

# Integration Testing

**Course:** Software Testing

**Worked by:** Evisa Nela

**Accepted by:** Ari Gjerazi, Jurgen Cama

## Integration Test Case 2: Supplier Delete Constraint Behavior (Parent–Child Relationship)

### Test Objective

The objective of this integration test is to verify that the system correctly enforces **referential integrity constraints** between the suppliers (parent table) and products (child table). Specifically, the test checks whether a supplier **cannot be deleted** while products referencing that supplier still exist, and that deletion becomes possible only after dependent products are removed.

### Type of Testing

Database Integration Testing

### Integration Method

Decomposition-Based Integration Testing (Bottom-Up)

This method was chosen because the test starts from **lower-level database components** (tables and constraints) and verifies correct behavior before higher-level business logic is considered.

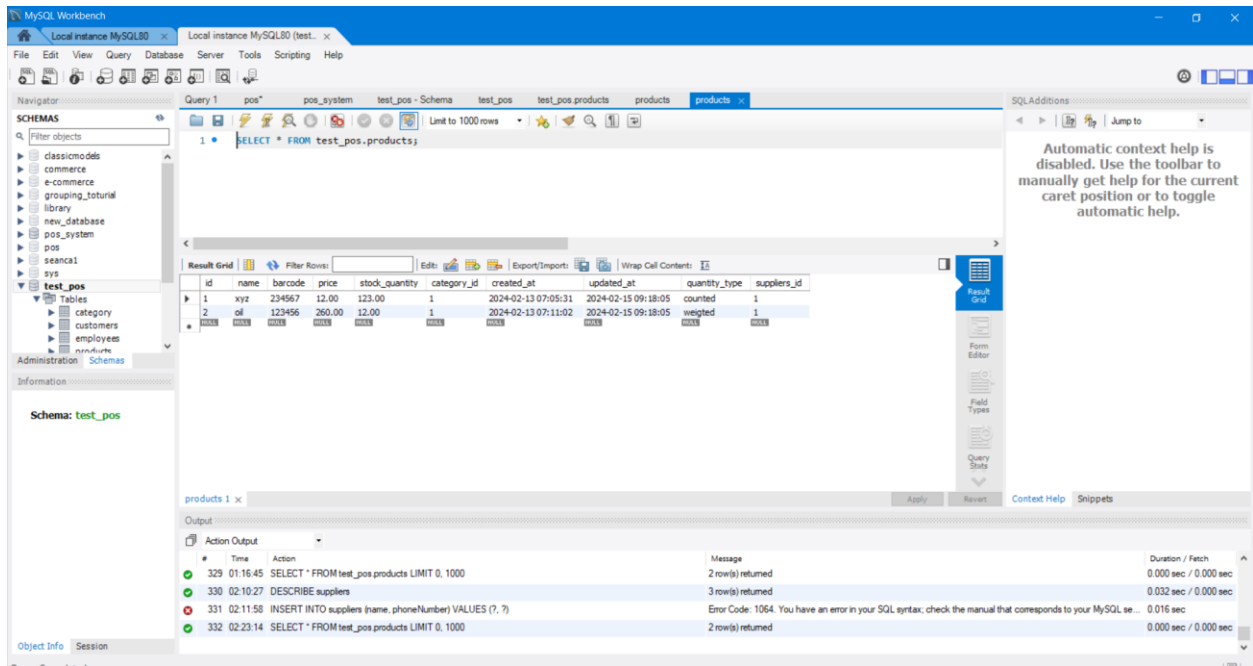
### Integrated Components

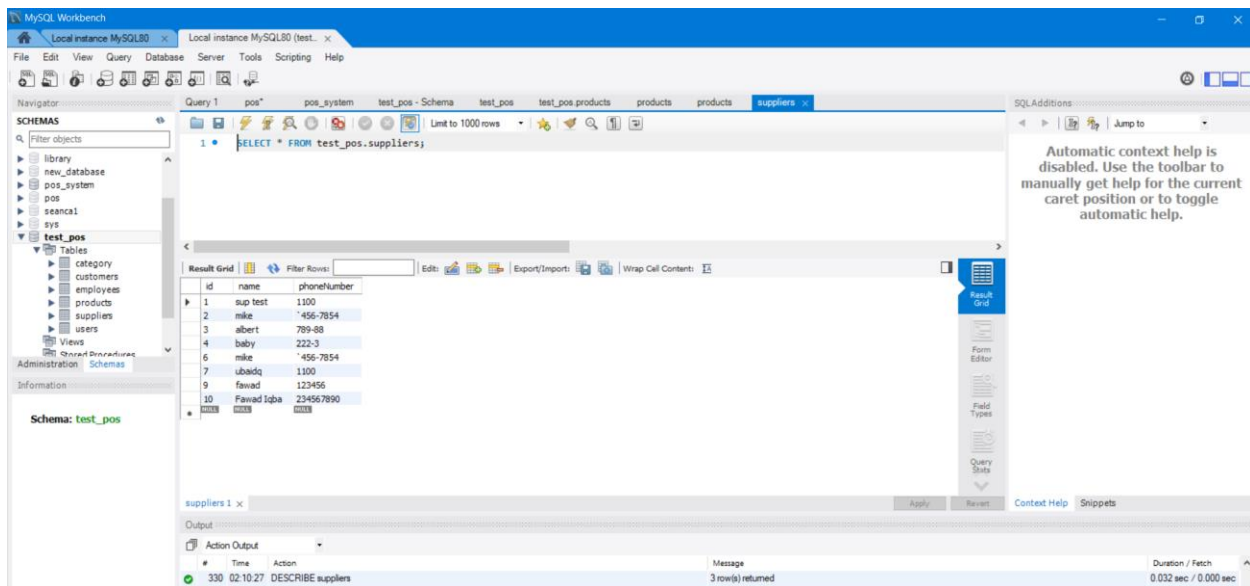
- **Tables:** suppliers, products
- **Database Constraint:** Foreign Key (products.suppliers\_id → suppliers.id)
- **Operations Integrated:**

- Insert supplier
- Insert product linked to supplier
- Attempt supplier deletion
- Delete product
- Delete supplier

## Test Environment

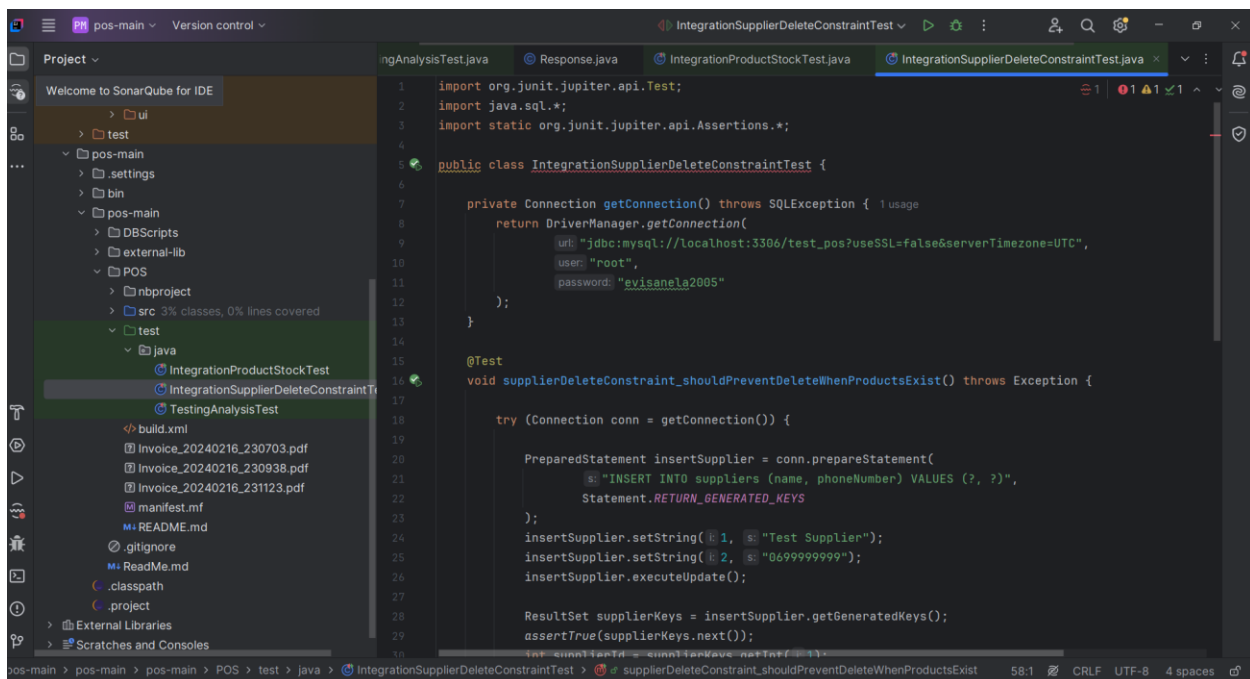
- **Database:** MySQL (Schema: test\_pos)
- **Programming Language:** Java
- **Testing Framework:** JUnit 5
- **Database Access:** JDBC (PreparedStatement)
- **IDE:** IntelliJ IDEA





## Test Flow

1. Insert a new supplier into the suppliers table.
2. Insert a product linked to that supplier using suppliers\_id.
3. Attempt to delete the supplier **while the product still exists**.
4. Verify that deletion is **prevented by the foreign key constraint**.
5. Delete the product referencing the supplier.
6. Delete the supplier successfully after dependency removal.





## Actual Result

- When attempting to delete the supplier while a product existed, the database **prevented the deletion**, enforcing referential integrity.
- After deleting the product, the supplier was deleted successfully.
- The behavior confirms that the foreign key constraint is working as expected.

The screenshot displays the IntelliJ IDEA IDE interface. The left sidebar shows a project structure with folders like 'pos-main', 'pos', 'ui', 'test', 'bin', 'DBScripts', 'external-lib', 'POS', 'nbproject', 'src', and 'test'. The main editor window shows the code for 'IntegrationSupplierDeleteConstraintTest.java'. The code includes imports for JUnit and JDBC, a private method 'getConnection()' that returns a database connection, and a test method 'supplierDeleteConstraint\_shouldPreventDeleteWhenProductsExist()' which uses the connection to execute a SQL statement. The bottom panel shows the 'Run' output, indicating that the test passed successfully with an exit code of 0.

```
1 import org.junit.jupiter.api.Test;
2 import java.sql.*;
3 import static org.junit.jupiter.api.Assertions.*;
4
5 public class IntegrationSupplierDeleteConstraintTest {
6
7     private Connection getConnection() throws SQLException {
8         return DriverManager.getConnection(
9             url: "jdbc:mysql://localhost:3306/test_pos?useSSL=false&serverTimezone=UTC",
10            user: "root",
11            password: "evisanela2005"
12        );
13    }
14
15    @Test
16    void supplierDeleteConstraint_shouldPreventDeleteWhenProductsExist() throws Exception {
17
18        try (Connection conn = getConnection()) {
19
20            PreparedStatement insertSupplier = conn.prepareStatement(
21                "INSERT INTO suppliers (name, phone_number) VALUES (?, ?)"
22            );
23            insertSupplier.setString(1, "Supplier 1");
24            insertSupplier.setString(2, "123456789");
25            insertSupplier.executeUpdate();
26
27            PreparedStatement insertProduct = conn.prepareStatement(
28                "INSERT INTO products (supplier_id, product_name, price) VALUES (?, ?, ?)"
29            );
30            insertProduct.setInt(1, 1);
31            insertProduct.setString(2, "Product 1");
32            insertProduct.setDouble(3, 10.0);
33            insertProduct.executeUpdate();
34
35            // Attempt to delete the supplier
36            PreparedStatement deleteSupplier = conn.prepareStatement(
37                "DELETE FROM suppliers WHERE id = 1"
38            );
39            deleteSupplier.executeUpdate();
40
41            // Verify that the supplier was not deleted
42            PreparedStatement selectSupplier = conn.prepareStatement(
43                "SELECT * FROM suppliers WHERE id = 1"
44            );
45            ResultSet rs = selectSupplier.executeQuery();
46            assertTrue(rs.next());
47
48            // Clean up
49            PreparedStatement deleteProduct = conn.prepareStatement(
50                "DELETE FROM products WHERE supplier_id = 1"
51            );
52            deleteProduct.executeUpdate();
53
54            // Finally delete the supplier
55            deleteSupplier.executeUpdate();
56
57            // Verify that the supplier was deleted
58            rs = selectSupplier.executeQuery();
59            assertFalse(rs.next());
60
61            conn.close();
62        }
63    }
64}
```

Run: IntegrationSupplierDeleteConstraintTest x

IntegrationSupplierDeleteConstraintTest 588 ms ✓ Tests passed: 1 of 1 test - 588 ms

supplierDeleteConstraint\_shouldPreventDeleteWhenProductsExist() 588 ms

Process finished with exit code 0

## Test Result

PASS

The test successfully validated correct enforcement of parent–child constraints between the suppliers and products tables.

## Observations

- The test confirms that **database-level constraints** protect data integrity even without additional application-level validation.
- This behavior prevents orphan records and ensures consistency between related entities.
- Proper test data setup (including mandatory fields such as phoneNumber) is required to avoid insertion errors.

Step	Operation	Expected Outcome	Actual Outcome
1	Insert supplier	Supplier added	Supplier added
2	Insert product with supplier ID	Product added	Product added
3	Delete supplier	Operation blocked	Blocked by FK
4	Delete product	Product removed	Product removed
5	Delete supplier	Supplier removed	Supplier removed

## Conclusion

This integration test demonstrates that the system correctly applies **foreign key constraints** to manage parent–child relationships. By using a **Bottom-Up integration strategy**, the test confirms reliable database behavior and ensures that invalid delete operations are safely prevented.