Database Implementation and Use

Section: Transaction Integrity

Business Transaction

- is a sequence of steps that constitute some well-defined business activity.

Examples:

- Admit the Patient in a hospital
- Enter Customer Order in a manufacturing company.

Normally, a business transaction requires several actions against the database.

For example, the transaction Enter Customer Order. When a new customer order is entered, an application program might perform the following steps:

1. Input the order data (keyed by the user).
2. Read the CUSTOMER record (or ask the user to key it in if a new customer).
3. Accept or reject the order. If the Balance Due plus the Order Amount does not exceed the Credit Limit, accept the order; otherwise, reject it.
4. If the order is accepted, increase the Balance Due by Order Amount. Store the updated (or new) CUSTOMER record. Insert the accepted ORDER record in the database.

A transaction is a complete set of closely related update commands that must all be done (or none of them done) for the database to remain valid.

When processing transactions, a DBMS must ensure that the transactions have four well-accepted characteristics, called the **ACID properties**

ACID Properties:

Atomic

- The transaction cannot be subdivided; hence, it must be processed in its entirety or not at all.
- Once the transaction is processed, we say the changes are *committed*.
- If the transaction fails at any midpoint, we say that it has aborted.
- For example, suppose the program accepts a new customer order, increases the Balance Due, and stores the updated CUSTOMER record. However, suppose that the new ORDER record is not inserted successfully (perhaps due to a duplicate Order Number key or insufficient physical file space). In this case, we want none of the parts of the transaction to affect the database.

ACID Properties:

Consistent

- Any database constraints that must be true before the transaction must also be true after the transaction.
- For example, if the inventory on-hand balance must be the difference between total receipts minus total issues, this will be true both before and after an order transaction, which reduces the on-hand balance to satisfy the order.

ACID Properties:

Isolated

- Changes to the database are not revealed to users until the transaction is committed.
- For example, this property means that other users do not know what the on-hand inventory is until an inventory transaction is complete; this property then usually means that other users are prohibited from simultaneously updating and possibly even reading data that are in the process of being updated.
- A consequence of transactions being isolated from one another is that concurrent transactions (i.e., several transactions in some partial state of completion) all affect the database as if they were presented to the DBMS in serial fashion.

ACID Properties:

Durable

- Changes are permanent. Thus, once a transaction is committed, no subsequent failure of the database can reverse the effect of the transaction.

Transaction Boundaries

The logical beginning and end of a transaction.

Supporting SQL statements:

1. **Begin Transaction**: starts a new transaction.
2. **SQL Operations**: Execute your SQL queries
3. **Commit:** If all SQL statements execute successfully, make the changes permanent.
4. **Rollback**: If any of the statements fail or an exception is thrown, undo all changes made within the transaction.

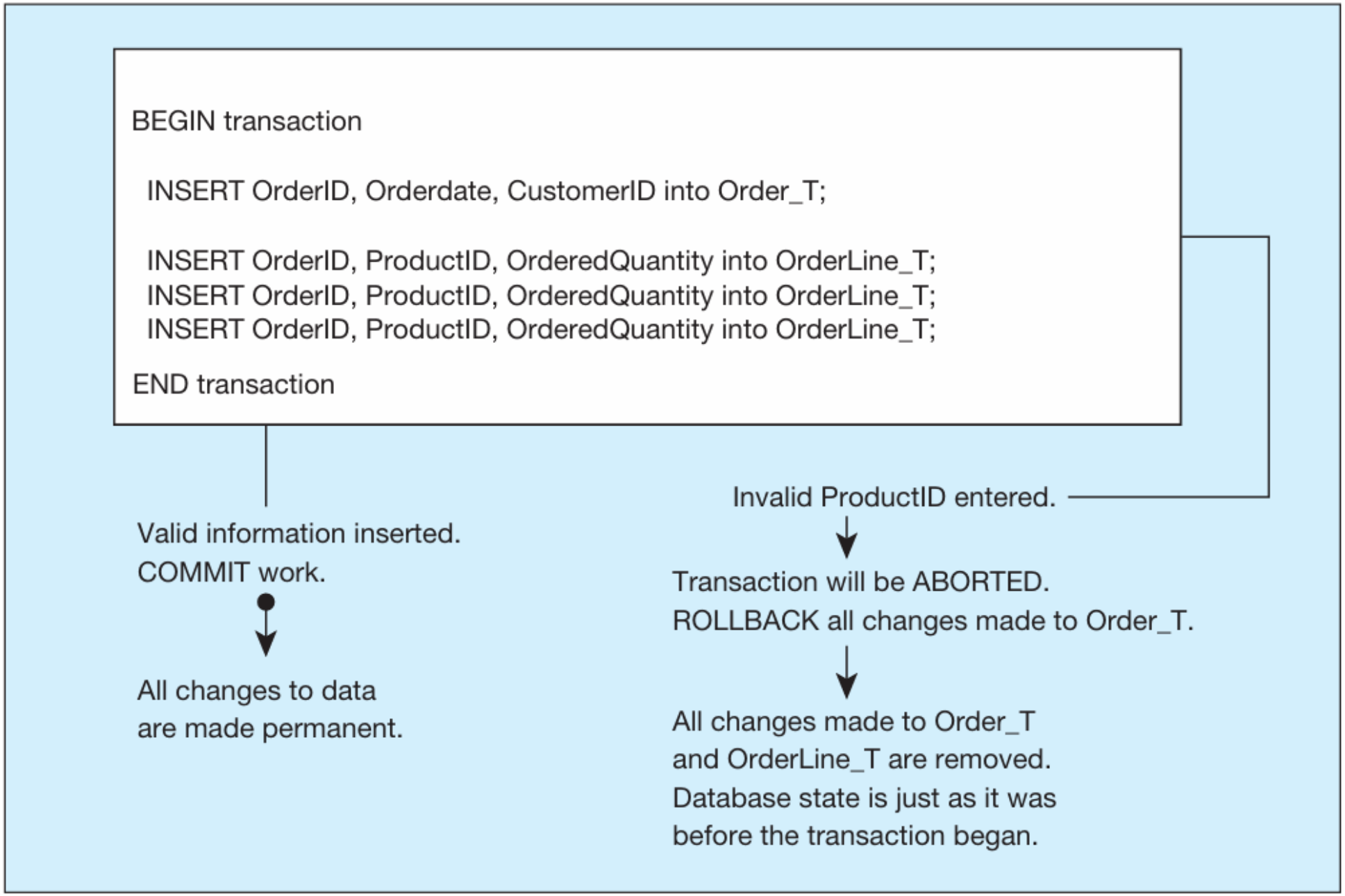


FIGURE 7-19 An SQL transaction sequence (pseudocode)

```php
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error)
 {
  die("Connection failed: " . $conn->connect_error);

}

// Disable autocommit mode
$conn->autocommit(FALSE);

try {
  // SQL statements for the transaction
  $conn->query("UPDATE accounts SET balance = balance - 100 WHERE accoun
  $conn->query("UPDATE accounts SET balance = balance + 100 WHERE accoun

  // Commit the transaction
  $conn->commit();
  echo "Transaction successful!";
} catch (Exception $e) {
  // Rollback the transaction if something failed
  $conn->rollback();
  echo "Error: " . $e->getMessage();
}
```

Considerations when creating transactions

Some database data are used frequently, so it is important to complete transactional work on these so-called hot spot data as quickly as possible.

For example, a primary key and its index for bank account numbers will likely need to be accessed by every ATM transaction, so the database transaction must be designed to use and release these data quickly.

- Also, remember that **all the commands between the boundaries of a transaction must be executed, even those commands seeking input from an online user**.
- If a user is slow to respond to input requests within the boundaries of a transaction, other users may encounter significant delays. Thus, **if possible, collect all user input before executing a database transaction.**

- To minimize the length of a transaction, check for possible errors, such as duplicate keys or insufficient account balance, as early in the transaction as possible so that portions of the database can be released as soon as possible for other users if the transaction is going to be aborted

In some cases..

- Some constraints (e.g., balancing the number of units of an item received with the number placed in inventory less returns) cannot be checked until many database commands are executed, so the transaction must be long to ensure database integrity

The general guideline is to make a database transaction as short as possible while still maintaining the integrity of the database.

- Database administrators must expect and plan for the likelihood that several users will attempt to access and manipulate data at the same time.
- Concurrent processing involving updates
- A database without concurrency control will be compromised due to interference between users.

GCash Incident (November 2024)

- On November 9th, 2024, several GCash users reported unauthorized deductions from their accounts, with some claiming losses amounting to thousands of pesos.
- GCash attributed the issue to "errors" during a system **reconciliation process**.
- While they initially assured users that no funds were lost, they later acknowledged the problem and worked to rectify the balances.
- The incident prompted investigations by the Bangko Sentral ng Pilipinas (BSP) and the National Privacy Commission (NPC).

Reconciliation process

is a crucial accounting procedure where two sets of records are compared to ensure accuracy and consistency.

This process aims to identify and resolve any discrepancies between these records, often involving internal accounts and external statements like bank statements.

Concurrency control

- The process of managing simultaneous operations against a database so that data integrity is maintained and the operations do not interfere with each other in a multi-user environment

Lost Updates

- The most common problem encountered when multiple users attempt to update a database without adequate concurrency control is lost updates.

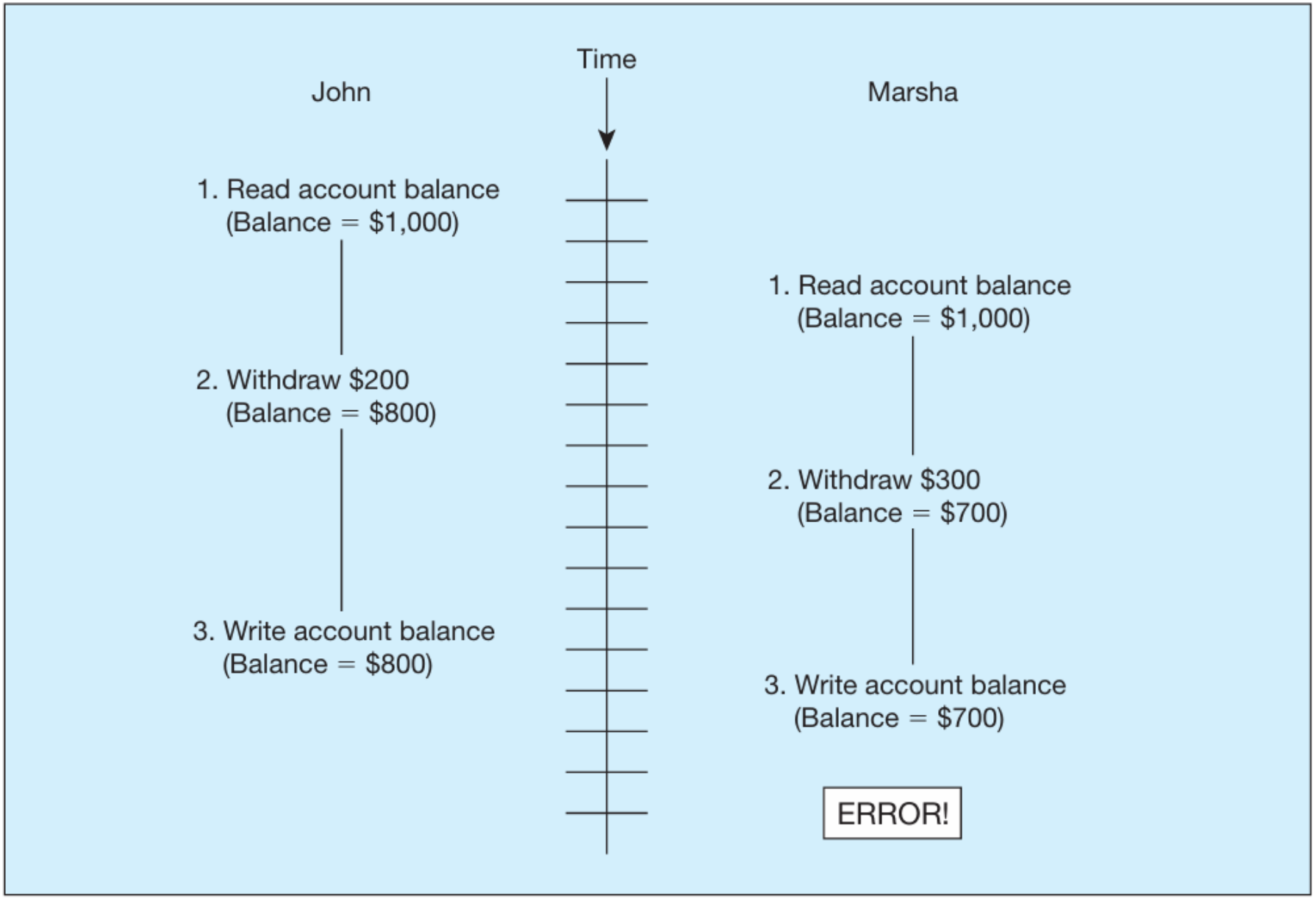


FIGURE 7-20 Lost update (no concurrency control in effect)

Inconsistent read problem

- An unrepeatable read, that occurs when one user reads data that have been partially updated by another user.

- Arises when the DBMS does not isolate transactions

Basic Approaches

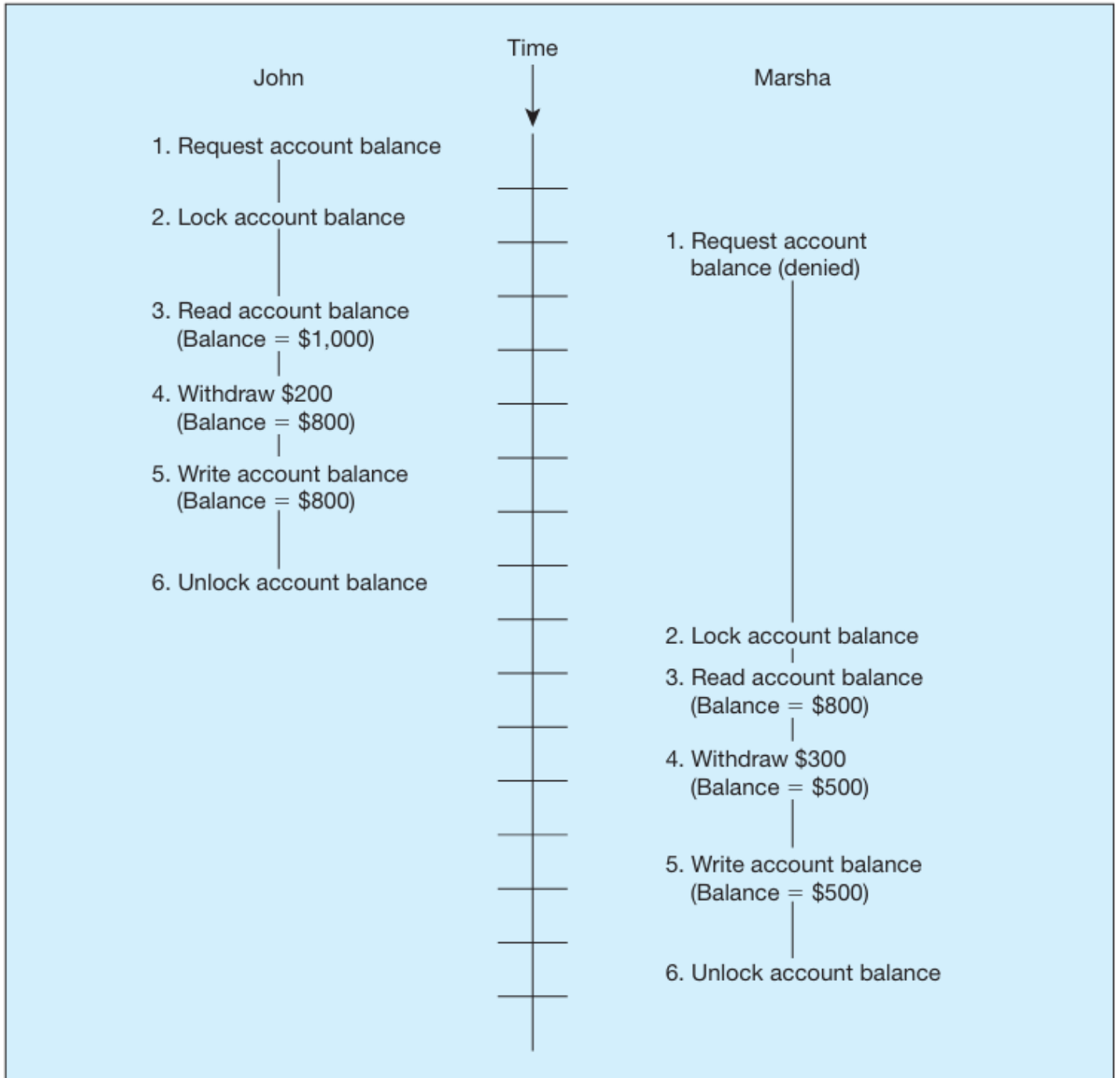
1. Pessimistic approach (involving locking)
2. Optimistic approach (involving versioning).

Locking

- A process in which any data that are retrieved by a user for updating must be locked, or denied to other users, until the update is completed or aborted.

Locking Mechanisms

FIGURE 7-21 Updates with locking (concurrency control)



Locking level (lock granularity)

The extent of a database resource that is included with each lock.

- **Database**. The entire database is locked and becomes unavailable to other users. This level has limited application, such as **during a backup of the entire database** (Rodgers, 1989)
- **Table**. The entire table containing a requested record is locked. This level is **appropriate mainly for bulk updates that will update the entire table, such as giving all employees a two percent raise.**

- **Block or page.** The physical storage block (or page) containing a requested record is locked. **This level is the most commonly implemented locking level**. A page will be a fixed size (4K, 8K, and so forth) and may contain records of more than one type.

- **Record.** Only the requested record (or row) is locked. All other records, even within a table, are available to other users. It does impose some overhead at run time when several records are involved in an update.

- **Field.** Only the particular field (or column) in a requested record is locked. This level may be appropriate when most updates affect only one or two fields in a record. For example, in inventory control applications, the quantity-on-hand field changes frequently, but other fields (e.g., description and bin location) are rarely updated. Field-level locks require considerable overhead and are seldom used.

TYPES OF LOCKS

1. Shared lock (S lock or read lock)

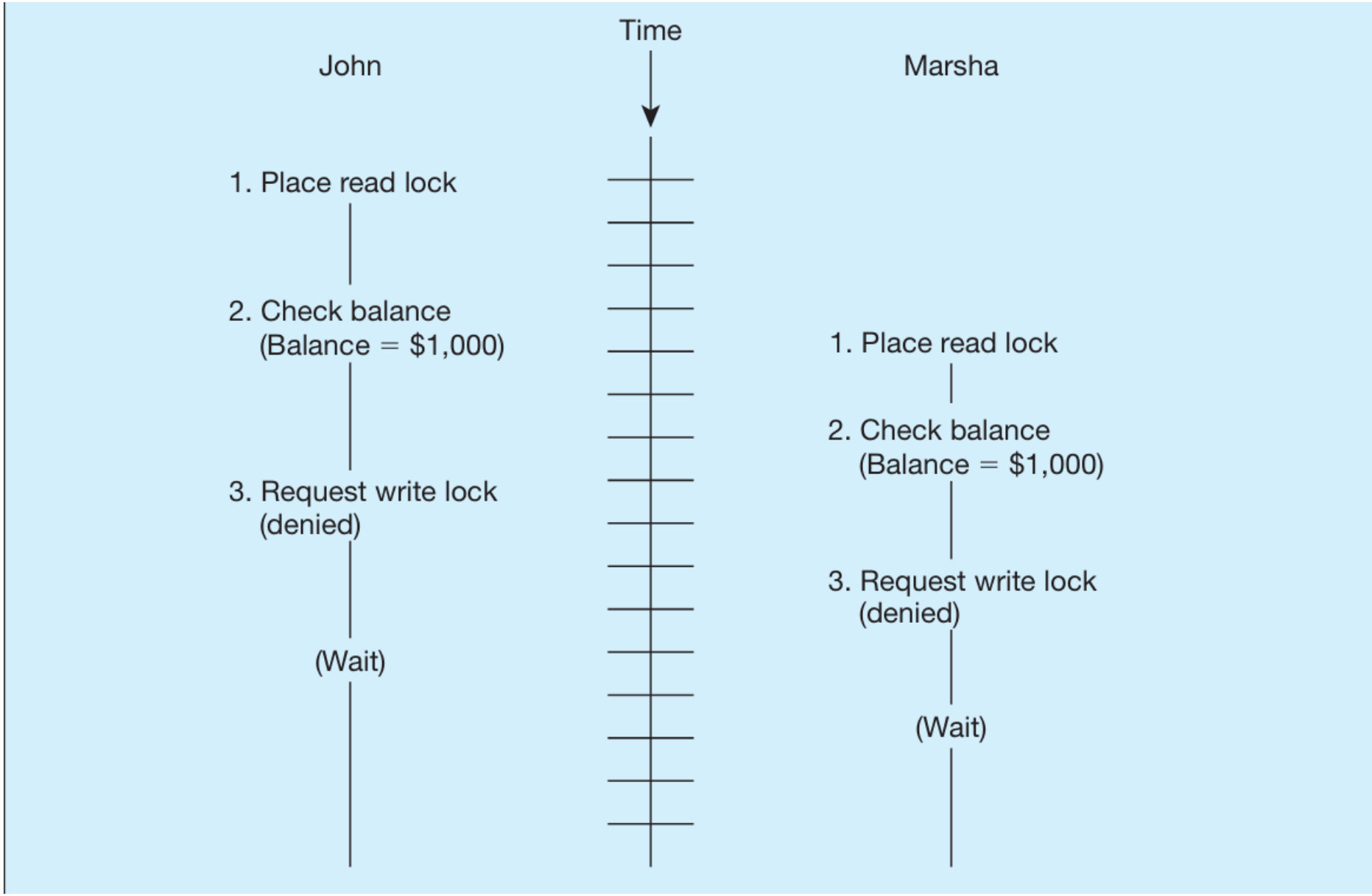
A technique that allows other transactions to read but not update a record or another resource.

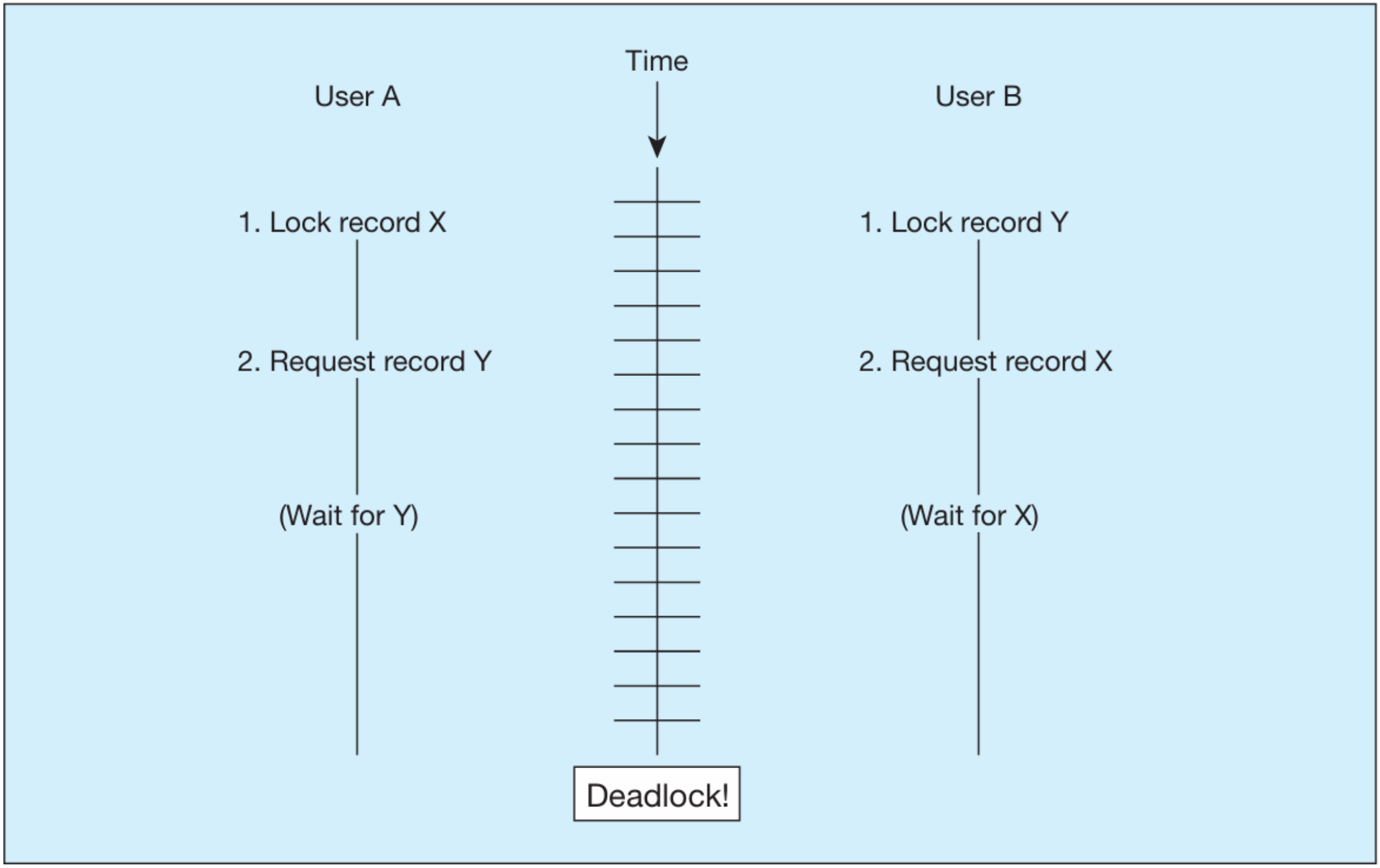
2. **Exclusive lock (X lock or write lock)**

A technique that prevents another transaction from reading and therefore updating a record until it is unlocked.

Deadlock

- An impasse that results when two or more transactions have locked a common resource and each waits for the other to unlock that resource.




Time
User A
User B
1. Lock record X
1. Lock record Y
2. Request record Y
2. Request record X
(Wait for Y)
(Wait for X)
Deadlock!

MANAGING DEADLOCK

1. Deadlock prevention

A method for resolving deadlocks in which user programs must lock all records they require at the beginning of a transaction (rather than one at a time).

2. Deadlock resolution

An approach to dealing with deadlocks that allows deadlocks to occur but builds mechanisms into the DBMS for detecting and breaking the deadlocks.

Versioning

- An approach to concurrency control in which each transaction is restricted to a view of the database as of the time that transaction started, and when a transaction modifies a record, the DBMS creates a new record version instead of overwriting the old record. Hence, no form of locking is required.

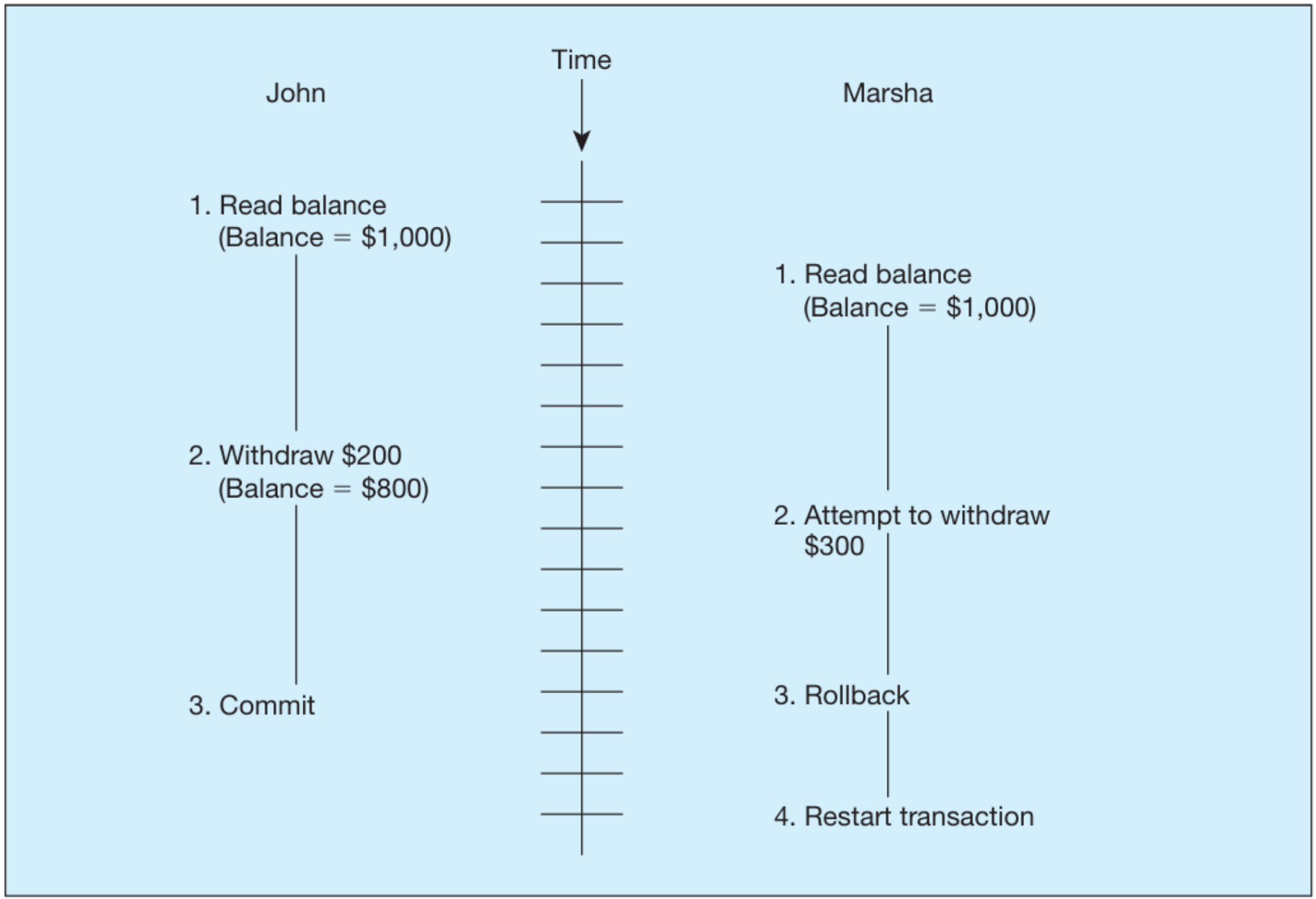


FIGURE 7-24 The use of versioning

Content source:

- Hoffer, J. A., Venkataraman, R., & Topi, H. (2022). *Modern database management* (13th ed., pp. 350–358). Pearson.
- Press Statement on the Alleged GCash Unauthorized Transactions: National Privacy Commission (NPC) https://privacy.gov.ph/press-statement-on-the-alleged-gcash-unauthorized-transactions/
- Statement on GCash Incident: Bangko Sentral ng Pilipinas (BSP) https://www.bsp.gov.ph/SitePages/MediaAndResearch/MediaDisp.aspx?ItemId=7314
- No external hacking in GCash issue, says DICT: ABS-CBN News https://news.abs-cbn.com/business/2024/11/13/no-external-hacking-in-gcash-issue-says-dict-1908
- GCash says missing funds issue fixed; DICT starts probe: BusinessWorld Online https://www.bworldonline.com/corporate/2024/11/11/633883/gcash-says-missing-funds-issue-fixed-dict-starts-probe/
- NPC to conduct data privacy probe on GCash glitch: GMA News Online https://www.gmanetwork.com/news/money/companies/926950/npc-to-conduct-data-privacy-probe-on-gcash-glitch/story/