# DB2 HADR Galileo Update January 2012

**Dale McInnis**
*IBM Canada Ltd.*

Platform: Linux, UNIX and Windows

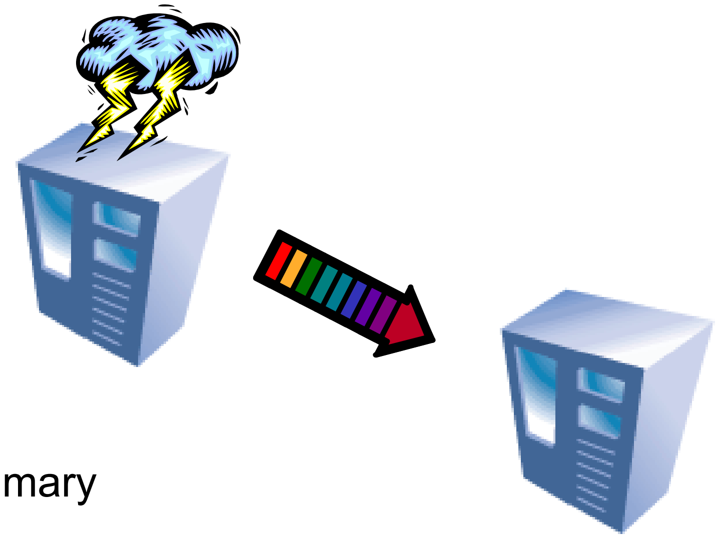
## Agenda

- HADR Review
- HADR Super Async mode
- HADR Log Spooling
- HADR Multiple Standby
- HADR Time Delay
- HADR Configuration
- HADR Monitoring
- HADR Application implications
- Comparison to other DB2 HA offerings

## Main Goals of the Design

- Ultra-fast failover
- Easy administration
- Handling of site failures
- Negligible impact on performance
- Configurable degree of consistency
- Protection against errant transactions
- Software upgrades without interruption
- Very easy integration with HA-software
- Eventually, no need for HA-software at all
- Transparent failover and failback for applications (combined with “client re-route”)

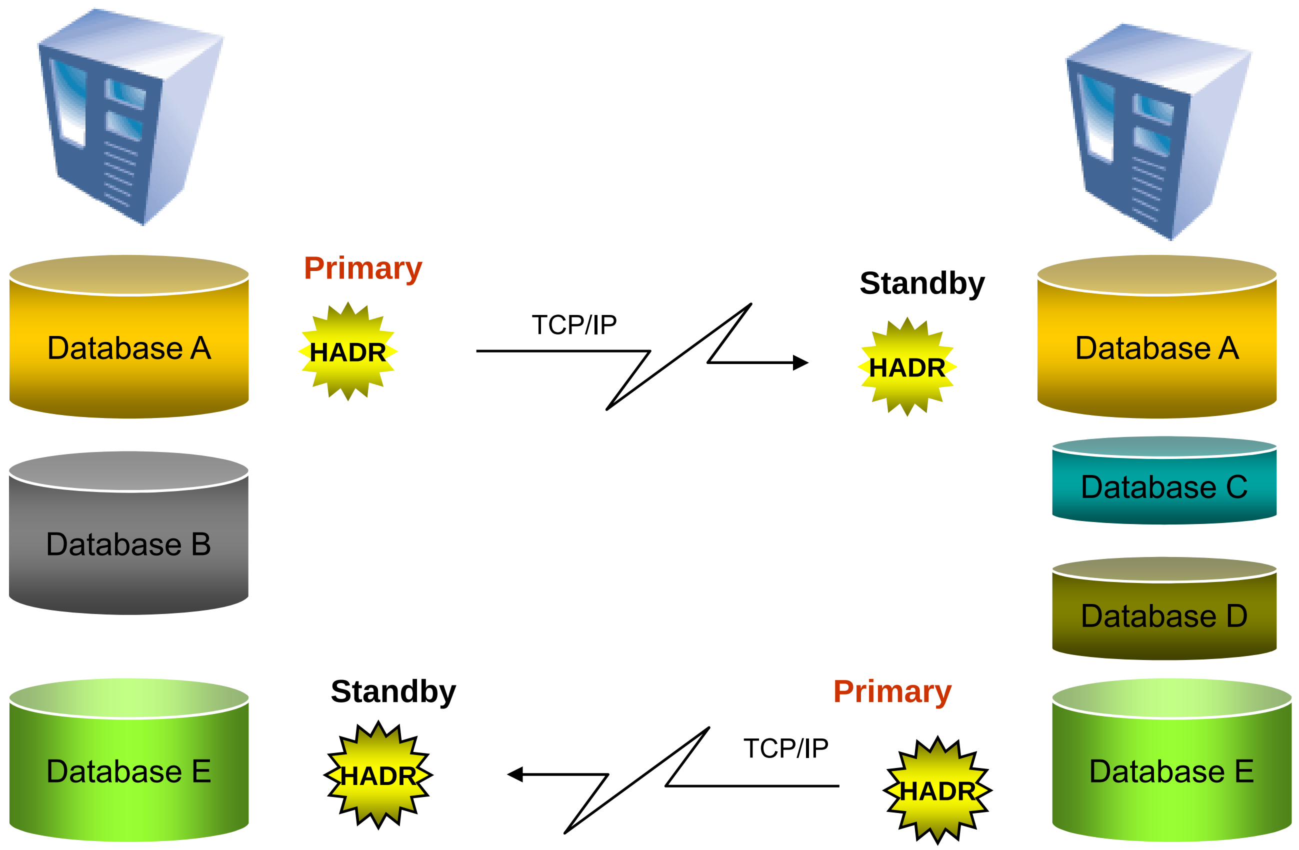
# Basic Principles of HADR



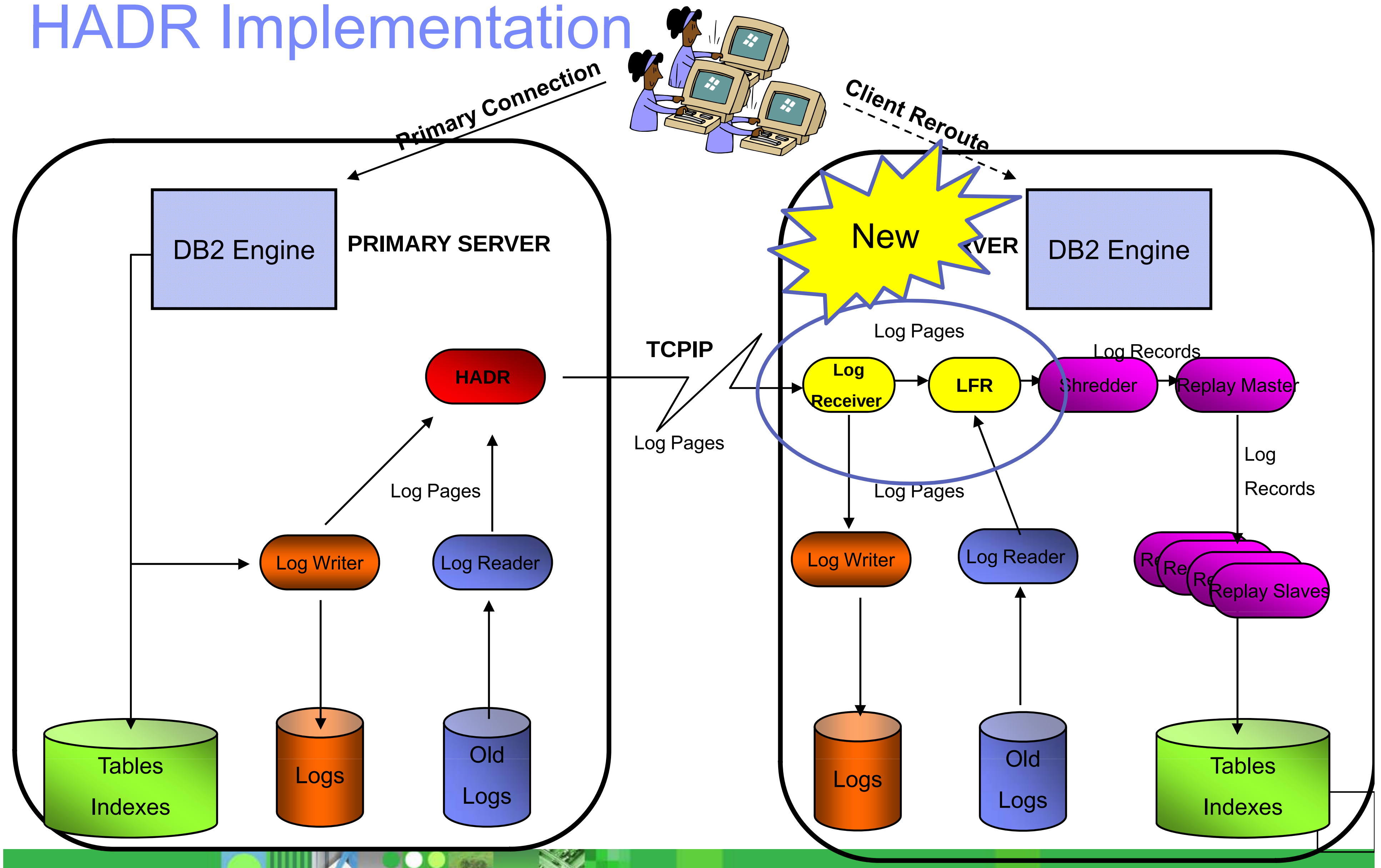
- Two active machines
  - Primary
    - Processes transactions
    - Ships log entries to the other machine
  - Standby
    - Cloned from the primary
    - Receives and stores log entries from the primary
    - Re-applies the transactions
- If the primary fails, the standby can take over the transactional workload
  - The standby becomes the new primary
- If the failed machine becomes available again, it can be resynchronized
  - The old primary becomes the new standby

## Scope of Action

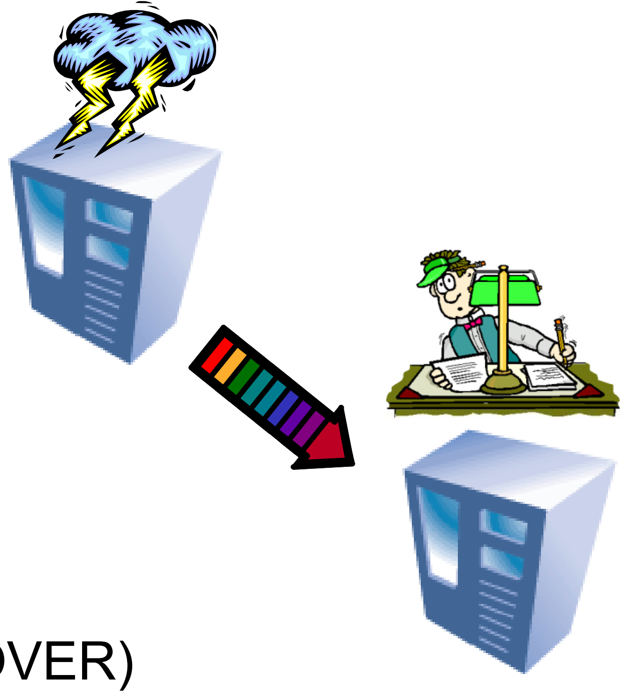
HADR replication takes place at the database level.



## HADR Implementation



## Failing Over : Simple “TAKEOVER” Command



- Normal TAKEOVER
  - Primary and standby switch roles as follows:
    1. Standby tells primary that it is taking over.
    2. Primary forces off all client connections and refuses new connections.
    3. Primary rolls back any open transactions and ships remaining log, up to the end of log, to standby.
    4. Standby replays received log, up to end of the log.
    5. Primary becomes new standby.
    6. Standby becomes new primary

- Emergency TAKEOVER (aka ‘Forced’ TAKEOVER)
  - The standby sends a notice asking the primary to shut itself down.
  - The standby does NOT wait for any acknowledgement from the primary to confirm that it has received the takeover notification or that it has shut down
  - The standby stops receiving logs from the primary, finishes replaying the logs it has already received, and then becomes a primary.

```
TAKEOVER HDR ON DATABASE <dbname>
        <USER <username> [USING <password>]]      [BY FORCE]
```

## Primary Reintegration

- After primary failure and takeover, allow old primary to reintegrate as a standby with the new primary (saves user from having to reinitialize standby from scratch)
- Differentiating feature for DB2 HADR – competitors do not support this
- Reintegration possible if old primary can be made consistent with new primary
- Some conditions to satisfy, e.g. old primary crashed in peer state and had no disk updates that were not logged on old standby; some other details.
- Successful reintegration is most likely in SYNC mode, least likely in ASYNC mode
- Synchronization with tail of the log file

## HADR Setup Fits on One Slide

**Primary Setup**

```
db2 backup db hadr_db to backup_dir

db2 update db cfg for hadr_db using
    HADR_LOCAL_HOST    host_a
    HADR_REMOTE_HOST   host_b
    HADR_LOCAL_SVC     svc_a
    HADR_REMOTE_SVC    svc_b
    HADR_REMOTE_INST   inst_b
    HADR_TIMEOUT       120
    HADR_SYNCMODE      ASYNC

db2 start hadr on database hadr_db as
  primary
```

**Standby Setup**

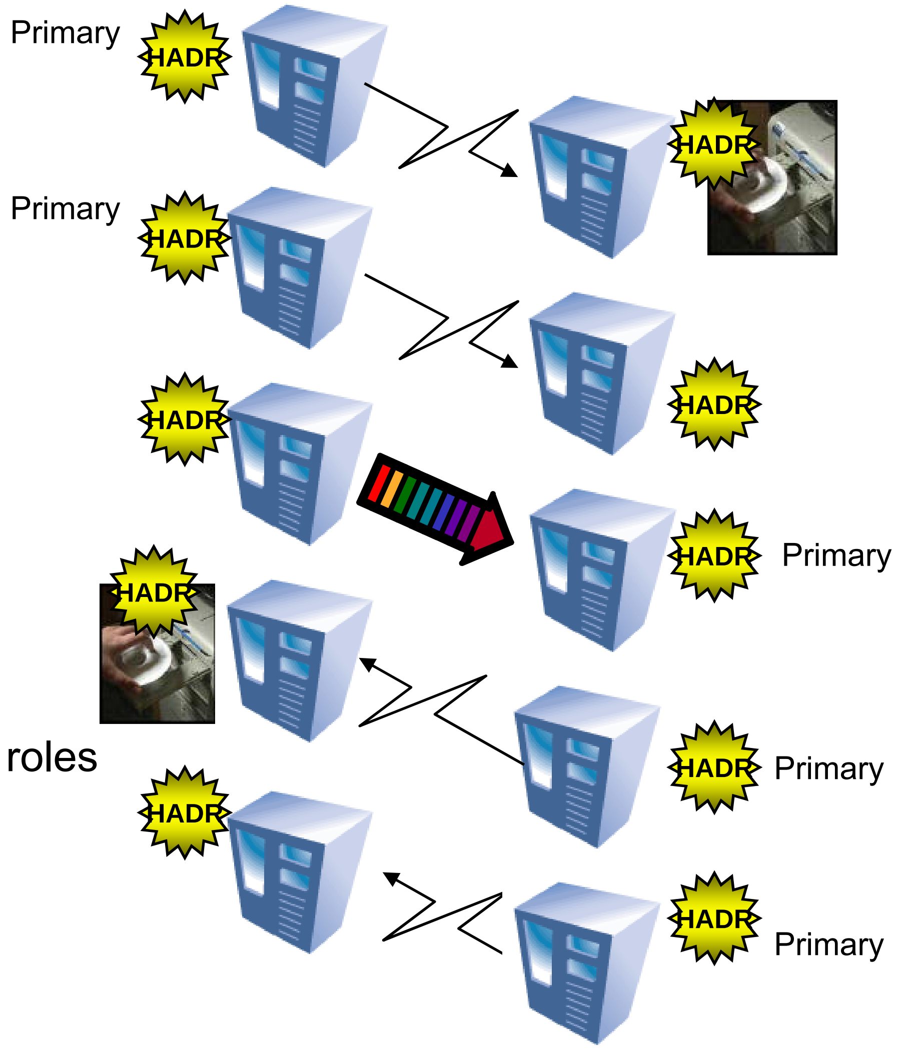
```
db2 restore db hadr_db from
  backup_dir

db2 update db cfg for hadr_db using
    HADR_LOCAL_HOST    host_b
    HADR_REMOTE_HOST   host_a
    HADR_LOCAL_SVC     svc_b
    HADR_REMOTE_SVC    svc_a
    HADR_REMOTE_INST   inst_a
    HADR_TIMEOUT       120
    HADR_SYNCMODE      ASYNC

db2 start hadr on database hadr_db as
  secondary
```

## HADR configuration, HADR_TIMEOUT

- If one database does not receive any message from the other for HADR_TIMEOUT period, connection is closed.
  - Connection will also close upon any TCP errors.
- Performance implications of HADR_TIMEOUT
  - Too long is bad: In peer state, if standby is not responding, transactions on primary can hang for HADR_TIMEOUT period.
  - Too short is bad too: You get false alarm on the connection. Frequent disconnection and reconnection wastes resource. HA protection also suffer as disconnection brings primary out of peer state.
  - Recommend at least 10 seconds.

## Software upgrades on the fly

1. HADR in peer state
2. Deactivate HADR on the Standby
3. Upgrade the standby
4. Start the standby again
   - Let it catch-up with primary
5. Issue a normal TAKEOVER
   - The primary and standby change roles
6. Suspend the new standby
7. Upgrade the new standby
8. Reactivate the new standby
   - Let it catch-up with primary
9. Optionally, TAKEOVER again
   - The primary and standby play their original roles



## Diagnostics

- db2diag.log is the most important diagnostic tool.
  - Look for hdrSetHdrState() trace points. All HADR state transition goes through this function.
  - You can also search for all messages produced by the HADR EDU.
  - If there are multiple HADR databases in the instances, be sure to distinguish messages from different databases.
  - What about the “This operation is not allowed on a standby database” message?
    - It indicates that a client (possibly an admin tool) is trying to connect to the standby database.

## What's replicated, what’s not?

- Logged operations are replicated
  - Example: DDL, DML, table space and buffer pool creation/deletion.
- Not logged operations are not replicated.
  - Example: database configuration parameter. not logged initially table, UDF libraries.
- Index pages are not replicated unless LOGINDEXBUILD is enabled

## Are LOBs replicated?

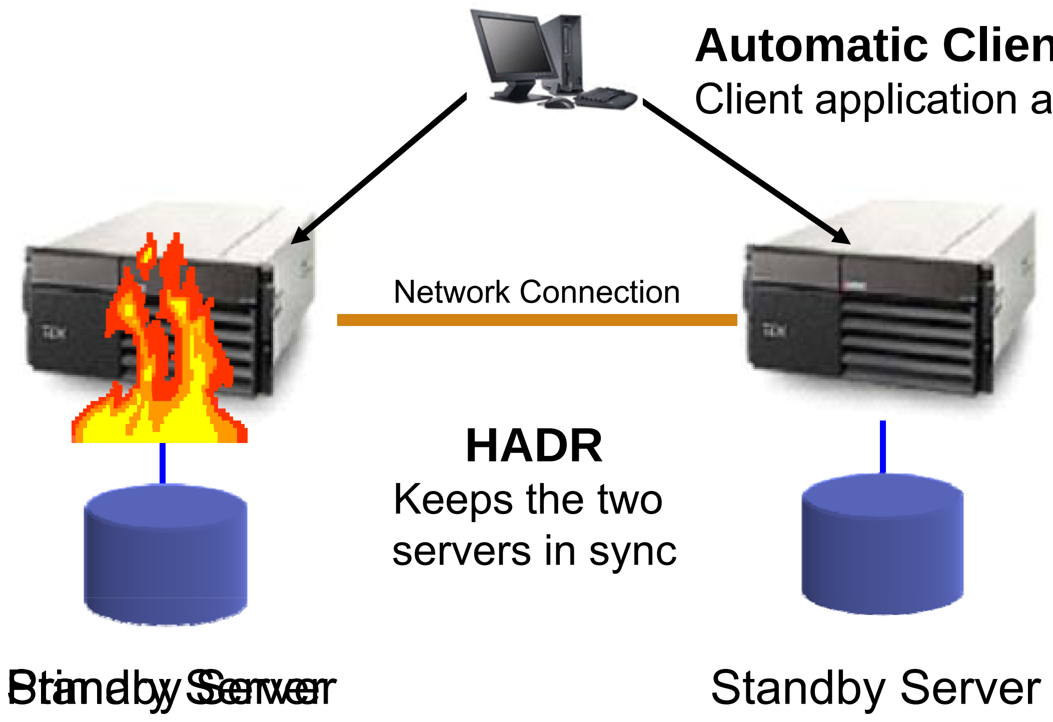
- User can define LOBs as logged or not logged
- Logged LOBs are replicated
- Not logged LOBs: data is not replicated, but LOB space is allocated on standby. The LOBs on the standby will have the right size, but the content will be binary zero

## HADR Restrictions

- Same OS on primary and standby.
- Same endianess
- Same db2 major version.
- Same minor version (fix packs) recommended.
  - Different minor version is allowed because it is needed for rolling upgrade. But it is not recommended for normal operation.
  - When minor versions are different, the primary can not be newer because a newer primary might generate log records the standby can not understand.

## DB2 Delivers fast failover at a fraction of the cost

- Redundant copy of the database to protect against site or storage failure
- Support for Rolling Upgrades
- **Failover in under 15 seconds**
  - **Real SAP workload with 600 SAP users – database available in 11 sec.**
- 100% performance after primary failure



**Automatic Client Reroute**
Client application automatically resumes on Standby

**HADR**
Keeps the two servers in sync

**TSA for server monitoring**
Monitors the primary and initiates the takeover.
-could also use heartbeat, TSA, MSCS, HACMP, etc

## Agenda

- HADR Review
- HADR Super Async mode
- HADR Log Spooling
- HADR Multiple Standby
- HADR Time Delay
- HADR Configuration
- HADR Monitoring
- HADR Application Implications
- Comparison to other DB2 HA offerings

## HADR superasync mode : also in V 9.5 + V 9.7

- All sync modes except superasync can result in back pressure being applied to the primary due to a slow network or lack of resources on the standby

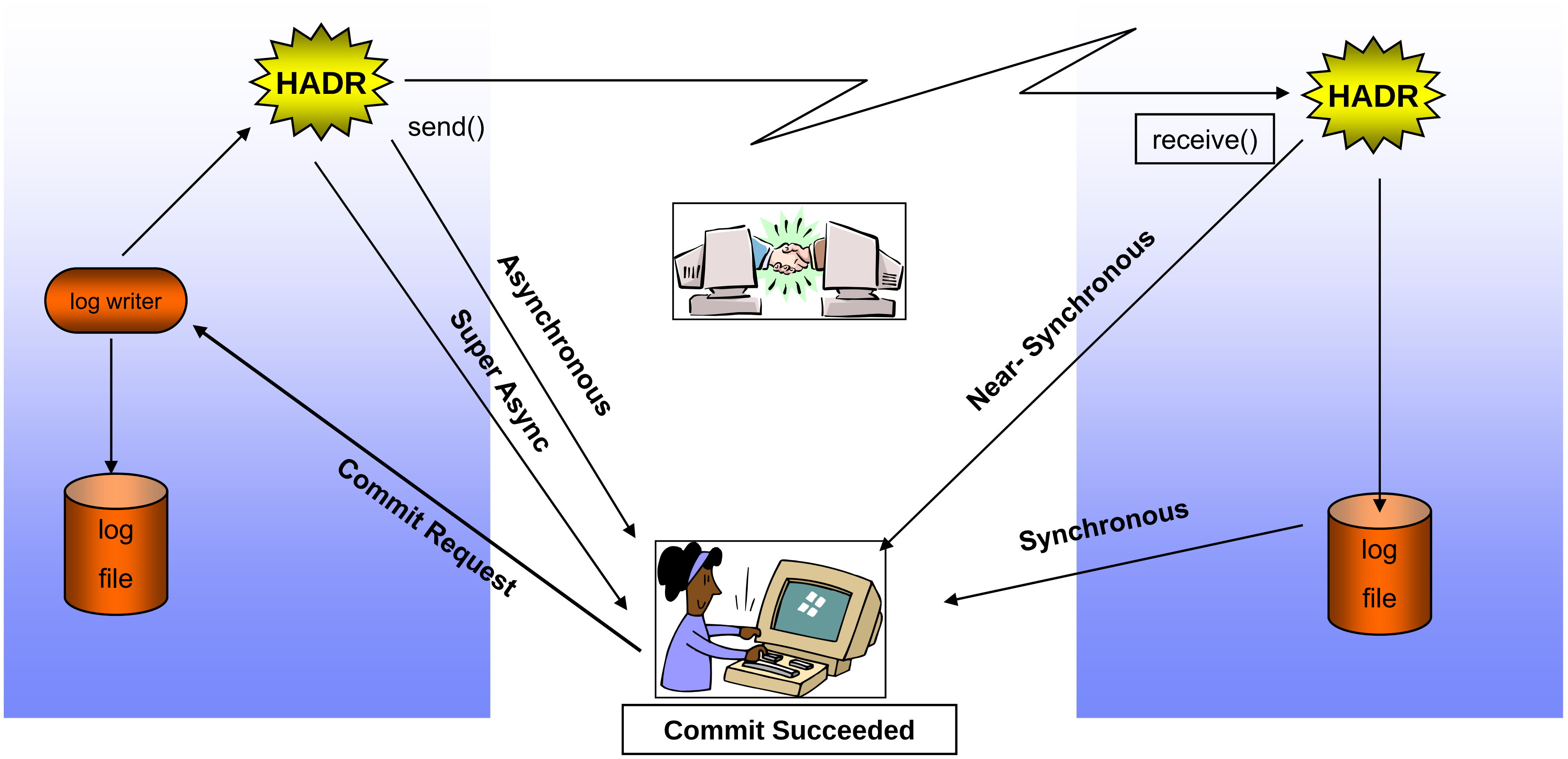
- superasync mode will prevent any back pressure by never entering peer state.

  - The HADR state will move from local catch up to remote catch up. It then stays in remote catch up. HADR will always ship logs from primary on disk or archived logs. It will not enter peer state, where logs are shipped from the primary database log buffer and the primary database log writer can be slowed down.

- Takeover behavior the same as other synchronization modes
  - Normal takeover is supported even though the standby state is Remote Catch Up pending state.

## Synchronization modes

Sync, Near Sync, Async, Super Async



## Sync mode review

- Transactions on primary will commit only after logs have been written to disk on both primary and standby.
- Maximum protection, with performance cost.
- After writing logs to local disk, primary sends a copy to standby. Primary will then wait for ”log written” ack message from standby.
- Serial write and send on primary.
- In peer state, any transaction committed on primary is guaranteed to have been committed on standby too.
- In peer state, if a failover occurs, you will not lose any committed transactions.

## Near Sync mode review

- Transactions on primary will commit only after logs have been written to disk on primary and received into memory on standby.
- Protection nearly as good as SYNC mode. Performance is better than SYNC mode.
- In peer state, you will lose data only if both primary and standby fail at the same time.
- When writing logs to local disk, primary also sends a copy to standby. Primary will then wait for ”log received” ack message from standby.
- Parallel write and send on primary.

## Async mode review

- Transactions on primary will commit only after logs have been written to local disk and sent to standby. Primary does not wait for or confirm reception.
- Performance better than SYNC and NEARSYNC. Less protection.
- When writing logs to local disk, primary also sends a copy to standby. Primary will go on as soon as send() call to TCP returns.
- In peer state, any transaction committed on primary is guaranteed to have been ”sent” to standby.
- In peer state, if a failover occurs, logs sent but not yet received can be lost.

## superasync Mode Details

- The log writes are considered successful as soon as the log records have been written to the log files on the primary database.
- Best performance. Least protection.
- Because the primary database does not wait for log send to the standby database, transactions are considered committed irrespective of the state of the replication of the transaction.
- Since the transaction commit operations on the primary are not affected by the relative slowness of the HADR network or the standby server, the log gap between the primary and standby may increase.

## Agenda

- HADR Review
- HADR superasync mode
- HADR Log Spooling
- HADR Multiple Standby
- HADR Time Delay
- HADR Configuration
- HADR Monitoring
- HADR Application Implications
- Comparison to other DB2 HA offerings

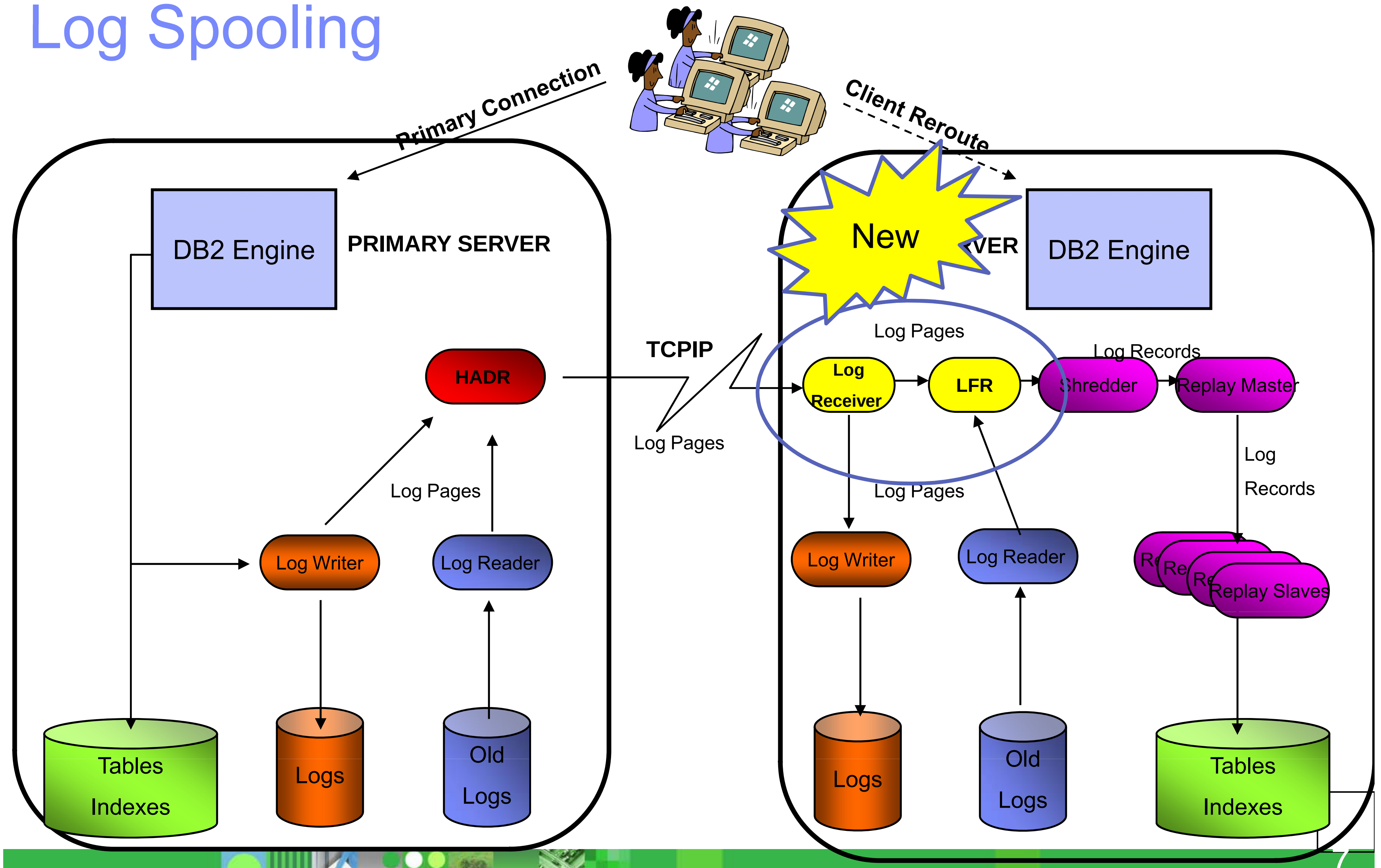
## HADR Standby Log Spooling

- When enabled, the log spooling feature will allow the standby to spool log records arriving from the primary
- This decouples log replay on the standby from receiving of the log data from the primary
- Logs will be spooled in the standby DB's active log path
- Set through a new DB CFG parameter, HADR_SPOOL_LIMIT
- Can limit the amount of space allocate to the log spool by specifying the maximum amount of disk space to use
  - Value of 0 disables spooling (default)
  - Value of -1 defines the spool to be unlimited (limited by file system free space)

## HADR Standby Log Spooling

- Log files will be automatically deleted when they are no longer required
- Log spooling will absorb load spikes in logging from the primary without providing any back pressure on the primary
- Takeover may take longer as the spool must be drained before takeover completes

# Log Spooling



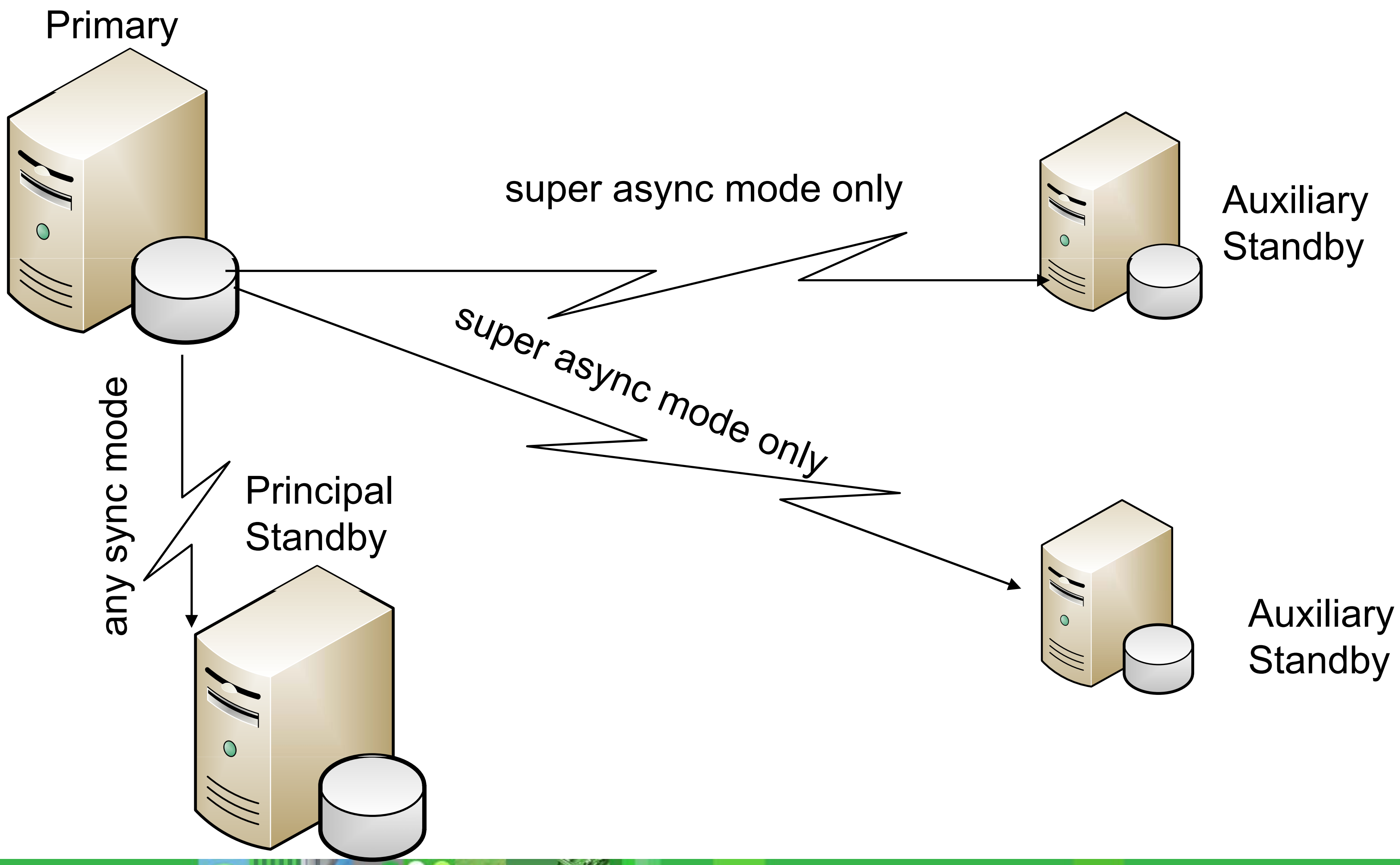
## Superasync mode or log spooling?

- Superasync mode
  - Eliminates back pressure on the primary
  - Useful network is slow
- Log spooling
  - Allows receipt of log records regardless of log replay on the standby
  - Useful with standby is slow
  - Can be used with any sync mode
- Both features can be combined if required
  - Slow network and a slow standby

## Agenda

- HADR Review
- HADR Super Async mode
- HADR Log Spooling
- HADR Multiple Standby
- HADR Time Delay
- HADR Configuration
- HADR Monitoring
- HADR Application Implications
- Comparison to other DB2 HA offerings

## HADR Multiple Standby Overview



## HADR Multiple Standby Features

- Principal Standby (PS) equivalent to standby today
  - PS supports any sync mode
  - Can automate takeover using integrated TSA

- Support for up to two(2) Auxiliary Standbys (AS)
  - AS supports super async mode only
  - No automated takeover supported
  - Always feed from the current primary
  - Can be added dynamically

## HADR Multiple Standby Enablement

- Principal Standby (PS) is specified via the HADR_REMOTE_HOST, HADR_REMOTE_SVC, HADR_REMOTE_INST

- HADR_TARGET_LIST is used to specify all standbys, both auxiliary as well as the principal standby

- HADR_TARGET_LIST uses a hostname or IP Address and port number format with the "|" character as a delimiter
  - E.g. host1.ibm.com:4000|host2.ibm.com:hadr_service|9.47.73.34:5000

- On each standby the HADR_REMOTE_HOST, HADR_REMOTE_INST, HADR_REMOTE_SVC must point to the current primary

- Primary will validate hostname and port number upon handshake from AS

- Existing single standby installations need no configuration change

## Multiple Standby Takeover processing on PS

- Control Takeover is issued on the Principal Standby
  - Primary and Principal Standby reverse roles
  - All Aux. Standbys will be automatically updated to point to the new primary
- Forced Takeover is issued on the Principal Standby
  - All Aux. Standbys will have to be manually updated to point to the new primary

## Multiple Standby Takeover processing on Aux S.

- Control Takeover is issued on any Auxiliary Standby
  - Old primary becomes the Principal Standby
  - Old principal standby becomes an Aux. Standby
  - All Aux. Standbys will be automatically updated to point to the new primary
- Forced Takeover is issued on the Auxiliary Standby
  - Old primary will need to be manually reconfigured to become the principal standby
  - Old principal standby will need to be manually reconfigured to become an auxiliary standby
  - Aux. Standby that did not become the primary must be manually reconfigured to point to the new primary

## Multiple Standby Restrictions

- You can have a maximum of three standby databases: one principal standby and one or two auxiliary standbys
- Only the principal standby supports all the HADR synchronization modes; all auxiliary standbys will be in SUPERASYNC mode
- IBM Tivoli System Automation for Multiplatforms (SA MP) support applies only between the primary HADR database and its principal standby
- You must set the **hadr_target_list database configuration parameter on all the** databases in the multiple standby setup. In addition, for each combination of primary and standby, role switch between those databases must be allowed. That is, each database in the target list of a particular database must also have that particular database in its target list.

## Multiple Standby Configuration Setup

On the primary, set the parameters to the corresponding values on the principal standby by issuing the following command:

```
DB2 "UPDATE DB CFG FOR dbname USING
HADR_REMOTE_HOST principalhostname
HADR_REMOTE_SVC principalservicename
HADR_REMOTE_INST principalinstname"
```

On each standby, set the parameters to the corresponding values on the primary by issuing the following command:

```
DB2 "UPDATE DB CFG FOR dbname USING
HADR_REMOTE_HOST primaryhostname
HADR_REMOTE_SVC primaryservicename
HADR_REMOTE_INST primaryinstname"
```

# Automatic reconfiguration of HADR Parameters

**Reconfiguration after HADR starts**

Configuration parameters that identify the primary database for the standbys and identify the principal standby for the primary are automatically reset when HADR starts if you did not correctly set them. This behavior applies to the following configuration parameters:

- **hadr_remote_host**
- **hadr_remote_inst**
- **hadr_remote_svc**

**Reconfiguration during and after a takeover**

After a forced or non-forced takeover, the values for the **hadr_remote_host, hadr_remote_inst, and hadr_remote_svc** configuration parameters are updated automatically on all the databases that are potentially a part of the new setup. Any database that is not a valid standby for the new primary, because they are not included in each other's target list, is not updated. If you want to include a database as a standby, you must ensure that it is in the target list of the primary and that the primary is in the target list of the new standby database. Otherwise, the standby database waits for the old primary to restart as a primary.

## Takeover Behavior in Multiple Standby Environment

Two types of takeovers as in traditional HADR

- Role Switch
  - sometimes called graceful takeover or non-forced takeover, can be performed only when the primary is available and it switches the role of primary and standby Provides zero data loss guarantee

- Failover
  - can be performed when the primary is not available. It is commonly used in primary failure cases to make the standby the new primary. The old primary remains in primary role in a forced takeover.

- Both types of takeover are supported in multiple standby mode, and any of the standby databases can take over as the primary. A crucial thing to remember, though, is that if a standby is not included in the new primary's target list, it is considered to be orphaned and cannot connect to the new primary.

## Takeover Behavior in Multiple Standby Environment

- Takeover (forced and non-forced) is supported on all standbys
- After a takeover, DB2 auto-redirects and makes necessary configuration changes (host/service/instance name of new primary) on the standbys that are in the new primary's target list (and vice versa)
- Standbys not in new primary's target list (and vice versa) are "orphaned" standbys
- Data loss (usually from a failover) complicates the picture:
  - If old primary has more data than new primary, it cannot be reintegrated without being reinitialized
  - If a standby has more data than new primary, it will not pass pair validation check and cannot become a standby for that primary
  - To avoid the latter, check the log positions of all standbys and fail over to the one with the most data

## Takeover Behavior in Multiple Standby Environment

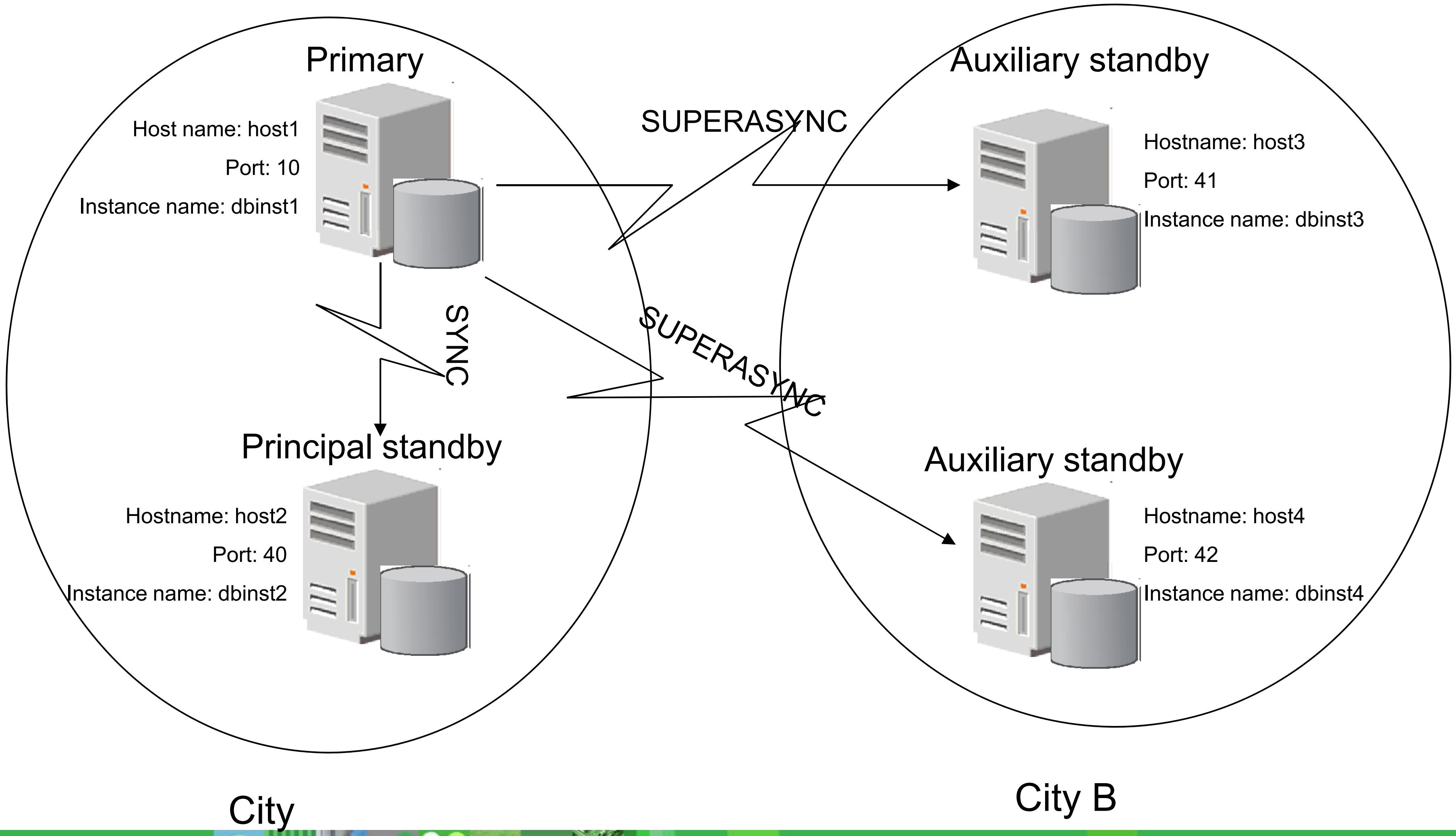
DB2 automatically makes a number of configuration changes for you so that the standbys listed in new primary's target list can connect to the new primary. The hadr_remote_host, hadr_remote_svc, and hadr_remote_inst configuration parameters are updated on the new primary and listed standbys in the following way

- **On the new primary:** They refer to the principal standby (the first database listed in the new primary's target list).
- **On the standbys:** They refer to the new primary. When an old primary is reintegrated to become standby, the START HADR AS STANDBY command first converts it to a standby. Thus it can also be automatically redirected to the new primary if it is listed in the target list of the new primary.
- Orphaned standbys are not automatically updated in this way. If you want them to join as standbys, you need to ensure they are in the new primary's target list and that they include the new primary in their target lists.

## Takeover Behavior in Multiple Standby Environment

- Role switch
  - Just as in single standby mode, role switch in multiple standby mode guarantees no data is lost between the old primary and new primary. Other standbys configured in the new primary’s hadr_target_list configuration parameter are automatically redirected to the new primary and continue receiving logs.
- Failover
  - Just as in single standby mode, if a failover results in any data loss in multiple standby mode (meaning that the new primary does not have all of the data of the old primary), the old and new primary's log streams diverge and the old primary has to be reinitialized. For the other standbys, if a standby received logs from the old primary beyond the diverge point, it has to be reinitialized. Otherwise, it can connect to the new primary and continue log shipping and replay. As a result, it is very important that you check the log positions of all of the standbys and choose the standby with the most data as the failover target. You can query this information using the db2pd command or the MON_GET_HADR table function.

## HADR Multiple Standby Example



## Multiple Standby Example

| Intended role | Host name | Port # | Instance name |
|---|---|---|---|
| Primary | host1 | 10 | dbinst1 |
| Principal standby | host2 | 40 | dbinst1 |
| Auxiliary standby | host3 | 41 | dbinst3 |
| Auxiliary standby | host4 | 42 | dbinst4 |

## Initial Setup

**Step 1: Create a backup image**:
On host1 (primary)
DB2 BACKUP DB HADR_DB to /nfs/db_backup

**Step 2: Initialize the standbys**:
on each of host2, host3, and host 4
DB2 RESTORE DB HADR_DB from /nfs/db_backup

**Step 3: Update the configuration**

the HADR_TARGET_LIST should be set to the following on all nodes:
PrincipalHost:Principalsvc | auxhost1:auxsvc1 |  auxhost2:auxsvc2

The following are not required to be set in a multiple standby environment as they will be automatically set:

- hadr_remote_host
- hadr_remote_svc
- hadr_remote_inst

## Initial Setup

**Step 4: Update the configuration optional step**

On the primary set the following:

| | |
|---|---|
| HADR_REMOTE_HOST | = principalhostname |
| HADR_REMOTE_SVC | = principalservicename |
| HADR_REMOTE_INST | = principalinstancename |

On each standby set the following:

| | |
|---|---|
| HADR_REMOTE_HOST | = primaryhostname |
| HADR_REMOTE_SVC | = primaryservicename |
| HADR_REMOTE_INST | = primaryinstancename |

## Initial Setup

On host1 (the primary):

```
DB2 "UPDATE DB CFG FOR hadr_db USING
HADR_TARGET_LIST host2:40|host3:41|host4:42
HADR_REMOTE_HOST host2
HADR_REMOTE_SVC 40
HADR_LOCAL_HOST host1
HADR_LOCAL_SVC 10
HADR_SYNCMODE sync
HADR_REMOTE_INST db2inst2"
```

## Initial Setup con’t

On host2 (the principal standby):

```
DB2 "UPDATE DB CFG FOR hadr_db USING
HADR_TARGET_LIST host1:10|host3:41|host4:42
HADR_REMOTE_HOST host1
HADR_REMOTE_SVC 10
HADR_LOCAL_HOST host2
HADR_LOCAL_SVC 40
HADR_SYNCMODE sync
HADR_REMOTE_INST db2inst1“
```

On host3 (an auxiliary standby):

```
DB2 "UPDATE DB CFG FOR hadr_db USING
HADR_TARGET_LIST host2:40|host1:10|host4:42
HADR_REMOTE_HOST host1
HADR_REMOTE_SVC 10
HADR_LOCAL_HOST host3
HADR_LOCAL_SVC 41
HADR_SYNCMODE superasync
HADR_REMOTE_INST db2inst1“
```

## Initial Setup con't

On host4 (an auxiliary standby):

```
DB2 "UPDATE DB CFG FOR hadr_db USING
HADR_TARGET_LIST host2:40|host1:10|host3:41
HADR_REMOTE_HOST host2
HADR_REMOTE_SVC 10
HADR_LOCAL_HOST host4
HADR_LOCAL_SVC 42
HADR_SYNCMODE su
HADR_REMOTE_INST db2i
HADR_REPLAY_DELAY 86400"
```

24 hour delay

## Starting HADR

**Starting the HADR databases**

The DBA starts the standby databases first, by issuing the following command on each of host2, host3, and host 4:

```
DB2 START HADR ON DB hadr_db AS STANDBY
```

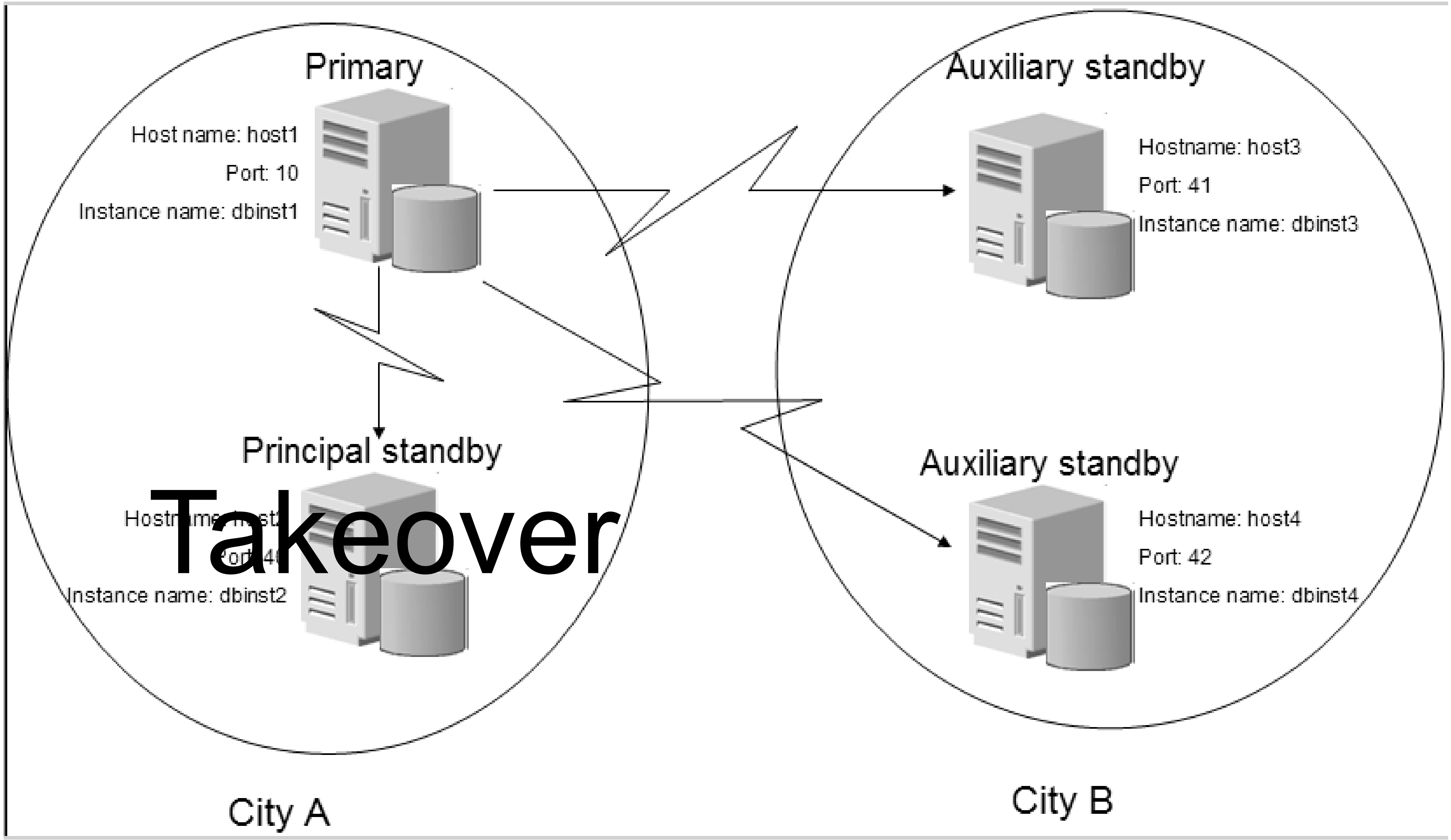
Next, the DBA starts HADR on the primary database, on host1:

```
DB2 START HADR ON DB hadr_db AS PRIMARY
```

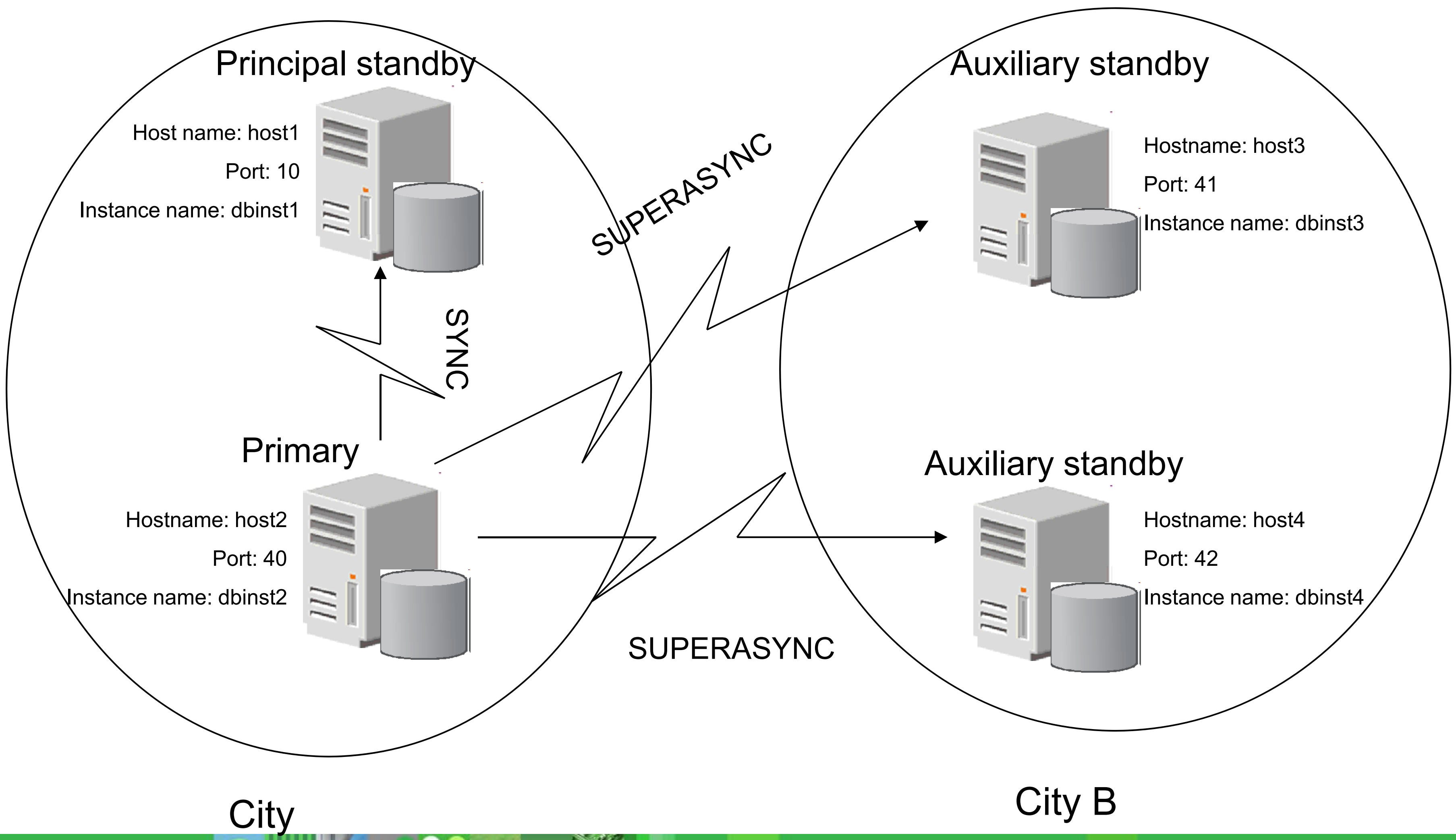
## Configuration values for each host

| Configuration parameter | Host1 | Host2 | Host3 | Host4 |
|---|---|---|---|---|
| Hadr_target_list | host2:40 \| host3:41 \| host4:42 | host1:10 \| host3:41 \| host4:42 | host2:20 \| host1:10 \| host4:42 | host2:40 \| host1:10 \| host3:41 |
| Hadr_remote_host | host2 | host1 | host1 | host1 |
| Hadr_remote_svc | 40 | 10 | 10 | 10 |
| Hadr_remote_inst | dbinst2 | dbinst1 | dbinst1 | dbinst1 |
| Hadr_local_host | host1 | host2 | host3 | host4 |
| Hadr_local_svc | 10 | 40 | 41 | 42 |
| Operational Hadr_syncmode | sync | sync | superasync | superasync |
| Effective Hadr_syncmode | N/A | sync | superasync | superasync |

# HADR Multiple Standby Example



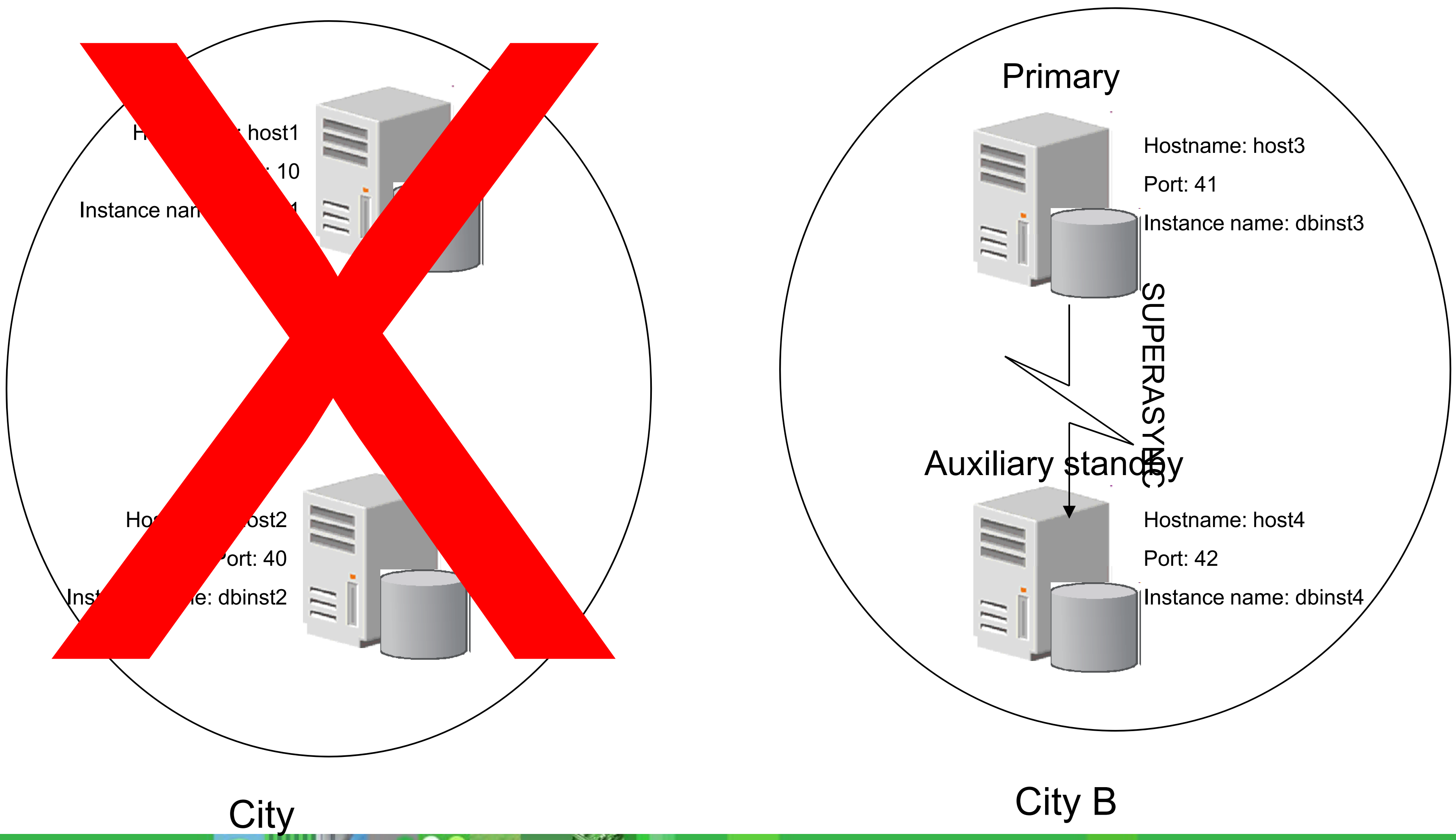
## HADR Multiple Standby Example



## After issuing takeover on host2 (auto reconfigured)

| Configuration parameter | Host1 | Host2 | Host3 | Host4 |
|---|---|---|---|---|
| Hadr_target_list | host2:40 \| host3:41 \| host4:42 | host1:10 \| host3:41 \| host4:42 | host2:40 \| host1:10 \| host4:42 | host2:40 \| host1:10 \| host3:41 |
| Hadr_remote_host | host2 | host1 | host2 | host2 |
| Hadr_remote_svc | 40 | 10 | 40 | 40 |
| Hadr_remote_inst | dbinst2 | dbinst1 | dbinst2 | dbinst2 |
| Hadr_local_host | host1 | host2 | host3 | host4 |
| Hadr_local_svc | 10 | 40 | 41 | 42 |
| Operational Hadr_syncmode | sync | sync | superasync | superasync |
| Effective Hadr_syncmode | sync | N/A | superasync | superasync |

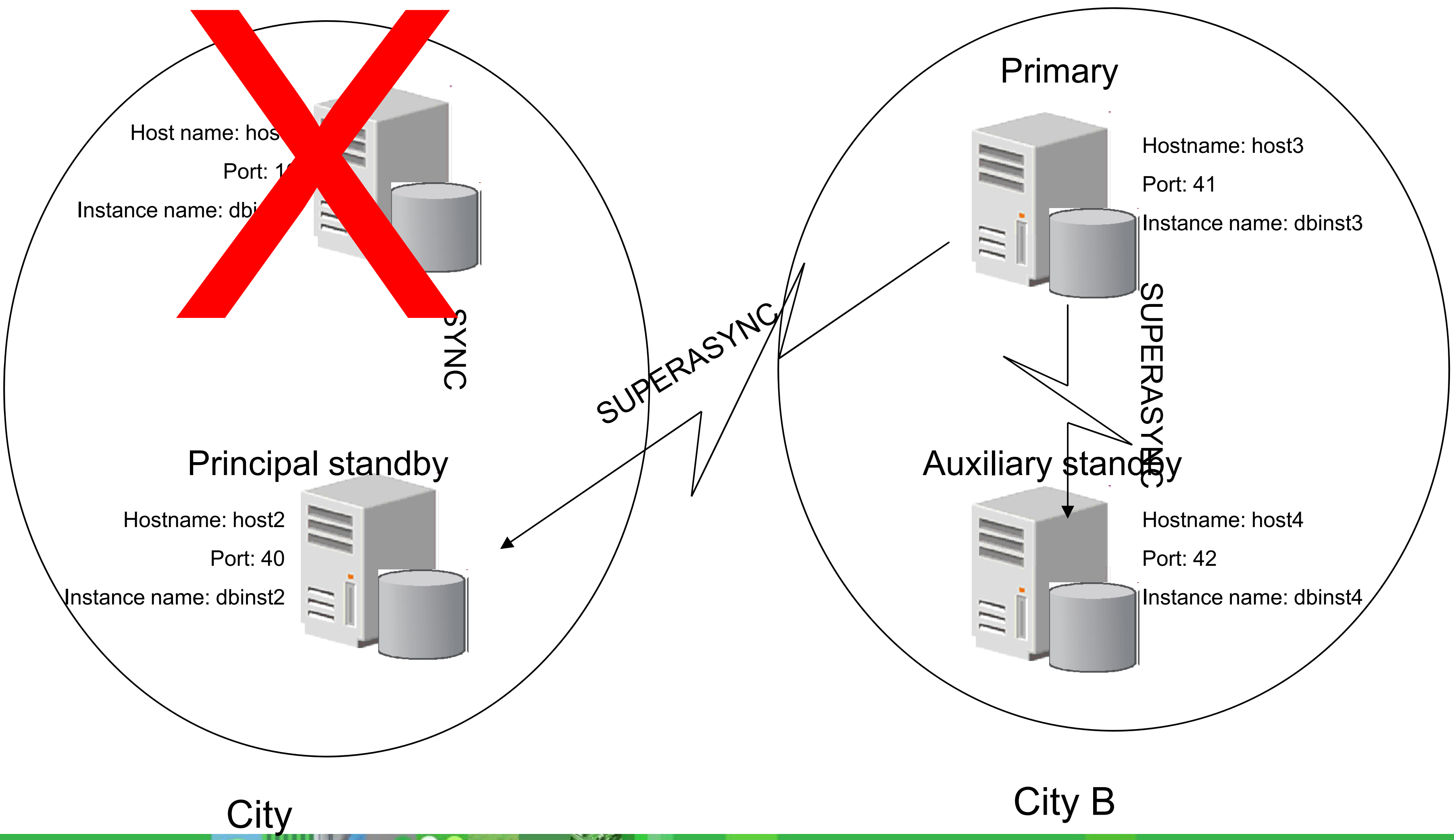
## HADR Multiple Standby Example



## After issuing takeover on host3 (host 1+2 are down)

| **Configuration parameter** | **Host1** | **Host2** | **Host3** | **Host4** |
|---|---|---|---|---|
| Hadr_target_list | host2:40 \| host3:41 \| host4:42 | host1:10 \| host3:41 \| host4:42 | host2:40 \| host1:10 \| host4:42 | host2:40 \| host1:10 \| host3:41 |
| Hadr_remote_host | host2 | host1 | host2 | host3 |
| Hadr_remote_svc | 40 | 10 | 40 | 41 |
| Hadr_remote_inst | dbinst2 | dbinst1 | dbinst2 | dbinst3 |
| Hadr_local_host | host1 | host2 | host3 | host4 |
| Hadr_local_svc | 10 | 40 | 41 | 42 |
| Operational Hadr_syncmode | sync | sync | superasync | superasync |
| Effective Hadr_syncmode | N/A | sync | n/a | superasync |

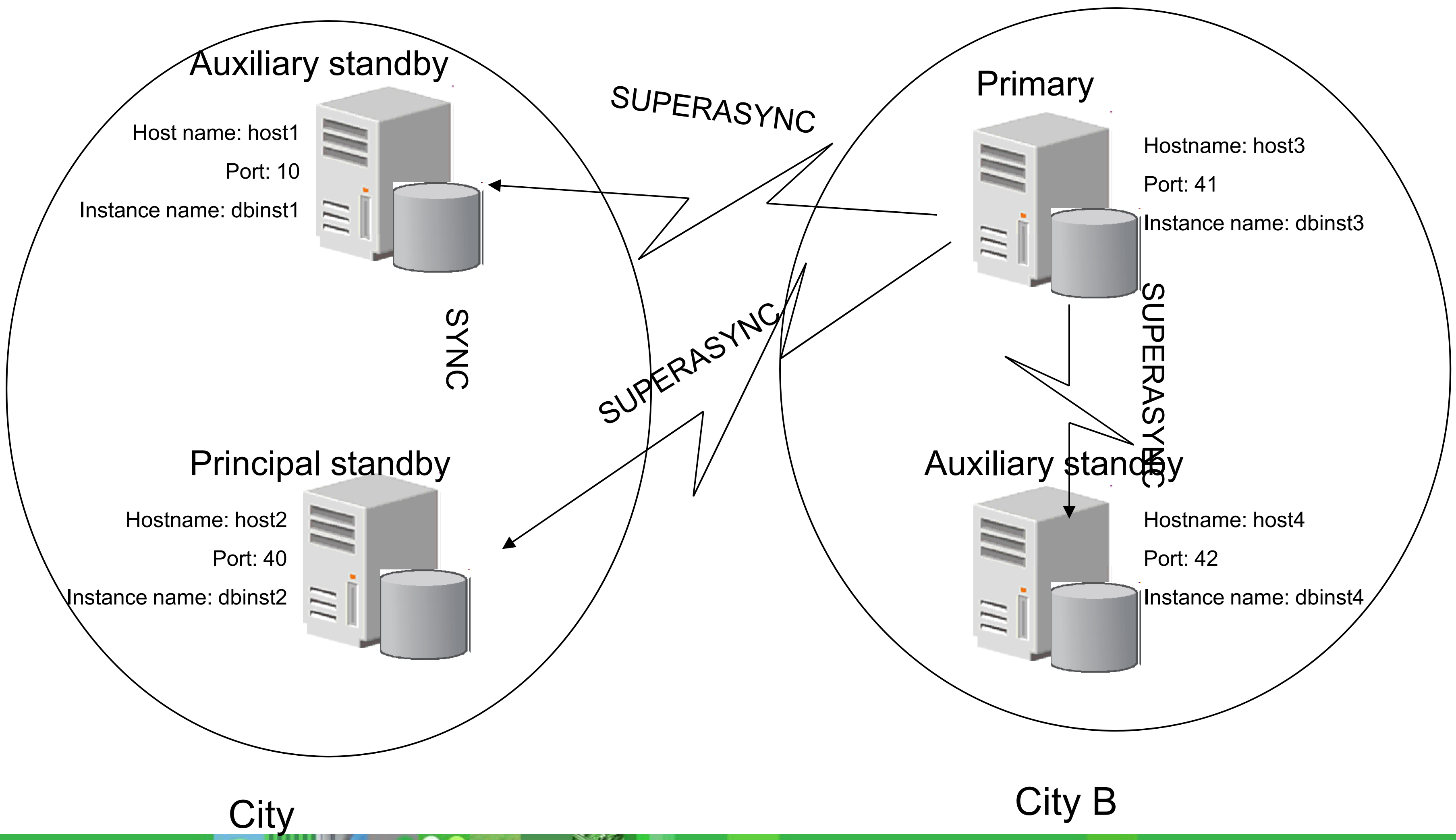
# HADR Multiple Standby Example



## After issuing takeover on host3 (host 2 online)

| Configuration parameter | Host1 | Host2 | Host3 | Host4 |
|---|---|---|---|---|
| Hadr_target_list | host2:40 \| host3:41 \| host4:42 | host1:10 \| host3:41 \| host4:42 | host2:40 \| host1:10 \| host4:42 | host2:40 \| host1:10 \| host3:41 |
| Hadr_remote_host | host2 | host3 | host2 | host3 |
| Hadr_remote_svc | 40 | 41 | 40 | 41 |
| Hadr_remote_inst | dbinst2 | dbinst3 | dbinst2 | dbinst3 |
| Hadr_local_host | host1 | host2 | host3 | host4 |
| Hadr_local_svc | 10 | 40 | 41 | 42 |
| Operational Hadr_syncmode | sync | sync | superasync | superasync |
| Effective Hadr_syncmode | N/A | superasync | n/a | superasync |

# HADR Multiple Standby Example



## After issuing takeover on host3 (host 1 online)

| Configuration parameter | Host1 | Host2 | Host3 | Host4 |
|---|---|---|---|---|
| Hadr_target_list | host2:40 \| host3:41 \| host4:42 | host1:10 \| host3:41 \| host4:42 | host2:40 \| host1:10 \| host4:42 | host2:40 \| host1:10 \| host3:41 |
| Hadr_remote_host | host3 | host3 | host2 | host3 |
| Hadr_remote_svc | 41 | 41 | 40 | 41 |
| Hadr_remote_inst | dbinst3 | dbinst3 | dbinst2 | dbinst3 |
| Hadr_local_host | host1 | host2 | host3 | host4 |
| Hadr_local_svc | 10 | 40 | 41 | 42 |
| Operational Hadr_syncmode | sync | sync | superasync | superasync |
| Effective Hadr_syncmode | superasync | superasync | n/a | superasync |

## Software upgrades in a Multi Standby Environment

The procedure is essentially the same as with single standby mode, except you should perform the upgrade on one database at a time and starting with an auxiliary standby. For example, consider the following HADR setup:\

- host1 is the primary
- host2 is the principal standby
- host 3 is the auxiliary standby

For this setup, perform the rolling upgrade or update according to the following sequence:

1. Deactivate host3, make the required changes, activate host3, and start HADR on host3 (as a standby).
2. After host3 is caught up in log replay, deactivate host2, make the required changes, activate host2, and start HADR on host2 (as a standby).
3. After host2 is caught up in log replay and in peer state with host1, issue a takeover on host2.
4. Deactivate host1, make the required changes, activate host1, and start HADR on host1 (as a standby).
5. After host1 is in peer state with host 2, issue a takeover on host1 so that it becomes the primary again and host2 becomes the principal standby again.

## Agenda

- HADR Review
- HADR Super Async mode
- HADR Log Spooling
- HADR Multiple Standby
- HADR Time Delay
- HADR Configuration
- HADR Monitoring
- HADR Application implications
- Comparison to other DB2 HA offerings

## HADR TIME DELAY

New configuration parameter which will control how far behind the standby will remain at all times to prevent data loss due to errant transactions

**hadr_replay_delay :**

- This parameter specifies the time that must have passed from when the data is changed on primary before these changes would be reflected on the standby database. The time is specified in number of seconds

## HADR TIME DELAY Restrictions

- Can only be set on a standby database.
- A TAKEOVER command on a standby with replay delay enabled will fail.
  - You must first set the hadr_replay_delay configuration parameter to 0 and then deactivate and reactivate the standby to pick up the new value, and then issue the TAKEOVER command.
- The delayed replay feature is supported only in SUPERASYNC mode

## Agenda

- HADR Review
- HADR Super Async mode
- HADR Log Spooling
- HADR Multiple Standby
- HADR Time Delay
- HADR Configuration
- HADR Monitoring
- HADR Application implications
- Comparison to other DB2 HA offerings

## HADR Configuration Parameters

**hadr_db_role**
This parameter indicates the current role of a database; Valid values are: STANDARD, PRIMARY, or STANDBY.

**hadr_local_host**
specifies the local host for high availability disaster recovery (HADR) TCP communication.

**hadr_local_svc**
specifies the TCP service name or port number for which the local high availability disaster recovery (HADR) process accepts connections.

**hadr_peer_window**
When you set hadr_peer_window to a non-zero time value, then a HADR primary-standby database pair continues to behave as though still in peer state, for the configured amount of time, if the primary database loses connection with the standby database. This helps ensure data consistency

**hadr_remote_host**
specifies the TCP/IP host name or IP address of the remote high availability disaster recovery (HADR) database server

## HADR Configuration Parameters

**hadr_remote_inst**
specifies the instance name of the remote server. High availability disaster recovery (HADR) also checks whether a remote database requesting a connection belongs to the declared remote instance.

**hadr_remote_svc**
specifies the TCP service name or port number that will be used by the remote high availability disaster recovery (HADR) database server.

**hadr_replay_delay**
specifies the number of seconds that must pass from the time that a transaction is committed on the primary database to the time that the transaction is committed on the standby database

**hadr_spool_limit**
determines the maximum amount of log data that is allowed to be spooled to disk on HADR standby

**hadr_syncmode**
specifies the synchronization mode, which determines how primary log writes are synchronized with the standby when the systems are in peer state

## HADR Configuration Parameters

**hadr_target_list**

This parameter, which enables HADR to run in multiple standby mode, specifies a list of up to three target *host*:*port* pairs that act as HADR standby databases.

**hadr_timeout**

specifies the time (in seconds) that the high availability disaster recovery (HADR) process waits before considering a communication attempt to have failed

**Blocknonlogged**

specifies whether the database manager will allow tables to have the NOT LOGGED or NOT LOGGED INITIALLY attributes activated.

**Logindexbuild**

specifies whether index creation, recreation, or reorganization operations are to be logged so that indexes can be reconstructed during DB2 rollforward operations or high availability disaster recovery (HADR) log replay procedures.

## HADR Configuration Parameters Updates

- you need only stop and start HADR for updates to some HADR configuration parameters for the primary database to take effect. You do not have to deactivate and reactivate the database. This dynamic capability affects only the primary database because stopping HADR deactivates any standby database.
- The affected configuration parameters are as follows:
  - **hadr_local_host**
  - **hadr_local_svc**
  - **hadr_peer_window**
  - **hadr_remote_host**
  - **hadr_remote_inst**
  - **hadr_remote_svc**
  - **hadr_replay_delay**
  - **hadr_spool_limit**
  - **hadr_syncmode**
  - **hadr_target_list**
  - **hadr_timeout**

# Frequently Used HADR Registry Variables

- DB2_HADR_NO_IP_CHECK
  - Registry variable to bypass IP address cross check in NAT (Network Address Translation) environment. Search "DB2_HADR_NO_IP_CHECK" in http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp
- DB2_HADR_PEER_WAIT_LIMIT
  - Registry variable for maximal primary logging wait time. Prevents primary logging from blocking because of slow or blocked standby. search "DB2_HADR_PEER_WAIT_LIMIT" in http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp
- DB2_HADR_SOSNDBUF and DB2_HADR_SORCVBUF
  - Registry variable to tune socket buffer size for HADR connections. This allows tuning TCP window size for HADR connection without impact on other TCP connections on the system.
- DB2_HADR_ROS
  - Enables read on the HADR standby
- DB2_STANDBY_ISO
  - coerces the isolation level requested by applications and statements running on an active HADR standby database to Uncommitted Read (UR)

## Agenda

- HADR Review
- HADR Super Async mode
- HADR Log Spooling
- HADR Multiple Standby
- HADR Time Delay
- HADR Configuration
- HADR Monitoring
- HADR Application implications
- Comparison to other DB2 HA offerings

## HADR Monitoring Changes

There are two preferred ways of monitoring HADR:

- The db2pd command
- The MON_GET_HADR table function
- From the primary = information about the primary and all standbys
- From a standby = information about that standby and the primary

You can also use the following methods, but have been deprecated and ma be removed in a future release:

The **GET SNAPSHOT FOR DATABASE command**

- The db2GetSnapshot API
- The SNAPHADR administrative view
- The SNAP_GET_HADR table function
- Other snapshot administrative views and table functions

- Two interfaces for monitoring HADR:
- db2pd command with -hadr option
- MON_GET_HADR table function (new for v10)
- From the primary = information about the primary and all standbys
- From a standby = information about that standby and the primary
- Other interfaces like GET DB CONFIG will still work, but only report the primary and the principal standby

## DB2 PD Changes – one entry for each primary-standby pair

```
db2pd -db HADRDB -hadr
Database Member 0 -- Database HADRDB -- Active -- Up 0 days 00:23:17 -- Date 06/08/2011
HADR_ROLE = PRIMARY
REPLAY_TYPE = PHYSICAL
HADR_SYNCMODE = SYNC
STANDBY_ID = 1
LOG_STREAM_ID = 0
HADR_STATE = PEER
PRIMARY_MEMBER_HOST = hostP.ibm.com
PRIMARY_INSTANCE = db2inst
PRIMARY_MEMBER = 0
STANDBY_MEMBER_HOST = hostS1.ibm.com
STANDBY_INSTANCE = db2inst
STANDBY_MEMBER = 0
HADR_CONNECT_STATUS = CONNECTED
HADR_CONNECT_STATUS_TIME = 06/08/2011 13:38:10.199479 (1307565490)
HEARTBEAT_INTERVAL(seconds) = 25
HADR_TIMEOUT(seconds) = 100
TIME_SINCE_LAST_RECV(seconds) = 3
PEER_WAIT_LIMIT(seconds) = 0
```

## DB2 PD Changes – one entry for each primary-standby pair con't

```
LOG_HADR_WAIT_CUR(seconds) = 0.000
LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.006298
LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.516
LOG_HADR_WAIT_COUNT = 82
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 50772
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87616
PRIMARY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
STANDBY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
HADR_LOG_GAP(bytes) = 0
STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
STANDBY_RECV_REPLAY_GAP(bytes) = 0
PRIMARY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
STANDBY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
STANDBY_REPLAY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
STANDBY_RECV_BUF_SIZE(pages) = 16
STANDBY_RECV_BUF_PERCENT = 0
STANDBY_SPOOL_LIMIT(pages) = 0
PEER_WINDOW(seconds) = 0
READS_ON_STANDBY_ENABLED = Y
STANDBY_REPLAY_ONLY_WINDOW_ACTIVE = N
```

## Monitoring Standby Servers

**db2pd command**
This command retrieves information from the DB2 memory sets. You can issue this command from either a primary database or a standby database. If you are using multiple standby mode and you issue this command from a standby, it does not return any information about the other standbys.

**MON_GET_HADR table function**
If you want to issue the MON_GET_HADR function against a standby database, be aware of the following points:

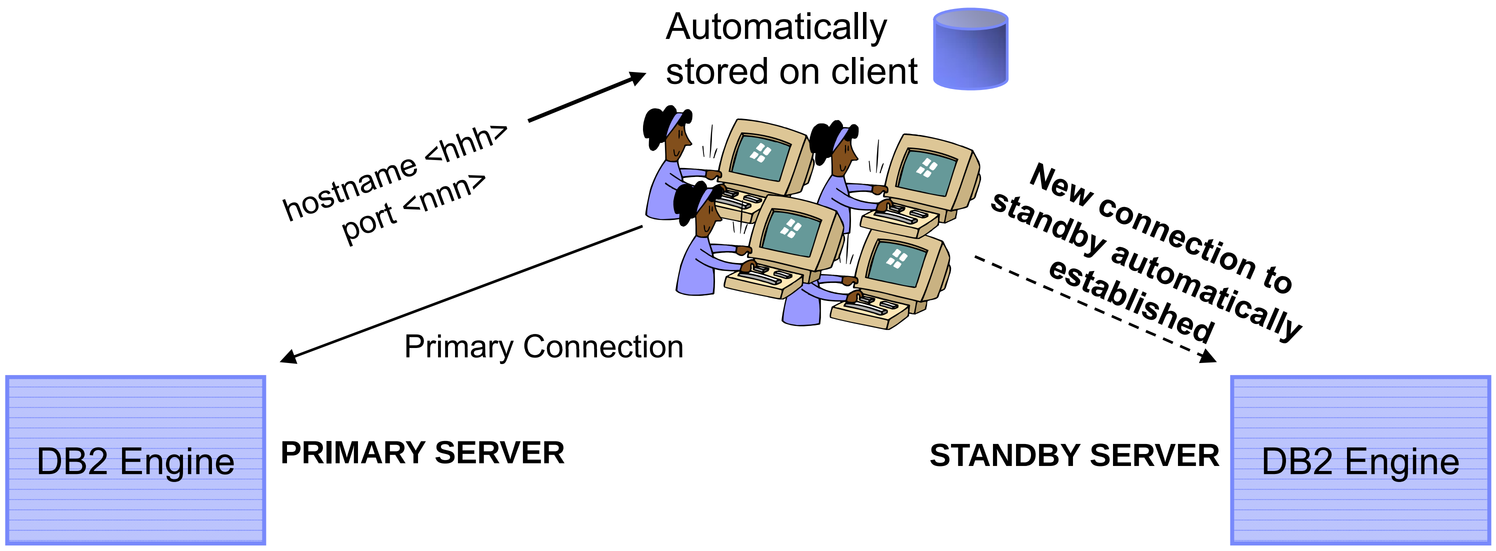
- You must enable reads on standby on the standby.
- Even if your HADR setup is in multiple standby mode, the table function does not return any information about any other standbys.

## Agenda

- HADR Review
- HADR Super Async mode
- HADR Log Spooling
- HADR Multiple Standby
- HADR Time Delay
- HADR Configuration
- HADR Monitoring
- HADR Application implications
- Comparison to other DB2 HA offerings

# Automatic Client Reroute

- Automatic, transparent connection to alternate server when primary connection fails
  - If there is a currently executing SQL statement, it will fail with sqlcode -30108
  - Transaction can then be re-driven without re-establishing a connection
- Alternate information Stored on client
  - System database directory
  - alternateDataSource property (Java Type 4 driver)
- Works with HADR, EE/ESE, EEE/DPF, Replication



db2 update alternate server for database <dbname> using hostname <hhh> port <nnn>

## Without DB2 Automatic Client Reroute Enabled

- During a server or database failure
  - **the current in-flight transaction will be rolled back**
  - Application receives SQLCODE:
    - -30081, -1224, -1776
    - java.sql.SQLException SQLCODE -4499 for JDBC and SQLJ clients
  - Application must
    - Re-establish connection
    - Setup environment
      - Create statement, prepare statement
    - Resubmit statement or transaction

## With DB2 Automatic Client Reroute Enabled

- During a server or database failure
  - **the current in-flight transaction will be rolled back**
  - Application receives connections re-established SQLCODE:
    - -30108
    - java.sql.SQLException SQLCODE -4498 for JDBC and SQLJ clients
  - Application connection is automatically re-established and environment maintained
    - Do not need to retry connect, create statement or prepare statement
  - Application need only resubmit the statement or transaction
    - Similar behavior to application getting lock timeout or deadlock sql error, -911 (ie. treat the event as a rollback)

## Considerations for WAS environments

- Ensure pool purge policy is set to entire pool
  - Facilitates entire pool getting re-established upon a communication error or reroute message
  - Reduces subsequent error paths and notifications
    - Which can occur later then when the original failure occurred

## Client Reroute Enhancements in DB2 9.5 FP1

- Seamless failover for JAVA applications
  - Suppress the SQLException with SQLCODE -4498
  - Driver re-issues the SQL statement that was being processed when the original connection failed
  - Conditions required for seamless failover to occur:
    - Set enableSeamlessFailover datasource property to True
    - The connection was not being used for a transaction at the time of failure
      - The failure must occur when the first SQL statement in the transaction is executed
    - There is no outstanding global resources in use
      - global temporary tables
      - open, held cursors

## Considerations for WAS environments

- With enableSeamlessFailover enabled
  - pool purge policy setting of “**failing connections only”**
    - only purge the failing connection out of the pool.
  - connections in the WAS in-use pool but that are not in a UOW will failover on next touch seamlessly (ie. no -4498 is throw)
  - connections in the WAS free pool will failover on first touch seamlessly
  - connections in the WAS in-use pool and that are in a UOW will get the -4498 (connections re-established) – other pool connections remain
  - any connections that have outstanding global resources that can't be restored will receive the -4498 – other pool connections remain

## Agenda

- HADR Review
- HADR Super Async mode
- HADR Log Spooling
- HADR Multiple Standby
- HADR Time Delay
- HADR Configuration
- HADR Monitoring
- HADR Application implications
- Comparison to other DB2 HA offerings

## High Availability Positioning

Bronze

**DB2 Integrated Clustering**

- Hot / Cold
- Basic availability solution
- Free is most cases
- Provides failover typically less than 1 or 2  minutes

Silver

**HADR**

- Hot/Warm (optional hot with read on standby option
- Provides very fast failover, typically less than 1 minute
- Easy to setup with turnkey availability
- Minimal licensing on the standby server
- Basic reporting (DB2 v9.7 FP1) requires full standby licensing
- Zero data loss option

Gold

**PureScale**

- Hot/Hot
- Online failover
- Transactions on surviving nodes are not impacted
- Provides the fastest failover, typically less than 30 seconds
- Online scaleout capability to address workload spikes

## Disaster Recovery Positioning

Bronze

**Log Shipping**
- Host/Cold
- No access to the target DB
- Target DB always at least 1 full log file behind
- Must complete rollforward before bringing target DB online
- Full DB replication only
- Requires all operations be logged

- DDL and DML replication support
- Async only
- Easy to setup

Silver

**Q Replication**
•Hot / Hot
•Unrestricted access to the target DB
•Supports different versions of DB2 on source and target
•Supports sub setting of data
•Multiple target support
•Online failover
•Can replicate to a different topology

•Async only
•Complex to setup
•No DDL support

Gold

**HADR Physical Replication**
•Hot / Warm
•Zero data loss option
•Provides very fast failover, typically under 1 minutes
•Easy to setup with turnkey availability
•DDL and DML replication support

•Limited use of the standby
•Single target support only
•Full DB replication only