX on expression, 8K, 16K, 32K
Randomized index key
Log Latch contention & spin relief, archiving
Not logged table space
- Partitioned table with segmented space
- Memory improvements 64 bit address space



DB2 9 for z/OS



DB2 9 for z/OS

❖ Business Insight



❖ Cost Savings through Optimization



❖ Business Resiliency



Disclaimer and Trademarks

Information contained in this material has not been submitted to any formal IBM review and is distributed on "as is" basis without any warranty either expressed or implied. Measurements data have been obtained in laboratory environment. Information in this presentation about IBM's future plans reflect current thinking and is subject to change at IBM's business discretion. You should not rely on such information to make business plans. The use of this information is a customer responsibility.

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, @server

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, WINDOWS 95

For additional information see ibm.com/legal/copytrade.phtml

MERGE

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
```

Example



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
  
```



:hv_id	:hv_amt
S.id	S.amt
1	30
5	10
10	40
5	20
1	50



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
  
```



:hv_id	:hv_amt
S.id	S.amt
1	30
5	10
10	40
5	20
1	50



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

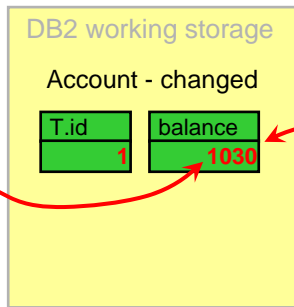
Does the source value have a matching value in the target?

```

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
    
```



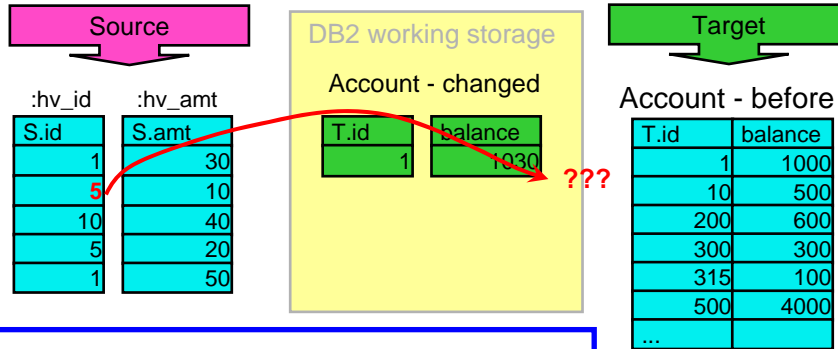
S.id	S.amt
1	30
5	10
10	40
5	20
1	50



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

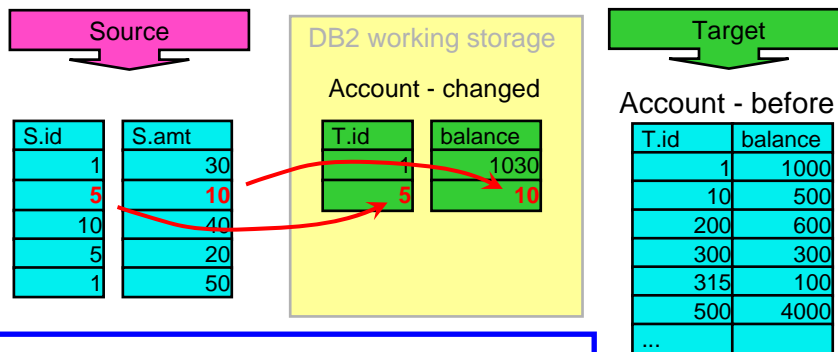
```

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
    
```



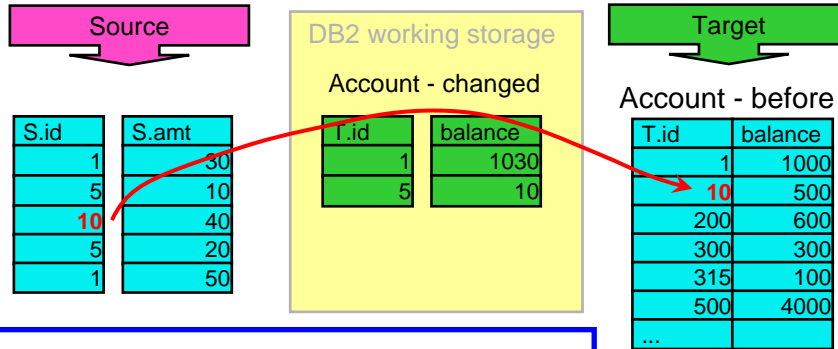
```

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
    
```



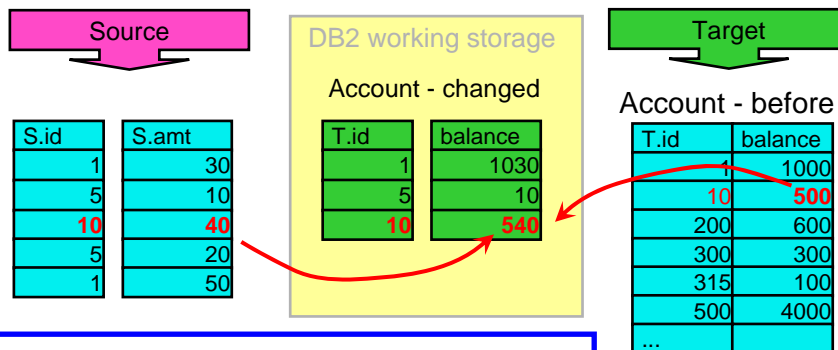
```

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
    
```

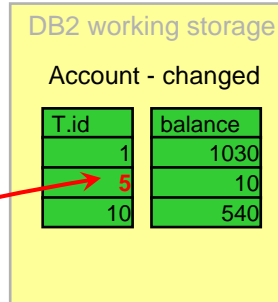


```

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
    
```



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



Account - before

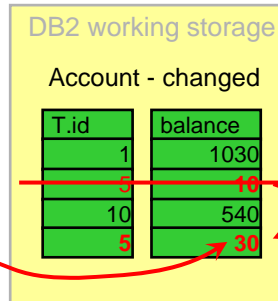
T.id	balance
1	1000
10	500
200	600
300	300
315	100
500	4000
...	

```

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
    
```



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



Account - before

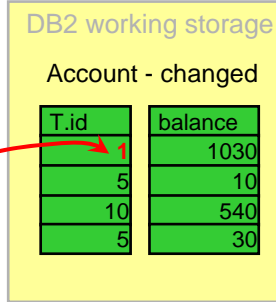
T.id	balance
1	1000
10	500
200	600
300	300
315	100
500	4000
...	

```

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
    
```



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



Account - before

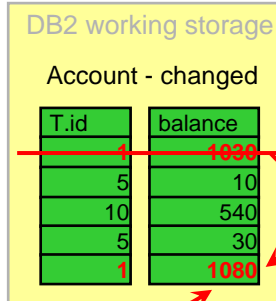
T.id	balance
1	1000
10	500
200	600
300	300
315	100
500	4000
...	

```

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
    
```



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



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

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
...	