INTRODUCTION:

A transaction is a logical unit of work that needs to be finished completely or not carried out at all. The University Waste Management System relies on transactional consistency to ensure reliable operations.

SCENARIO 1:

At the end of each month, salary payments are processed for the workers. The system calculates net salary of each worker, inserts record into salary_payment. It also updates worker salary history and marks the payment status as “Paid”. All of these must succeed together.

EXAMPLE:

BEGIN TRANSACTION;

INSERT INTO SALARY_PAYMENT $500;

UPDATE worker;

SET employment_status = ‘Active’;

WHERE worker_id = 12345;

COMMIT;

ACID:

Atomicity: If the salary insertion fails then no record should be created. If salary calculations fails then payment should not be partially stored. If the system crashes during payment then the transaction should roll back. This shows that atomicity ensures all or nothing.

Consistency: The net salary should be equal to basic salary + bonus – deductions, this ensures consistency. The payment status must be one of ‘Paid’, ‘Pending’, ‘Failed’. The foreign key must reference valid worker_id. If any of these constraints fail then the transaction aborts.

Isolation: If two admins attempt to process salary of the same worker then the system prevents duplicate entries for same worker at the same time. The use of composite index ensures uniqueness in the database. Isolation prevents dirty reads and lost updates.

Durability: Once we commit then the salary payment is permanently stored. Even if the server crashes, the record remains. This ensures permanent storage of data.

SCENARIO 2:

When a worker collects waste from a dustbin then waste_collection record is updated. We update the current_fill_level of the dustbin and last_emptied_date. The status of the dustbin is changed to 'Empty'. These all operations must execute together.

EXAMPLE:

BEGIN TRANSACTION;

INSERT INTO WASTE_COLLECTION;

UPDATE dustbin

SET current_fill_level = 0, last_emptied_date = CURRENT_DATE, status = 'Empty'

WHERE dustbin_id = 14567;

COMMIT;

ACID:

Atomicity: If insertion succeeds but dustbin update fails then entire transaction is rolled back. No partial update is allowed. This ensures data correctness.

Consistency: current_fill_level must be greater than or equal to zero. The status must be valid. If violation occurs, the transaction is aborted. This maintains logical correctness.

Isolation: If two workers try to update same dustbin at the same time, then row level locking prevents race conditions. One transaction waits until the other completes. This prevents dirty reads, lost updates and inconsistent fill levels.

Durability: Once committed then waste collection data cannot disappear. The status of the dustbin once updated is permanent.

CONCURRENCY:

We maintain concurrency by using row level locking and foreign keys to prevent the lost updates and dirty reads. We give the concurrent access to dustbin and salary tables which are controlled by using transactional statements. The use of foreign keys helps in insertion of correct worker's data. Composite indexes prevents the duplicate records.

ISOLATION LEVEL JUSTIFICATION:

It prevents dirty reads and non-repeatable reads. This is important because salary data must not change in between a transaction. Isolation is important as the leave approvals must remain stable. We do not serialize because it is too strict and reduces concurrency.

CONCLUSION:

The University Waste Management System ensures data integrity, reliability and correctness by strictly following ACID principles. This guarantees stability and strong referential integrity.