REPLICATION PART 4

Conflict Resolution

REPLICATION CONFLICT

- **Replication Conflicts Types**
  - INSERT Conflicts – INSERT same PK
  - UPDATE Conflicts(1) – UPDATE same PK*
  - UPDATE Conflicts(2) – UPDATE PK that does not exist
  - DELETE Conflicts – DELETE PK that does not exist

REPLICATION CONFLICT SCENARIO

Conflict Scenario of INSERT, UPDATE and DELETE

time	node A	node B
T1	**Insert(PK1) & commit**	**Insert(PK1) & commit**
T2	send log(T1)	send log(T1)
T3		receive log(node A's T2) ***INSERT CONFLICT**
T4	**delete(PK1) & commit**	
T5	send log(T4)	
T6		receive log(node A's T5) ***SUCCESS**
T7	**receive log(node B's T2) *SUCCESS**	
T8	**select(PK 1) - 1 rows**	**select(PK 1) - no rows**
T9	**update(PK1) & commit**	
T10	send log(T9)	
T11		receive log(node A's T10) ***UPDATE CONFLICT(2)**
T12	**delete(PK1) & commit**	
T13	send log(T12)	
T14		receive log(node A's T13) ***DELETE CONFLICT**

REPLICATION CONFLICT SCENARIO

❖ UPDATE Conflict Scenario(1)

- When there is no detection of UPDATE conflict
- The detection is advised to find out whether the DML operation of same PK in different nodes has succeeded or not

time	node A	node B
T1	**update(PK1, 'A') & commit**	
T2	send log(T1)	
T3		**update(PK1, 'B') & commit**
T4		send log(T3)
T5	receive log(node B's T4) ***SUCCESS**	
T6		**receive log(node A's T2) *SUCCESS**
T7	**select(PK1) - 'B' *UPDATE CONFLICT(1)**	**select(PK1) - 'A' *UPDATE CONFLICT(1)**

ALTIBASE CONFLICT RESOLUTION

Solution that ALTIBASE HDB provides for replication conflict

- DBMS Level
  - User-oriented scheme
  - Timestamps-based scheme
  - Master-Slave scheme
- Utility Level
  - Audit

Different processing in different replication conflict type

Conflict Type	Operation	Situation	Processing
INSERT	**INSERT**	INSERT same PK	**Follow a conflict policy configured in receiver***
UPDATE	**UPDATE**	UPDATE same PK	
		UPDATE PK that does not exist	Produce replication conflict report only
DELETE	**DELETE**	DELETE PK that does not exist	

USER-ORIENTED SCHEME

User-oriented scheme

- Replication conflict solution policy that is configured by default
- Ignores related operations and records when replication conflict occurs
    - It is only recorded in replication trace log file to let user to check and handle
        - $ALTIBASE_HOME/trc/altibase_rp.log

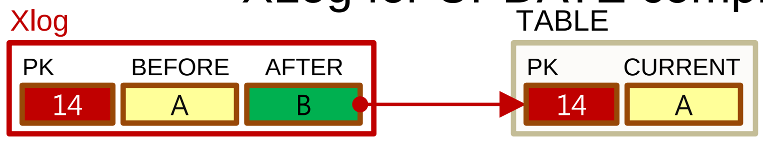
Detection and processing for different replication conflict type

Conflict Type	Situation	Processing
INSERT	INSERT same PK	Ignores all the related operations and produce the detection report only
UPDATE	UPDATE same PK	
	UPDATE PK that does not exist	
DELTE	DELETE PK that doest not exist	

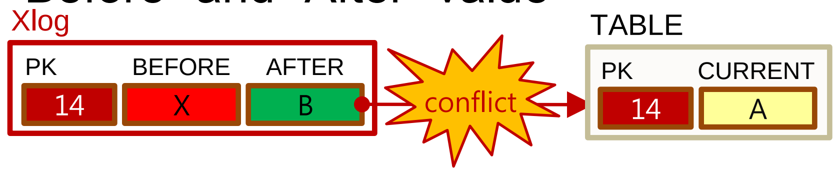
USER-ORIENTED SCHEME

Value-based Method

- Detects DML conflict(UPDATE same PK)
- Determined as a conflict when the current value is different to the previous value
  - XLog for UPDATE comprises “Before” and “After” value



[Fig1. When it is normal]



[Fig2. When it is conflict]

Handles by value-based method that detects DML conflict

- Processing DML conflict can be configured by property
  - When REPLICATION_UPDATE_REPLACE = 1

	Situation	Processing
DML Conflict	UPDATE same PK	When REPLICATION_UPDATE_REPLACE =0 , it writes to replication conflict report When REPLICATION_UPDATE_REPLACE =1, it ignores all the conflicts and applies contents

USER-ORIENTED SCHEME

- **Caution**
  - For LOB column, it does not detect the conflict of modifying same PK
    - LOB data type feature
      - LOB cannot be detected as previous value for LOB data type does not exist in redo log

MASTER-SLAVE SCHEME

Master-Slave Scheme

- Replication conflict solution policy that assigns both Master and Slave when creating replication object
- Always set Master as standard

Processing Methods

	Situation	Processing	
		Master	Slave
INSERT Conflict	INSERT same PK	Replication Conflict Report	INSERT is applied after DELETE is executed on the current record
UPDATE Conflict	UPDATE same PK	Replication Conflict Report	Applies UPDATE

Caution

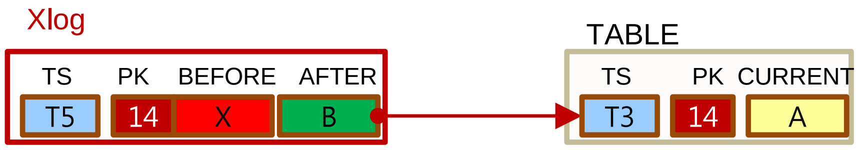
- Replication is possible when there is a Slave object that corresponds to single Master replication object

  Master-Master (X), Slave-Slave (X), Master or Slave-NONE (X)

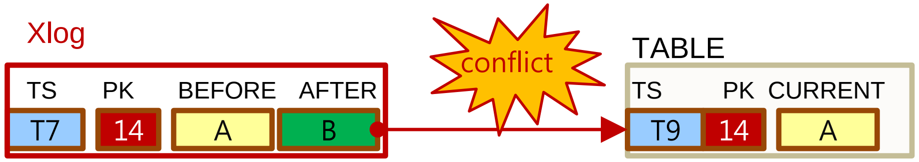
TIMESTAMPS-BASED SCHEME

Timestamps-based scheme

- Replication conflict solution policy that can be configured as property
  - Set REPLICATION_TIMESTAMP_RESOLUTION =1(Default 0)
- Identifies rank by using TIMESTAMP
  - Conflict can be resolved as the number is ordered by most recent time



[Fig1. When it is normal]



[Fig2. When it is conflict]

Processing methods

	Situation	Processing
INSERT Conflict	INSERT same PK	DELETE is executed on current records and INSERT is executed when TIMSTAMP is greater than or equal otherwise, it writes to replication conflict report
UPDATE Conflict	UPDATE same PK	UPDATE is applied when TIMESTAMP is greater than or equal otherwise it writes to replication conflict report

TIMESTAMPS-BASED SCHEME

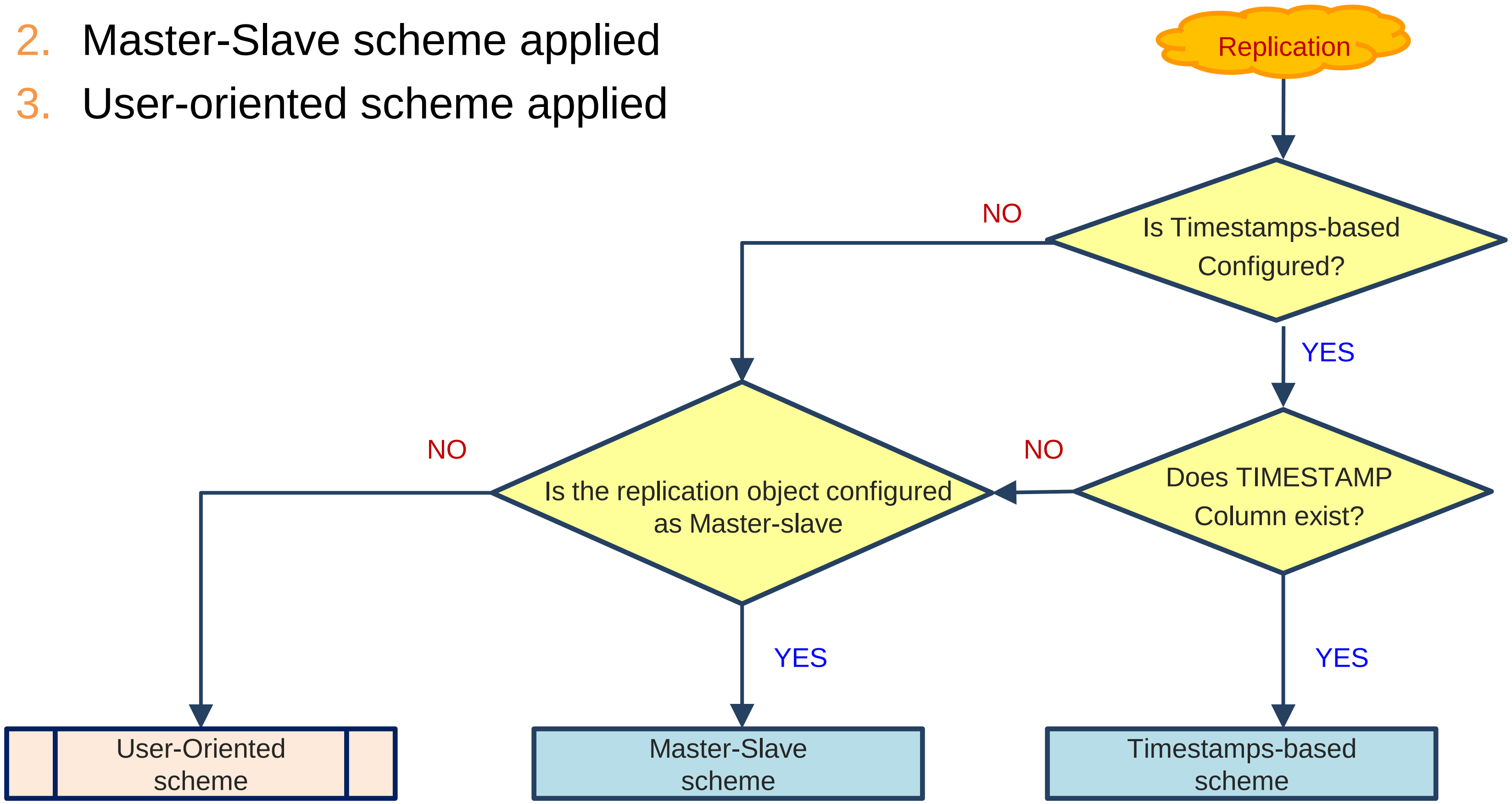
- **Caution**
  - Time setting for each replication server has to be the same
  - TIMESTAMP column is compulsory
    - Table that has no TIMESTAMP column is not applied even though the configured property is set to '1'
    - TIMESTAMP column has to be added by user manually

- **Note**
  - Additional 8 bytes of data space will be occupied
  - Communication cost of replication will be increased as the TIMESTAMP column is sent additionally

CONFLICT RESOLUTION FLOW

❖ **The flow of Processing when there is a confusion in solving conflicts**

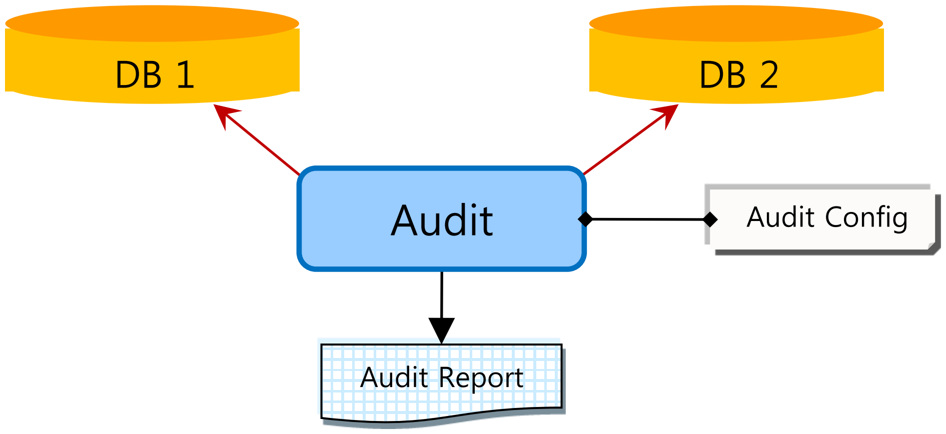
1. Timestamps-based scheme is firstly applied when the relevant property has to be set to “1” and when TIMESTAMP column exists
2. Master-Slave scheme applied
3. User-oriented scheme applied


Replication
NO
Is Timestamps-based Configured?
YES
NO
Is the replication object configured as Master-slave
NO
Does TIMESTAMP Column exist?
YES
YES
User-Oriented scheme
Master-Slave scheme
Timestamps-based scheme

AUDIT

❖AUDIT

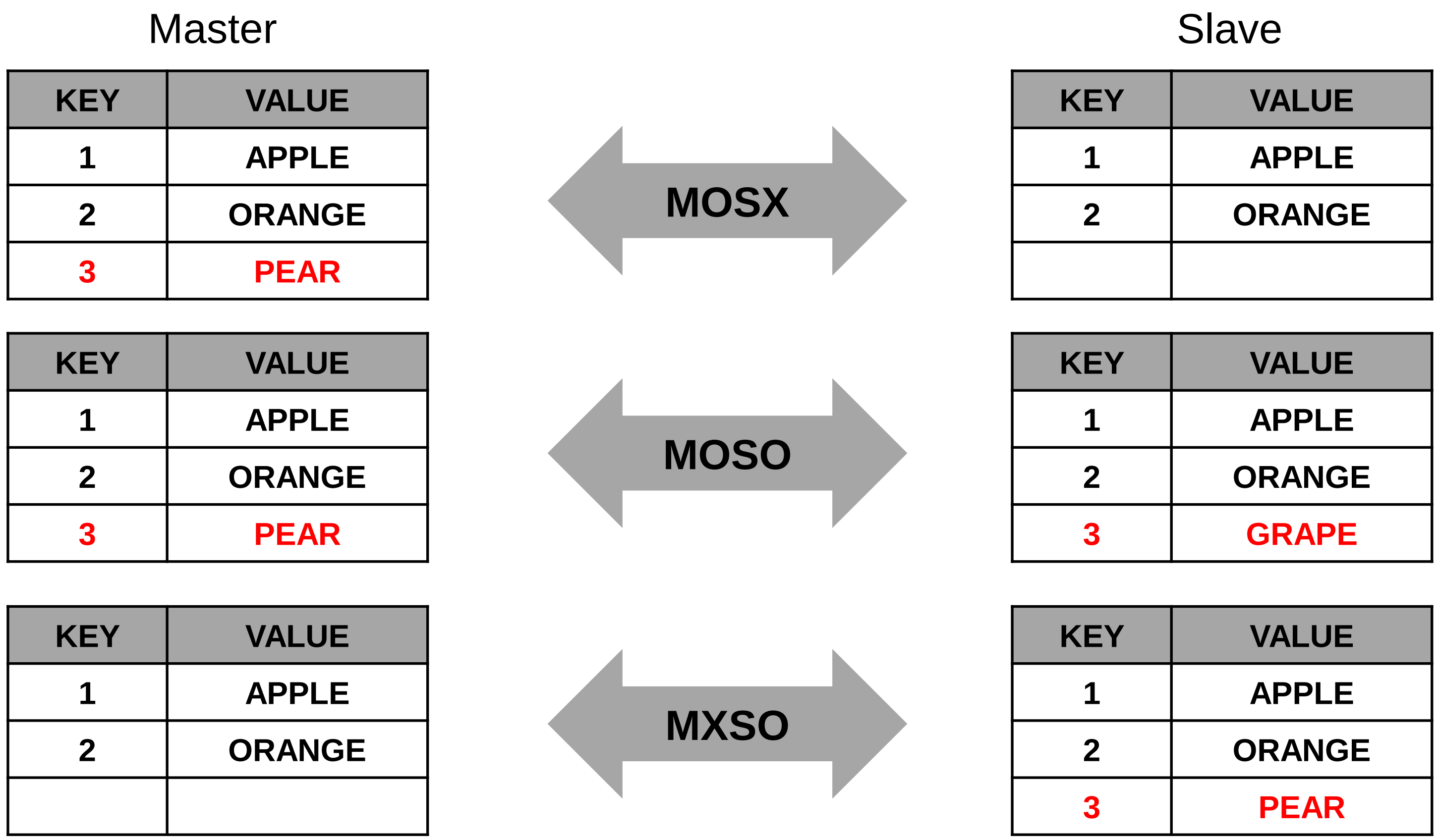
- Utility that compares and synchronizes two databases on a table-by-table basis
- Resolving data inconsistencies caused by replication conflicts
- Treats the (MASTER) DB as the reference DB and synchronizes the (SLAVE) DB with it
- It may not operate properly in the event that a target is DB is on modification process



AUDIT

Three types of Audit

- Three different cases where data inconsistencies are occurred between Master DB and Slave DB



Master

KEY	VALUE
1	APPLE
2	ORANGE
3	PEAR

MOSX

Slave

KEY	VALUE
1	APPLE
2	ORANGE

Master

KEY	VALUE
1	APPLE
2	ORANGE
3	PEAR

MOSO

Slave

KEY	VALUE
1	APPLE
2	ORANGE
3	GRAPE

Master

KEY	VALUE
1	APPLE
2	ORANGE

MXSO

Slave

KEY	VALUE
1	APPLE
2	ORANGE
3	PEAR

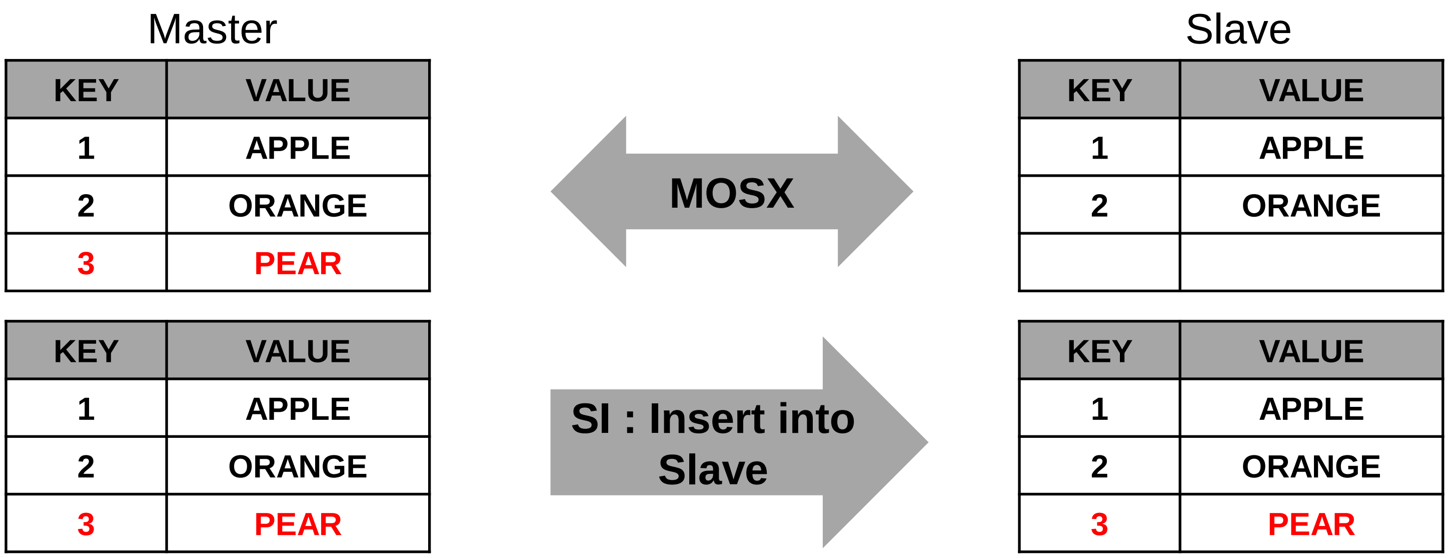
AUDIT

Data Synchronization Policy

- The following synchronization policies are used for three different data inconsistencies occurred between Master DB and Slave DB after a audit

SI (Slave Database Insert)

- This policy resolves MOSX inconsistencies by inserting records from the Master DB into the Slave DB



Master

KEY	VALUE
1	APPLE
2	ORANGE
3	PEAR

Slave

KEY	VALUE
1	APPLE
2	ORANGE

Master

KEY	VALUE
1	APPLE
2	ORANGE
3	PEAR

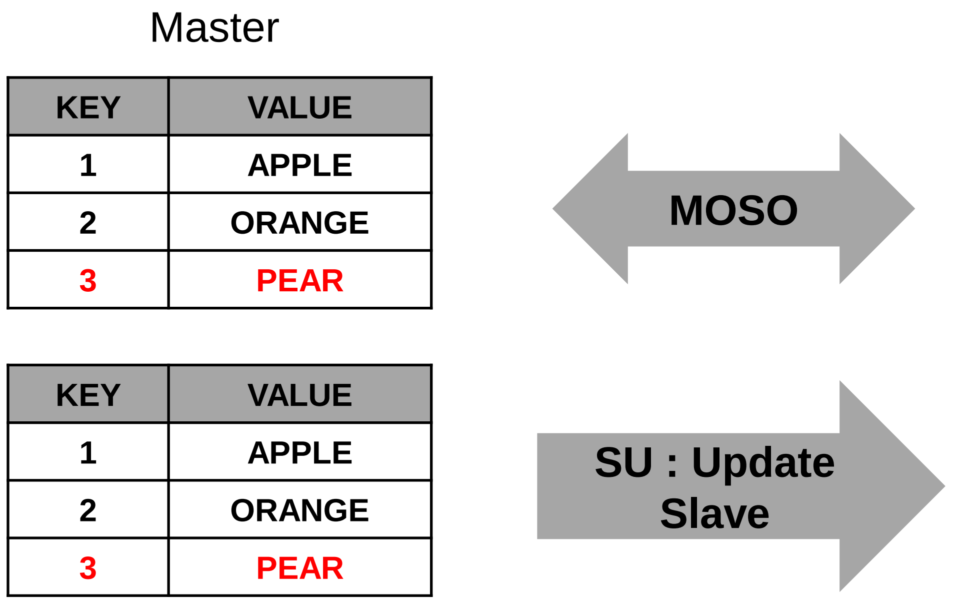
Slave

KEY	VALUE
1	APPLE
2	ORANGE
3	PEAR

AUDIT

SU (Slave Database Update)

- This policy resolves MOSO inconsistencies by updating the Slave DB with the contents of the Master DB



Master

KEY	VALUE
1	APPLE
2	ORANGE
3	PEAR

Slave

KEY	VALUE
1	APPLE
2	ORANGE
3	GRAPE

Master

KEY	VALUE
1	APPLE
2	ORANGE
3	PEAR

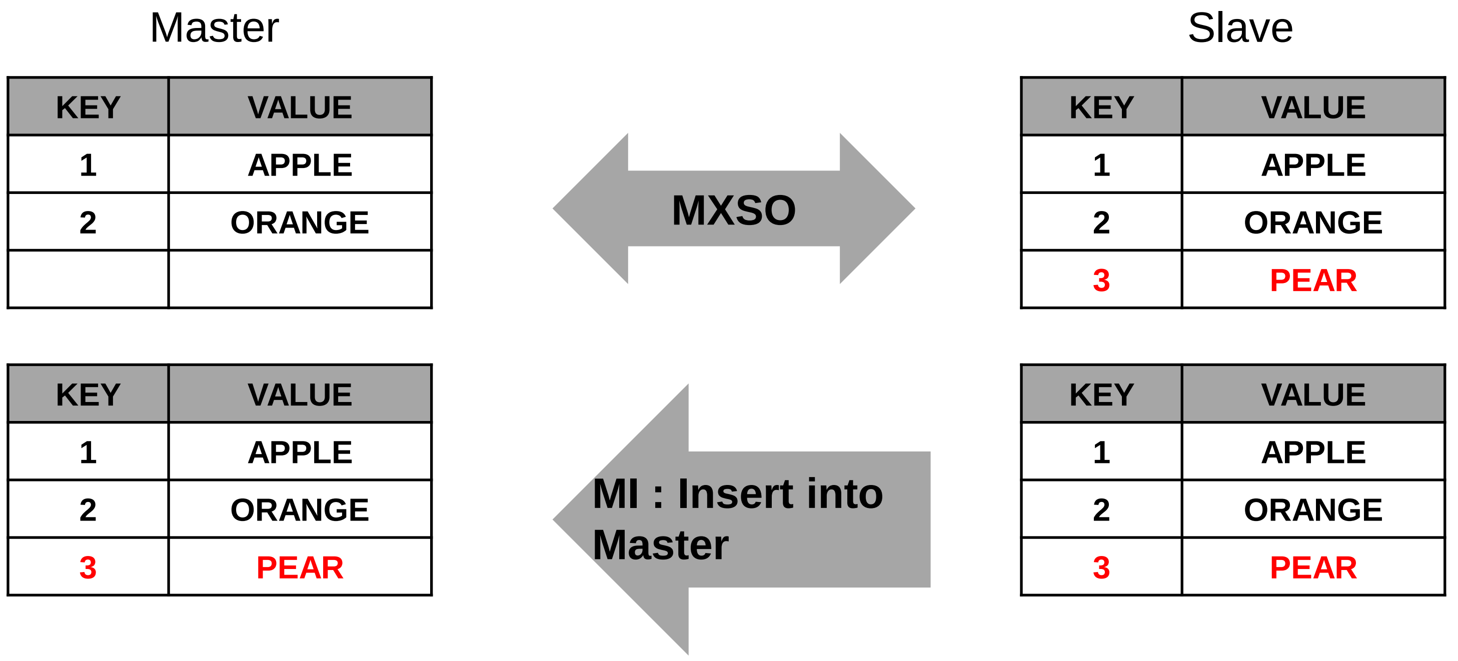
Slave

KEY	VALUE
1	APPLE
2	ORANGE
3	PEAR

AUDIT

MI (Master Database Insert)

- This policy resolves MXSO inconsistencies by inserting records from the Slave DB into the Master DB



Master (before)

KEY	VALUE
1	APPLE
2	ORANGE

Slave (before)

KEY	VALUE
1	APPLE
2	ORANGE
3	PEAR

Master (after)

KEY	VALUE
1	APPLE
2	ORANGE
3	PEAR

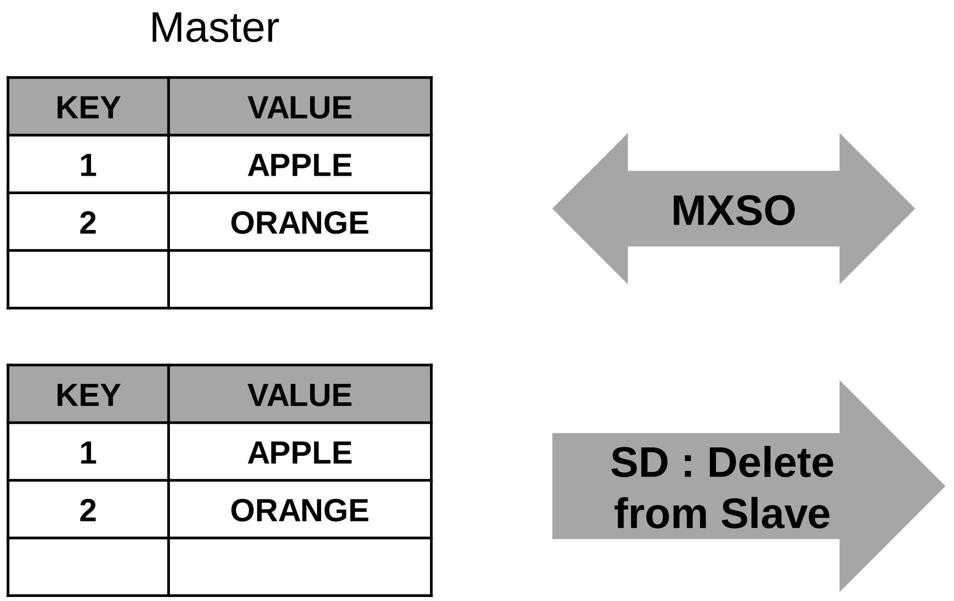
Slave (after)

KEY	VALUE
1	APPLE
2	ORANGE
3	PEAR

AUDIT

❖ SD (Slave Database Delete)

- ➢ This policy resolves MXSO inconsistencies by deleting records from the Slave DB



Master

KEY	VALUE
1	APPLE
2	ORANGE

Slave

KEY	VALUE
1	APPLE
2	ORANGE
3	PEAR

Master (after SD)

KEY	VALUE
1	APPLE
2	ORANGE

Slave (after SD)

KEY	VALUE
1	APPLE
2	ORANGE

AUDIT

AUDIT

- To execute the Audit, it requires the configuration file that contains audit polices which will be used for two different databases

- **Audit configuration file:** Contains Audit option such as connection, Audit function, consistency policies and it is recommended to use sample.cfg by copying the existing audit.cfg file($ALTIBASE_HOME/audit/sample.cfg)

- **Comparison(DIFF) Function :** Identifies inconsistent records between Master DB and Slave DB and creates the result file

- **Synchronization(SYNC) Function :** Resolves the inconsistencies between two databases and creates the result file

AUDIT

❖ Comparison(DIFF) Function

- Identifies inconsistent data between Master DB and Slave DB during the course of replication and creates the result file

```
# Audit environment file configuration for Comparison

# Configuration for Master DB Connection
DB_MASTER="altibase://sys:manager@DSN=host1;PORT_NO=10111;NLS_USE=MS949"

# Configuration for Slave DB Connection
DB_SLAVE="altibase://sys:manager@DSN=host2;PORT_NO=20111;NLS_USE=MS949"

# Audit Operation types (DIFF is configured as it's comparison)
OPERATION = DIFF

# Assign number of threads(Unlimited)
MAX_THREAD = -1
```

AUDIT

```
#Configuration of audit  policy for inconsistency ( Not supported in DIFF)
DELETE_IN_SLAVE = ON
INSERT_TO_SLAVE = ON
INSERT_TO_MASTER = OFF
UPDATE_TO_SLAVE = ON
AUTODETECT_UNIQ_INX = ON

# Where execution result file will be created
LOG_DIR = “./”
LOG_FILE = “sample.log”

# Configuration of Audit table target
# Comparing master table [EMP] with slave table EMPLOYEE

[EMP]
TABLE = EMPLOYEE
SCHEMA = SYS

# Comparing master table [DEPT] with slave table DEPARTMENT
[DEPT]
TABLE = DEPARTMENT
SCHEMA = SYS
#Audit environment file configuration finished
```

AUDIT

- Executing Audit command

```
$ audit –f sample.cfg
```

- When Audit command is executed successfully, the file named 'MasterTable-username.SlaveTable.log' is created about each tables along with execution log file (sample.log)

```
# Inside 'sample.log' (the execution log file)
INFO[ MNG ] Tread # 0 init is OK!
INFO[ MNG ] Tread # 0 start is OK!

[TAB_1->TAB_2]
Fetch Rec In Master: 3
Fetch Rec In Slave : 2
MOSX = DF, Count : 1
MXSO = DF, Count : 0
MOSO = DF, Count : 1
SCAN TPS: 20547.95
Time: 0.00 sec
```

It displays the contents of executed environmental file and the summary of Comparison(DIFF)operation about tables in each group

AUDIT

- Inside 'MasterTable-Username.SlaveTable.log'

MOSX

```
$ cat emp-sys.employee.log
MOSX[19,15]->ENO(19):PK->{19}
MOSX[20,15]->ENO(20):PK->{20}
```

MOSO

```
$ cat emp-sys.employee.log
MOSO[10,10]->ENAME('JJLEE          ','YHBAE          '):PK->{10}
MOSO[11,11]->ENAME('MJYOO          ','MSKIM          '):PK->{11}
```

MXSO

```
$ cat emp-sys.employee.log
MXSO[8,8]->ENO(8):PK->{8}
MXSO[8,9]->ENO(9):PK->{9}
```

AUDIT

Synchronization (SYNC) Function

- Resolves the inconsistency by identifying the inconsistent data between Master DB and Slave DB according to the Audit environment file match policy

```
# Audit environment file configuration for Synchronization

# Configuration for Master DB Connection
DB_MASTER=“altibase://sys:manager@DSN=host1;PORT_NO=10111;NLS_USE=MS949”

# Configuration for Slave DB Connection
DB_SLAVE=“altibase://sys:manager@DSN=host2;PORT_NO=20111;NLS_USE=MS949”

# Audit Operation Types(SYNC is configured as it’s synchronization)
OPERATION = SYNC

# Assign the number of threads(Unlimited)
MAX_THREAD = -1
```

AUDIT

```
# Configuration of inconsistency audit  policy
DELETE_IN_SLAVE = ON
INSERT_TO_SLAVE = ON
INSERT_TO_MASTER = OFF
UPDATE_TO_SLAVE = ON
AUTODETECT_UNIQ_INX = ON

# Assign the location where execution result file will be created
LOG_DIR = “./”
LOG_FILE = “sample.log”

# Configuration of target Audit table
# Comparing master table [EMP] with slave table EMPLOYEE
[EMP]
TABLE = EMPLOYEE
SCHEMA = SYS

# Comparing master table [DEPT] with slave table DEPARTMENT
[DEPT]
TABLE = DEPARTMENT
SCHEMA = SYS
#Audit environment file configuration finished
```

AUDIT

- Executing Audit command

```
$ audit –f sample.cfg
```

- The contents of execution result file are as follows. If a failure occurs for any record, the cause of the error and the record contents are written to the log

```
INFO[ MNG ] Tread # 0 init is OK!
INFO[ MNG ] Tread # 0 start is OK!

[TAB_1->TAB_2]
Fetch Rec In Master: 3
Fetch Rec In Slave : 2
MOSX = -, SI
MXSO = -, -
MOSO = -, SU
MXSX = -, -
```

AUDIT

```
------------------------------------------------------------
Operation   Type      MASTER    SLAVE
------------------------------------------------------------
INSERT        Try           0              1
              Fail          0              0

UPDATE        Try           X              1
              Fail           X              0

DELETE        Try           X              0
              Fail           X              0
------------------------------------------------------------
UPDATE        Try           0              2
              Fail          0               0

OOP   TPS: 13698.63
SCAN TPS: 20547.95
          Time: 0.00 sec
```

AUDIT

Synchronization(SYNC) Operation Procedure

1. Stop all the related applications
   Data might be inconsistent in the event of modification transaction during AUDIT SYNC
2. Check replication gap

Check whether everything is reflected to remote server (rep_gap=0)

```
iSQL> SELECT rep_gap FROM v$repgap;
```

1. Stop replication

```
iSQL> ALTER REPLICATION replication_name  STOP;
```

2. Execute AUDIT
3. REPLICATION QUICK START

Prevent the transaction log is sent to replication

```
iSQL> ALTER REPLICATION replication_name  QUICKSTART;
```

AUDIT

Cautions

- Error recorded in Logfile when there is no PK
  - FATAL[TASK] Process failure! [SCANER]: [ERR-910D8 : No Primary Key Column exist (T1:T1)]
- Error when there is a conflict between SD and MI policy
  - Invalid Property Value SD and MI Incompatible was defined.
- Same value with different data type recognized differently
  - char(10) vs. varchar(10)

Q & A

Thank you!

Altibase Education Center

Tel : 02-2082-1451
Fax : 02-2082-1459
E-mail : education@altibase.com
Homepage : http://edu.altibase.com