VIETNAM NATIONAL UNIVERSITY HCMC
UNIVERSITY OF INFORMATION TECHNOLOGY

Subject: Distributed Database System

# Topic: Database Management of Pen Stock

Lecturer: Assoc. Dr. Do Phuc

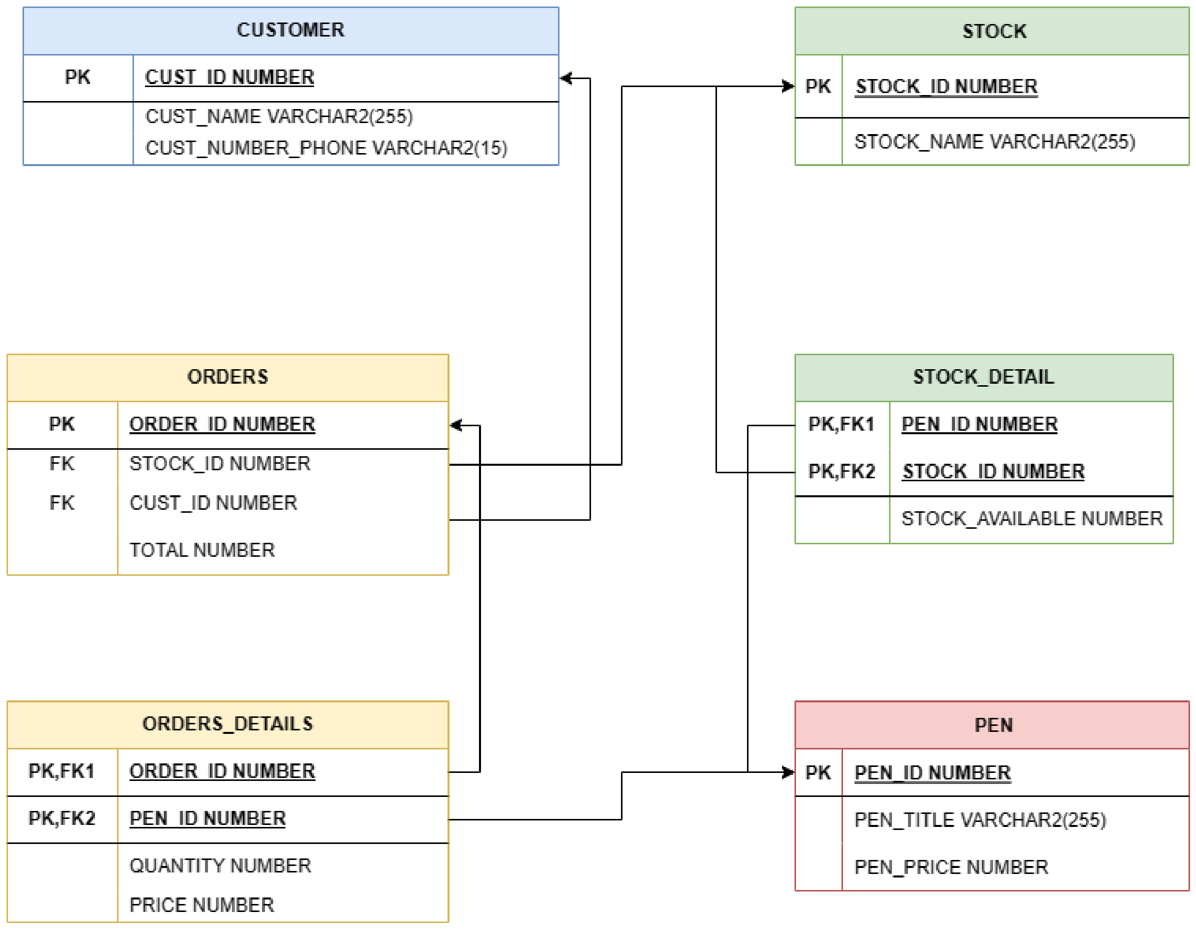
Lecturer: MSc. Nguyen Thi Kim Phung

Members:

- Phạm Khánh Hòa - 19521519
- Phạm Thuỳ Dung- 20521214

# Table Of Contents

## 1. Global Database Tables



| Table | Attributes |
|---|---|
| **customer** | • cust_id number<br>• cust_city_id number<br>• cust_name varchar2(255 byte)<br>• cust_number_phone varchar2(15 byte) |
| **orders** | • order_id number<br>• stock_id number<br>• cust_id number<br>• total number<br>• order_date date |

| | |
|---|---|
| **order detail** | • order_id number<br>• pen_idnumber<br>• quatity number<br>• price unmber |
| **pen** | • pen_id number<br>• pen_title varchar2(255 byte)<br>• pen_price number |
| **stock** | • stock_id number<br>• stock_city_id number<br>• stock_name varchar2(255 byte) |
| **stock detail** | • pen_idnumber<br>• stock_id number<br>• stock_available number |

*Table 1. the global tables*

Table Description

- **Stock ( id number, city_id number, name varchar2(255), available number, capacity number)**

| Item | Description | Type | Value |
|---|---|---|---|
| id (key) | The ID of stock | Integer | |
| Name | The name of stock | String | |

*Table 2. stock description*

- **Pen (id number, title varchar2(255), price number)**

| Item | Description | Type | Value |
|---|---|---|---|

| | | | |
|---|---|---|---|
| **ID** | The ID of pen | Integer | |
| **Title** | Name of pen | String | - |
| **Price** | Price of pen | Integer | >0 |

*Table 3. pen description*

- **Stock_detail (pen_id number, stock_id number, stock_available number)**

| Item | Description | Type | Value |
|---|---|---|---|
| Pen_id | The ID of customer placing the pen order | Integer | pen.id |
| stock_id | The ID of a ordered pen | Integer | stock.id |
| stock_available | Amount of ordered pen | Integer | - |

*Table 4. stock detail description*

- **Customer (Id number , name varchar2(255), city_id number, number_phone varchar2(15))**

| Item | Description | Type | Value |
|---|---|---|---|
| id | The ID of customer | Integer | customer.id |
| name | The name of customer | String | - |
| score | Points that customers accumulate | Integer | Score >0 |

*Table 5. customer descriptions*

- **Orders (order_id number, stock_id number, cust_id number, total number, order_date date default sysdate not null))**

| Item | Description | Type | Value |
|---|---|---|---|
| Id (key) | The ID of order | Integer | |
| Stock_id | The ID of stock export product for this order | Integer | Stock.id |
| Cust_id | The ID of customer who order a pen | Integer | Customer.id |
| Total | The total money of order | Integer | - |
| Order_date | The date create order | Date | - |

*Table 6. order description*

- **Orders_details (order_id number, pen_id number, quatity number, price number)**

| Item | Description | Type | Value |
|---|---|---|---|
| Order_id | The ID of customer who order a pen | Integer | order.id |
| Pen_id | The ID of a ordered pen | Integer | pen.id |
| Quantity | Amount of ordered pens | Integer | - |
| Price | Price of ordered pen | Integer | - |

*Table 7. order detail description*

## 2. Fragmentation

- **Stock ( id number, city_id number, name varchar2(255), available number, capacity number)**

| Fragmentation Name | Fragmentation Condition |
|---|---|
| Stock.1 | 0 <= id <=2 |
| Stock.2 | 2<id<=4 |

*Table 8. Horizontal Fragmentation of Table "Stock"*

- **Customer (Id number , name varchar2(255), city_id number, number_phone varchar2(15), score number)**

| Fragmentation Name | Fragmentation Condition |
|---|---|
| customer.1 | 0 < id <= 100 |
| customer.2 | 100< id <= 200 |

*Table 10. Horizontal Fragmentation of Table "customer"*

- **Orders (order_id number, stock_id number, brand_store_id number, cust_id number, total number, order_date date default sysdate not null))**

| Fragmentation Name | Fragmentation Condition |
|---|---|
| orders.1 | **id** <= 100 |
| orders.2 | 100 < **id** <= 200 |

*Table 18. Horizontal Fragmentation of Table "orders"*

## 3. Allocation

Site Configuration: 2 sites deployed at 2 computers.

| At Site Name | Fragmentation Name |
|---|---|
| DB1 at Site 1 | • Stock.1<br>• pen_id.1<br>• customer.1<br>• orders.1 |
| DB2 at Site 2 | • Stock.2<br>• pen_id.2<br>• customer.2<br>• orders.2 |

## 4. Functions

### 4.1 Trigger

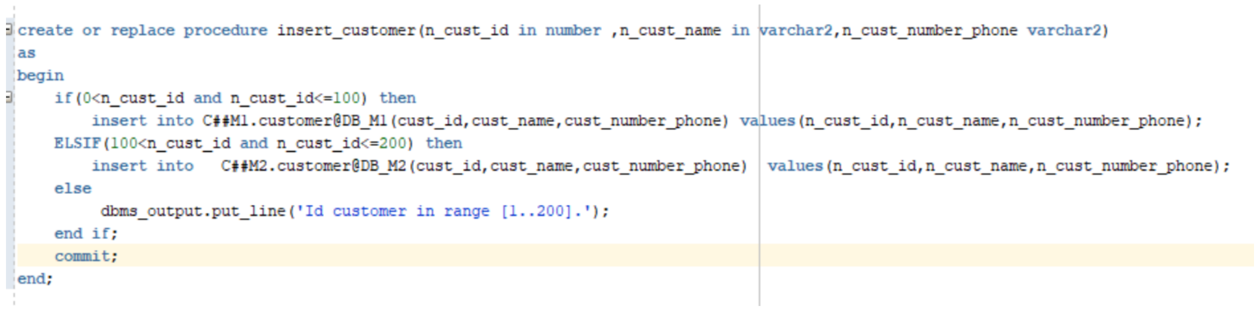
It Includes: (Detail in the code file)

#### 4.1.1 Order Detail Before Delete
#### 4.1.2 Order Detail After Delete
#### 4.1.3 Order Detail Before Insert
#### 4.1.4 Order Detail After Insert
#### 4.1.5 Order Detail After Update

### 4.2 Procedure

It includes:

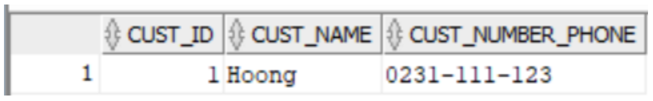
#### 4.2.1 Insert Customer



```
create or replace procedure insert_customer(n_cust_id in number ,n_cust_name in varchar2,n_cust_number_phone varchar2)
as
begin
    if(0<n_cust_id and n_cust_id<=100) then
        insert into C##M1.customer@DB_M1(cust_id,cust_name,cust_number_phone) values(n_cust_id,n_cust_name,n_cust_number_phone);
    ELSIF(100<n_cust_id and n_cust_id<=200) then
        insert into   C##M2.customer@DB_M2(cust_id,cust_name,cust_number_phone)  values(n_cust_id,n_cust_name,n_cust_number_phone);
    else
         dbms_output.put_line('Id customer in range [1..200].');
    end if;
    commit;
end;
```

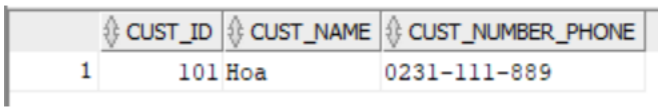
Result:

Site 1:



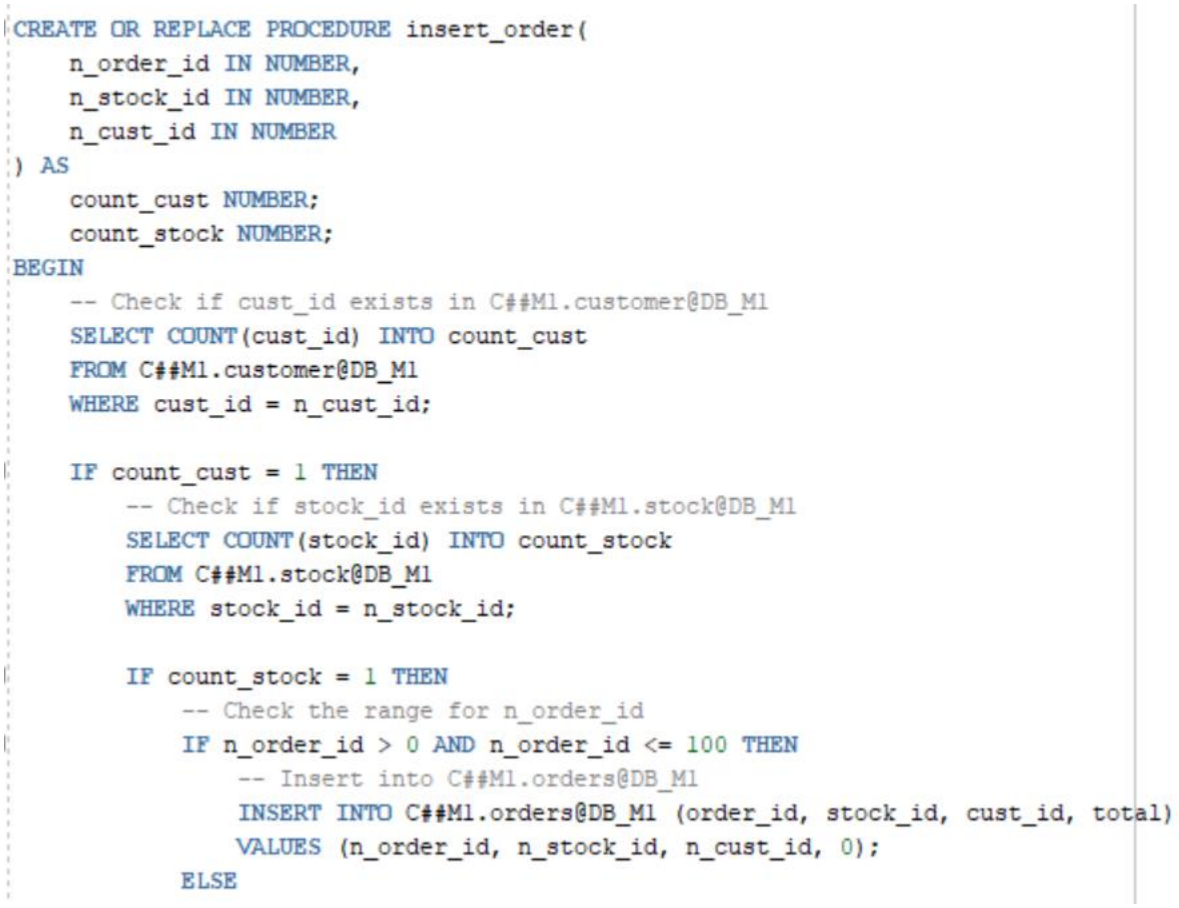
| | CUST_ID | CUST_NAME | CUST_NUMBER_PHONE |
|---|---|---|---|
| 1 | 1 | Hoong | 0231-111-123 |

Site 2:



| | CUST_ID | CUST_NAME | CUST_NUMBER_PHONE |
|---|---|---|---|
| 1 | 101 | Hoa | 0231-111-889 |

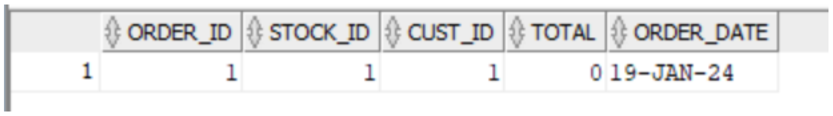
### 4.2.2 Insert Order



```
CREATE OR REPLACE PROCEDURE insert_order(
    n_order_id IN NUMBER,
    n_stock_id IN NUMBER,
    n_cust_id IN NUMBER
) AS
    count_cust NUMBER;
    count_stock NUMBER;
BEGIN
    -- Check if cust_id exists in C##M1.customer@DB_M1
    SELECT COUNT(cust_id) INTO count_cust
    FROM C##M1.customer@DB_M1
    WHERE cust_id = n_cust_id;

    IF count_cust = 1 THEN
        -- Check if stock_id exists in C##M1.stock@DB_M1
        SELECT COUNT(stock_id) INTO count_stock
        FROM C##M1.stock@DB_M1
        WHERE stock_id = n_stock_id;

        IF count_stock = 1 THEN
            -- Check the range for n_order_id
            IF n_order_id > 0 AND n_order_id <= 100 THEN
                -- Insert into C##M1.orders@DB_M1
                INSERT INTO C##M1.orders@DB_M1 (order_id, stock_id, cust_id, total)
                VALUES (n_order_id, n_stock_id, n_cust_id, 0);
            ELSE
```

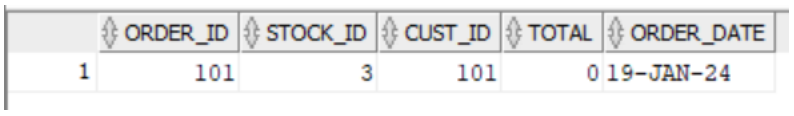
(Detail in the code file)

Site 1:



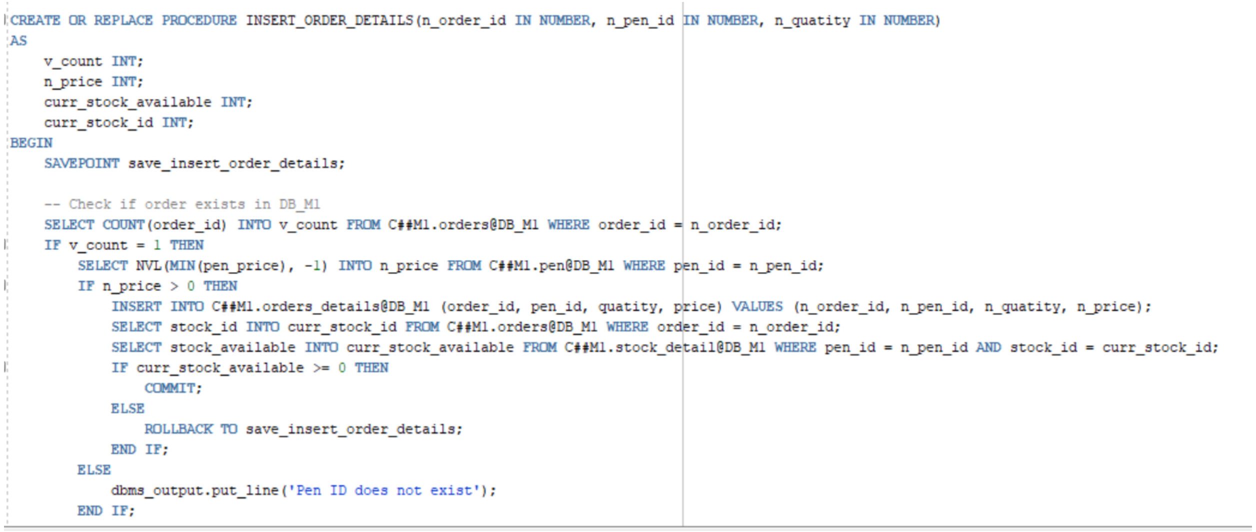
| | ORDER_ID | STOCK_ID | CUST_ID | TOTAL | ORDER_DATE |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 19-JAN-24 |

Site 2:



| | ORDER_ID | STOCK_ID | CUST_ID | TOTAL | ORDER_DATE |
|---|---|---|---|---|---|
| 1 | 101 | 3 | 101 | 0 | 19-JAN-24 |

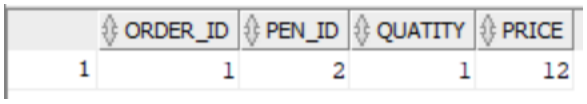
### 4.2.3 Insert Order Detail



```sql
CREATE OR REPLACE PROCEDURE INSERT_ORDER_DETAILS(n_order_id IN NUMBER, n_pen_id IN NUMBER, n_quatity IN NUMBER)
AS
    v_count INT;
    n_price INT;
    curr_stock_available INT;
    curr_stock_id INT;
BEGIN
    SAVEPOINT save_insert_order_details;

    -- Check if order exists in DB_M1
    SELECT COUNT(order_id) INTO v_count FROM C##M1.orders@DB_M1 WHERE order_id = n_order_id;
    IF v_count = 1 THEN
        SELECT NVL(MIN(pen_price), -1) INTO n_price FROM C##M1.pen@DB_M1 WHERE pen_id = n_pen_id;
        IF n_price > 0 THEN
            INSERT INTO C##M1.orders_details@DB_M1 (order_id, pen_id, quatity, price) VALUES (n_order_id, n_pen_id, n_quatity, n_price);
            SELECT stock_id INTO curr_stock_id FROM C##M1.orders@DB_M1 WHERE order_id = n_order_id;
            SELECT stock_available INTO curr_stock_available FROM C##M1.stock_detail@DB_M1 WHERE pen_id = n_pen_id AND stock_id = curr_stock_id;
            IF curr_stock_available >= 0 THEN
                COMMIT;
            ELSE
                ROLLBACK TO save_insert_order_details;
            END IF;
        ELSE
            dbms_output.put_line('Pen ID does not exist');
        END IF;
```

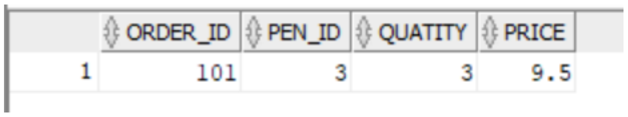
(Detail in the code file)

Site 1:



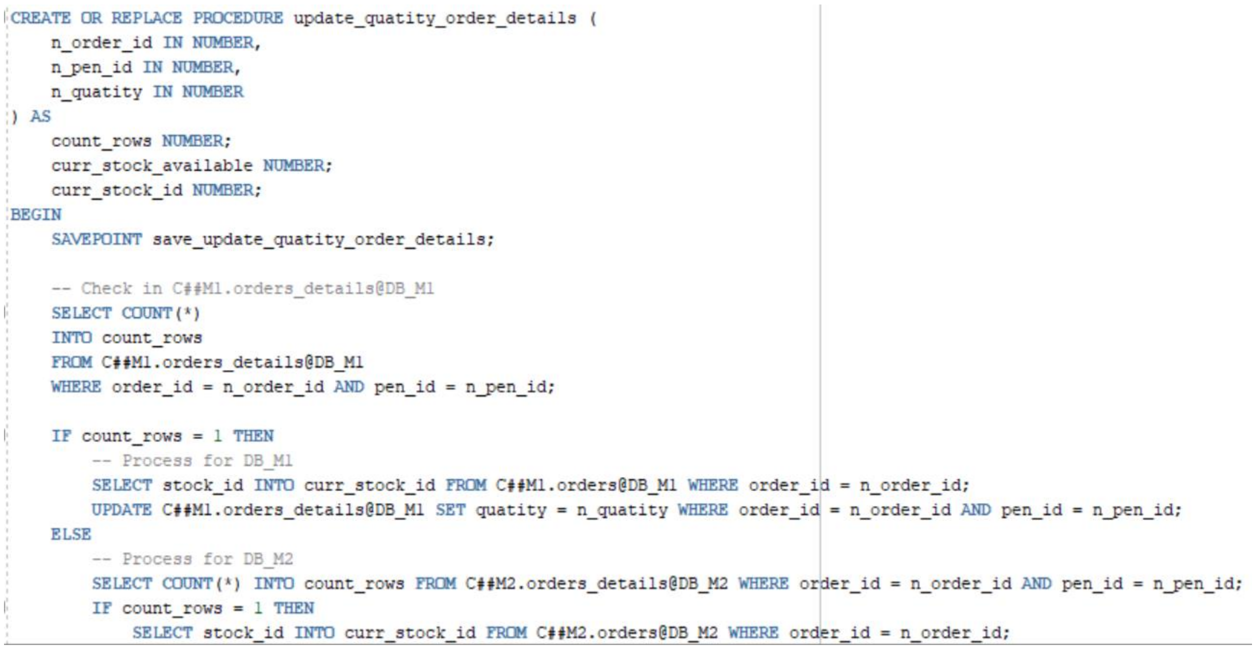
| | ORDER_ID | PEN_ID | QUATITY | PRICE |
|---|---|---|---|---|
| 1 | 1 | 2 | 1 | 12 |

Site 2:



| | ORDER_ID | PEN_ID | QUATITY | PRICE |
|---|---|---|---|---|
| 1 | 101 | 3 | 3 | 9.5 |

### 4.2.4 Update Quantity in Order detail

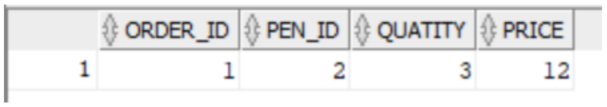


```
CREATE OR REPLACE PROCEDURE update_quatity_order_details (
    n_order_id IN NUMBER,
    n_pen_id IN NUMBER,
    n_quatity IN NUMBER
) AS
    count_rows NUMBER;
    curr_stock_available NUMBER;
    curr_stock_id NUMBER;
BEGIN
    SAVEPOINT save_update_quatity_order_details;

    -- Check in C##M1.orders_details@DB_M1
    SELECT COUNT(*)
    INTO count_rows
    FROM C##M1.orders_details@DB_M1
    WHERE order_id = n_order_id AND pen_id = n_pen_id;

    IF count_rows = 1 THEN
        -- Process for DB_M1
        SELECT stock_id INTO curr_stock_id FROM C##M1.orders@DB_M1 WHERE order_id = n_order_id;
        UPDATE C##M1.orders_details@DB_M1 SET quatity = n_quatity WHERE order_id = n_order_id AND pen_id = n_pen_id;
    ELSE
        -- Process for DB_M2
        SELECT COUNT(*) INTO count_rows FROM C##M2.orders_details@DB_M2 WHERE order_id = n_order_id AND pen_id = n_pen_id;
        IF count_rows = 1 THEN
            SELECT stock_id INTO curr_stock_id FROM C##M2.orders@DB_M2 WHERE order_id = n_order_id;
```

(Detail will show in the code file)

Result:

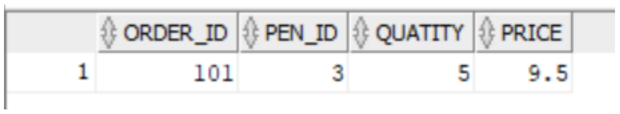
Site 1:



| | ORDER_ID | PEN_ID | QUATITY | PRICE |
|---|---|---|---|---|
| 1 | 1 | 2 | 3 | 12 |

Pen Id 2 with quatity 1 update 3

Site 2:

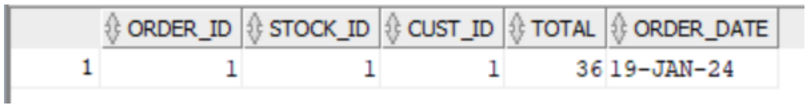


| | ORDER_ID | PEN_ID | QUATITY | PRICE |
|---|---|---|---|---|
| 1 | 101 | 3 | 5 | 9.5 |

Pen Id 3 with quantity 3 updates to 5

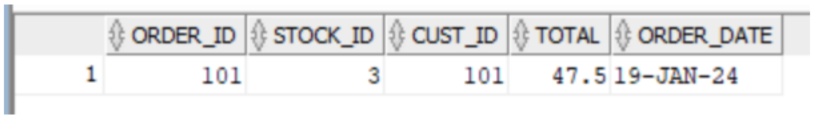
And here is total price in order

Site 1:



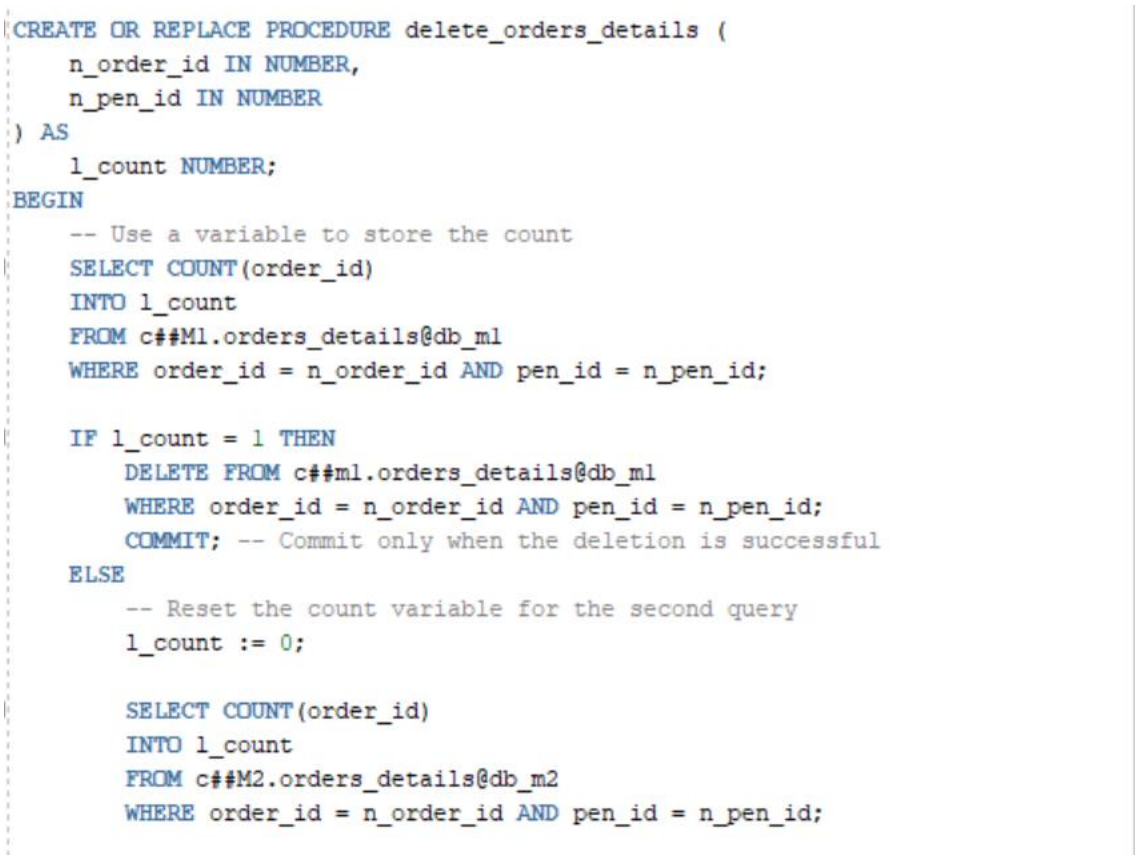
| | ORDER_ID | STOCK_ID | CUST_ID | TOTAL | ORDER_DATE |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 36 | 19-JAN-24 |

Site 2:



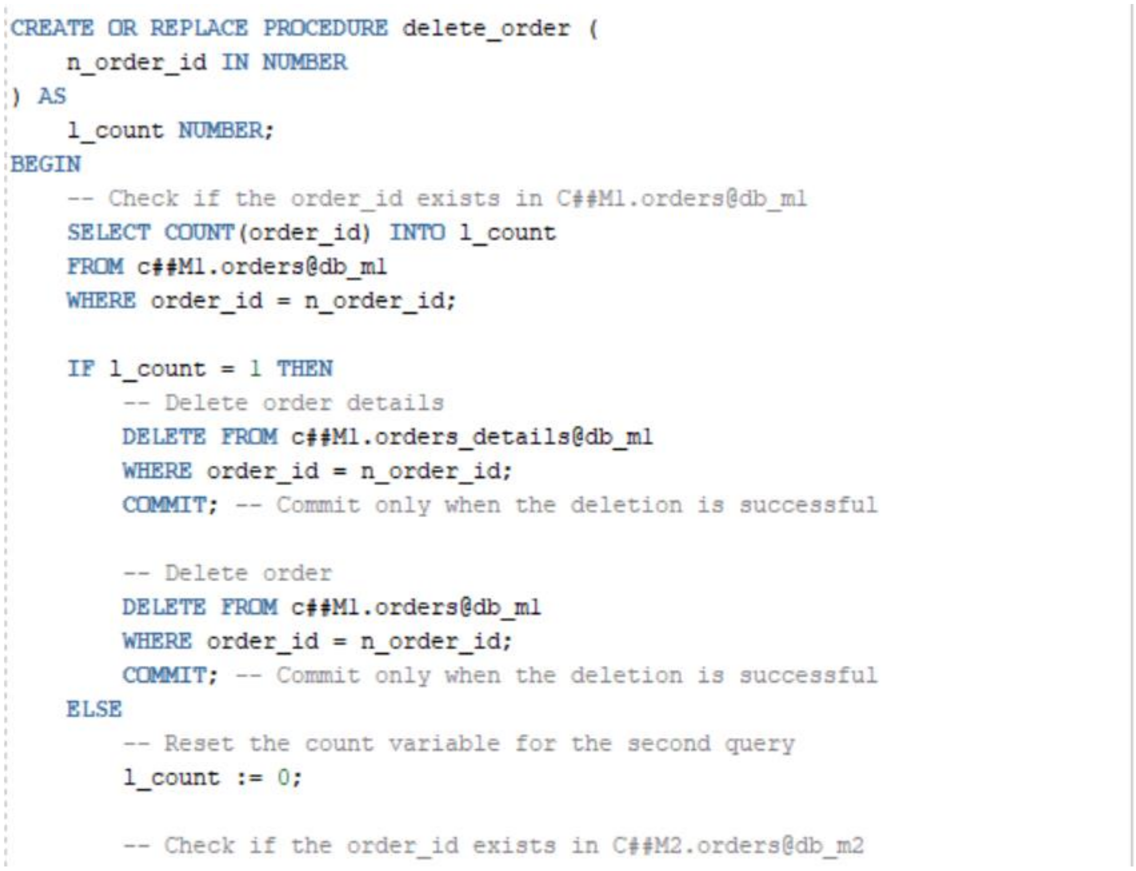
| | ORDER_ID | STOCK_ID | CUST_ID | TOTAL | ORDER_DATE |
|---|---|---|---|---|---|
| 1 | 101 | 3 | 101 | 47.5 | 19-JAN-24 |

### 4.2.5 Delete Order Detail



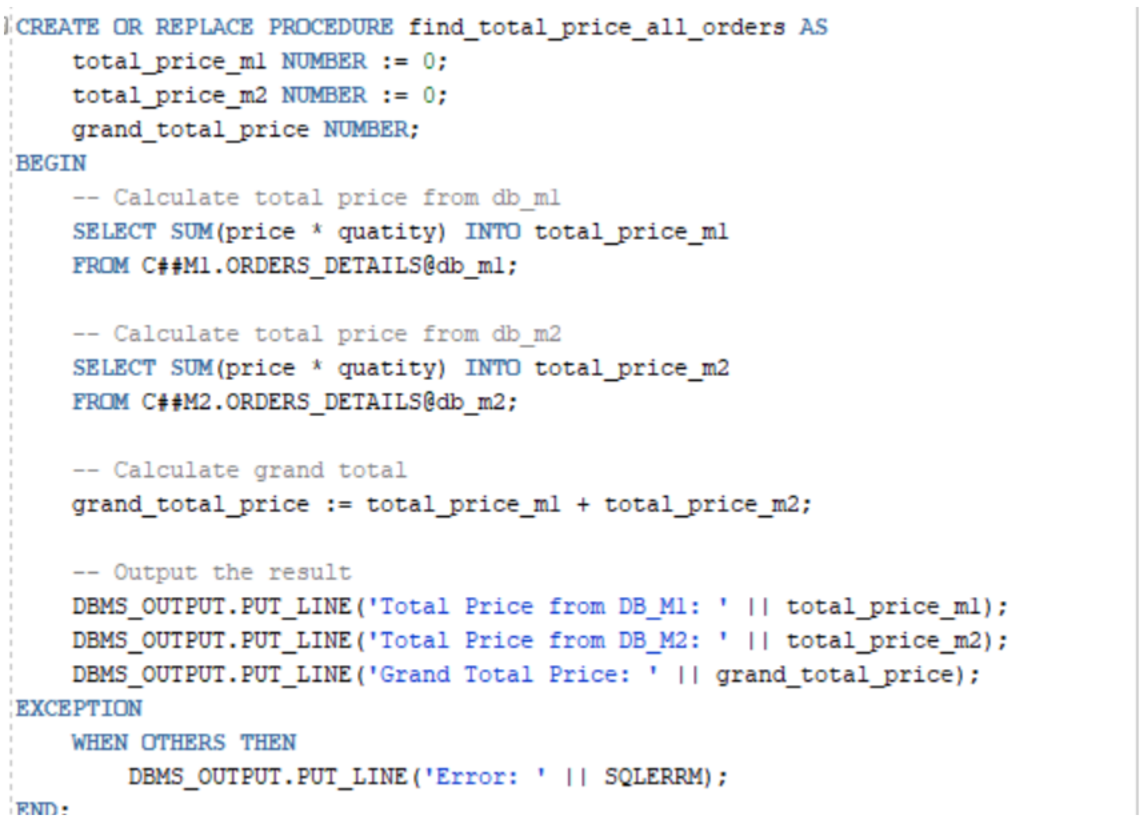
```
CREATE OR REPLACE PROCEDURE delete_orders_details (
    n_order_id IN NUMBER,
    n_pen_id IN NUMBER
) AS
    l_count NUMBER;
BEGIN
    -- Use a variable to store the count
    SELECT COUNT(order_id)
    INTO l_count
    FROM c##M1.orders_details@db_m1
    WHERE order_id = n_order_id AND pen_id = n_pen_id;

    IF l_count = 1 THEN
        DELETE FROM c##m1.orders_details@db_m1
        WHERE order_id = n_order_id AND pen_id = n_pen_id;
        COMMIT; -- Commit only when the deletion is successful
    ELSE
        -- Reset the count variable for the second query
        l_count := 0;

        SELECT COUNT(order_id)
        INTO l_count
        FROM c##M2.orders_details@db_m2
        WHERE order_id = n_order_id AND pen_id = n_pen_id;
```

### 4.2.6 Delete Order



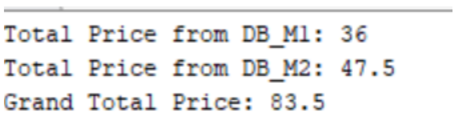
```
CREATE OR REPLACE PROCEDURE delete_order (
    n_order_id IN NUMBER
) AS
    l_count NUMBER;
BEGIN
    -- Check if the order_id exists in C##M1.orders@db_m1
    SELECT COUNT(order_id) INTO l_count
    FROM c##M1.orders@db_m1
    WHERE order_id = n_order_id;

    IF l_count = 1 THEN
        -- Delete order details
        DELETE FROM c##M1.orders_details@db_m1
        WHERE order_id = n_order_id;
        COMMIT; -- Commit only when the deletion is successful

        -- Delete order
        DELETE FROM c##M1.orders@db_m1
        WHERE order_id = n_order_id;
        COMMIT; -- Commit only when the deletion is successful
    ELSE
        -- Reset the count variable for the second query
        l_count := 0;

        -- Check if the order_id exists in C##M2.orders@db_m2
```

## 4.2.7 Find Total Price all order order from site 1 and site 2



```
CREATE OR REPLACE PROCEDURE find_total_price_all_orders AS
    total_price_m1 NUMBER := 0;
    total_price_m2 NUMBER := 0;
    grand_total_price NUMBER;
BEGIN
    -- Calculate total price from db_m1
    SELECT SUM(price * quatity) INTO total_price_m1
    FROM C##M1.ORDERS_DETAILS@db_m1;

    -- Calculate total price from db_m2
    SELECT SUM(price * quatity) INTO total_price_m2
    FROM C##M2.ORDERS_DETAILS@db_m2;

    -- Calculate grand total
    grand_total_price := total_price_m1 + total_price_m2;

    -- Output the result
    DBMS_OUTPUT.PUT_LINE('Total Price from DB_M1: ' || total_price_m1);
    DBMS_OUTPUT.PUT_LINE('Total Price from DB_M2: ' || total_price_m2);
    DBMS_OUTPUT.PUT_LINE('Grand Total Price: ' || grand_total_price);
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
```

Result



```
Total Price from DB_M1: 36
Total Price from DB_M2: 47.5
Grand Total Price: 83.5
```