ACID Documentation

Transaction Scenario 1: Placing an Order

Steps:

1. Insert Order
2. Insert OrderItems
3. Insert Payment

ACID Mapping:

- Atomicity: All operations commit or rollback together
- Consistency: FK constraints ensure valid customer and dish
- Isolation: READ COMMITTED prevents dirty reads
- Durability: DB logs ensure persistence

Transaction Scenario 2: Table Reservation

Steps:

1. Insert Reservation
2. Update DiningTable status

Concurrency Strategy:
Row-level locking on DiningTable prevents double booking.
Serializable isolation level ensures correctness.

Transaction Scenario 3: Employee Shift Assignment

When assigning a shift to an employee, the system must ensure the employee exists and shift times do not overlap incorrectly.

Steps

1. Check employee existence
2. Insert new shift record
3. Validate shift start and end time

ACID Property Mapping

Atomicity:
All operations execute as a single transaction. If shift insertion fails, no partial data is stored.

Consistency:
Foreign key constraints ensure that the shift is assigned to a valid employee. CHECK constraints ensure valid time intervals.

Isolation:
READ COMMITTED isolation level prevents dirty reads during concurrent shift assignments.

Durability:
Once committed, the shift assignment persists even after system crashes due to SQL Server logging mechanisms.

Transaction Scenario 4: Inventory Update After Order

Description

When an order is placed, ingredient quantities must be reduced from inventory. This ensures real-time stock tracking.

Steps

1. Retrieve ingredients for ordered dishes
2. Deduct ingredient quantities from Inventory
3. Commit changes

ACID Property Mapping

Atomicity:
All inventory updates must complete successfully; otherwise, the transaction is rolled back to prevent inconsistent stock levels.

Consistency:
CHECK constraints ensure inventory quantity does not become negative. Foreign keys ensure valid ingredient references.

Isolation:
REPEATABLE READ isolation level ensures consistent stock values during concurrent order processing.

Durability:
Committed inventory updates are permanently stored in the database transaction log.

Concurrency Control Strategy

The system uses transaction isolation levels such as READ COMMITTED and REPEATABLE READ to prevent dirty reads and lost updates. Row-level locking is applied on critical tables such as Order, Reservation, and Inventory to ensure data integrity during concurrent operations.