Distributed Database Systems and Data Warehouses

Dr. Volodymyr Sokol
(vlad.sokol@gmail.com)

LECTION 8

Types of Transactions

- Remote SQL statements: *Remote query* selects data from one or more remote tables, all of which reside at same remote node. *Remote update* modifies data in one or more tables, all of which are located at same remote node .
- Distributed SQL statements: *Distributed query* retrieves data from two or more nodes. *Distributed update* modifies data on two or more nodes.
- Remote transactions: Contains one or more remote statements, all of which reference a single remote node.

Types of Transactions

- Distributed transactions: Includes one or more statements that, individually or as a group, update data on two or more distinct nodes of a distributed database. Oracle ensures integrity of distributed transactions using 2PC.

Referential Integrity

- Oracle does not permit declarative referential integrity constraints to be defined across databases.
- However, parent-child table relationships across databases can be maintained using triggers.

Introduction to Database Replication

- Functionality of DDBMS is attractive. However, implementations of required protocols and algorithms are complex and can cause problems that may outweigh advantages.
- Alternative and more simplify approach to data distribution is provided by a replication server.
- Every major database vendor has replication solution.

Co-funded by the
Erasmus+ Programme
of the European Union

Introduction to Database Replication

- Database Replication is the process of copying and maintaining database objects, such as relations, in multiple databases that make up a distributed database system.

Introduction to Database Replication

- Replication can be described using the publishing industry metaphor:
  - *Publisher:* a DBMS that makes data available to other locations through replication.
  - *Distributor:* a DBMS that stores replication data and metadata about the publication and in some cases acts as a queue for data moving from the publisher to the subscribers.
  - *Subscriber:* a DBMS that receives replicated data. A subscriber can receive data from multiple publishers and publications.

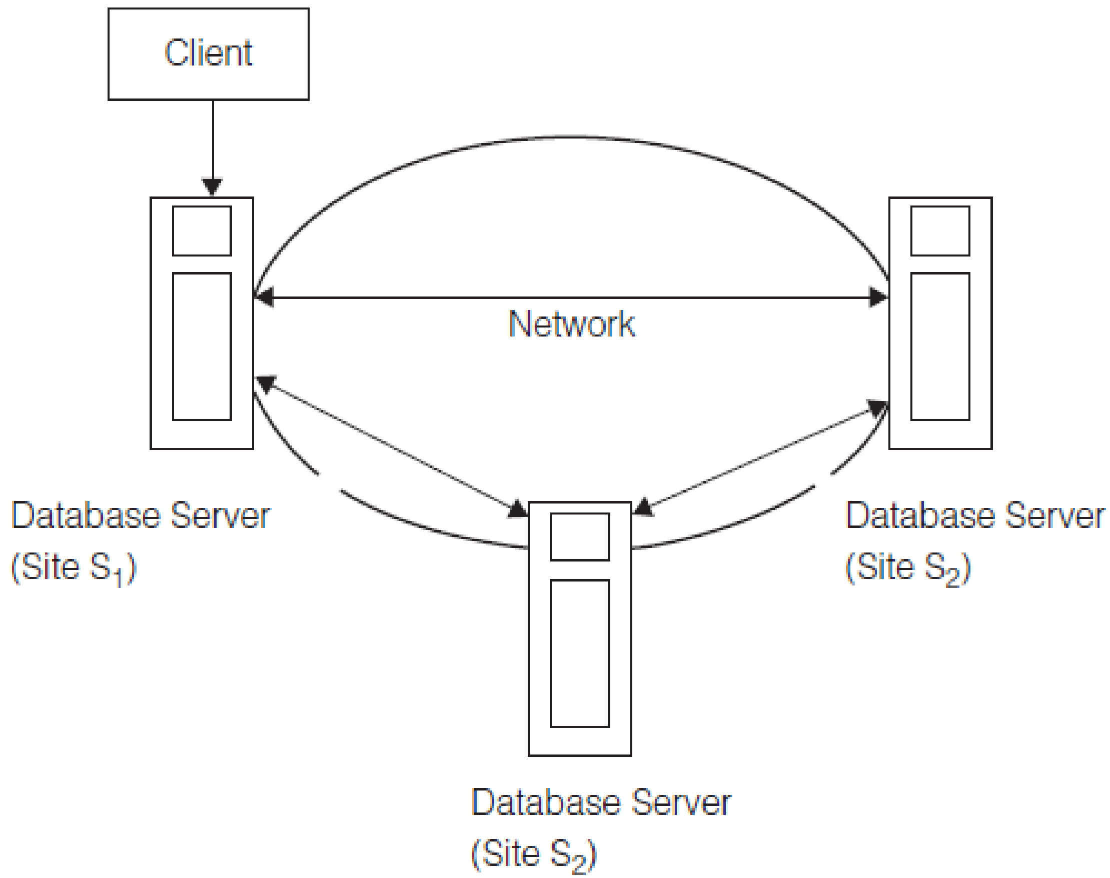
Introduction to Database Replication

- Replication has similar advantages to DDBMS:
  - *Reliability and availability*
  - *Improved performance*
  - *Supports disconnected computing model.*

Applications of Replication

- Replication supports a variety of applications that have very different requirements.
- Some applications are supported with only limited synchronization between the copies of the database and the central database system.
- Other applications demand continuous synchronization between all copies of the database.

Replication Model



Replication Model

- Replicated database system consists of several databases, called replicas or copies.
- As each site is also a backup site and backups are sometimes used interchangeably, a backup can also be used in combination with recovery aspects.

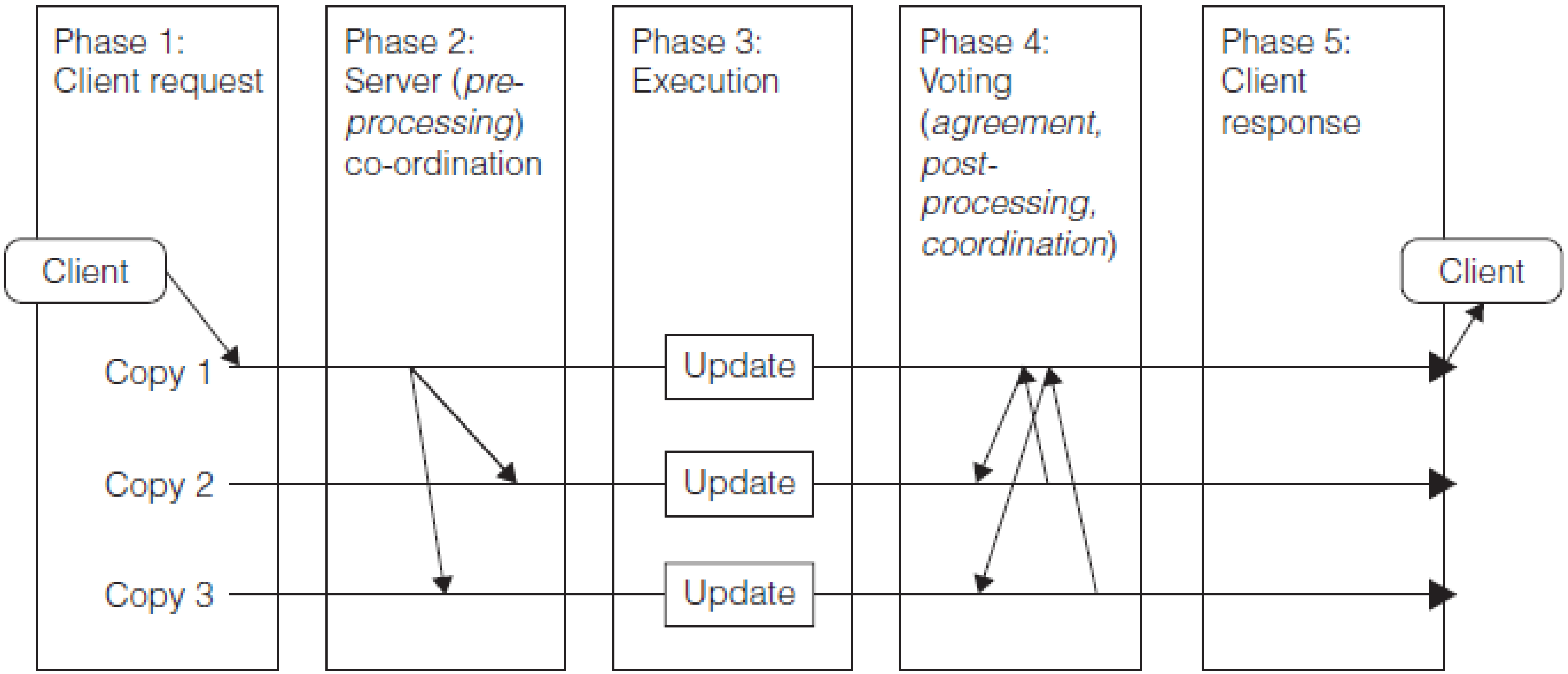
Replication Model

- Formally, replicated database consists of a set of *n* sites $S = \{S_1, S_2, \ldots, S_n)$, where *n* >= 2.
- A site hosts a set of copies of data items $x_1, x_2, x_3, \ldots$; we assume for the remainder of this presentation that each site is a complete copy of the database.

Replication Model

- To distinguish between physical copies and the logical data item itself, a copy is denoted with the site identifier; eg. a copy of data item x at site S_1 is denoted as x^1.
- Since many transactions might concurrently update copies at different sites, need a criterion (1CSR) to determine whether concurrent execution of transactions accessing copies at different sites is correct.
  - A replicated data history is one-copy serializable if it is equivalent to a serial one-copy history.

Functional Model of Replication Protocols



Functional Model of Replication Protocols

- Phase 1: A client submits its request to one site, called the local site.
- Phase 2: Depending on replication scheme, requests are forwarded to the other sites, called the remote sites.
- Phase 3: The request is processed.
- Phase 4: After all affected sites have processed request, sites communicate again, eg. to detect inconsistencies, propagate modifications, aggregate results, form a quorum or ensure atomicity of distributed transaction by running a concurrency control protocol, such as 2PC.
- Phase 5: Result is send to the client.

Consistency

- Transaction in a replicated database is an ACID unit of work, although, different definitions of consistency exist.
- Strongest form of consistency, 1CSR, degrades performance of a replicated database.
- It has been suggested that a replicated system can only choose two out of the properties: consistency, availability, and partition tolerance (CAP theorem).

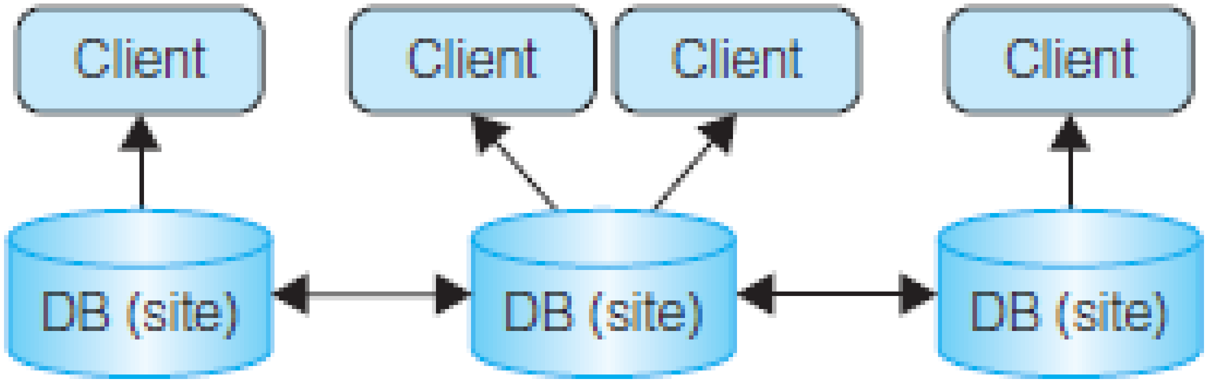
Consistency Types

- *Strong and Weak Consistency*:
  - Strong - all copies of a data item have same value at end of update.
  - Weak consistency - values eventually become identical and there is some time where replicas might have different values.
- *Transaction and Mutual Consistency*:
  - Mutual - copies converge to the same value
  - Transaction - global execution history is 1CSR.
  - A system can be mutually consistent but not transactional consistent, although the opposite is not true.

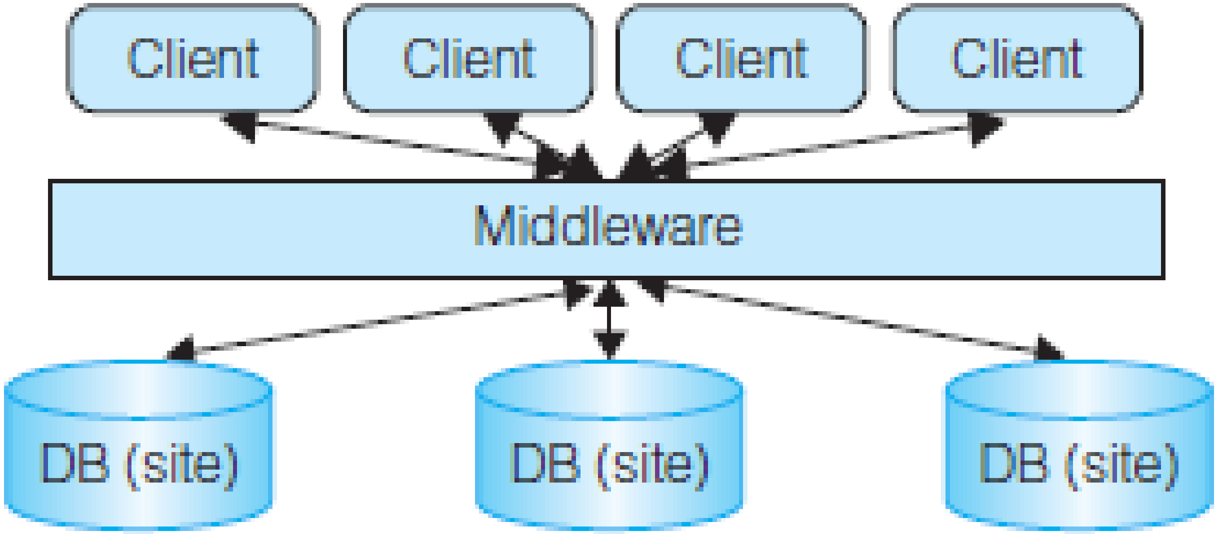
Consistency Types

- *Session Consistency*:
  - A basic property for each replication technique.
  - Guarantees that a client observes its own updates, also known as *read-your-own-writes*.
  - If clients do not observe their own updates a serious race condition arises. A *race condition* is where a transaction writes data item x on S_1 and a subsequent read of x within the same transaction on site S_2 does not reflect the write.

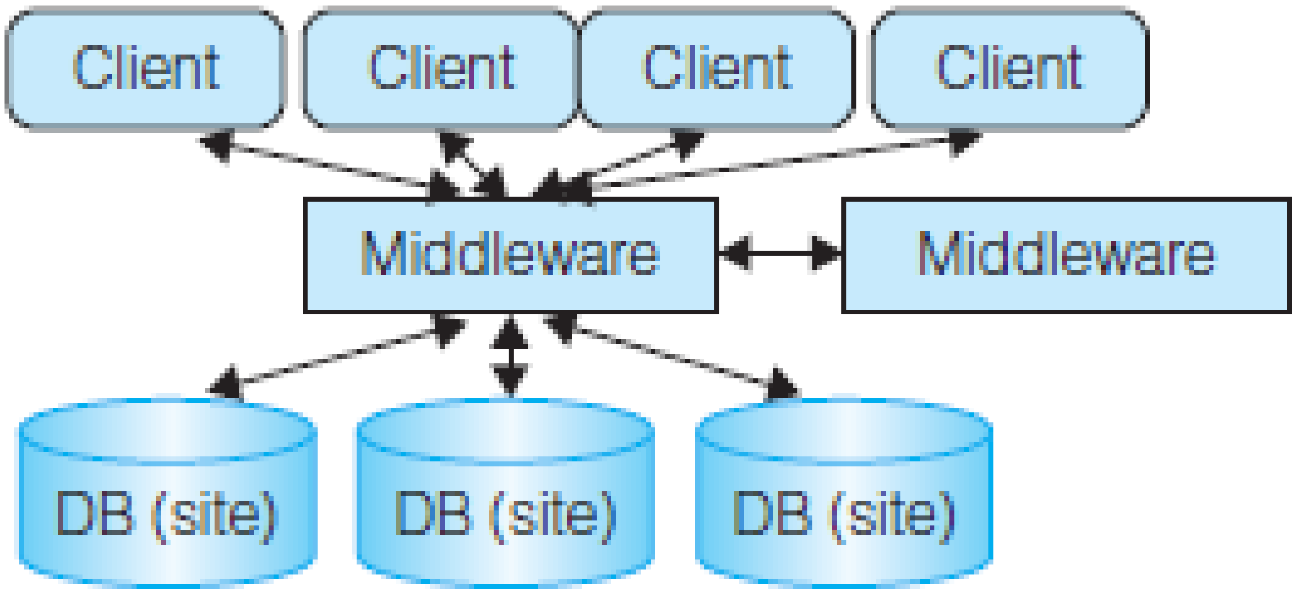
Kernel-Based Replication



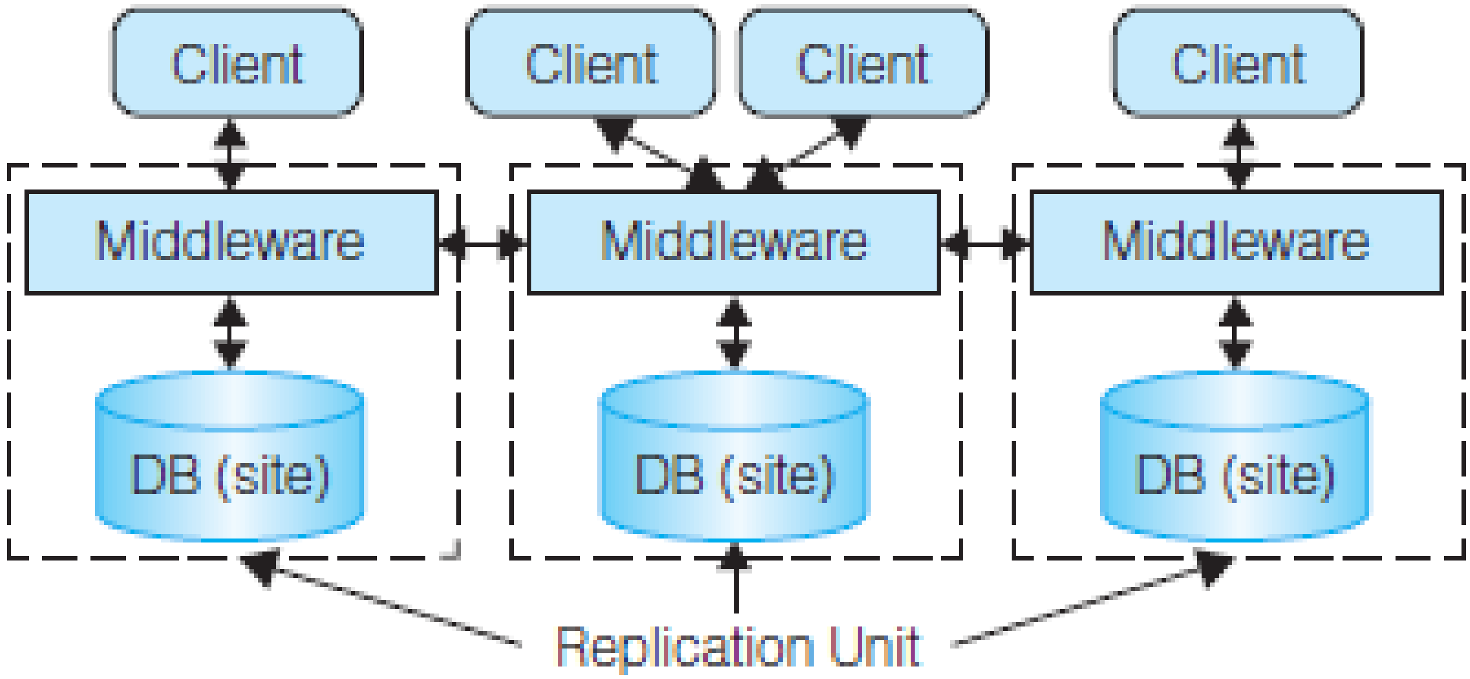
Middleware-Based Replication



Middleware-Based Replication



Decentralized Middleware-Based Replication



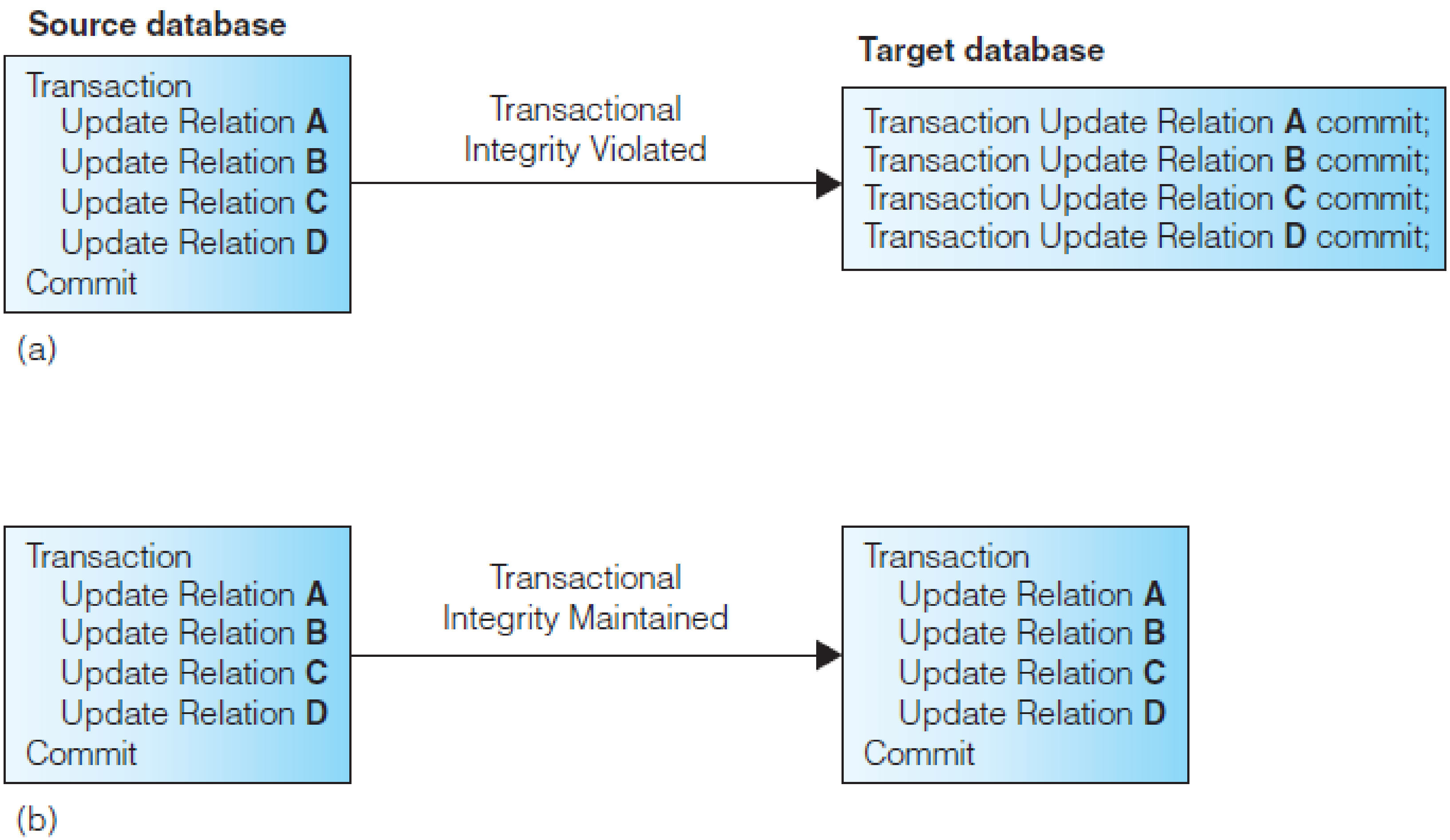
Replication Servers Functionality

- Basic function is copy data from one database to another (synchronously or asynchronously).
- Other functions include:
  - Scalability
  - Mapping and Transformation
  - Object Replication
  - Specification of Replication Schema
  - Subscription mechanism
  - Initialization mechanism
  - Easy Administration

Non-Transactional versus Transactional Update

- Early replication mechanisms were non-transactional.
- Data was copied without maintaining atomicity of transaction.
- With transactional-based mechanism, structure of original transaction on source database is also maintained at target site.

Non-Transactional versus Transactional Update



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.

Co-funded by the
Erasmus+ Programme
of the European Union
MASTIS

Data Ownership

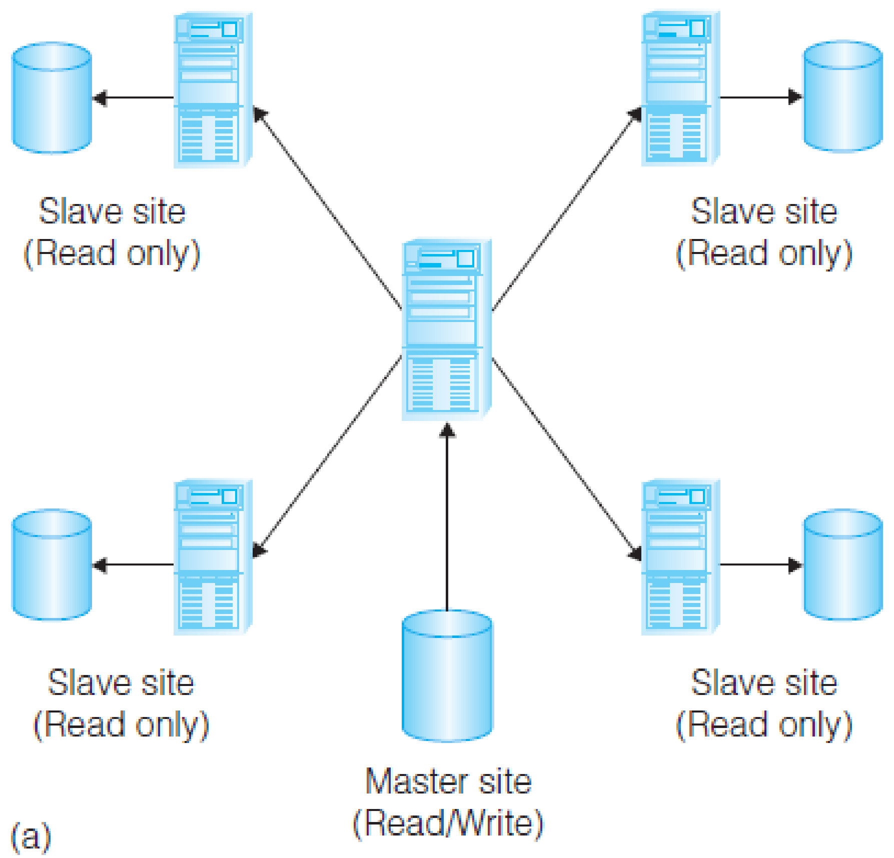
- Ownership relates to which site has privilege to update the data.

- Main types of ownership are:
  - Primary and secondary copy (or master/slave),
  - Workflow,
  - Update-anywhere (or peer-to-peer or symmetric replication).

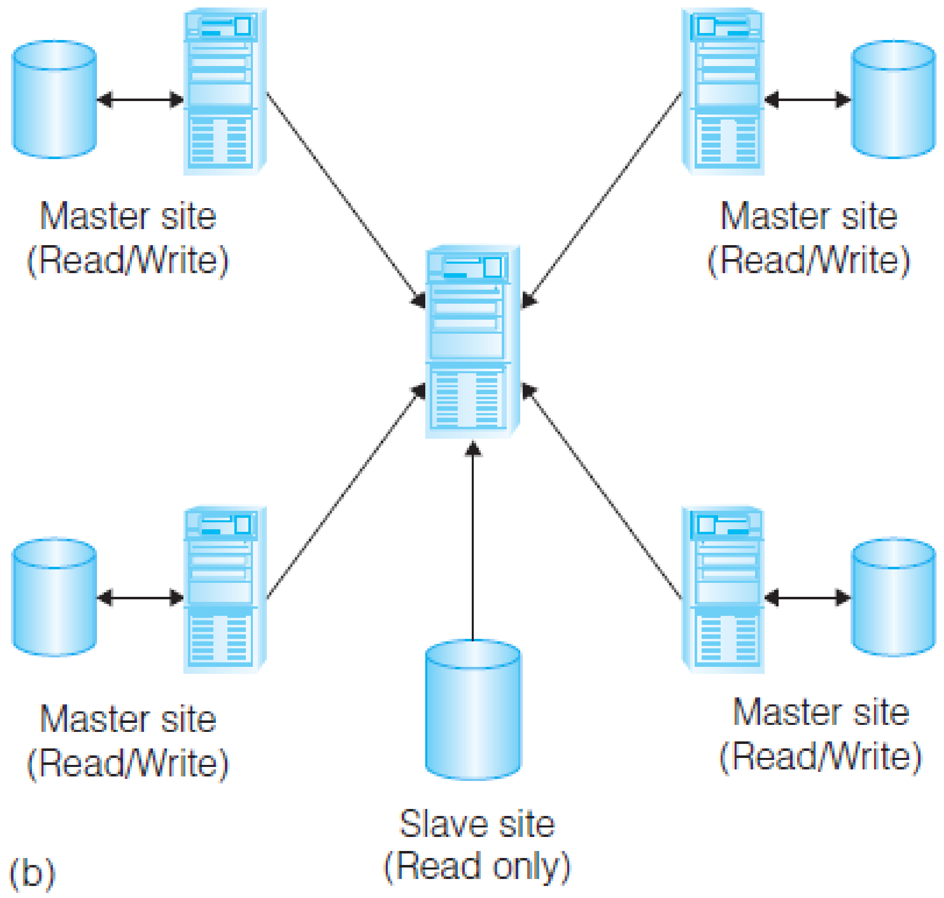
Primary Copy Ownership

- Asynchronously replicated data is owned by one (master) site, and can be updated by only that site.
- Using 'publish-and-subscribe' metaphor, master site makes data available.
- Other sites 'subscribe' to data owned by master site, receiving read-only copies.
- Potentially, each site can be master site for non-overlapping data sets, but update conflicts cannot occur.

Primary Copy Ownership – Data Dissemination



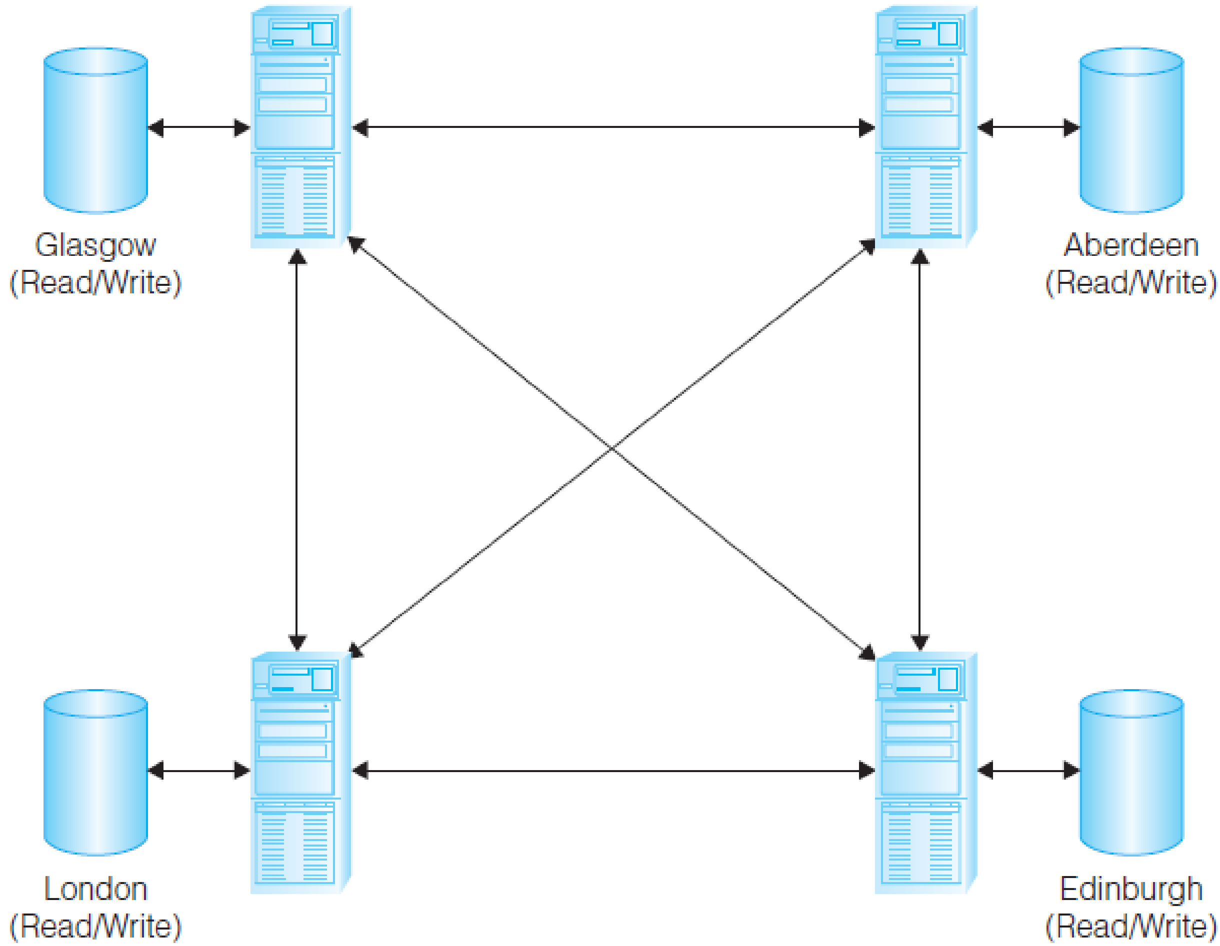
Primary Copy Ownership – Data Consolidation



Update-Anywhere Ownership

- Creates peer-to-peer environment where multiple sites have equal rights to update replicated data.
- Allows local sites to function autonomously, even when other sites are not available.
- Shared ownership can lead to conflict scenarios and have to employ methodology for conflict detection and resolution.

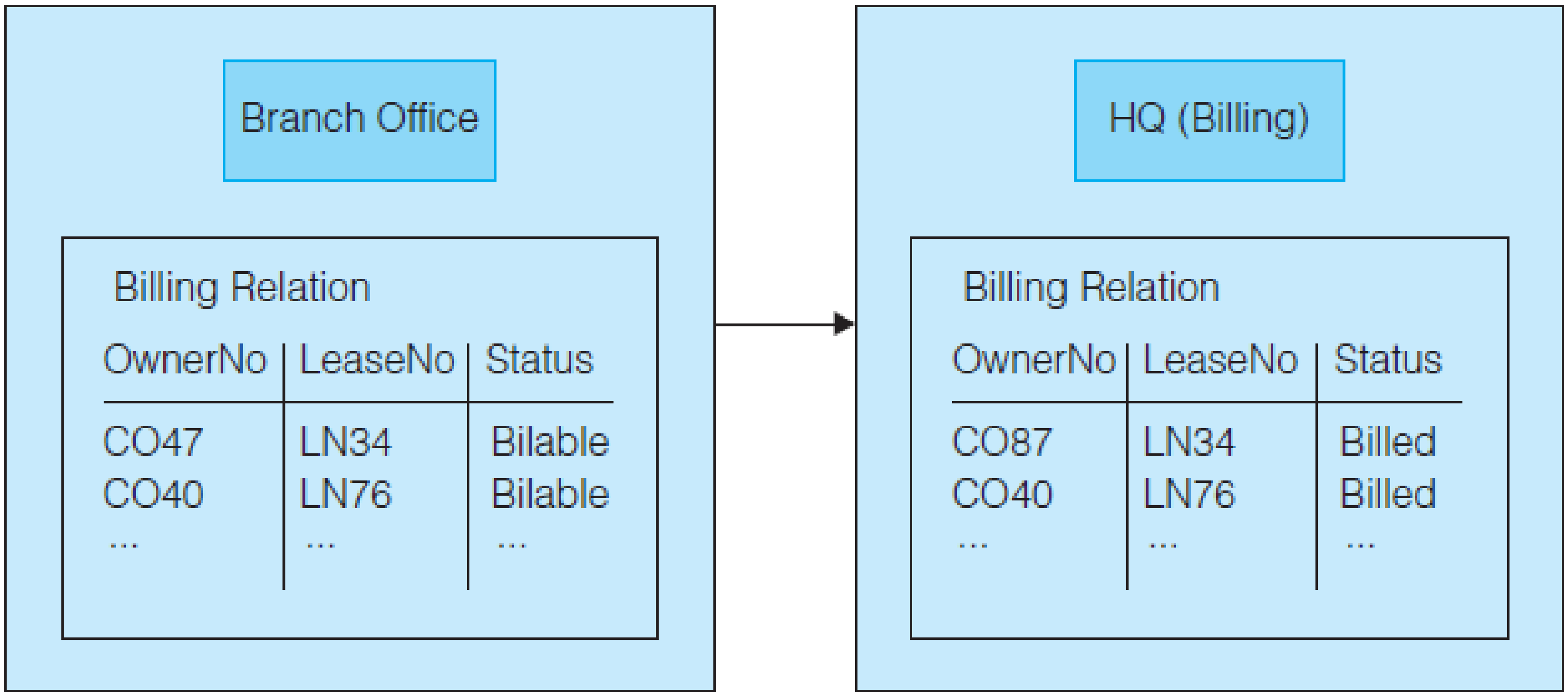
Update-Anywhere Ownership



Workflow Ownership

- Avoids update conflicts, while providing more dynamic ownership model.
- Allows right to update replicated data to move from site to site.
- However, at any one moment, only ever one site that may update that particular data set.
- Example is order processing system, which follows series of steps, such as order entry, credit approval, invoicing, shipping, and so on.

Workflow Ownership



Termination Protocols

- Voting:
  - As in DDB, a voting protocol (eg. 2PC) ensures atomicity of a transaction executed across sites.
  - Voting also affects fault tolerance of the system; eg. if T_1 updates data item x on S_1 and the installation of this update at S_2 is not confirmed by a vote protocol, there is no guarantee that other sites have been updated as part of this transaction and if S_1 fails, the update of T_1 is lost.
  - Execution of remote transactions not within the boundary of the local transaction is called *1-safe*; if local site fails the update is lost; *n-safe - n*-1 sites can fail but the update is not lost.

Termination Protocols

- Nonvoting:
  - Some replication techniques avoid voting to reduce message overhead and increase performance and scalability.
  - However, no voting phase means atomicity of transaction has to be ensured some other way (no atomicity is not an option as it violates consistency).
  - In an update-anywhere architecture, one solution is to use *group communication protocols*, as we discuss shortly.

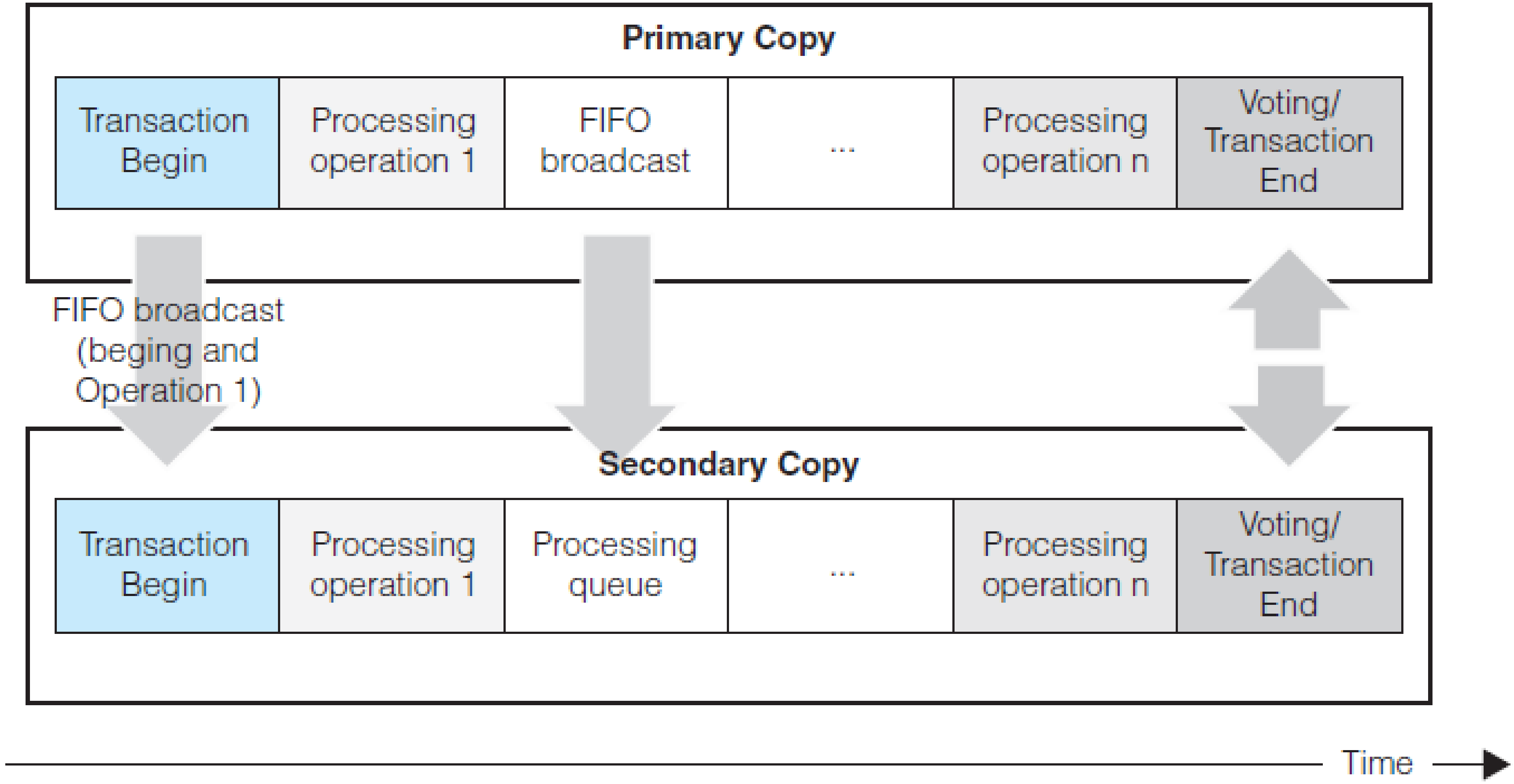
Replication Schemes

- Discuss 4 combinations of properties: update propagation and update location (called scheme):
  - Eager and primary copy, called *eager primary copy*;
  - Eager and update-anywhere, called *eager update anywhere*;
  - Lazy and primary copy, called *lazy primary copy*;
  - Lazy and update anywhere, called *lazy update anywhere*.

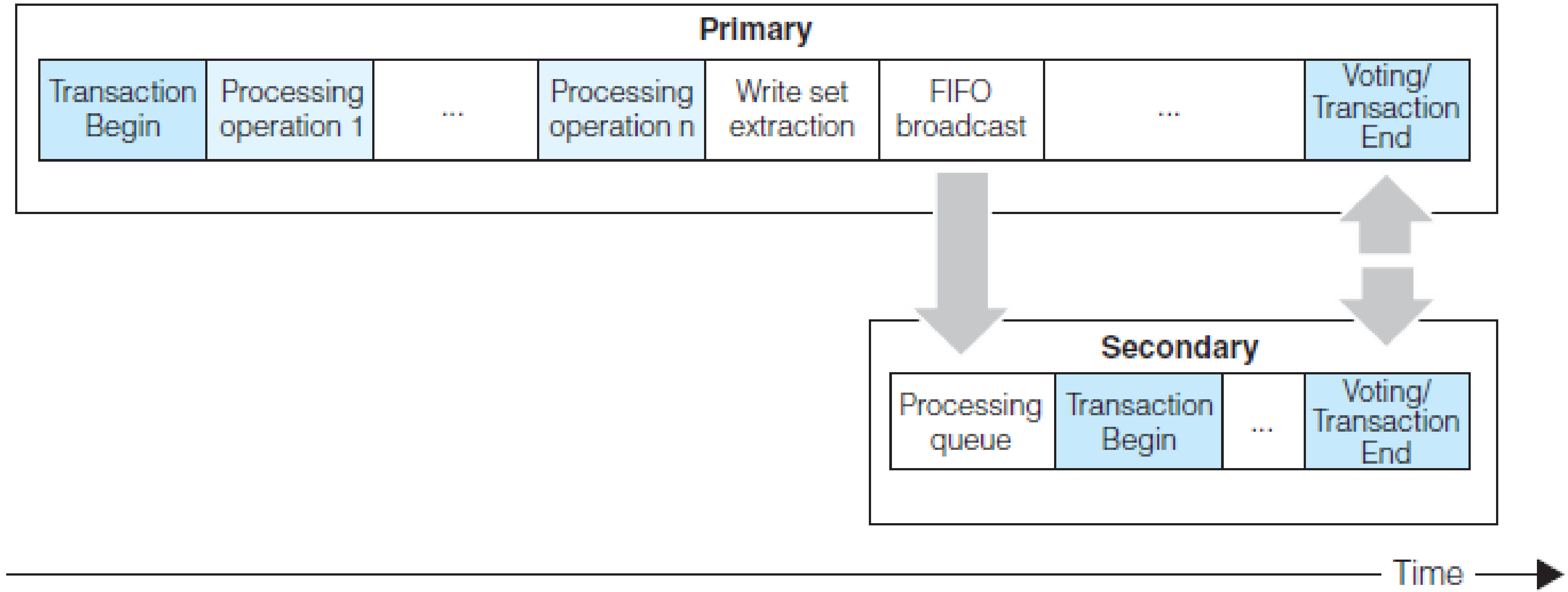
Eager Primary Copy

- Updates take place at primary copy only, which eagerly propagates them to each secondary copy.
- A secondary copy is only allowed to process read-only transactions and, to ensure atomicity, all sites run a voting phase.
- The primary site can propagate either:
  - update by update
  - wait until transaction has executed all operations, extract write-set, and propagate all modifications in one message to each secondary copy.

Eager Primary Copy – Update by Update



Eager Primary Copy – Propagate All



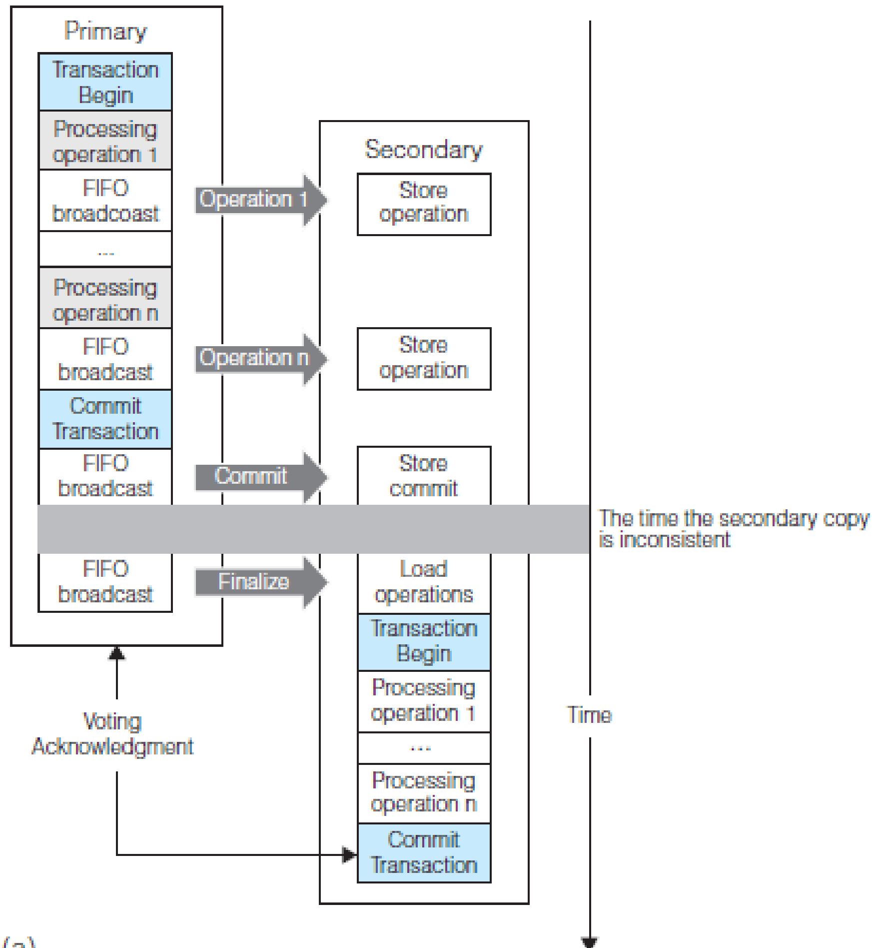
Lazy Primary Copy

- Lazy propagation increases performance at the primary site by allowing it to unilaterally decide whether to commit or abort a transaction; ie., primary site does not have to wait for any secondary sites.
- Since the update propagation is not within the transaction boundary, response time is shorter than with eager replication (the higher the network latency, the bigger is this effect).

Lazy Primary Copy

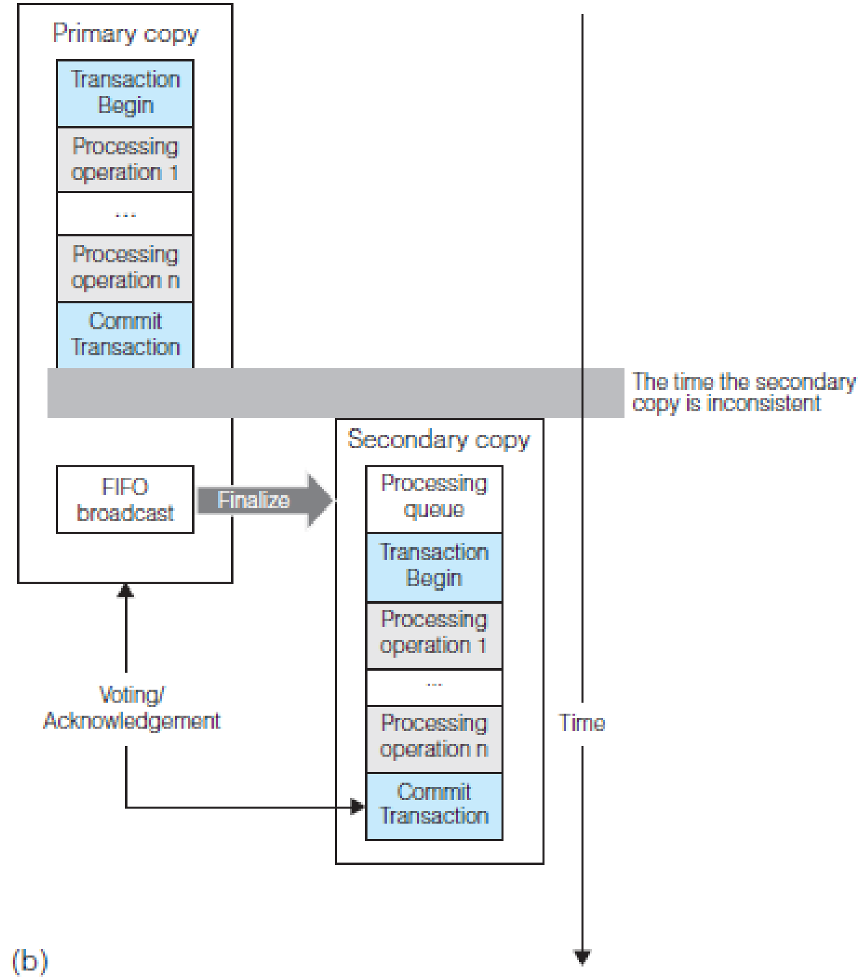
- To maintain transaction's execution order, FIFO (first-in-first-out) message delivery is used.
- A primary site can choose to propagate:
  - update by update
  - entire write-set.

Lazy Primary Copy – Update by Update



(a)

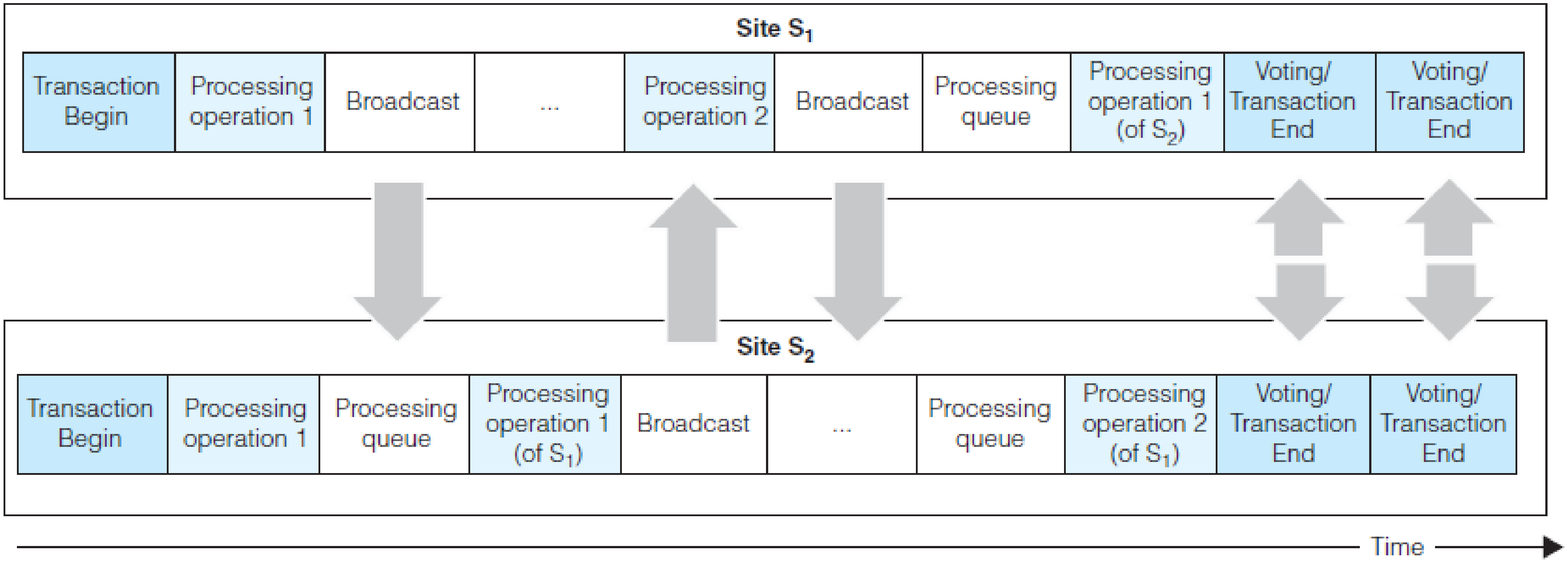
Lazy Primary Copy – Propagate All



Eager Update Anywhere

- Present a ROWA scheme where updates are processed by some site and are then eagerly broadcast to all other sites.
- Propagation of updates takes place within the boundary of local transaction and atomicity is ensured by a final voting phase.
- Consider a linear interaction only.

Eager Update Anywhere



Lazy Update Anywhere

- ROWA scheme where updates are allowed at any site but are lazily propagated to remote sites.
- Need a mechanism to detect conflicting updates and restore data consistency.
- Problem is any site can decide whether to commit or abort and might have 2 conflicting sites that have already committed.

Lazy Update Anywhere

- In a lazy primary copy scheme can remove a secondary site that does not accept an update.
- This is not possible here, because every site is a primary site and due to the laziness, any site might have locally committed, but conflicting transactions, not propagated yet.
- To resolve conflicts, mechanisms to detect and resolve conflicts are key to make this scheme feasible.

Conflict Detection and Resolution

- Some of most common mechanisms are:
  - Earliest and latest timestamps.
  - Site Priority.
  - Additive and average updates.
  - Minimum and maximum values.
  - User-defined.
  - Hold for manual resolution.