Architetture dati

Replication

Contents

1. Definitions
2. Contexts motivating replication
3. IBM replication technologies
4. Microsoft SQL Server replication technologies

Part 1 - Definitions

Database replication

Database replication is

- the process of creating and maintaining multiple instances of the same database and
- the process of **sharing data** or **database design changes** between databases in different locations **without having to copy the entire database**.

Syncronization

- **Synchronization** is the process of ensuring that every copy of the database contains sooner or later the same objects and data.

Classifications of replication

- Syncronous vs Asyncronous
- IBM Topology based and Mode based -> in detail
- Microsoft SQL Server Snapshot vs Transactional vs Merge → in detail

Synchronous Versus Asynchronous Replication

- *Synchronous* – updates to replicated data are part of enclosing transaction.
  - If one or more sites that hold replicas are unavailable transaction cannot complete.
  - Large number of messages required to coordinate synchronization.
- *Asynchronous* - target database updated **after** source database modified.
  - Delay in regaining consistency may range from few seconds to several hours or even days.

Syncronous vs Asyncronous

- **Synchronous Replication**:

  What is: updating two storages at the same time; roll back if one fails

  Benefits: High availability/auto fail-over/minimal data loss

  Usages: Disaster recover

  Drawbacks: Network efficiency /scalability/cost/less flexibility

- **Asynchronous Replication**:

  What is: changes are captured on the primary storage and immediately / timely propagated

  Benefits: low cost / scalability /flexibility

  Usages: load balance/off-line access/access efficiency

  Drawbacks: data lost

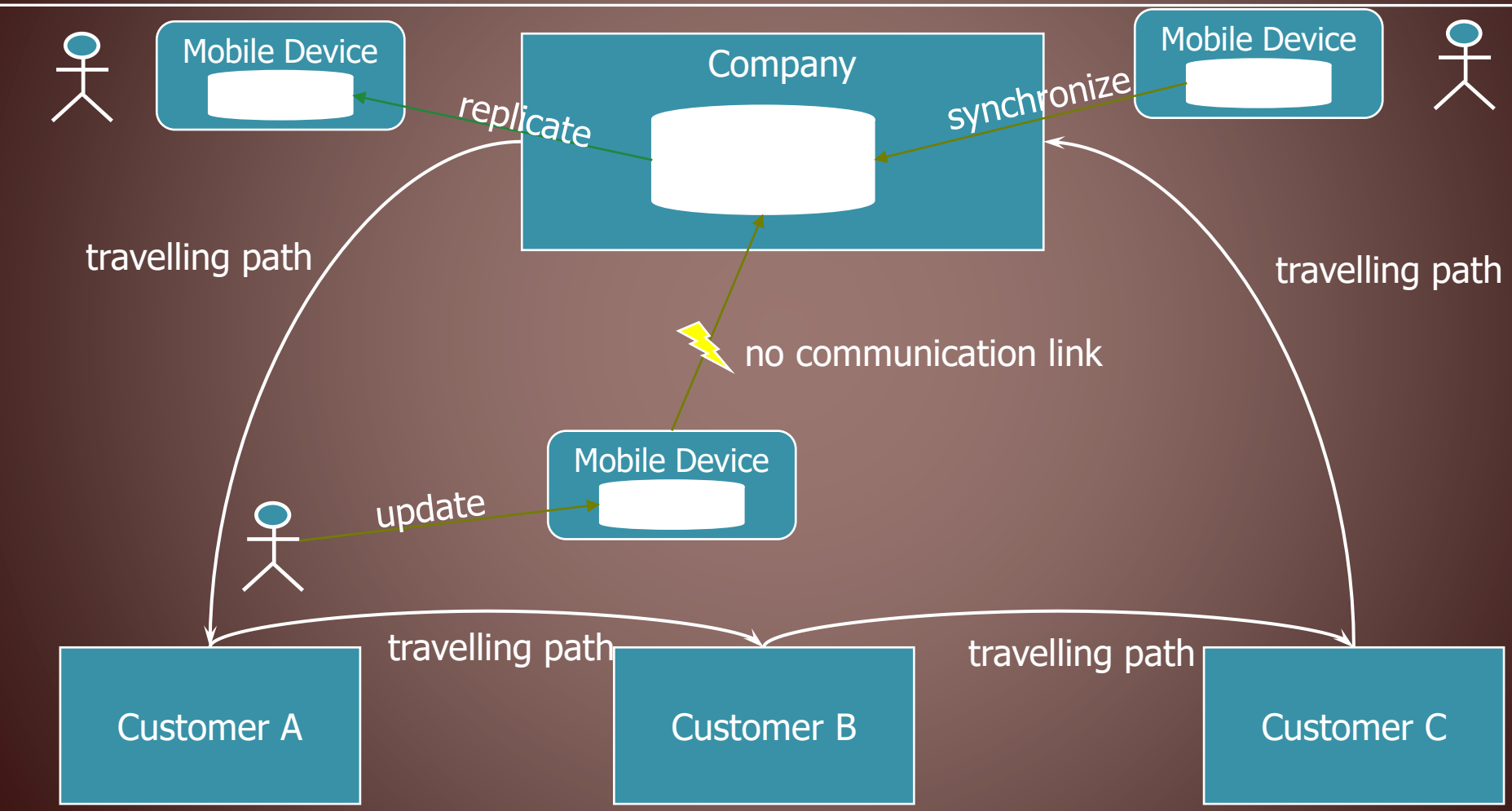
Part 2 - Contexts motivating replication

Contexts motivating replication

1. Sharing of data among dispersed users
2. Data consolidation (central audit & analyse)
3. Data distribution (for branch offices)
4. Performance
    1. Access efficiency (moving data near applications)
    2. Load balance (distributing access load)
    3. Availability (off-line access)
5. Separating Data Entry and Reporting
6. Application co-existence

1. **Share data** among dispersed users

The disconnected sales-person problem: a typical day



1. **Share data** among dispersed users Traveling sales force - 1

- A first scenario in which replication is often utilized is that of **a traveling sales force**.
- Sales people usually carry their own laptops, which can contain important company data, and they often update that data throughout the day.
- The new information is then fed into the database back at headquarters. Because the laptops are often not connected, a continuous replication setup is not an option.

1. **Share data** among dispersed users Traveling sales force - 2

- What further complicates the issue is that **the same data can sometimes be updated in different places**, creating conflicts that must be resolved.
- With **merge replication**, SQL Server provides a replication topology that was built for exactly this type of scenario → see later

2. **Consolidation** of data from remote systems

- An enterprise may have data on many different distributed systems
  - Retail companies have data at each store.
  - Manufacturing companies have data at each plant.
  - Insurance companies have data at each branch office or on each salesperson's laptop computer.
- Replication can **copy changes from each of the distributed sites to a central site**
  - for analysis and reporting, and
  - for data warehouse & business intelligence applications such as OLAP or Data Mining
  - for enterprise application processing.

3. **Data Distribution**
E-commerce applications

- Continuous bidirectional synchronization between web-based applications and mission-critical business applications in e-commece (e.g. continuous changing pricing, provisions, orders)
- It helps organizations **improve customer online shopping experience** with improved visibility into inventory and customer shopping activities (e.g. only two left copies!)
- E.g. Amazon

3. Data distribution **among remote offices**

- You can use database replication to **create copies** of a corporate database to send to each satellite office across a wide area network (WAN).
- Each location enters data in its replica, and **all remote replicas are synchronized** with the replica at corporate headquarters.
- Individual replicas can maintain **local tables** that contain information not included in the other replicas in the set

4.1 Improve **data accessibility**

- If your solution **does not need to have immediate updates** to data, you can use database replication to **reduce the network load** on your primary server.
- Introducing **a second server with its own copy** of the database improves response time.
- Replication requires **less centralized administration** of the database while offering **greater access** to centralized data.

4.2. **Load Balance** Across Servers

- As your organization grows, you might find yourself in a situation in which **a single database server is utilized by too many users**.
- If **CPU utilization** on your database servers is constantly **over 80 percent** and you have tuned database design and queries appropriately, chances are you could benefit by **spreading the user base over multiple servers**.

4.2. **Load Balance** Across Servers Example

- For instance, a server named South could serve all employees working in the southern United States, and a server called North could serve all Northerners.
- If you need to combine all data for reporting, you could use replication to **move transactions from North and South** to a server named Central_Reporting.

4.3. **Improve Availability**

- Occasionally, you might consider using replication for **high availability**; that is, to replicate transactions from the main server to a standby server.
- If the main server fails, you can then point your data sources to the standby server.
- Be aware that **using replication for high availability takes careful planning and testing**.

5. **Separating Data Entry and Reporting**

- If you have worked in an environment in which **the same database is used for data entry and reporting**, you probably know that things aren't always rosy.
- Constantly reading (reporting) and modifying data (entry) in the same set of tables just doesn't work very well if you care about data integrity.
- **Transactions that run against a set of tables prevent reading the locked data rows and pages**, or perhaps prevent even an entire table from being read by a report.

5. **Separating Data Entry and Reporting**

- In such an environment, **you are bound to see blocking locks**. Although there are ways to avoid blocking, **it is best to separate data entry and reporting databases**.
- Transactional replication works well by delivering data changes from the data entry server to the reporting server.

6. Application co-existence

- There may be **complex data transformation** needed to fit new application requirements.
- The new application may be
  - a Web application,
  - a purchased package, or
  - an application distributed on multiple laptop computers.
- The data that is copied may **need to be filtered and/or transformed** for the target application.

6. Application co-existence Migrating between environments

- Distribution of data can also be used to provide **application co-existence when migrating from one environment to another**.
- Legacy data can be copied to the new environment for reference by the new applications **until such time as** the legacy applications are migrated to the new environment.

When Database Replication should not be used

Although database replication has many benefits and can solve many problems in distributed-database processing, in some situations replication is less then ideal.

Two relevant cases:

1. There are **frequent updates** of existing records at multiple replicas
2. Data consistency is critical at all times

1. There are frequent updates of existing records at multiple replicas

- Solutions that have a large number of **record updates on the same records** in different replicas are likely to have **more record conflicts** than solutions that **simply insert new records** in a database.
- If changes are made to the same record by different users and at the same time then record conflicts will definitely appear.
- This can be real time consuming because the conflicts must be resolved manually.

2. Data consistency is critical at all times

Solutions that rely on information being correct at all times, such as

- funds transfers,
- airline reservations, and
- the tracking of package shipments

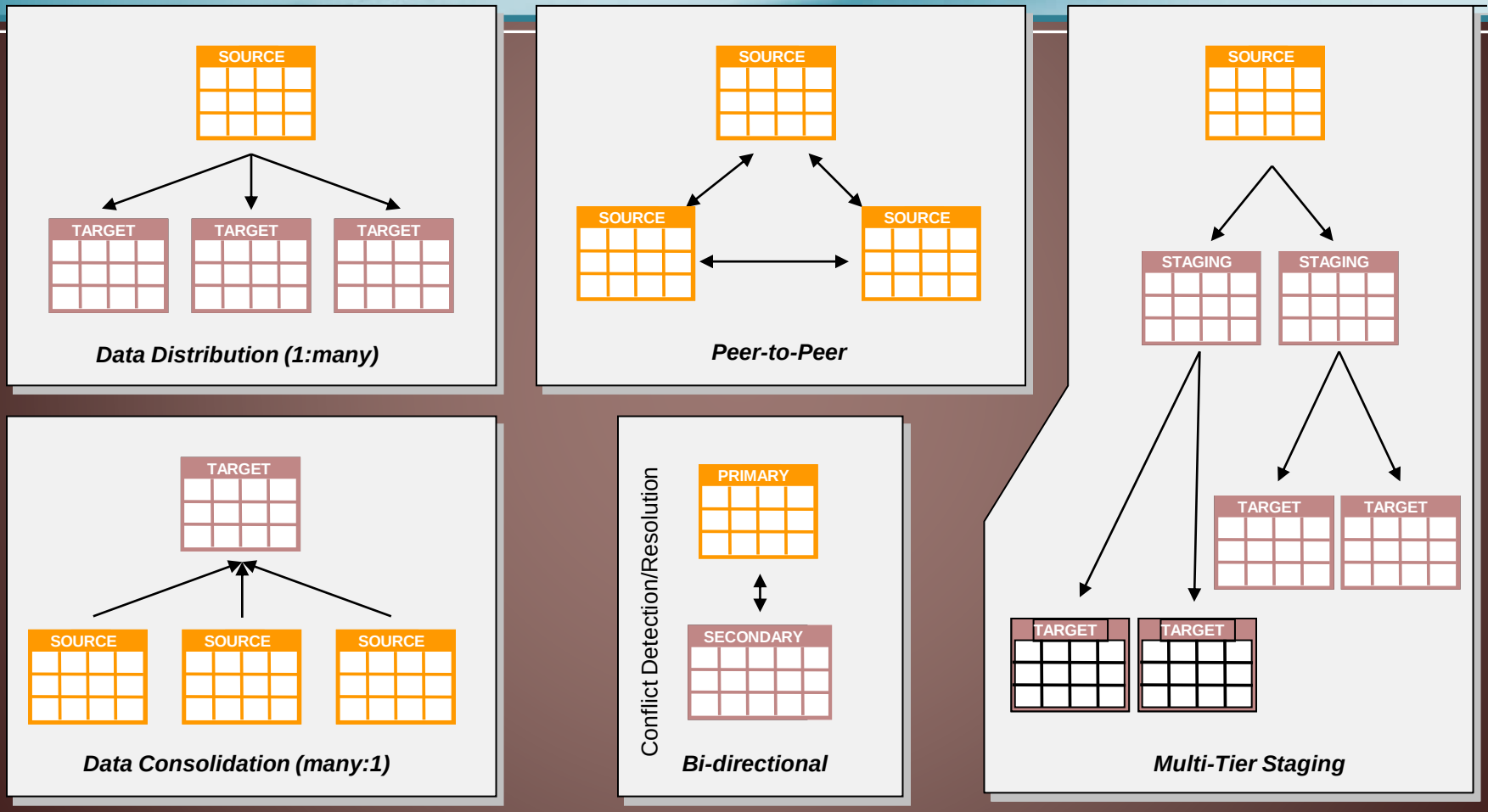
usually use a transaction method ACID compliant.

Although transactions can be processed within a replica, there is no support for processing transactions across replicas.

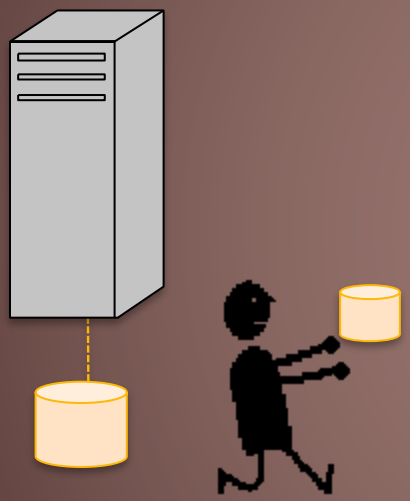
In conclusion
Benefits of replication

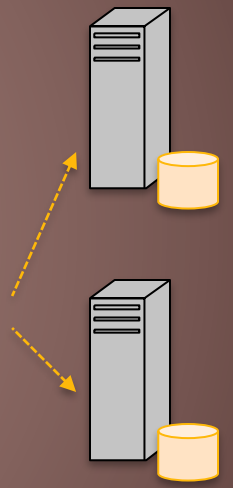
- Availability
- Reliability
- Performance
- Load reduction
- Disconnected computing
- Supports many users

Tipologie di replica



Come realizzare una replica



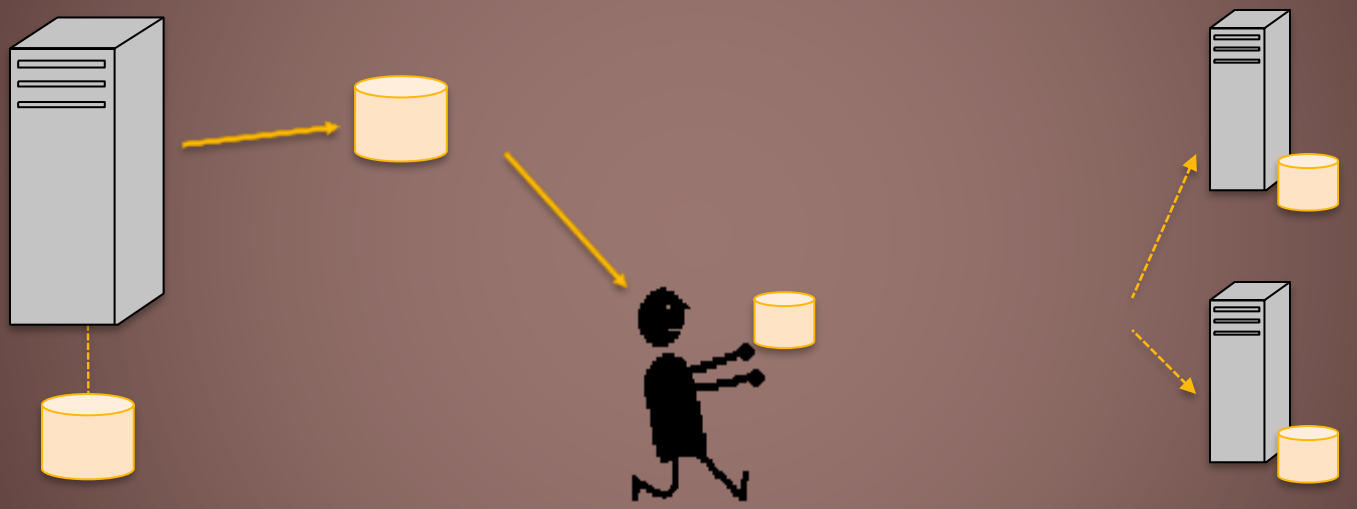


1. Detach
3. Attach

2. Copy

4. Attach

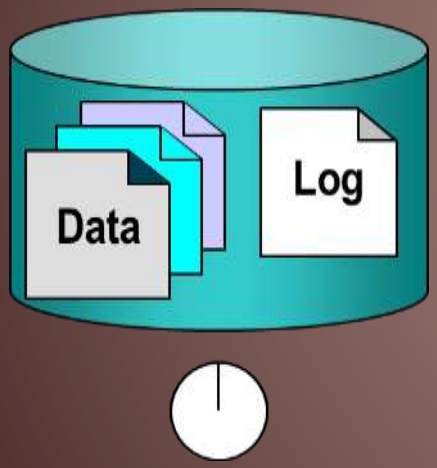
Come realizzare una replica



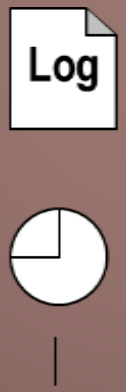
1. Backup **(2. Copy)** **3. Restore**

Come creare le repliche

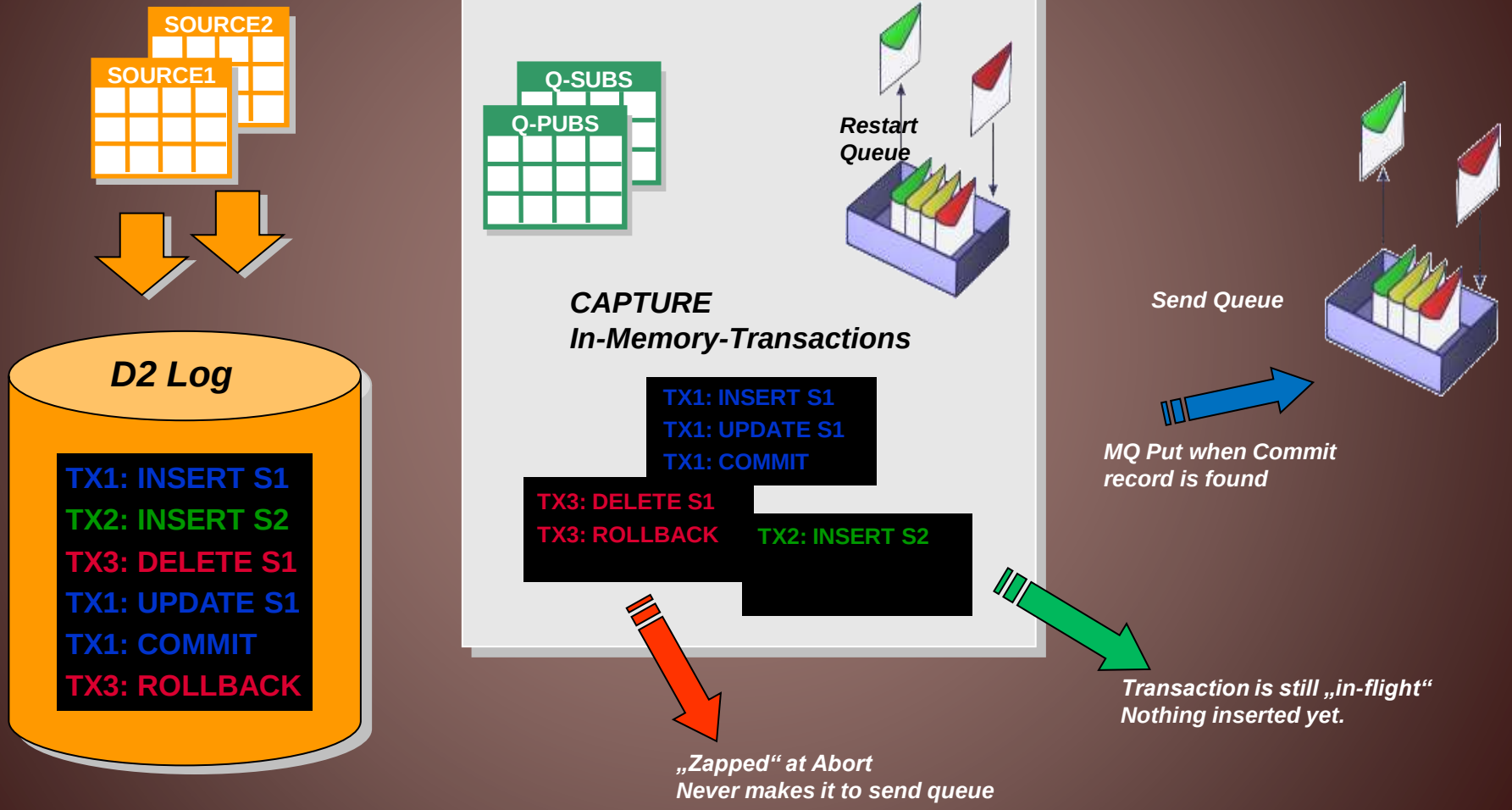
Full backup



Transaction log

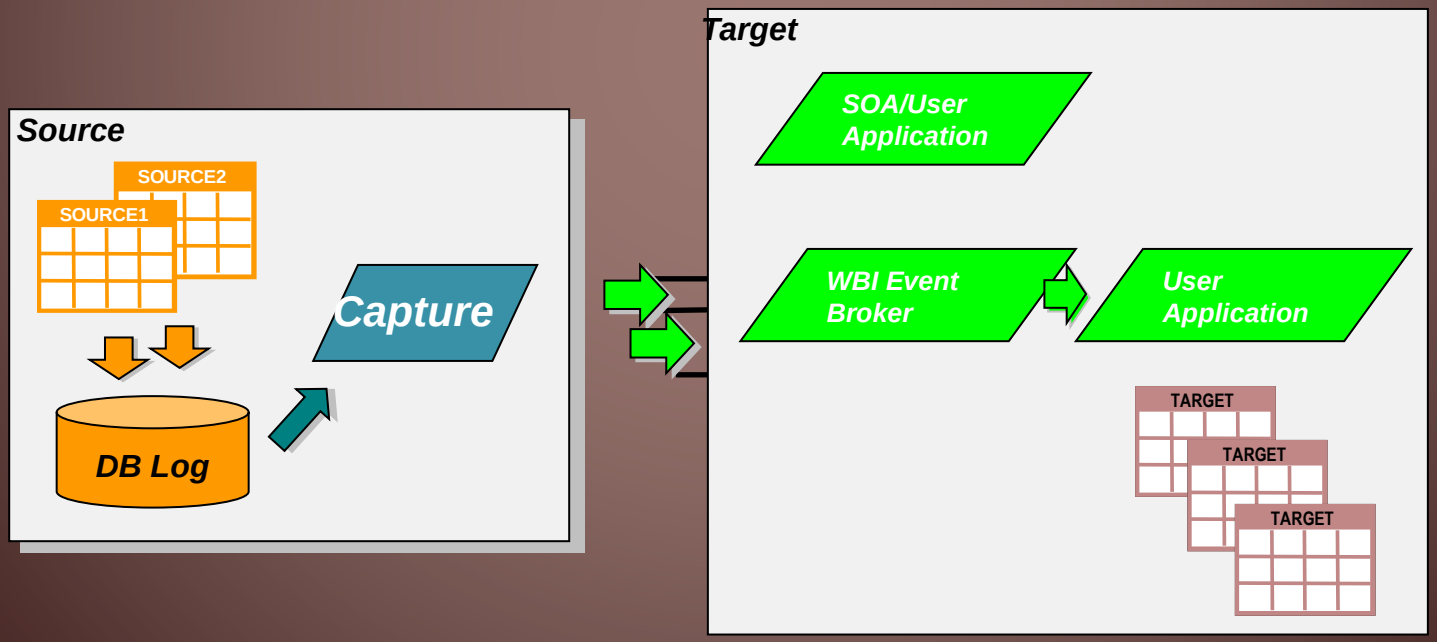


Come costruire una replica

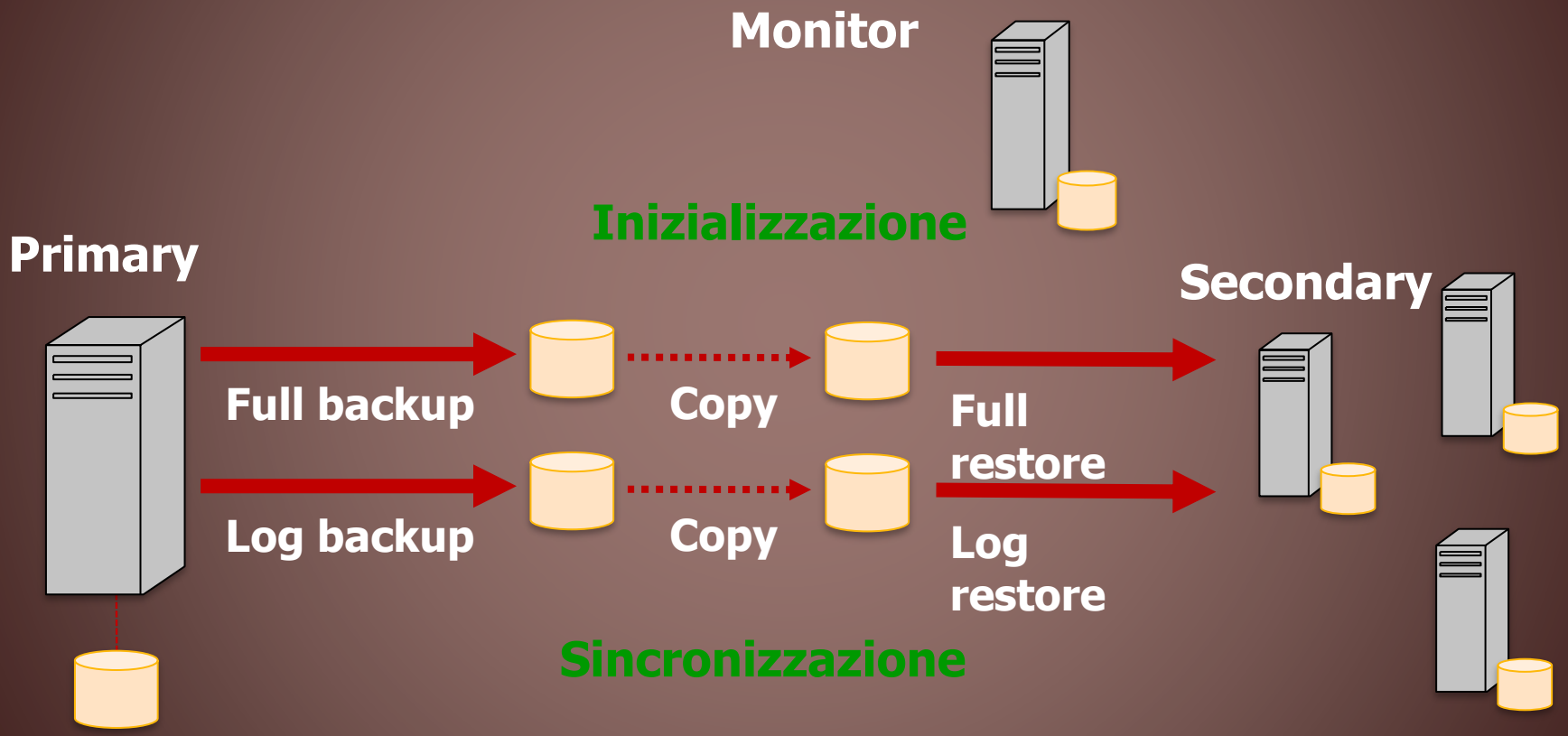


Event Publishing

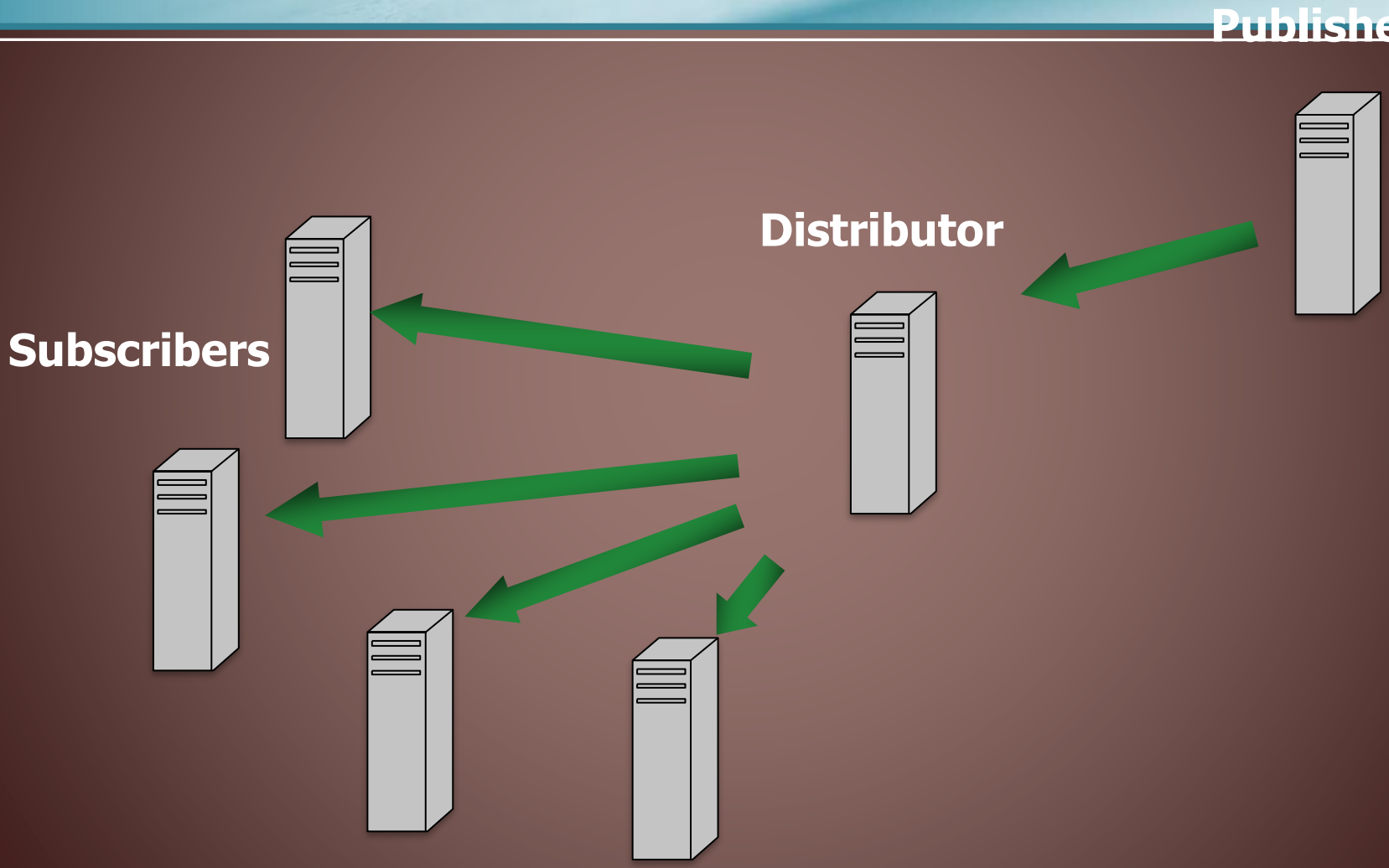
- Dal punto di vista concettuale è una replica senza apply



Eseguire repliche



Altre modalità



Part 3 - IBM replication technologies

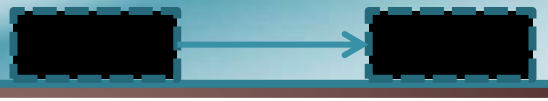
IBM Replication Technologies

- Replication **topologies**
- Replication **modes**
- **Transformational** capabilities

IBM Replication topologies

IBM replication topologies

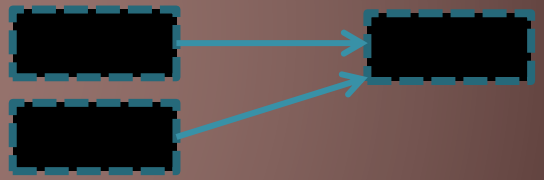
1. Unidirectional



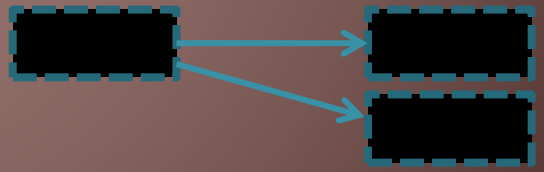
2. Cascading replication

3. Bidirectional replication

4. Consolidation replication

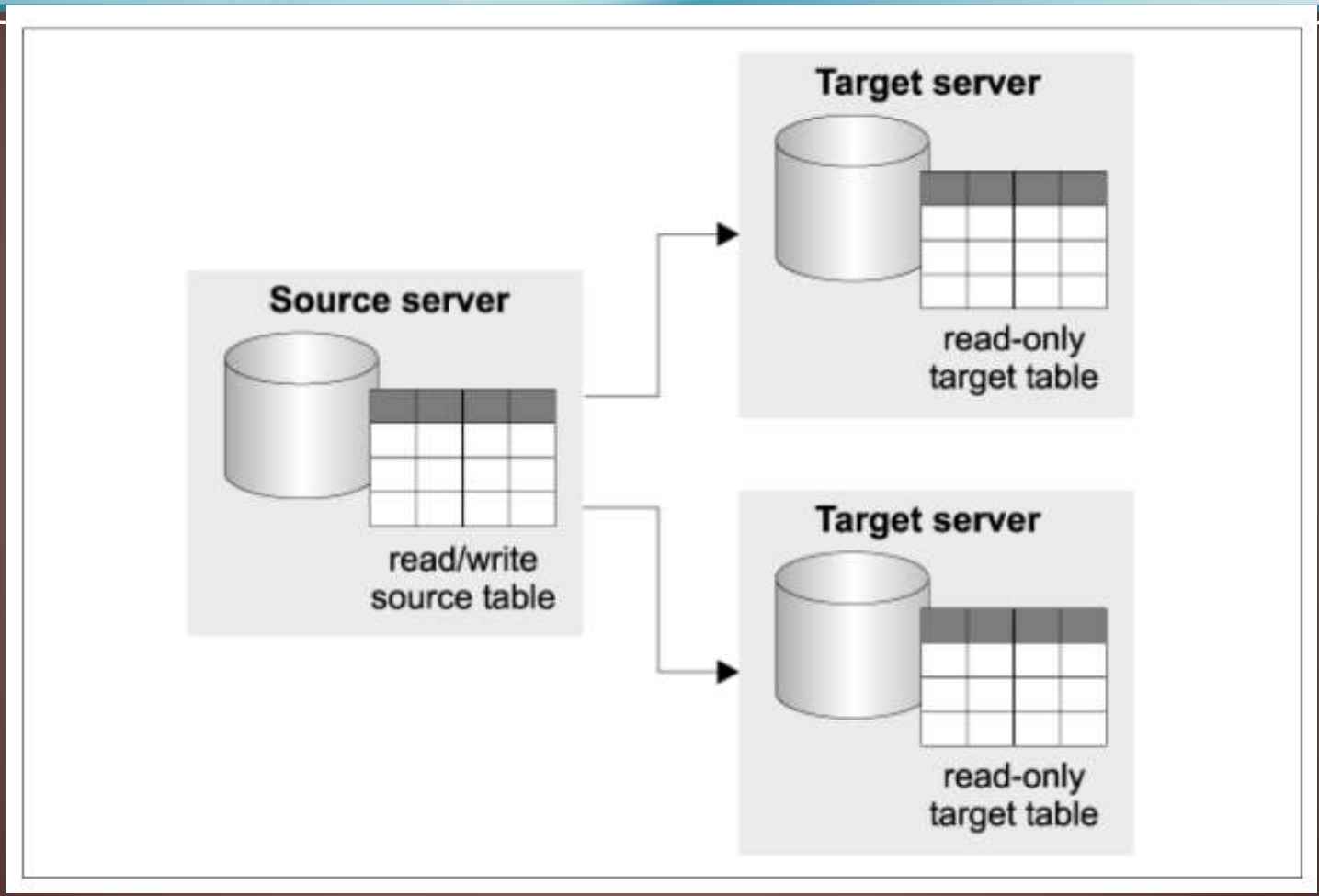


5. Data distribution



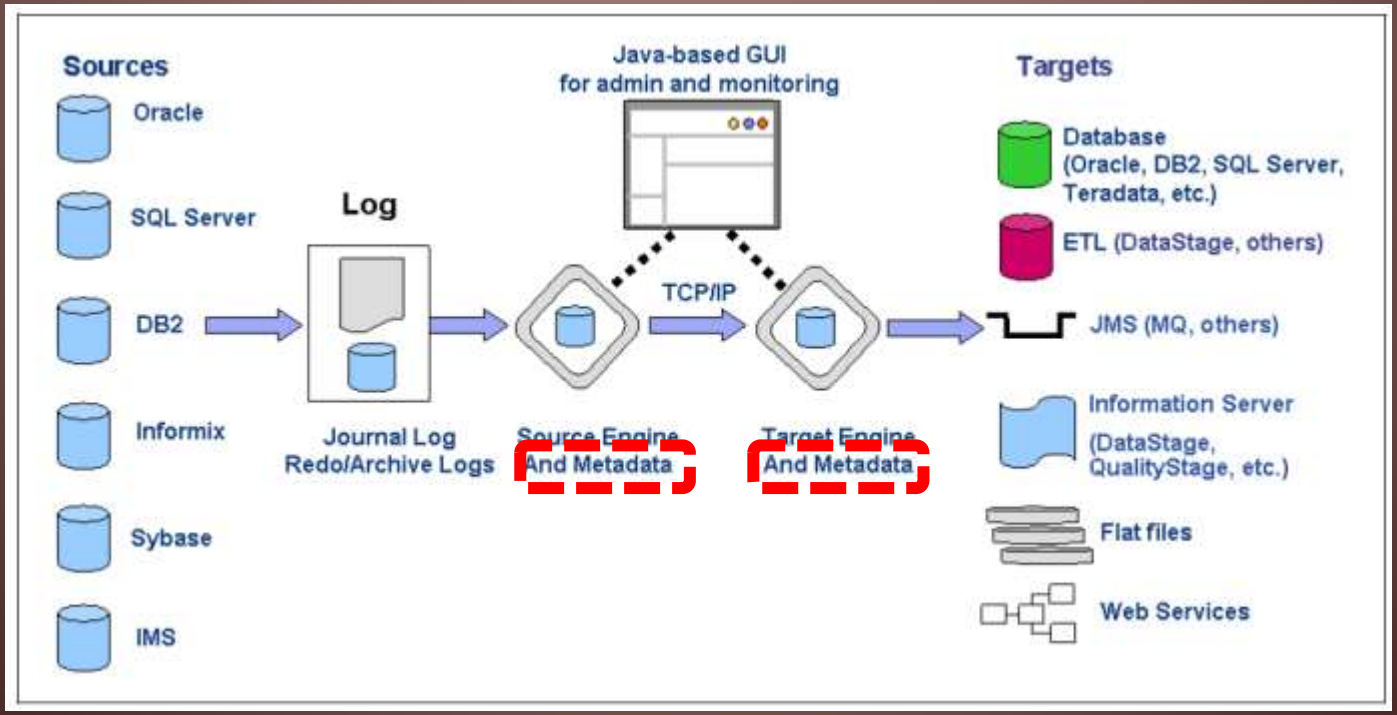
6. Hub-and-Spoke replication with propagation

1. Unidirectional replication



1. Unidirectional replication

- Unidirectional replication is the movement of data in one direction from the source tables to the target tables, and is used for data redundancy and synchronization

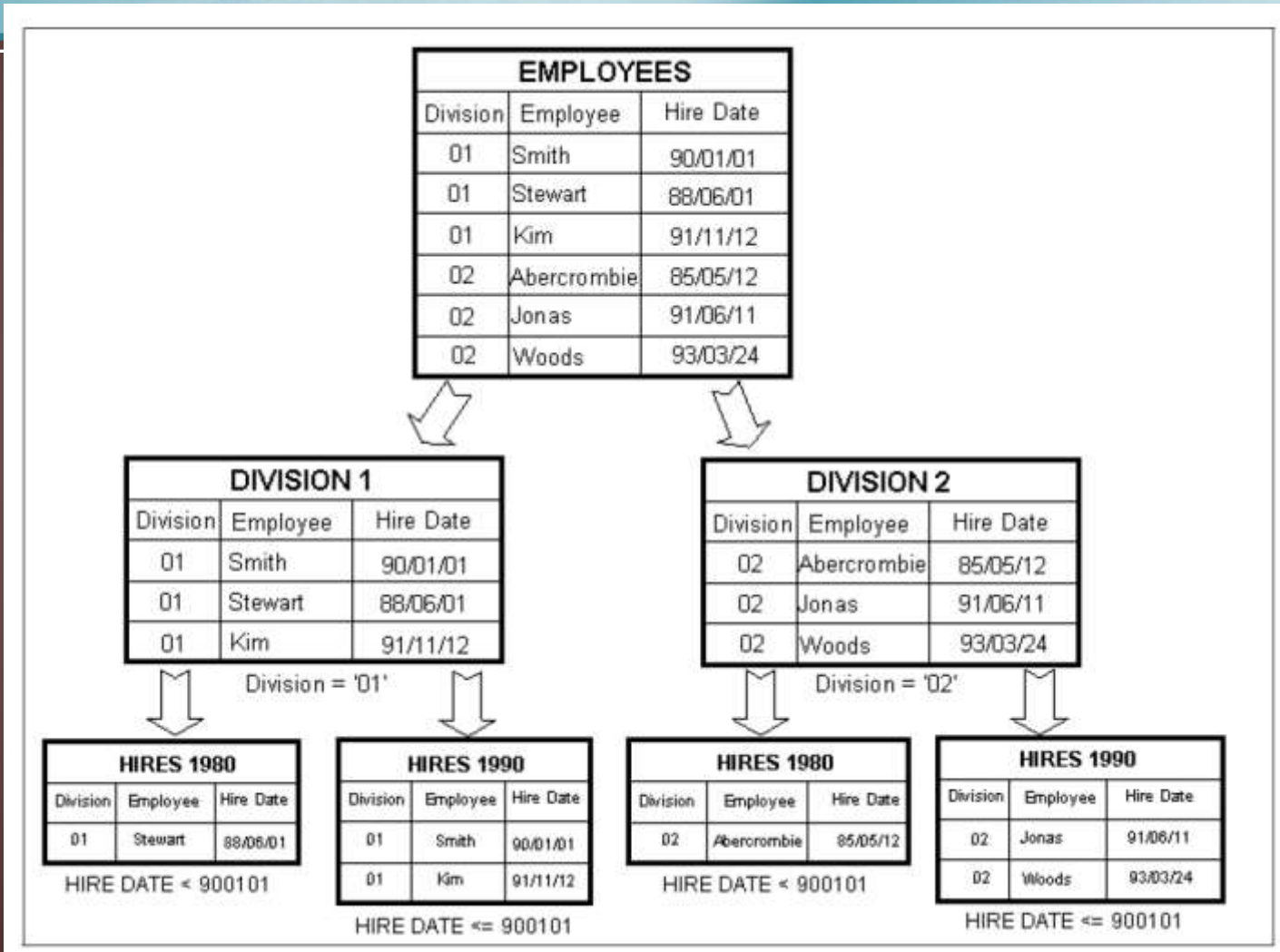


Two mechanisms

1. **Custom:** Map tables individually when you want to map **only one source table to one target table** at a time. These tables are source tables that **might not share** a table structure or similar table names as the target tables. This option is the option to map each source table in a one at a time fashion.

2. **One-to-one:** Map tables using one-to-one replication when you want to map multiple source tables to **multiple target tables** at a time. These tables **share a table structure** and similar table names. The Map Tables wizard automatically maps tables based on an example mapping you define and set up.

2. Cascading replication - 1

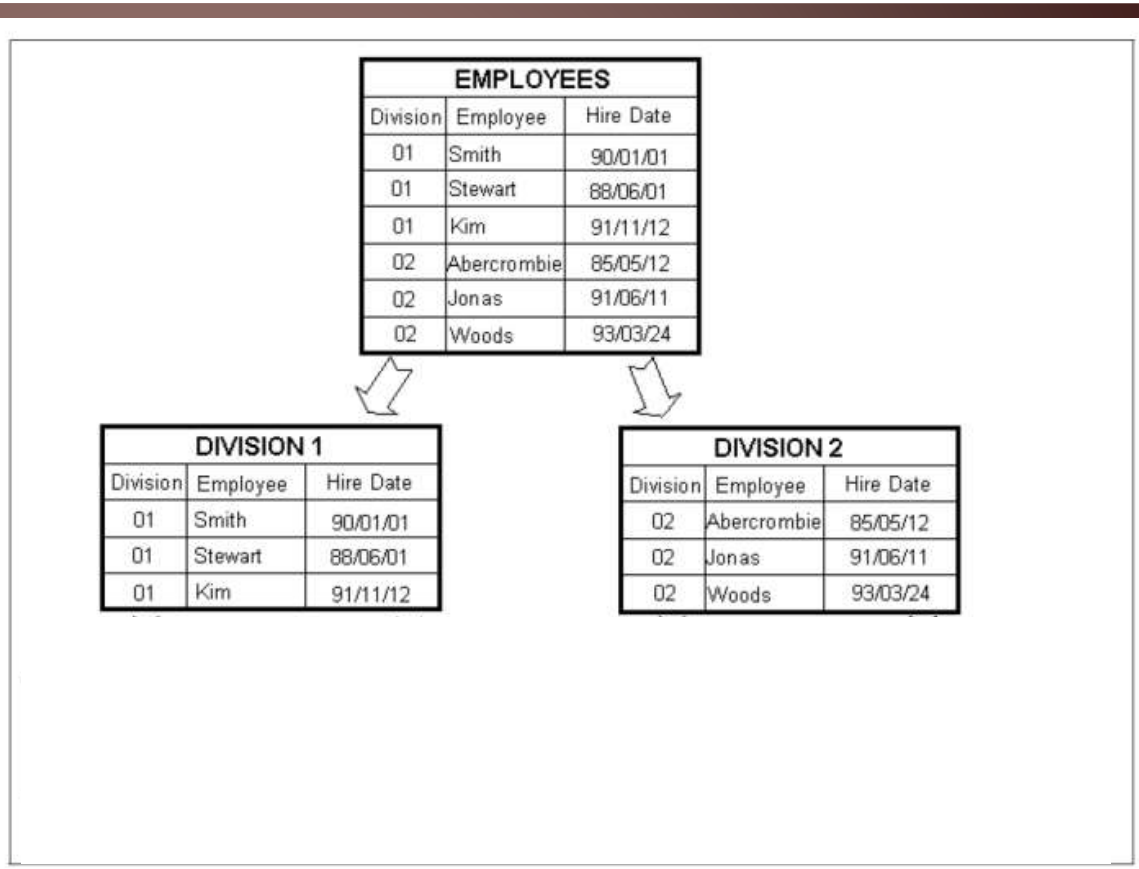


Cascading replication - 2

- Cascading replication involves a source system transmitting data to a target system which, in turn, serves as a source for the next system in the replication chain
- Cascading integration **streamlines the integration process** by enabling organizations to select regional cascade points.

Cascading replication - 3

- You can use this type of replication to **distribute changes across many servers** using a multiplier effect.
- In Figure →, Employee data (EMPLOYEES) is replicated to two separate tables (DIVISION1 and DIVISION2), where each table contains data about employees **in a specific division**.



EMPLOYEES

Division	Employee	Hire Date
01	Smith	90/01/01
01	Stewart	88/06/01
01	Kim	91/11/12
02	Abercrombie	85/05/12
02	Jonas	91/06/11
02	Woods	93/03/24

DIVISION 1

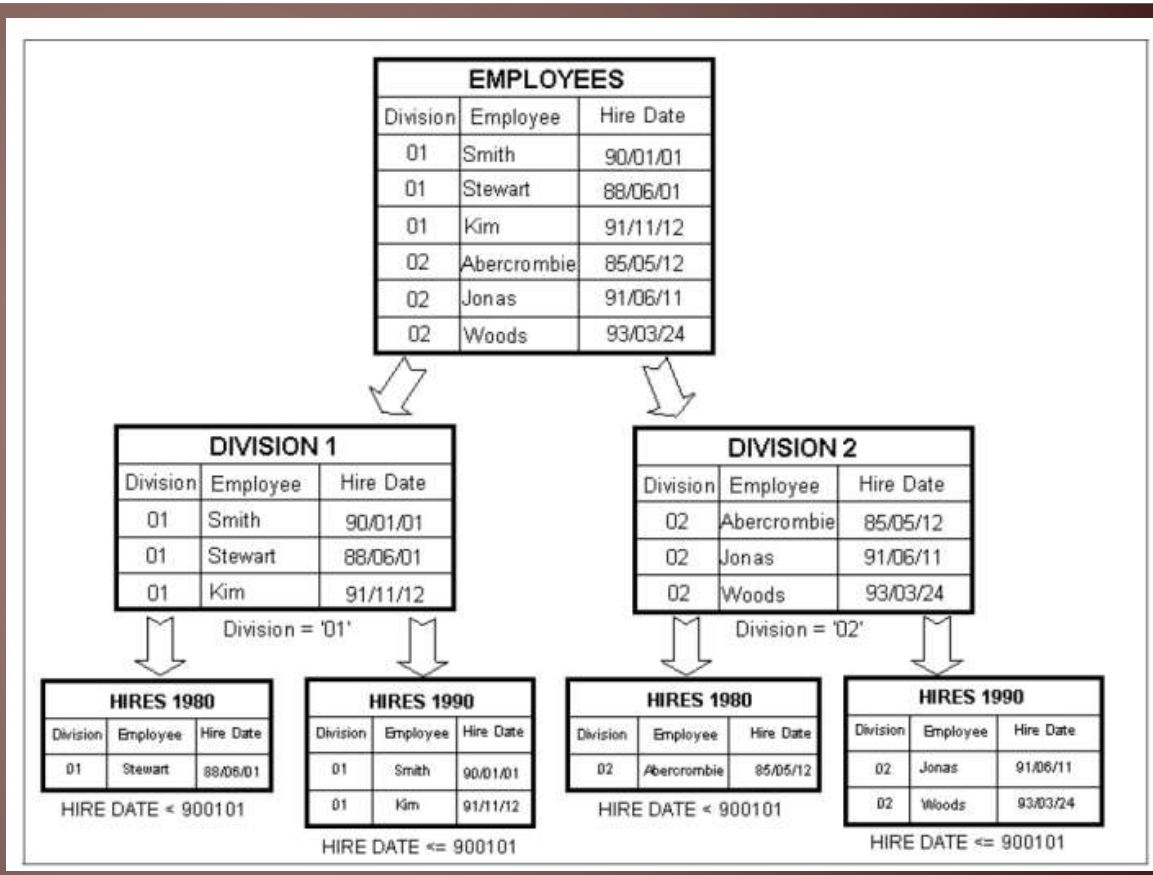
Division	Employee	Hire Date
01	Smith	90/01/01
01	Stewart	88/06/01
01	Kim	91/11/12

DIVISION 2

Division	Employee	Hire Date
02	Abercrombie	85/05/12
02	Jonas	91/06/11
02	Woods	93/03/24

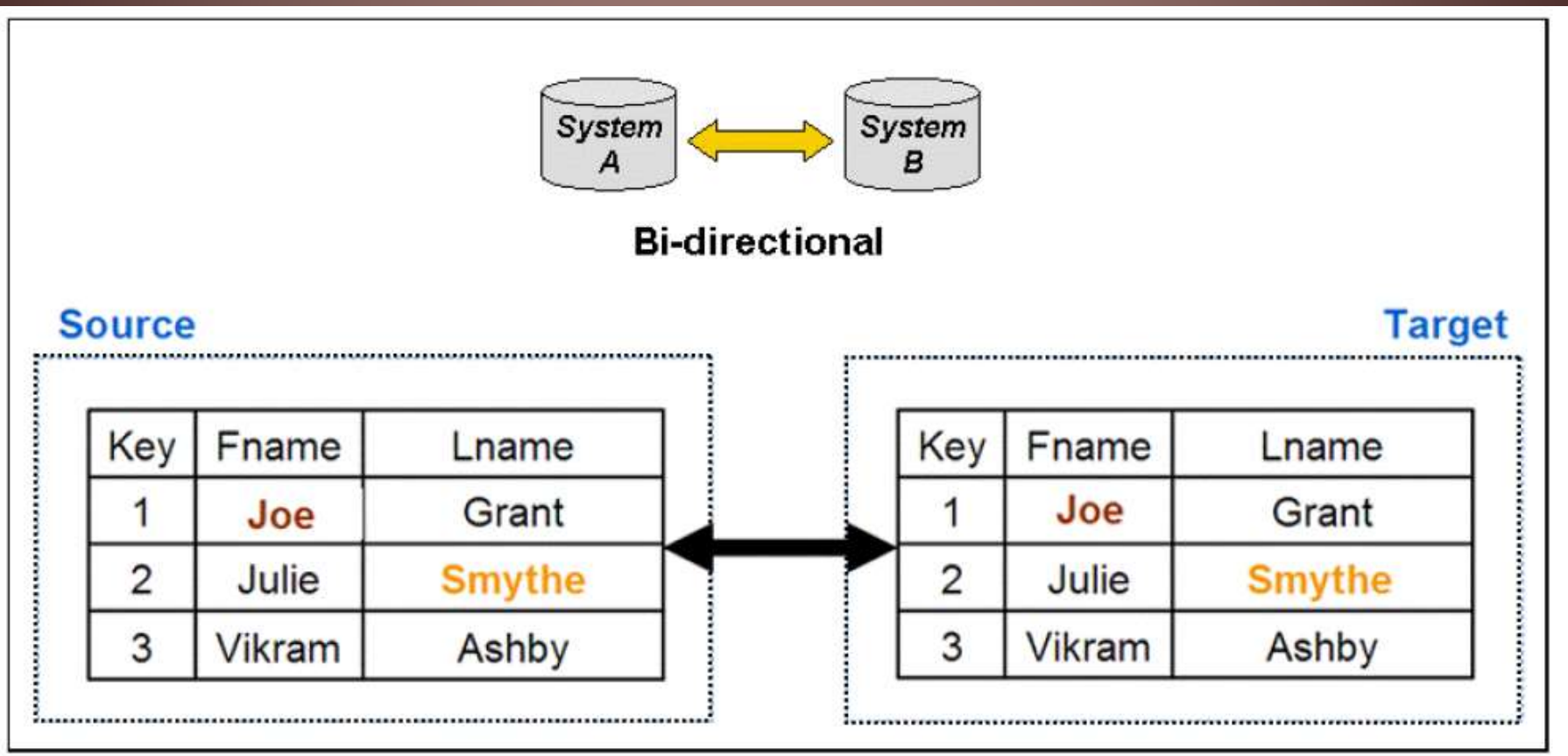
Cascading replication - 4

Data from each division table is then replicated to other tables (HIRES1980 and HIRES1990), where **the separation is based on hiring dates**.

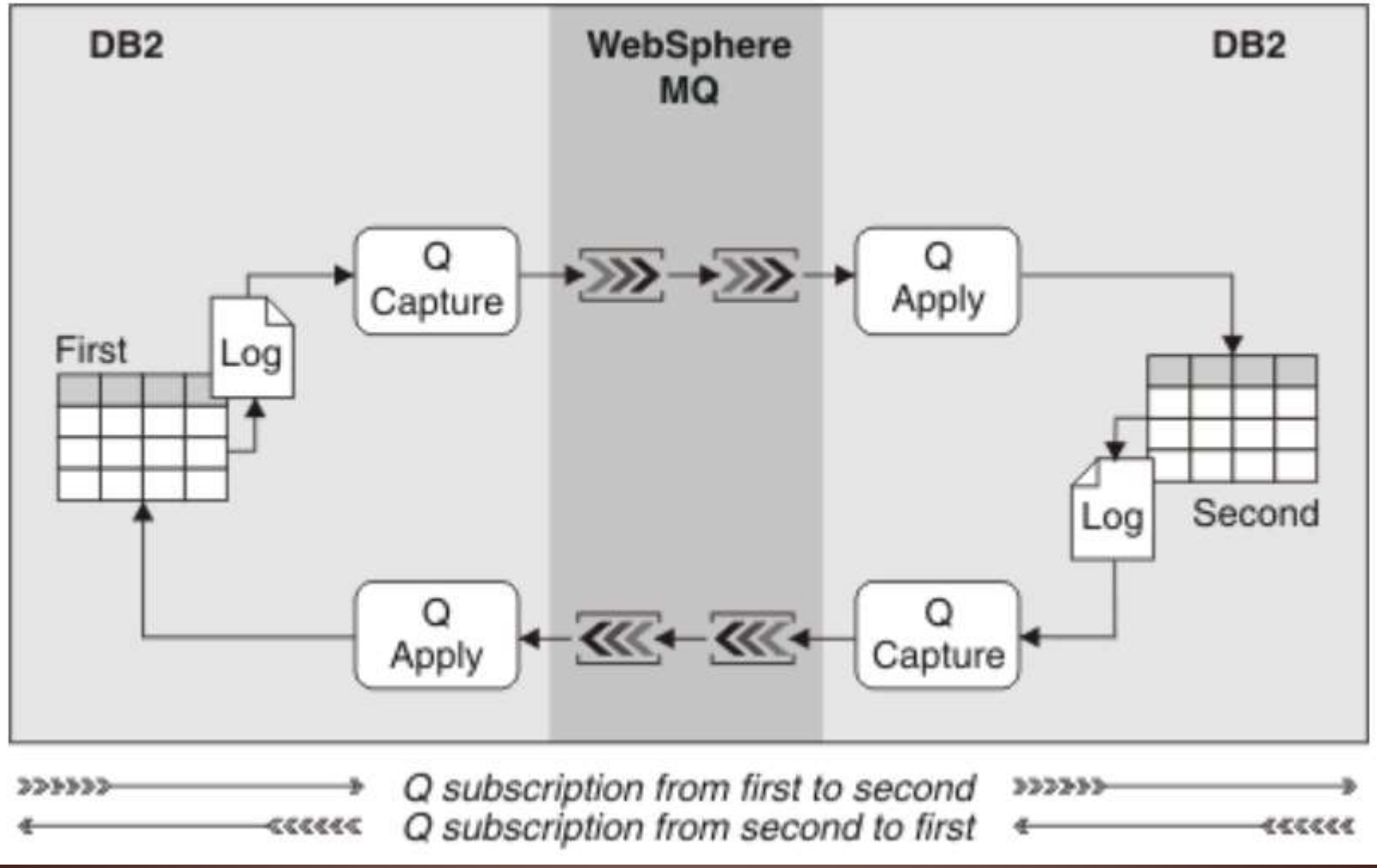


3. Bidirectional replication -1

- Bidirectional replication involves **replication from the publication server to the subscription server, and replication in the opposite direction**

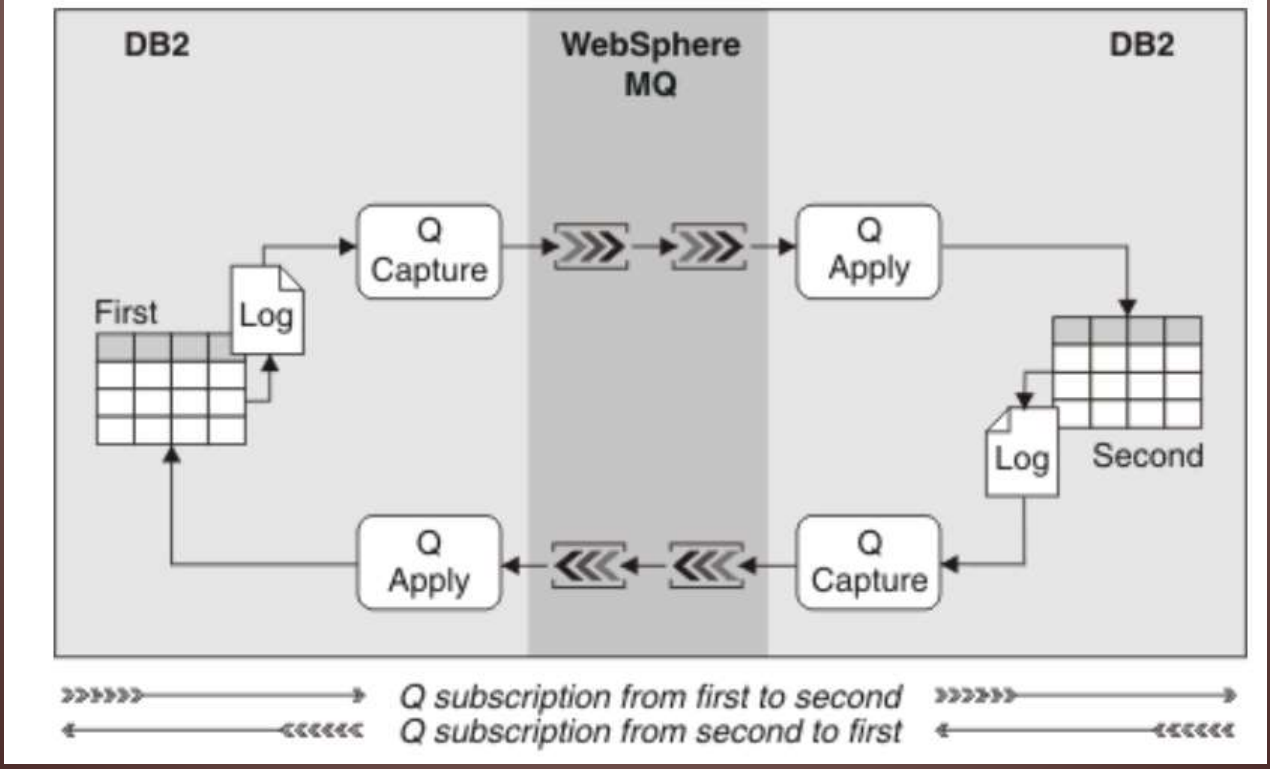


Bidirectional replication in DB2



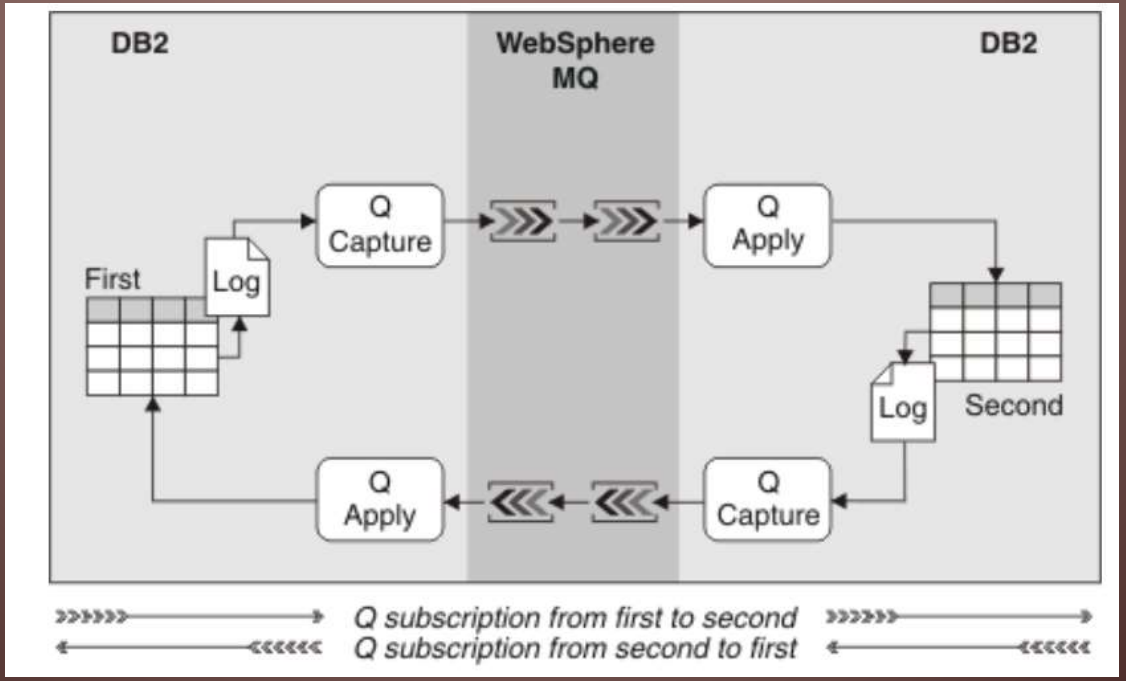
Conflict detection in bidirectional replication

- In bidirectional replication, it is possible for data that is replicated from the source table *First* in one subscription to conflict with changes made to the corresponding target table *Second* by another application.



Conflict detection in bidirectional replication

- Bidirectional replication uses data values to detect and resolve conflicts.
- You can choose which data values are used to detect conflicts. These data values can be
  - key column values only,
  - changed column values, or
  - all column values.



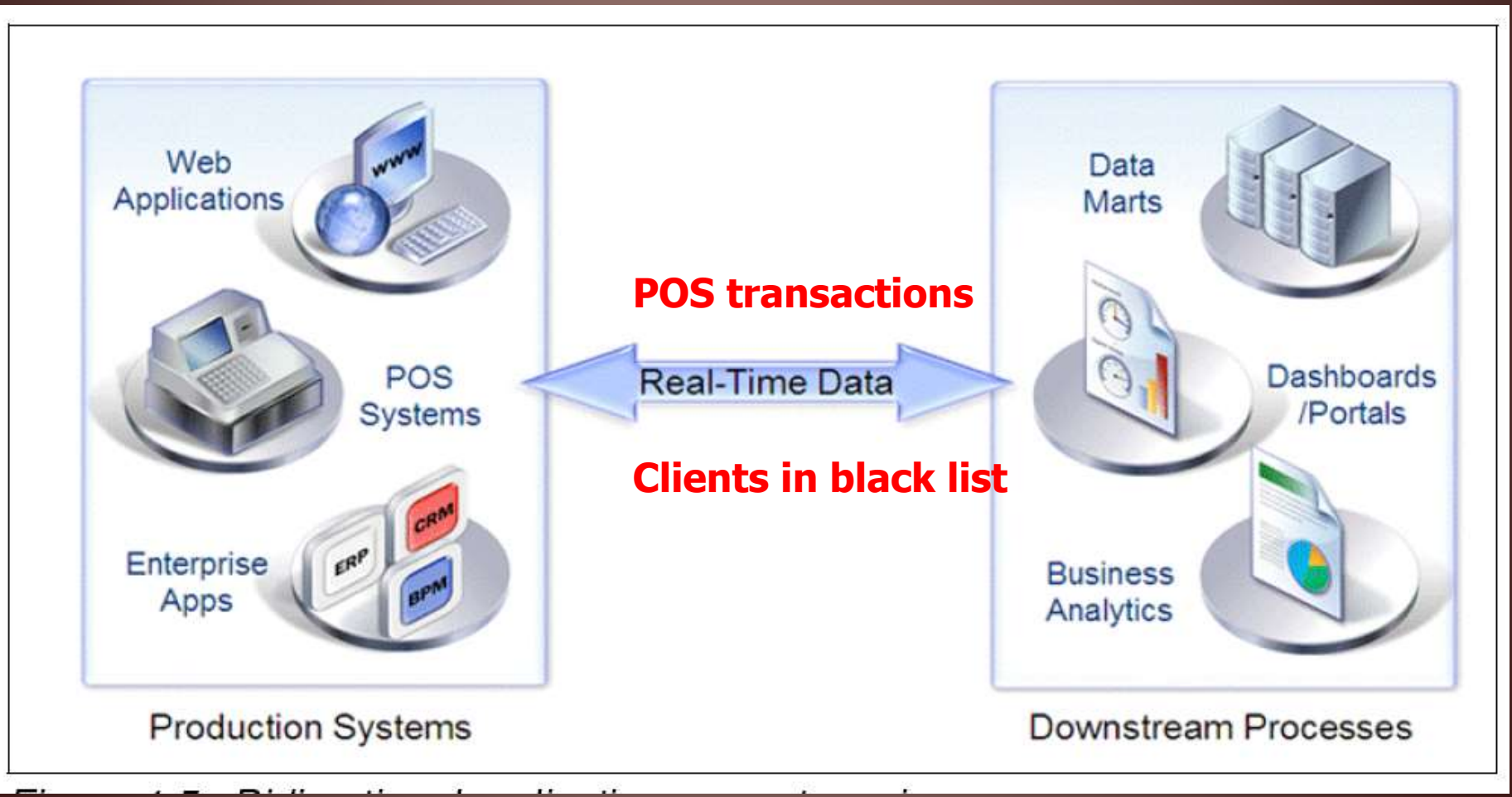
Conflict detection in bidirectional replication and system failures

- Imagine a scenario in which applications on one system make changes to tables in a server (SERVER_RED) and **that server replicates those changes to identical tables** in a server (SERVER_GREEN) on a standby system.
- The first system fails, at which time your applications start using the tables on SERVER_GREEN.

Conflict detection in bidirectional replication and system failures

- When the first system comes back online, you want to replicate changes **from SERVER_GREEN to SERVER_RED**.
- However, when the first system shut down, it could have failed to replicate some data to the second system.
- That data, which is now old, should be replaced by the data replicated from SERVER_GREEN.
- When you replicate the new data, the program for SERVER_RED recognizes the conflicts and forces the changes that come from SERVER_GREEN to SERVER_RED.

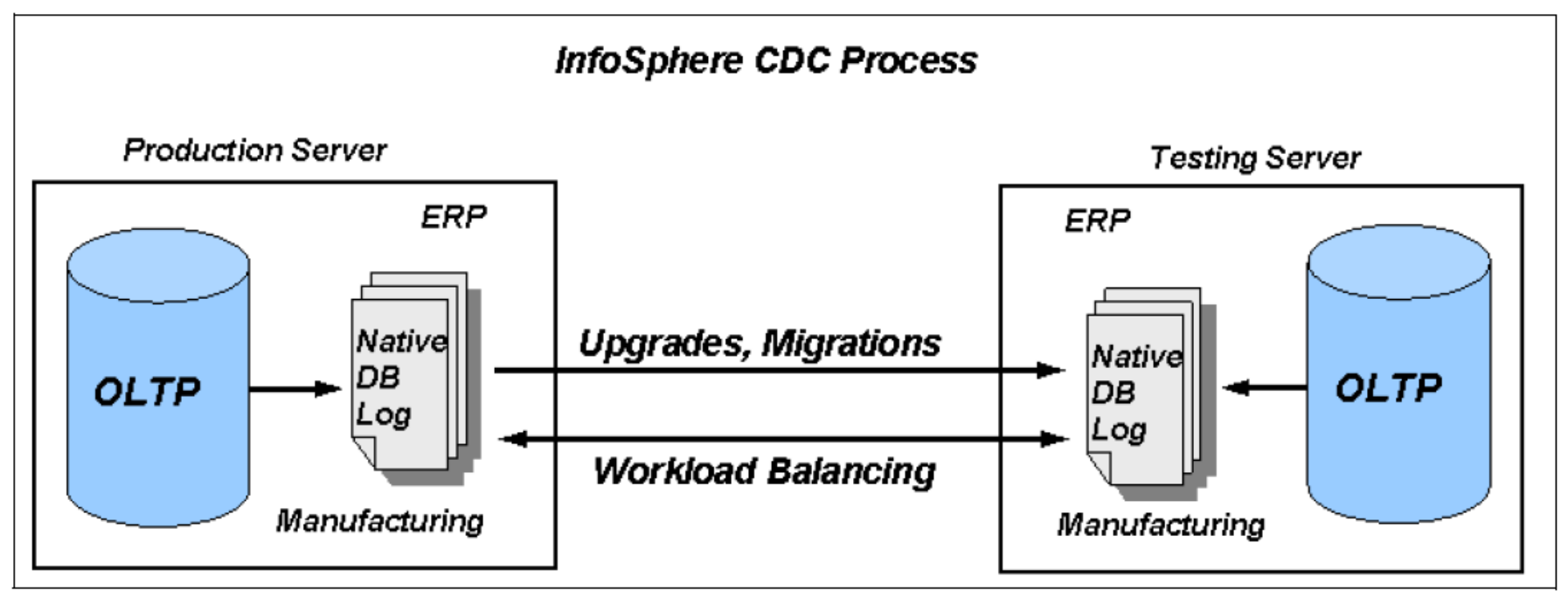
Another view of bidirectional replication from a systems and processes point of view



Other contexts of BR

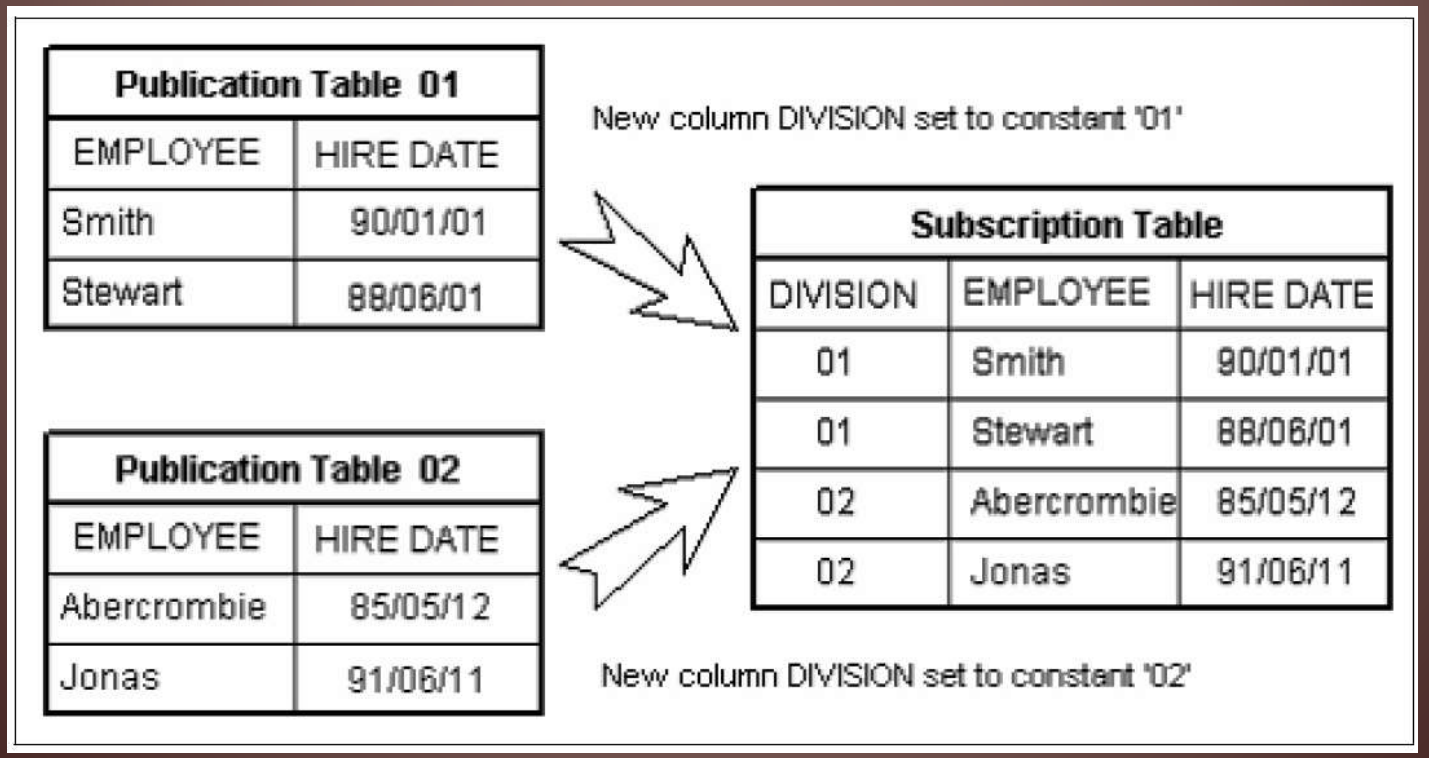
- Another excellent use for bidirectional topology is **data synchronization for upgrades and migrations**.
- This capability keeps data synchronized between the current **production server** and a **testing server**, for example, **to test a new application version upgrade** or a hardware or OS upgrade.
- The **workload balancing capability** (master to master support) allows database instances to remain synchronized where dual or double data entry is required (such as when data entry is occurring on both systems at the same time) →

Data synchronization for upgrades, migrations, and workload balancing



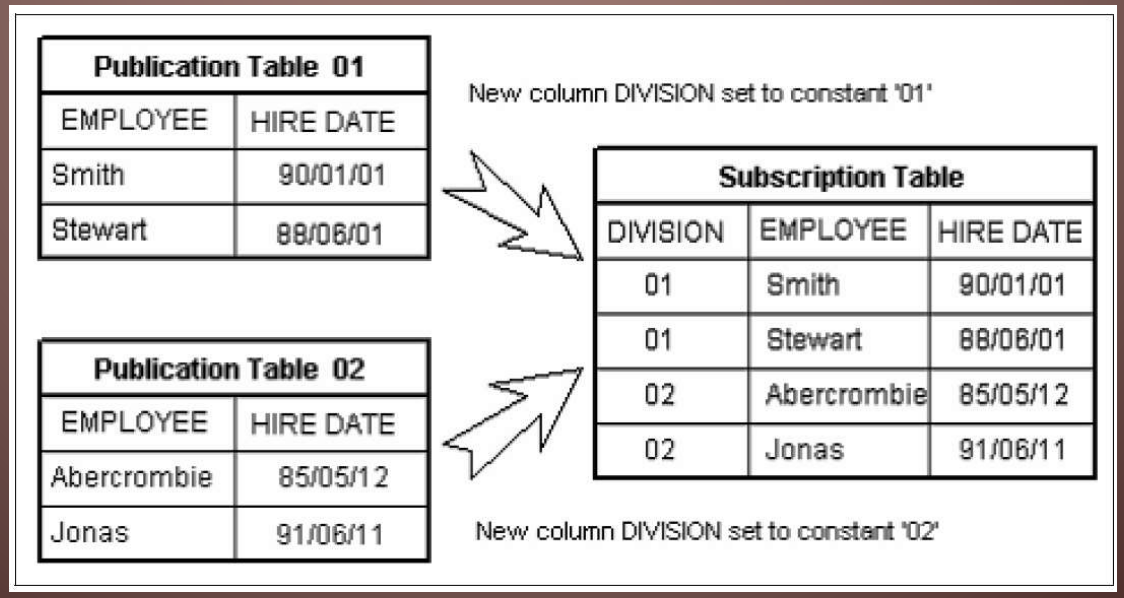
4. Consolidation replication - 1

- IBM InfoSphere also supports the implementation of consolidation replication. In this implementation, **data from multiple publication servers update a single subscription table**.



Consolidation replication - 2

- You must define each publication server separately, and then assign the publication tables to the subscription table **that serve as the data warehouse**.
- Because multiple publication tables are updating a single subscription table, **the publication tables must have the same attributes**.

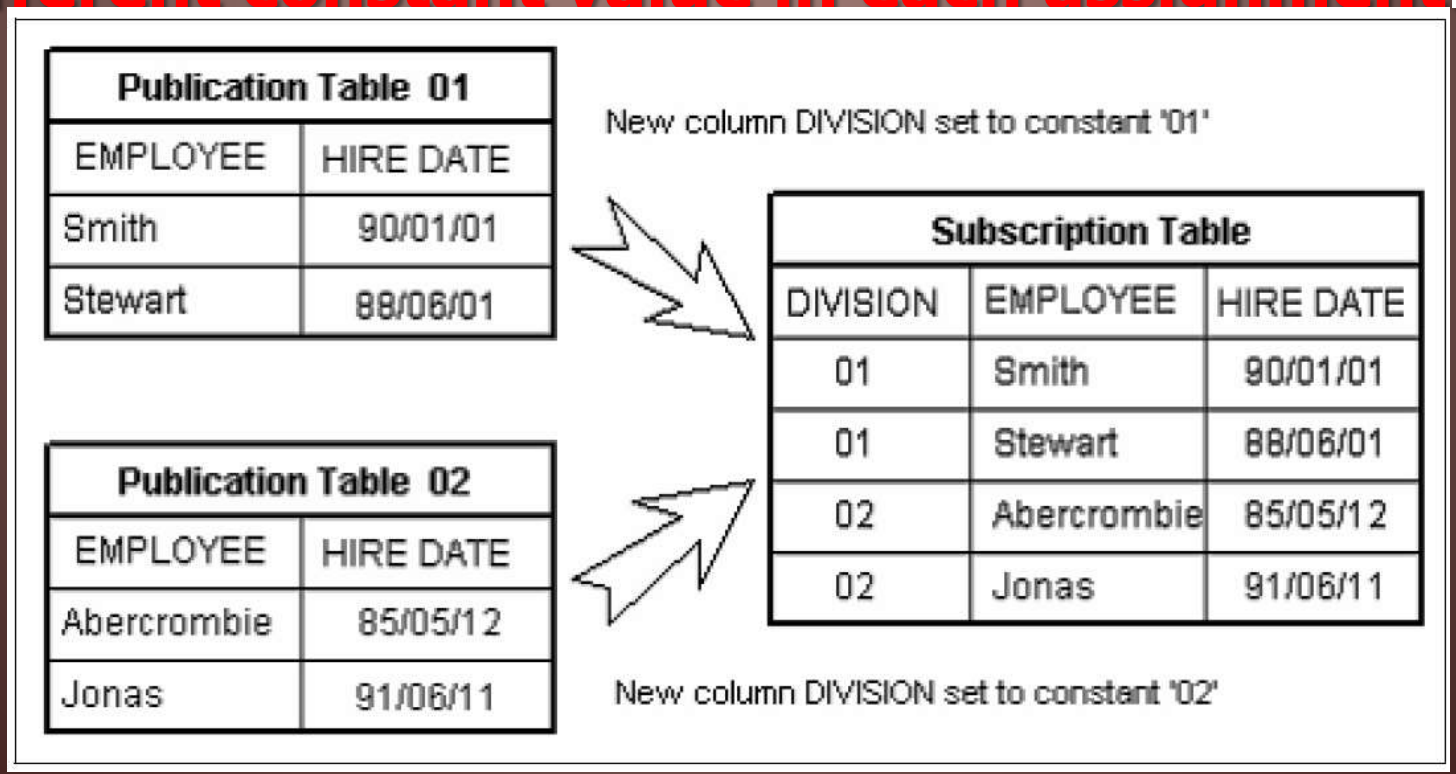


Consolidation replication - 3

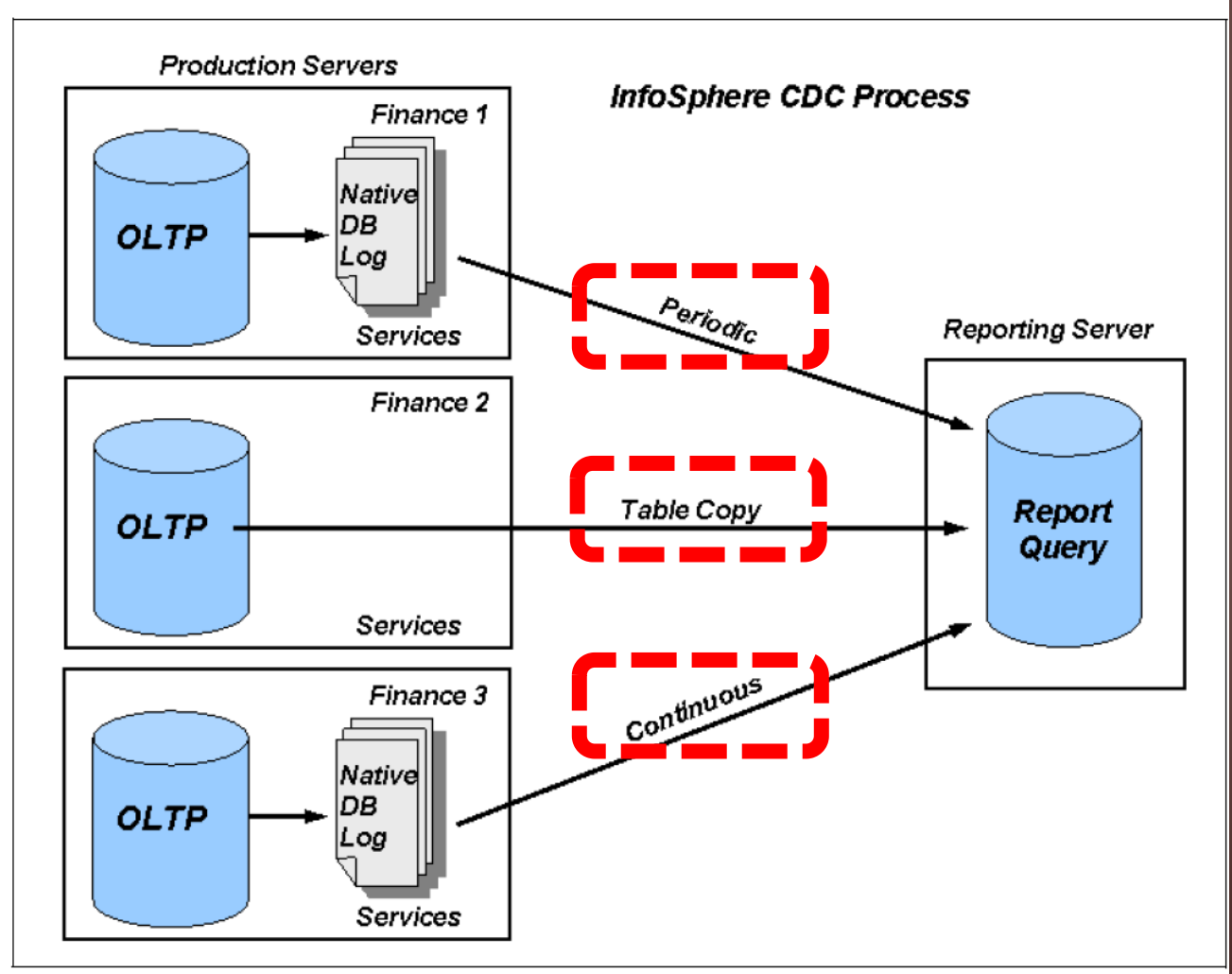
- For example, **suppose that you need to create a data warehouse to consolidate employee records created and used by two separate divisions**.
- Each of these divisions has **a publication table that is maintained separately**.
- A **subscription table is used to consolidate** the data from these two tables.
- On each publication server, you define the publication tables **to be mirrored**, and then transfer the publication table definitions to the subscription.

Consolidation replication - 4

- When defining the subscription table, **a new column** is added for the division data.
- After assigning the publication tables to the subscription table, **map the new column to a different constant value in each assignment**.



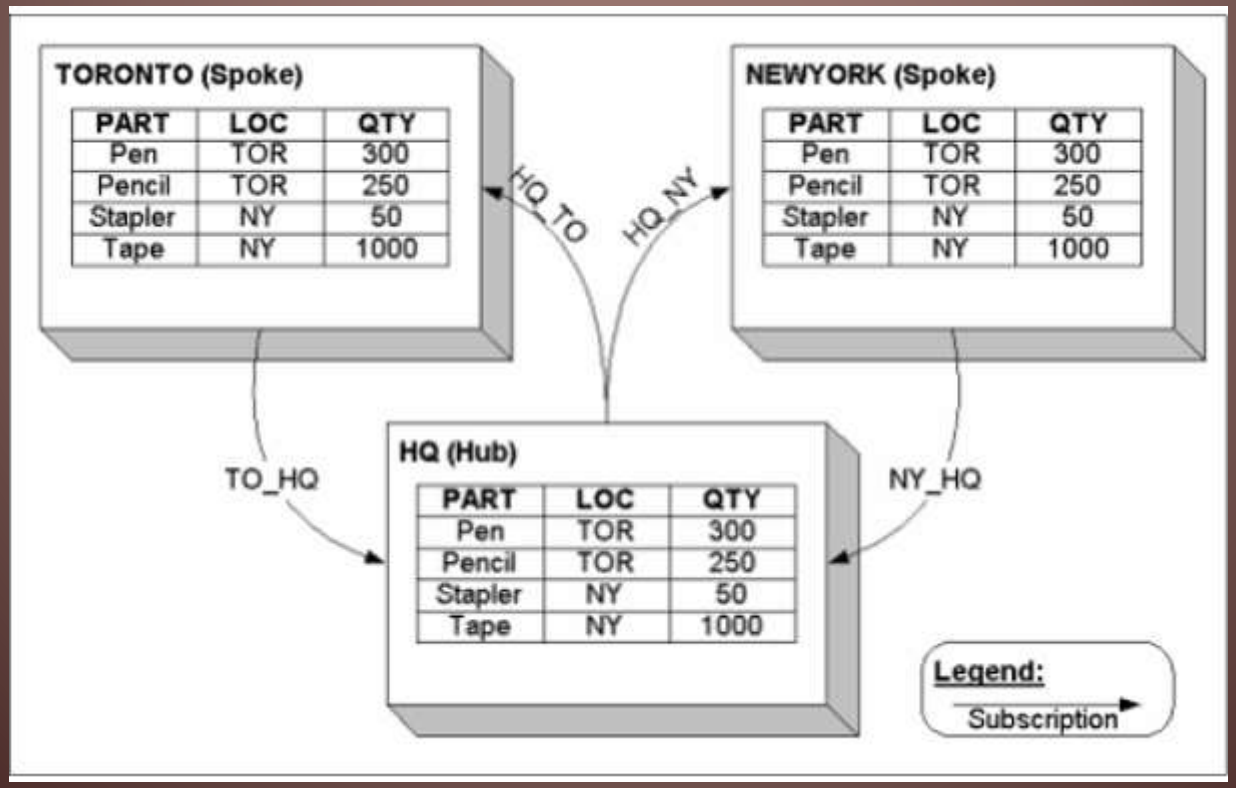
Replication frequency generally varies from continuous (near real time) to periodic. In addition to **log based change data capture, table level refresh or copy** can be used



We skip 5. Data Distribution since we addressed this topic before in the course

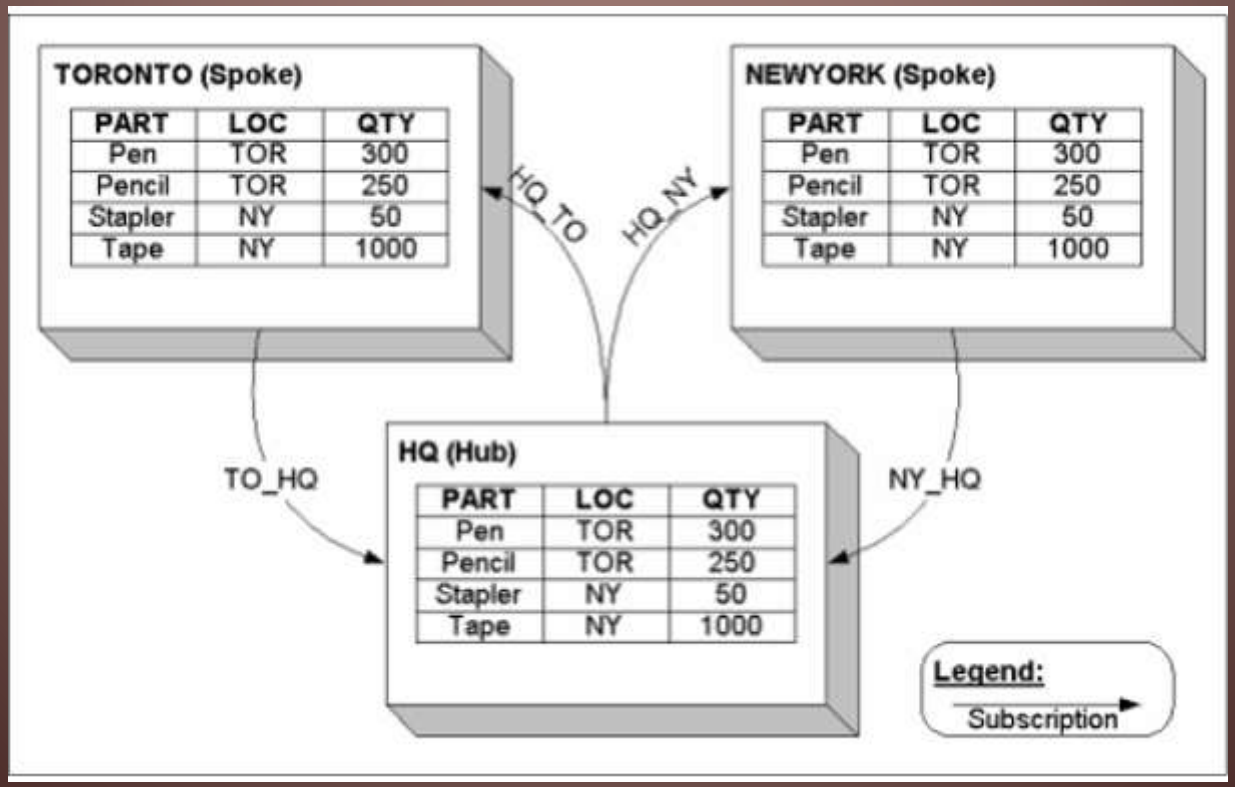
6. Hub-and-Spoke replication - 1

- Hub-and-Spoke replication requires a configuration consisting of **centrally-administered tables on a hub server** that are **simultaneously maintained on multiple subscription (spoke) servers**



Hub-and-Spoke replication - 2

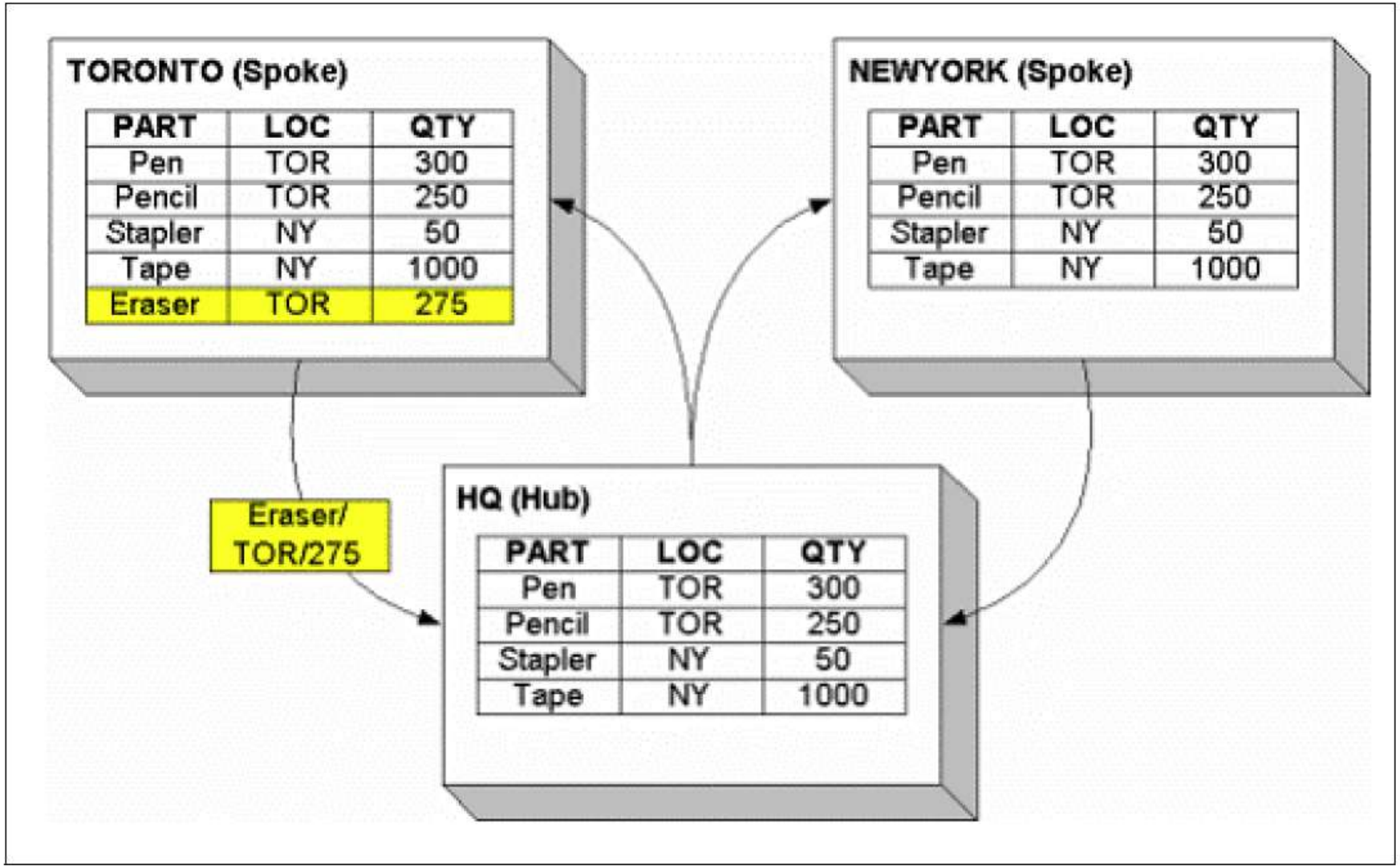
- **Changes applied to the tables on subscription servers** (in this example, TORONTO and NEWYORK) **are replicated** to the same tables on the hub server (HQ), and **then routed** to the same tables on all other subscription servers in the Hub-and-Spoke configuration



Hub-and-Spoke replication - 3

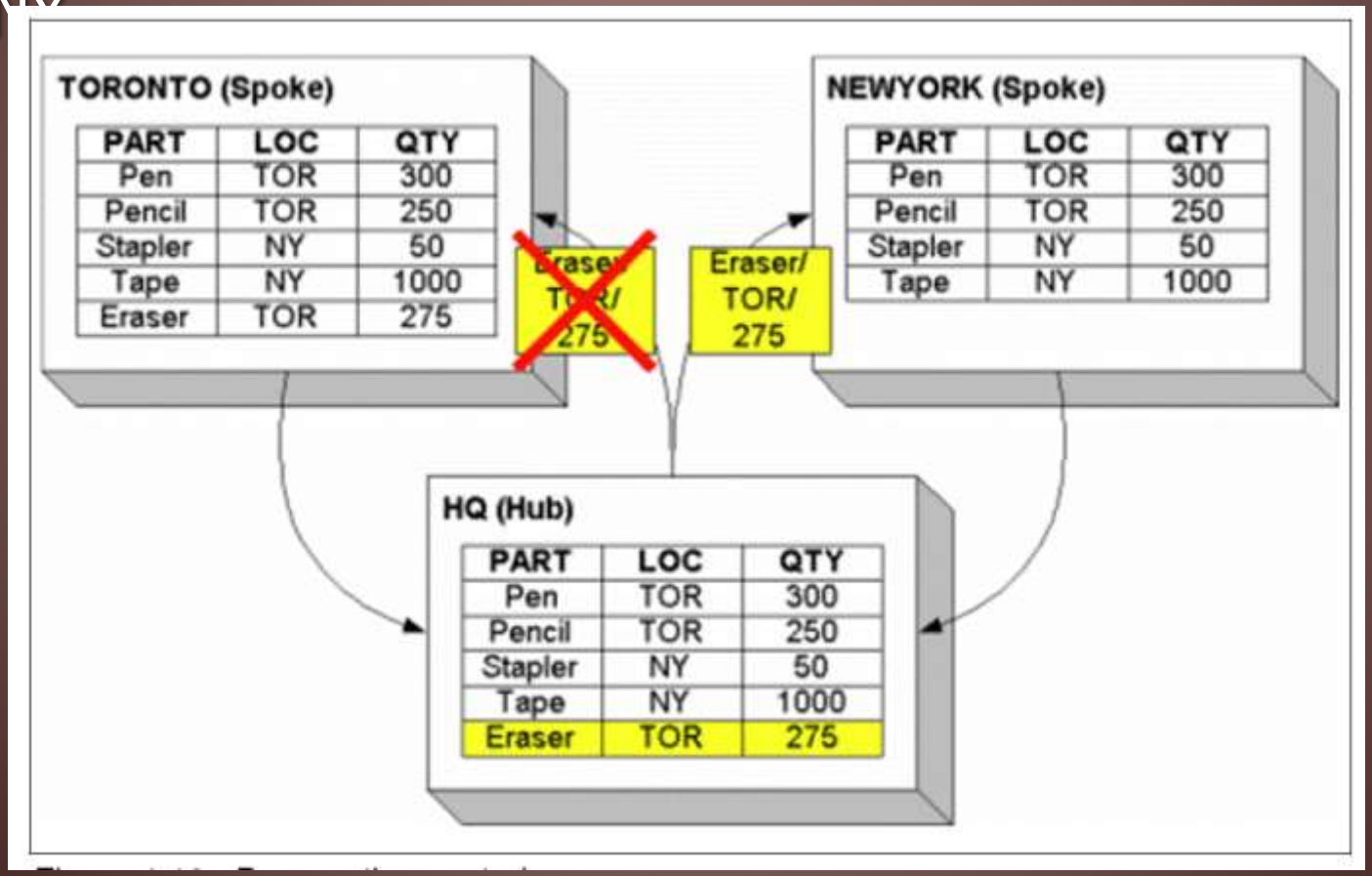
- In a Hub-and-Spoke configuration, it is possible for changes originating on one spoke to be replicated back to that spoke recursively.
- To prevent this situation from happening, you must configure propagation control for each hub-to-spoke subscription.
- **Propagation control** allows you **to specify the corresponding spoke-to-hub subscription** when defining **the hub-to-spoke subscription** →

Figure shows a change on the TORONTO spoke being replicated along the spoke-to-hub subscription to the HQ hub



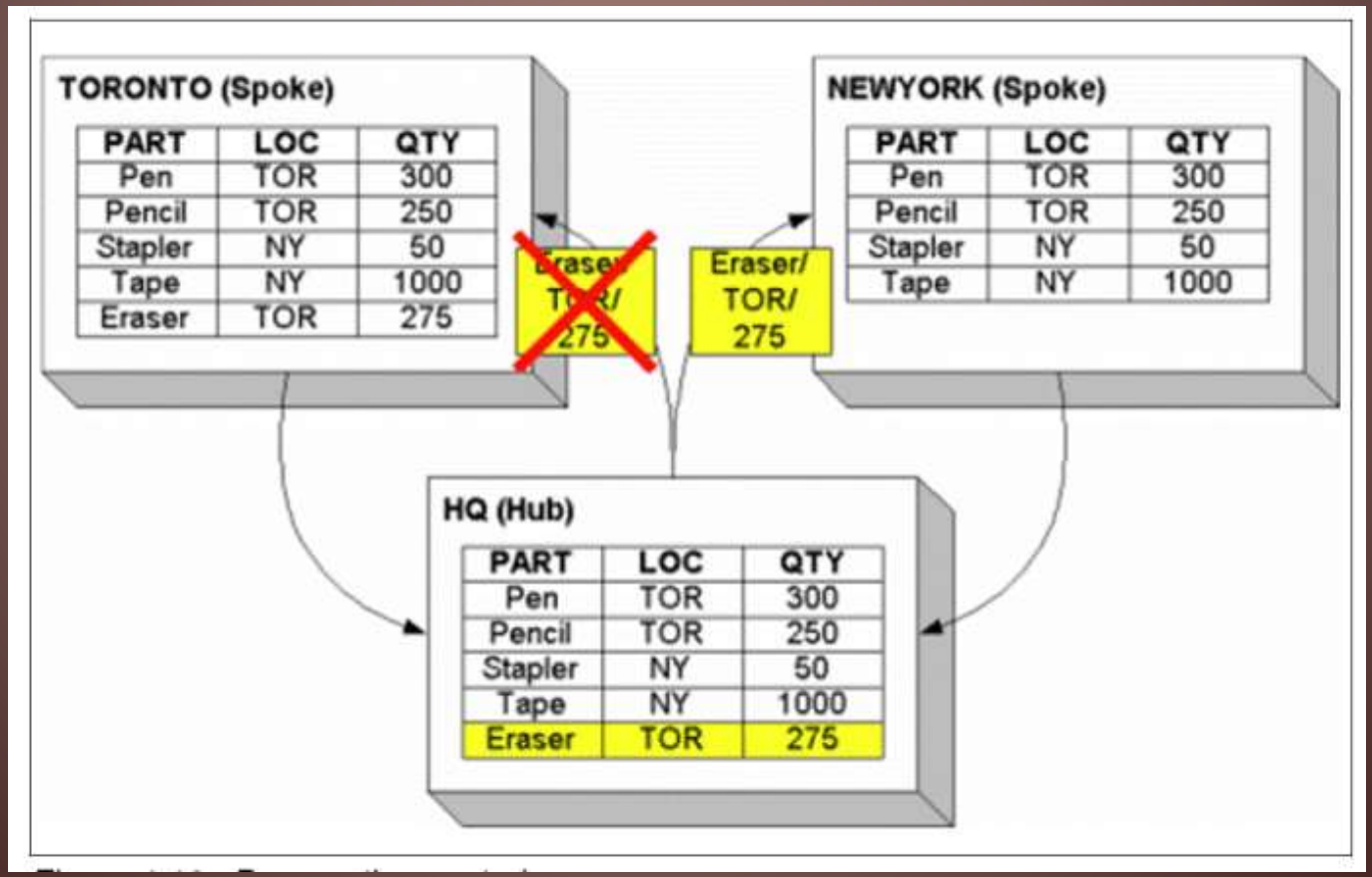
Hub-and-Spoke replication - 4

- After applying the change to the HQ hub, the hub server then needs to replicate the change to its spokes. There are two subscriptions that replicate data from the hub, HQ_TO and HQ_NY



Hub-and-Spoke replication - 5

- When defining these subscriptions, **you must declare that you do not want HQ_TO to replicate changes** that were originally applied to the hub by the TO_HQ subscription. The same is true for the HQ_NY and NY_HQ subscriptions.



E-commerce application synchronization

- A good example to use for this topology is for **e-commerce application synchronization**.
- Here the topology provides **continuous bidirectional synchronization** between **web-based applications** and **mission-critical business applications**.
- It also helps organizations **improve customer online shopping experience** with **improved visibility into inventory and customer shopping activities**.

IBM **Modes** of replication

1. Refresh →
2. Mirroring →
3. Continuous
4. Scheduled End (Net Change)

1. Refresh

- The IBM InfoSphere CDC **refresh operation** is the replication mode used to **capture a complete copy of the data in the source table and transfer that data to the target table**.
- Each refresh **applies all features of replication** such as
  - row and column filtering,
  - column mappings,
  - and so on

during the transfer.

Types of refresh
Standard and differential refresh

- A **standard** refresh results in a complete copy of the data in a source table being sent to the target table. This type of refresh is typically performed to bring the entire source and target tables into synchronization.
- A **differential** refresh updates the target table **by applying only the differences** between it and the source table. This type of refresh is typically performed when the target table is already synchronized with the source table.

Differential refresh

- A differential refresh updates the target table by applying only the differences between it and the source table.
- Instead of the target table being cleared at the start of the refresh and repopulated with data, the differential refresh compares each row in the target table with each row in the source table to identify missing, changed, or additional rows.
- The **primary advantage** of the differential refresh is that **the target table stays online** during the refresh operation.

Types of differential refresh

1. **Refresh Only**: Performs a differential refresh by changing any target rows that differ from the source rows.

Types of differential refresh

2. Refresh + Log Differences: Performs a differential refresh and also creates a log table in the target replication engine metadata to track all changes during the refresh.

- The log table is identical to the target table, **with the addition of a column to indicate** the actions taken during the refresh, such as

- inserting a row,
- deleting a row, or
- updating a row.

Types of differential refresh

3. Only Log Differences: Creates and populates a log table **in the target replication engine metadata** to identify all differences between the source and target tables. The target table is not updated.

- This method allows you to evaluate what the differences are between the target and the source.
- If you then decide to refresh the table, you can go back to the subscription and select Refresh Only to update the target table or update the target table manually **based on the contents of the log table**.

Referential Integrity

- **If Referential Integrity (RI) is in effect on the target tables** being replicated to, it is important to set the refresh order.
- If the tables are refreshed in the incorrect order, **database failures can result** when InfoSphere CDC tries to rebuild the foreign key constraints. It is also important to make sure all tables that make up the RI are part of the same subscription.

Exercise (in *italics* Referential integrities)

Source

- Table A (A1, A2, A3)
- Table B (B1, B2, *A1*)
- Table C (C1, C2, *B1*)

Target

- Table A1 (A1, A2, A3)
- Table B1 (B1, B2, *A1*)
- Table C1 (C1, C2, *B1*)

Question - In a refresh (standard or differential) operating on A,B,C to be replicated into A1, B1, C1, which are the correct orders achieving RIs among

1. A, B, C
2. A, C, B
3. B, A, C
4. B, C, A
5. C, A, B
6. C, B, A

?

Hint – distinguish between insert and delete commands.

Solution

Delete command should be applied in the following order

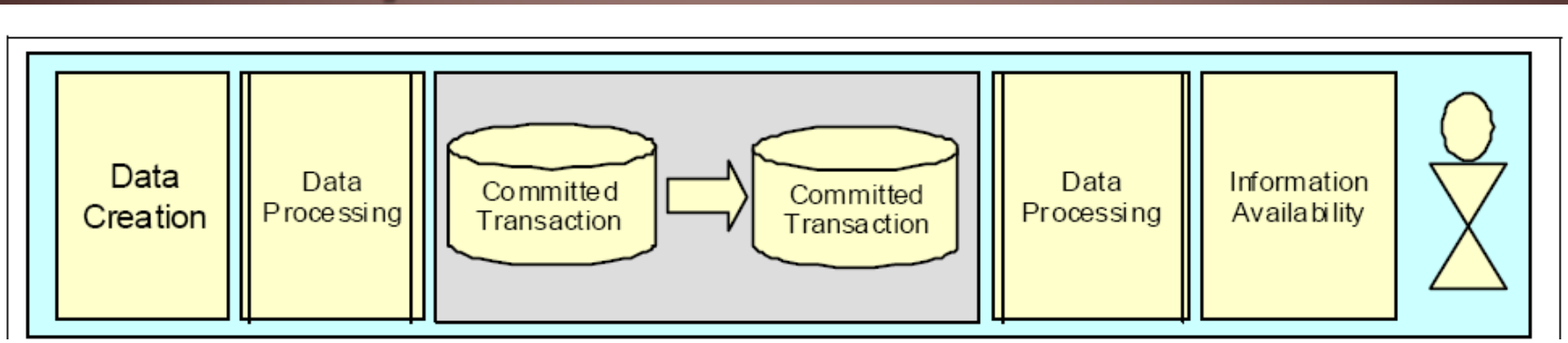
- First on table C1, then on table B1, finally on Table A1

So that first is eliminated the data that are in the left part of the referential integrity constraint and the data in the right part.

Insert commands should be applied in the reverse order A1, B1, C1.

Latency and thrughput

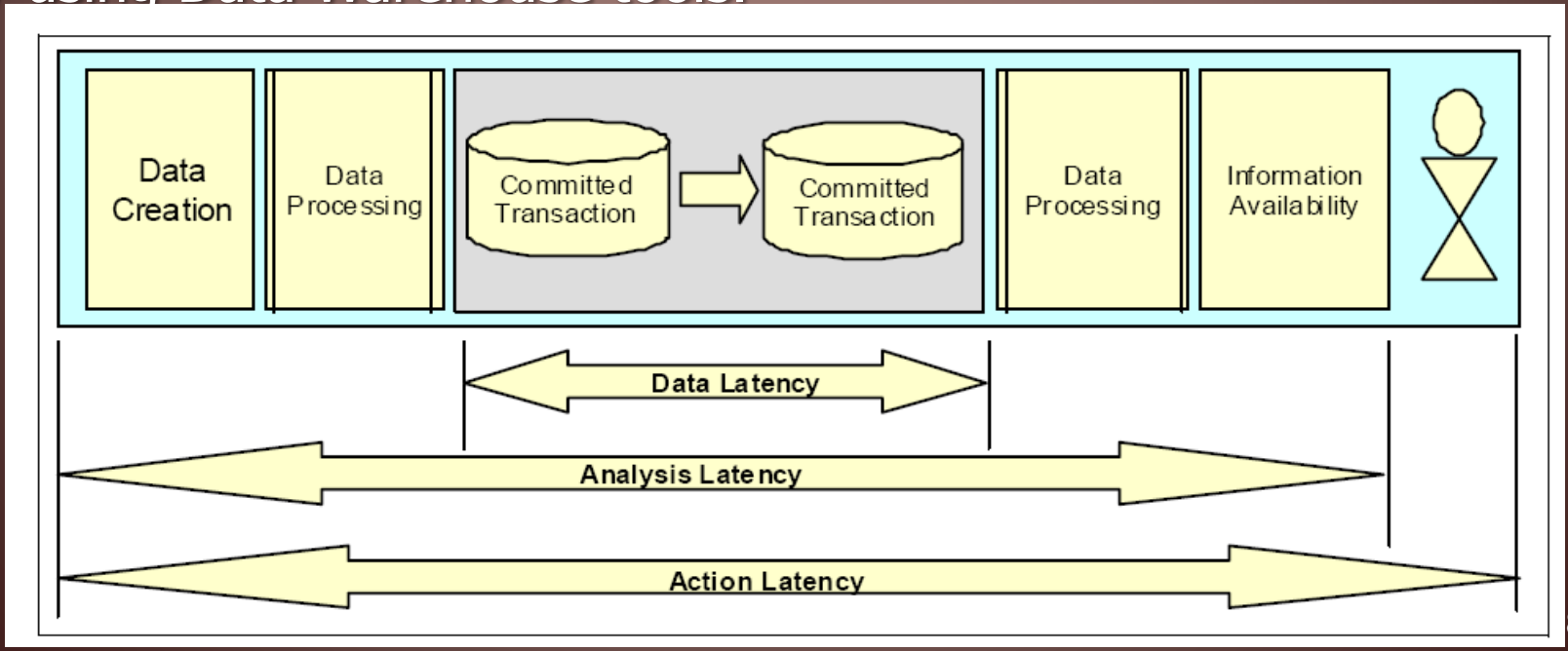
- **Latency**, sometimes referred to as replication lag, is the amount of time between an update applied to the source system and the same update applied to the target system. **The shorter the duration, the lower the latency**.



Latency and thrughput

There are three types of latency:

- **Data latency** is the time delay in data transfer from source systems to target systems
- **Analysis latency** is the time delay that includes data creation and processing (Extract Transform Load process in data warehouses)
- **Action latency** is the time delay in getting information fro data by using Data Warehouse tools.



Transformational capabilities - 1

- InfoSphere CDC provides the ability to **manipulate data** during the replication flow.
- **These transformations can be made against any or all columns** included in a single operation. They provide functionality such as
  - table joins,
  - date and string manipulation, and
  - IF / THEN / ELSE logic.

Transformational capabilities

InfoSphere CDC allows businesses to share data between heterogeneous sources and targets. This heterogeneity often involves sources and targets that have different data structures, so it might require some of the following:

- Mapping between different column names
- Conversions between data types
- Changes in data sizing
- Default values
- Creating column values through derived expressions
- Accessing other tables for additional data
- Running user written code for processing outside of InfoSphere CDC

Part 4 - Microsoft SQL Server Replication Technologies

Replication Terms

Publisher
Each Ad is an Article

Distributor

Subscriber

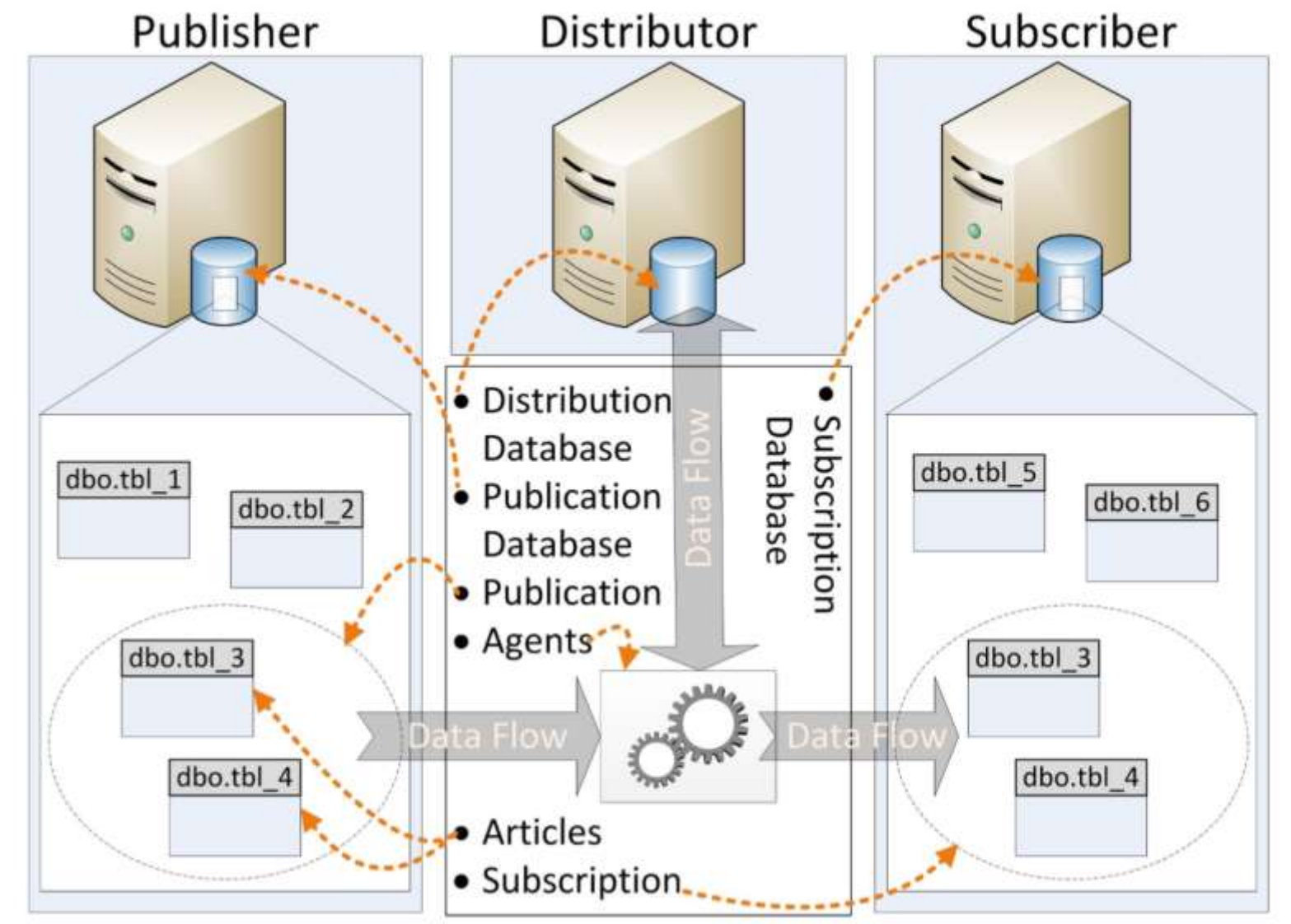
Components of a Replication setup

- Publications
- Articles
- Publisher
- Distributor
- Subscriber

Publications

- A **publication** (generalization of database) is a collection of articles (generalization of tables, files, documents) grouped together as one unit
- Every **article** is defined to be part of exactly one publication. However, you can define different articles on the same object in separate publications.
- A publication supports several **configurable options** that apply to all its articles, e.g. **the type of replication** to use.

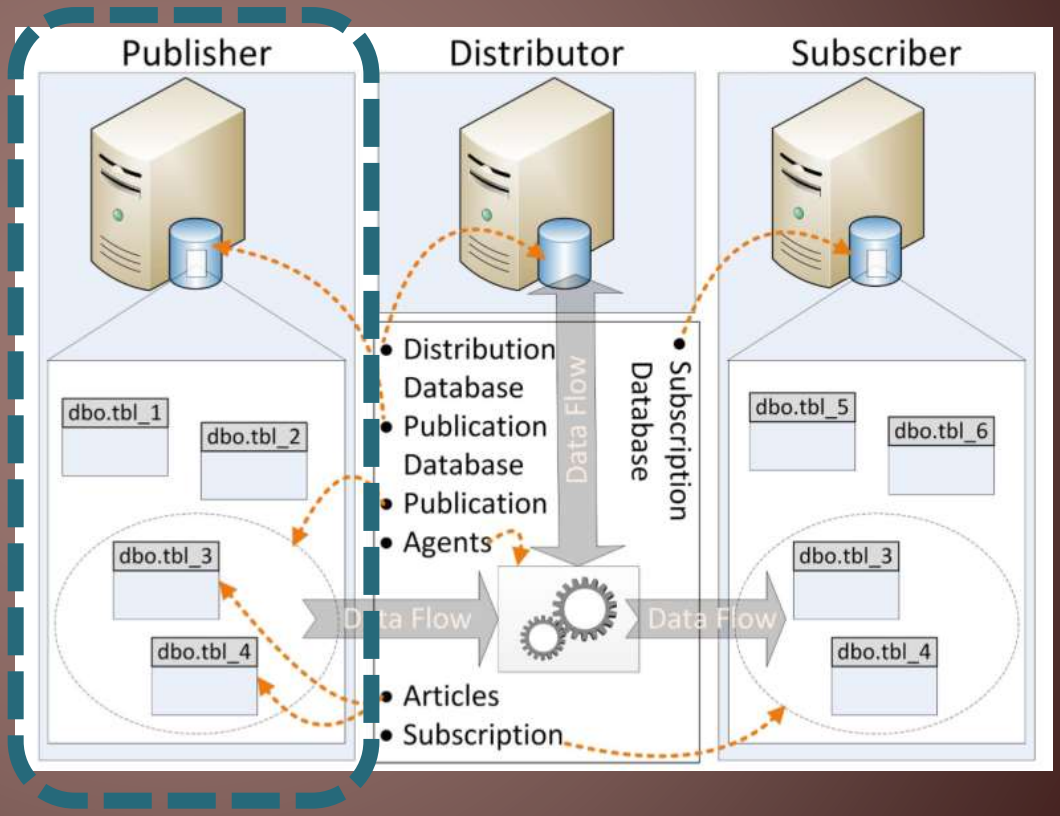
Components of a replication setup - 1



Components of a replication setup - 2

The components in the replication setup include a **Publisher** and its **publication database**.

The highlighted publication database contains a publication that includes two articles.



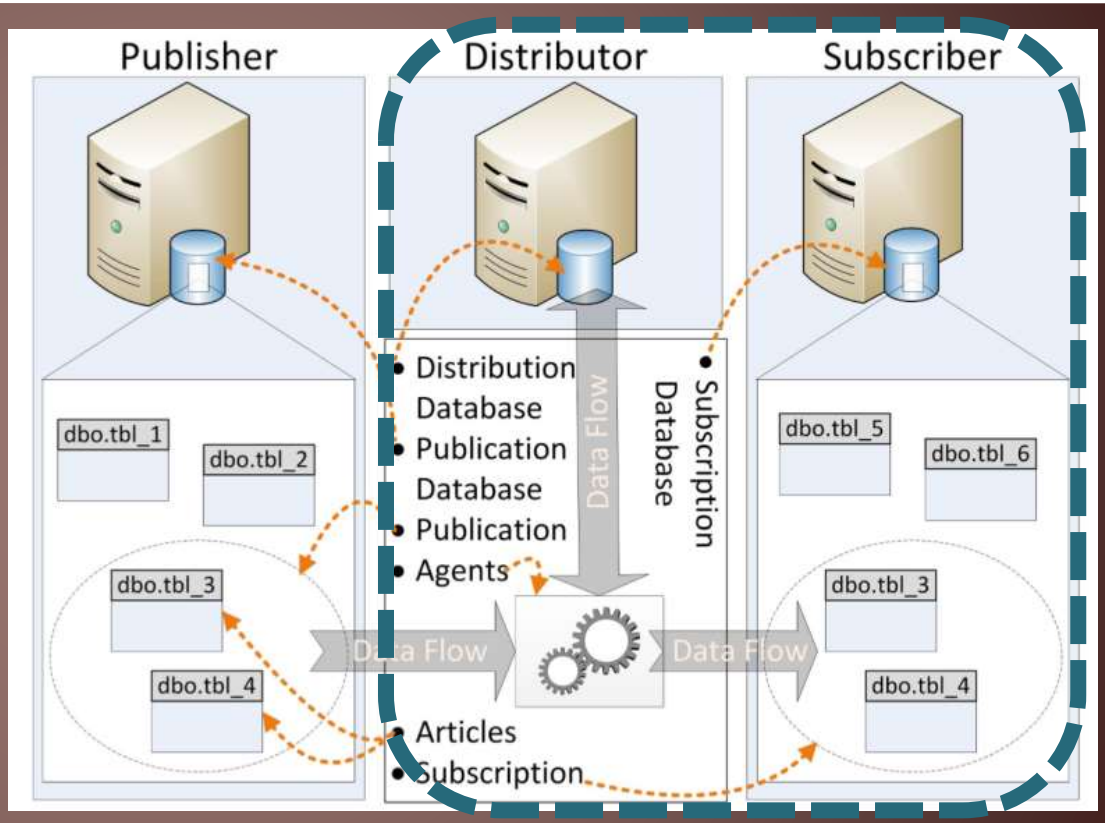
Publisher

- The **Publisher** is the SQL Server instance that makes a publication available for replication; however, the Publisher itself **doesn't actually have an active role** in a replication setup.
- After the publication is defined, the Distributor and sometimes the Subscriber do all the heavy lifting.

Distributor & Subscriber Components of a Replication setup

The setup also includes a **Distributor** and its distribution database as well as

a **Subscriber** and its subscription database, which contains the **subscription**.



Distributor

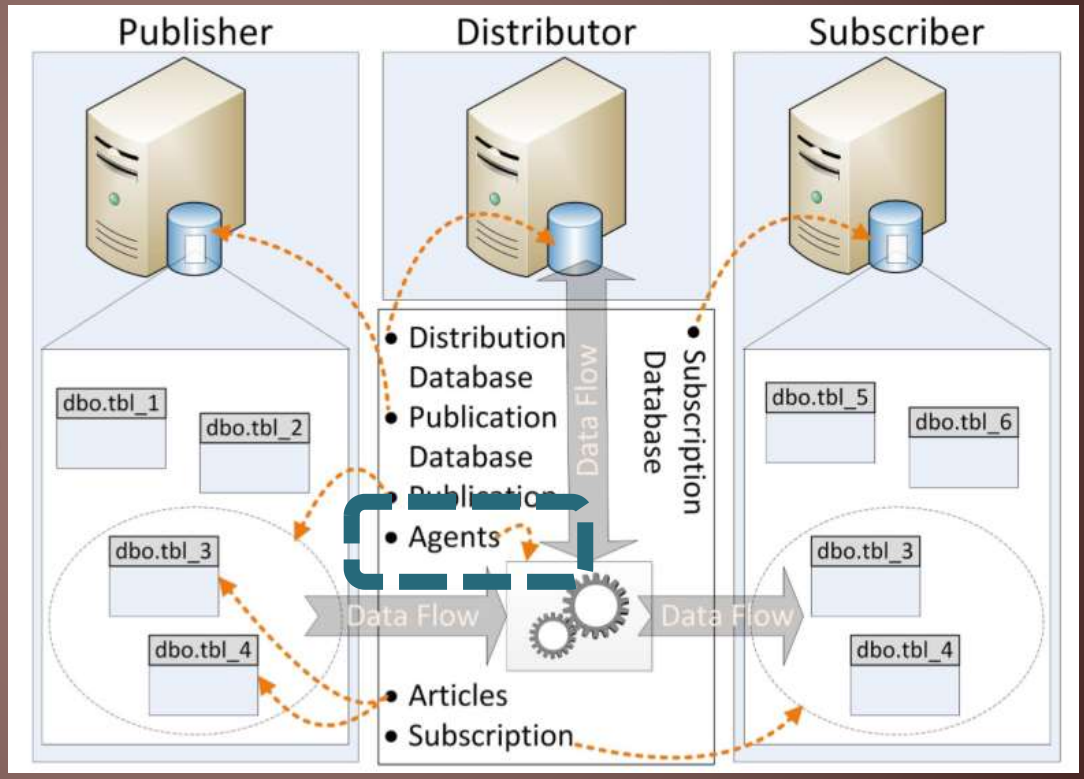
- Each Publisher is linked to a single Distributor. The Distributor is **an SQL Server instance that identifies changes to the articles** on each of its Publishers.
- Depending on the replication setup, the Distributor might also be **responsible for notifying** the Subscribers that have subscribed to a publication that an article has changed.
- The information about these changes is stored **in the distribution database** until
  - all Subscribers have been notified or
  - the retention period has expired.

Subscriber

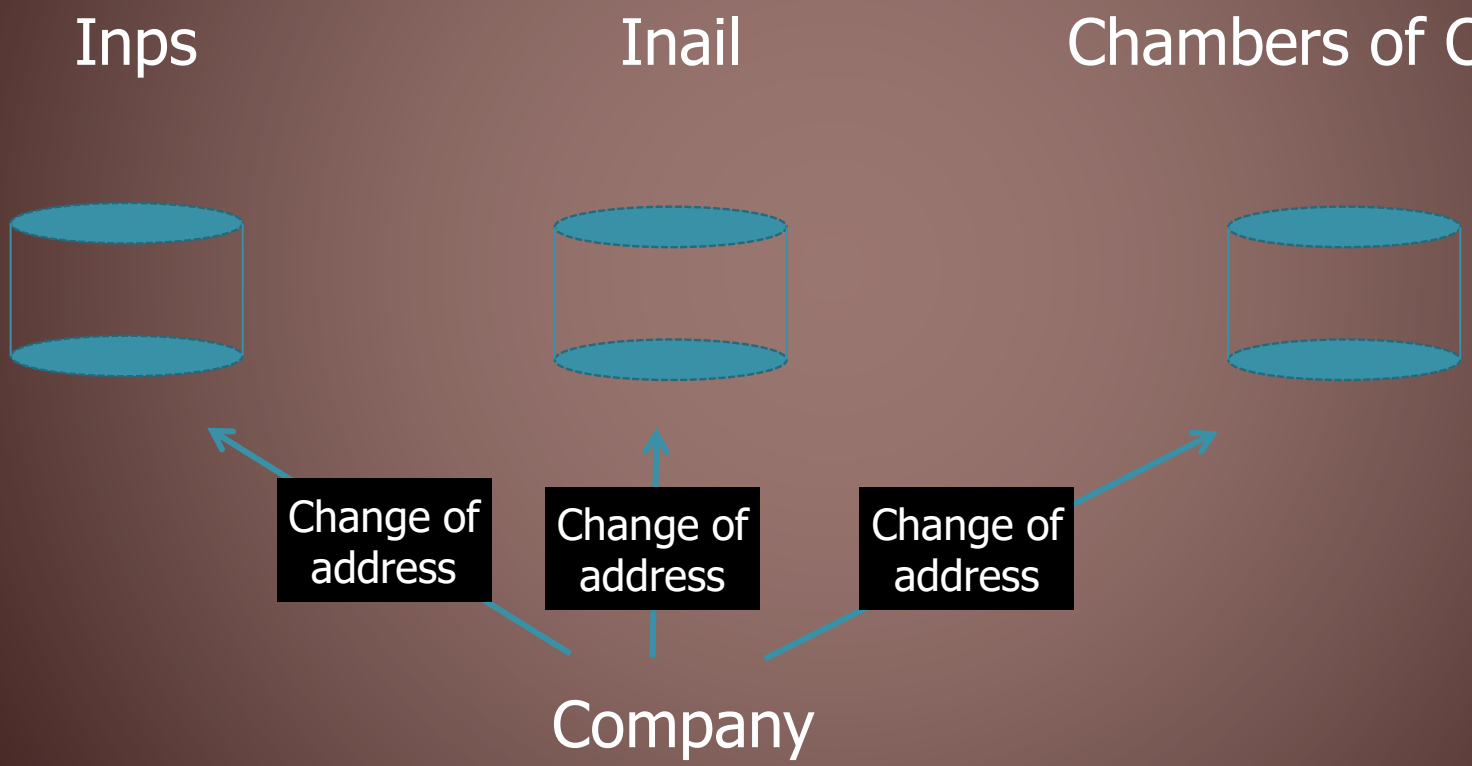
- Each SQL Server instance that subscribes to a publication is called a **Subscriber**.
- The Subscriber receives changes to a published article through that publication.
- A Subscriber **does not necessarily play an active role** in the replication process.
- Depending on the settings selected during replication setup, it might receive the data passively.

Components of a replication setup Agents

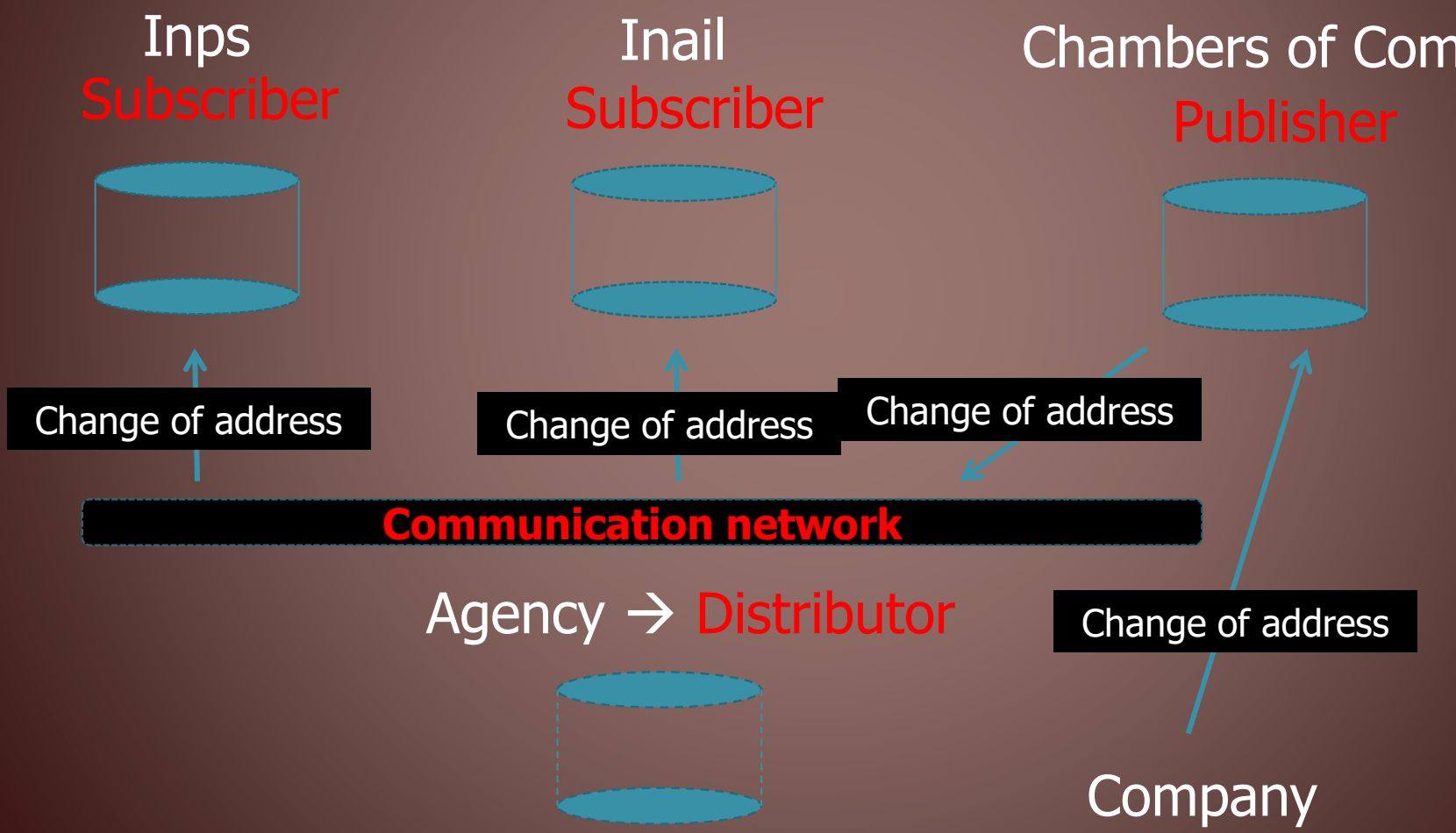
Finally, the setup includes the replication agents necessary to drive these processes.



Old scenario of interactions among administrations and companies in Italy



New scenario of interactions among administrations and companies



Microsoft SQL Server Types of replication

1. Snapshot
2. Transactional
3. Merge

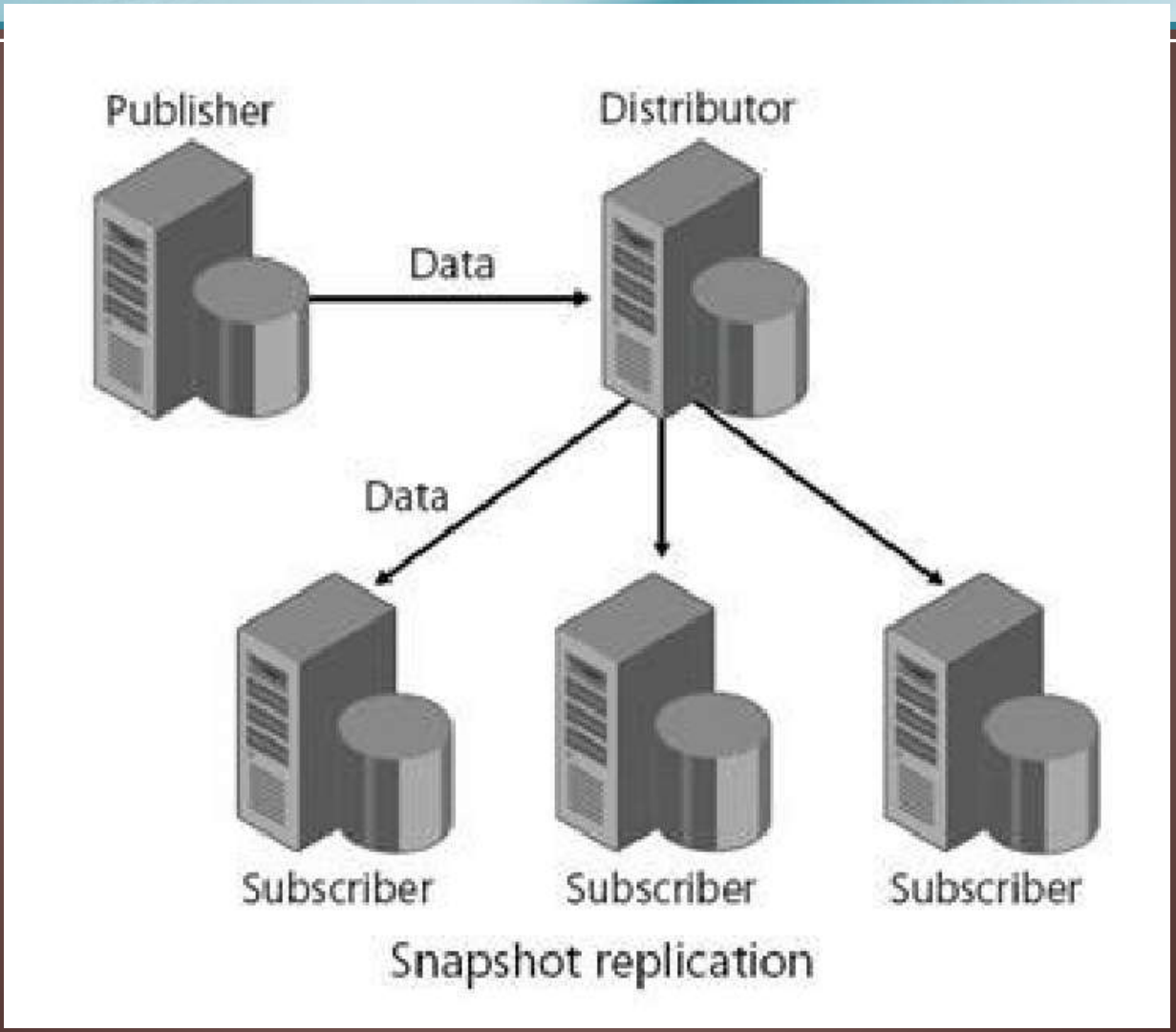
→

Types of replication

- **Snapshot replication**: Data on one database server **is plainly copied** to another database server
- **Merging replication**: Data from two or more databases is **combined** into a single database.
- **Transactional replication**: Users obtain complete initial copies of the database and then obtain **periodic updates** as data changes.

1. Snapshot replication

Snapshot replication



Snapshot replication – 1 Scenarios

- The snapshot replication method functions by periodically sending data in bulk format.
- when Usually it is used when the subscribing servers need to browse data such as
  - price lists,
  - online catalogs, or
  - data for decision support,
- and as a consequence can function in **read only** environment, and also when **can function for some time without updated data**.
- Functioning without updated data for a period of time is referred to as **latency**.

Snapshot replication – 2 Scenarios

- EXAMPLE - a retail store uses replication as a means of maintaining an **accurate inventory** throughout the district.
- Since the inventory can be managed on a weekly or even monthly basis, the retail stores **can function without updating the central server for days at a time**.
- This scenario has a high degree of latency and is a **perfect candidate** for snapshot replication.

Snapshot replication – 3 Scenarios

- Additional reasons to use this type of replication include scenarios with **low bandwidth connections**.
- Since the subscriber **can last for a while** without an update, this provides a solution that is **lower in cost than other methods while still handling the requirements**.

Snapshot replication – 5 Intervals

- Replication can be scheduled to run **at certain intervals**. For example, you can run replication **nightly** to provide a data snapshot to the reporting environment.
- As a result, your reports will contain data from completed days only, rather than containing data from a part of a day.
- That means **you don't have to build extra logic** into your system **to deal with today's incomplete data**.

Snapshot replication – 6 Limits and inefficiencies

- Although snapshot replication is the easiest type to set up and maintain, **it requires copying all data** each time a table is refreshed.
- This may lead to significant inefficiencies

Conclusions on snapshot replication - 1

Snapshot replication is helpful when:

- Data is **mostly static** and does not change often.
- It is acceptable to have copies of data that are **out of date for a period of time**.

Conclusions on snapshot replication – 2 When to use snapshot replication

- Snapshot replication is most appropriate when data changes are **substantial but infrequent**.
- EXAMPLE – If a sales organization maintains a product price list and the prices are all updated at the same time once or twice each year, replicating the entire snapshot of data after it has changed is recommended.
- **Given certain types of data, more frequent snapshots may also be appropriate**.

2. Transactional replication

Transactional replication - 1

Involves copying data from the publisher to the subscriber(s) once and then delivering transactions to the subscriber(s) as they occur on the publisher.

1. The **initial copy** of the data is transported by using the same mechanism as with snapshot replication: SQL Server takes a snapshot of data on the publisher and moves it to the subscriber(s).
2. As database users insert, update, or delete records on the publisher, **transactions are forwarded** to the subscriber(s).

Transactional replication - 2

- Transactional replication works, as the name suggests, on a transaction basis.
- The Log Reader Agent **scans the transaction log** of the publication database and **examines each committed transaction** to determine whether any changes affect the replicated articles. If they do, those changes are logged to the distribution database. The Distribution Agent then replicates those changes to the Subscriber.
- The data changes are applied to the Subscriber **in the same order** and **within the same transaction boundaries** as they occurred at the Publisher; therefore, within a publication, **transactional consistency is guaranteed**.

Transactional replication - 3

In what could be considered the opposite of snapshot replication, transactional replication **works by sending changes** to the subscriber.

You can

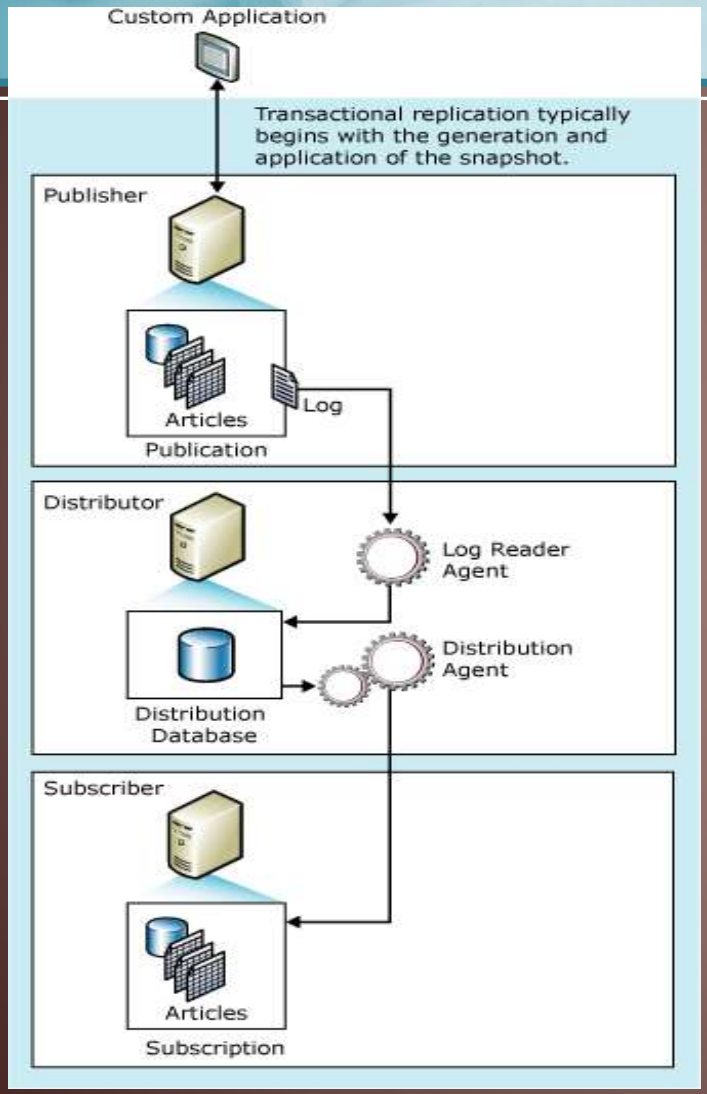
1. control the replication process so that it will accumulate transactions and send them **at timed intervals**, or
2. **transmit all changes as they occur**.

WHEN - You use this type of replication in environments having a **lower degree of latency** and **higher bandwidth connections**.

Transactional replication - 4

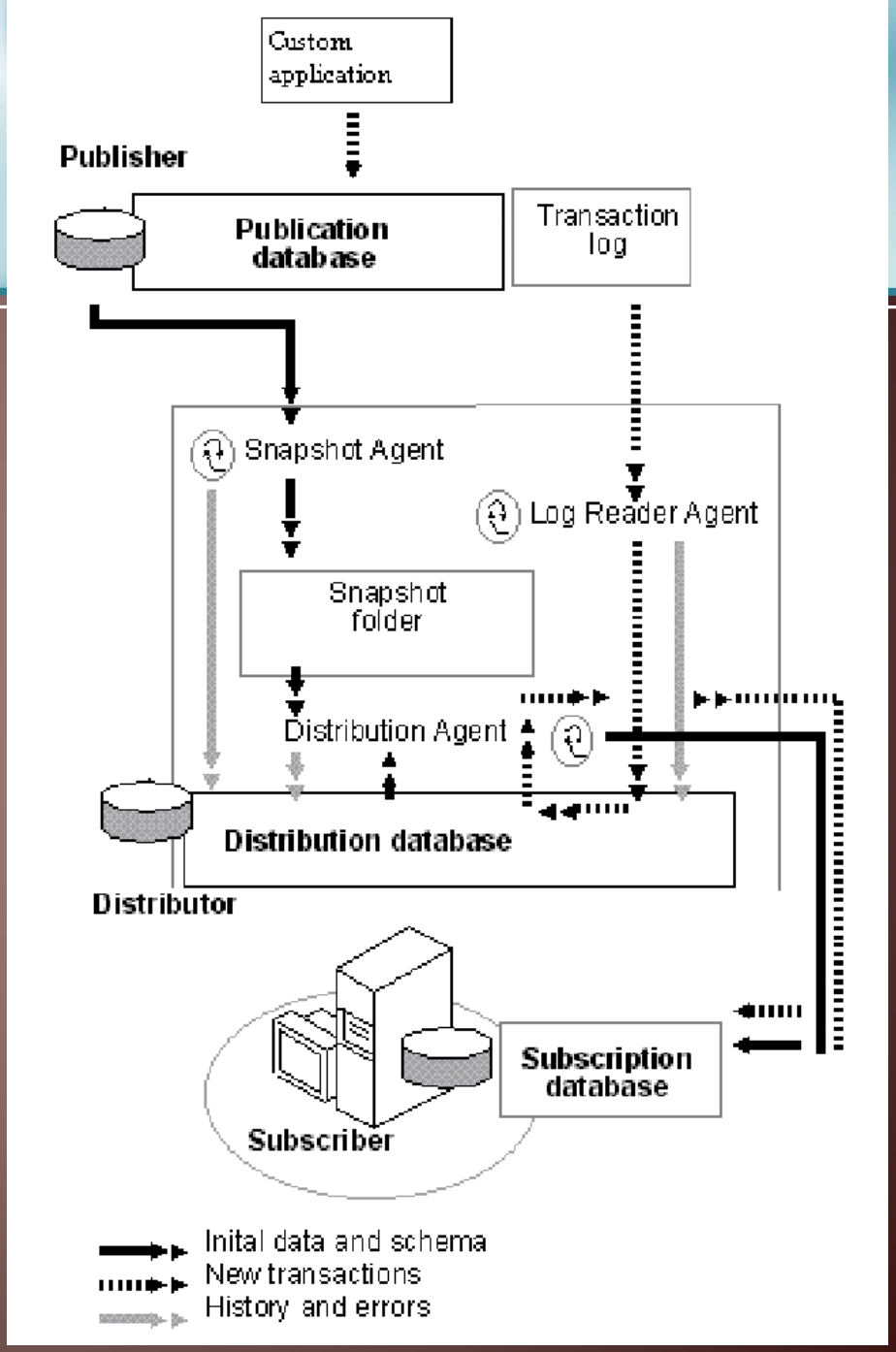
- You can set up transactional replication to transfer data in near real-time to your reports.
- Although SQL Server **doesn't offer synchronous replication (namely, ACID properties),** there is usually only a short delay for changes in the source data to show up in the replicated data.
- Of course, **latency depends on a lot of factors**, but most of the time it is in the range of only **a few seconds**.

Transactional Replication - 5



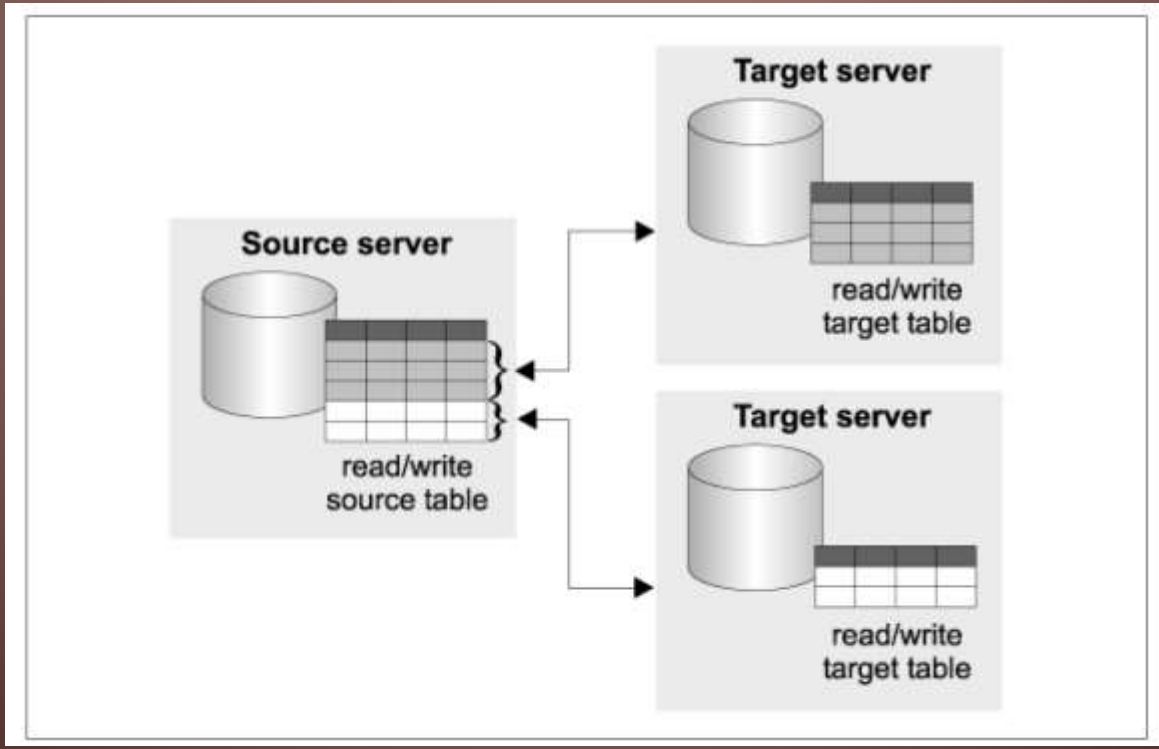
- LogReader reads publisher log file
- Transactions flow through distribution database
- Distribution agent reads distribution database and propagates changes to the subscribers

Transactional replication: how it works details



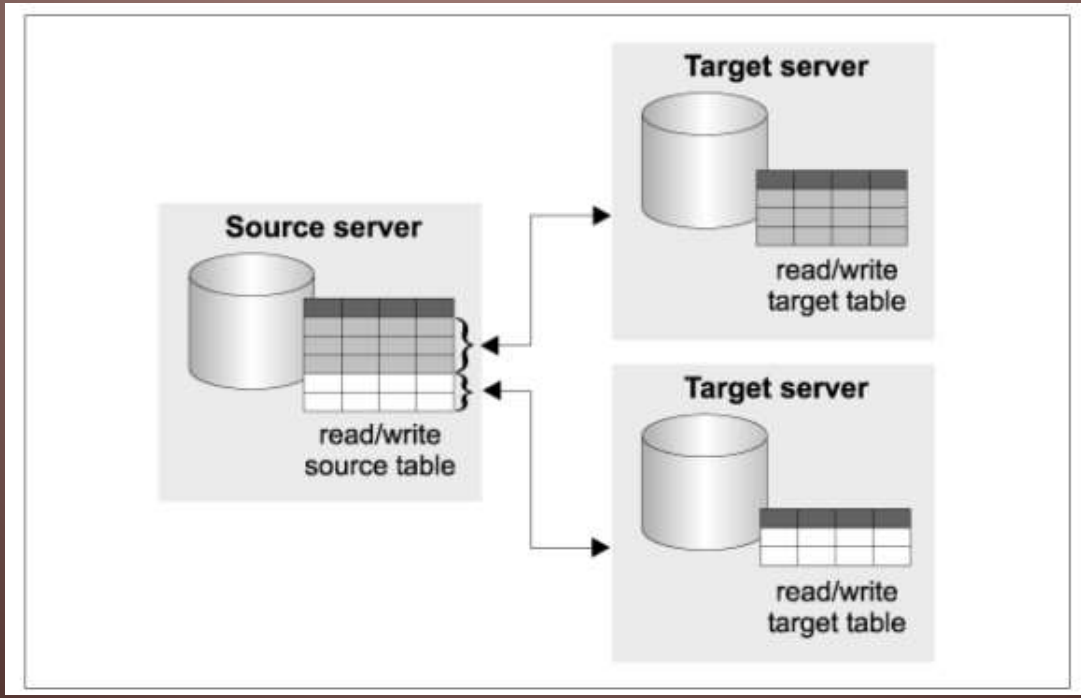
Transactional replication **with updating subscribers** - 1

An offshoot of standard transactional replication, this method of replication basically works the same way, but **adds to subscribers the ability to autonomously update data**.



Transactional replication with updating subscribers - 2

- This process allows for replication scenarios in which the published data is considered **read-only most of the time**, but **can be changed at the subscriber on occasion if needed**.
- Transactional replication with updating subscribers requires a permanent and reliable connection of medium to high bandwidth.



When Transactional replication is helpful

- You want incremental changes to be propagated to Subscribers as they occur.
- Subscribers are reliably and/or frequently connected to the Publisher

Snapshot vs transactional replication

Snapshot vs transactional replication

- Snapshot replication has an **overhead** on the Publisher than transactional replication, because incremental changes are not tracked.
- However, if the dataset set being replicated is **very large**, it will require substantial resources to generate and apply the snapshot.
- METHODOLOGY - Consider the size of the entire data set and the frequency of changes to the data when evaluating whether to utilize snapshot replication →

Exercise

- Given a Table A with n tuples
- In the snaphot replication A is copied every S seconds
- K transactions tr_j occur in a given TINT = S * m seconds time interval
- $C(tr_j)$ is the cost of transferring and applying a transaction to a Publisher P
- $C(tu_i)$ is the cost of copying a tuple on the Publisher
- When transactional replication is convenient?
- HINT evaluate the two costs in the interval TINT

3. Merge replication

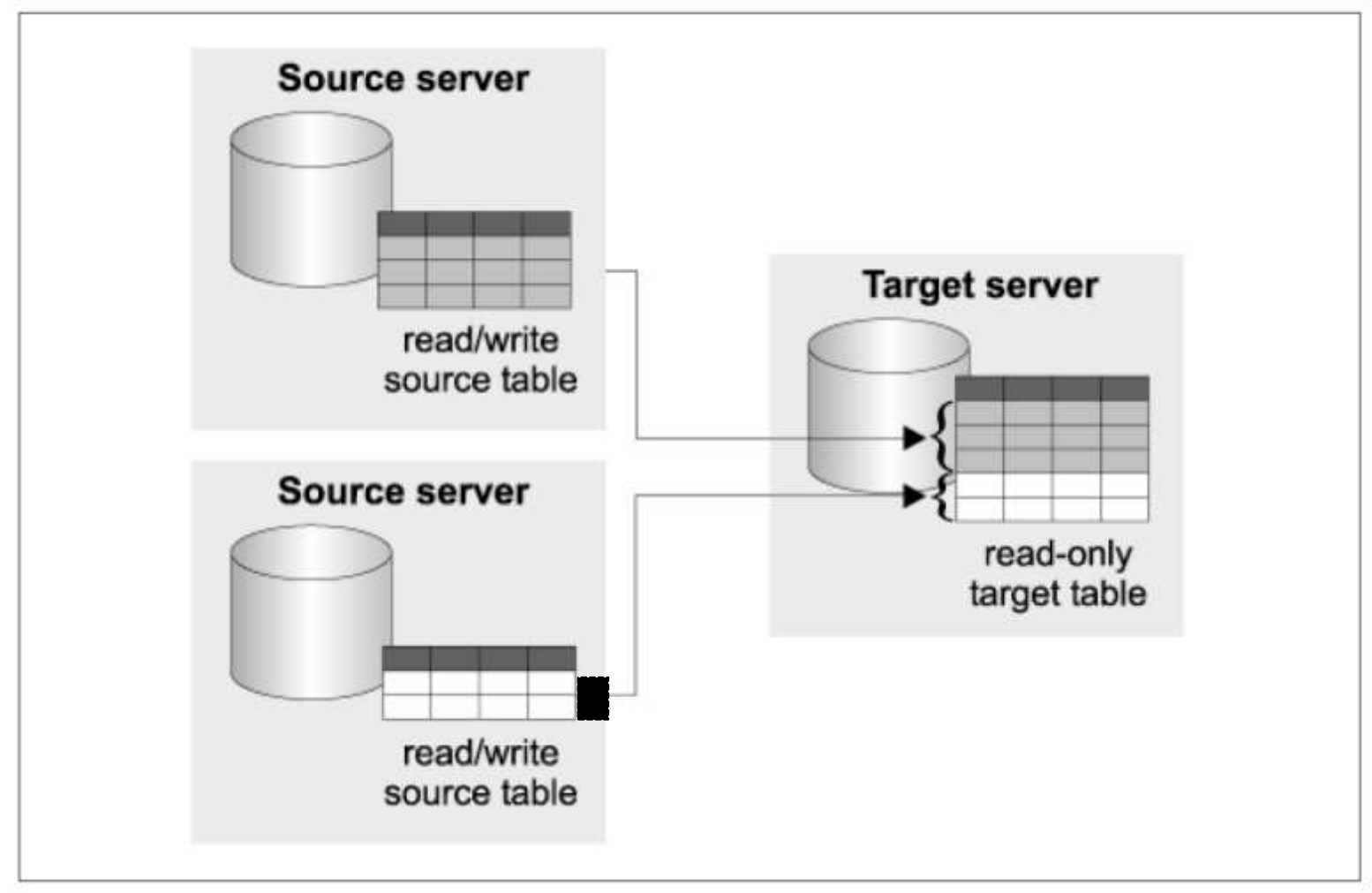
Merge replication -1

- Merge replication is the process of distributing data from Publisher to Subscribers, allowing the Publisher and Subscribers to make updates while connected or disconnected, and **then merging the updates** between sites when they are connected.
- Merge replication allows various sites to work autonomously and **at a later time merge updates** into a single, uniform result.
- The **initial snapshot** is applied to Subscribers, and then **changes are tracked to published data** at the Publisher and at the Subscribers.

Merge replication types

- Unidirectional
- Bidirectional

Unidirectional merge replication



Unidirectional merge replication - 2

- **Combines data from multiple sources** into a single central database.
- Unlike transactional replication, merge replication allows **changes of the same data** on publishers and subscribers, even when subscribers are not connected to the network.
- When subscribers connect to the network, replication will detect and combine changes from all subscribers and change data on the publisher accordingly.

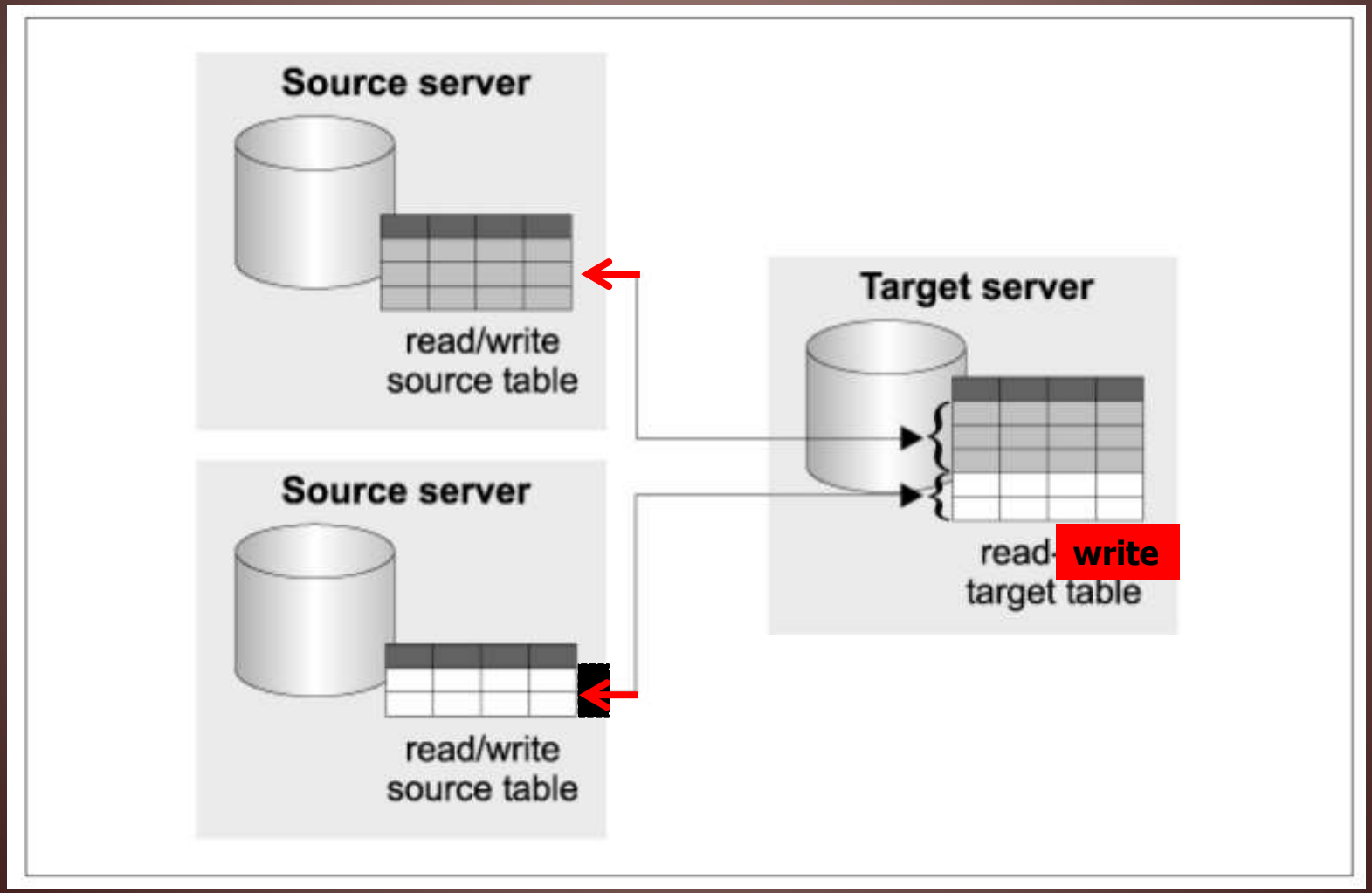
Unidirectional merge replication - 3

- Merge replication allows **two or more** databases to be kept in sync.
- Any change applied to one database will automatically be applied to the other databases – and vice versa
- Merge replication was designed to allow for **data changes on the Publisher as well as the Subscriber**, but merge replication also allows for disconnected scenarios.
- For example, if a Subscriber is disconnected from a Publisher during part of the day, the Subscriber and Publisher are synchronized when they are reconnected.

Bidirectional Merge replication

- The data is synchronized between servers continuously, **at a scheduled time, or on demand**.

Bidirectional merge replication



Bidirectional Merge replication

- Because **updates are made at more than one server**, the same data may have been updated by the Publisher or by more than one Subscriber. Therefore, **conflicts can occur when updates are merged**.
- Merge replication includes **default and custom choices for conflict resolution that you can define as you configure a merge publication**.
- When a conflict occurs, a **resolver** is invoked by the Merge Agent and **determines which data will be accepted and propagated to other sites**
- **See later →**

Choosing among types of replications

The type of replication you choose for an application depends on many factors, including

- the physical replication environment,
- the type and quantity of data to be replicated, and
- whether the data is updated at the Subscriber.

The physical environment includes

- the number and location of computers involved in replication and
- whether these computers are clients (workstations, laptops, or handheld devices) or servers.