**CSCI235: Database Systems**

**Assignment 2**

**Nicholas Monteleone 5055076**

**Task 3**



|  |  |
| --- | --- |
| Transaction 1 | Transaction 2 |
| SELECT NVL(MAX(unit\_price), 0)  INTO max\_unit\_price  FROM PRODUCT; |  |
| UPDATE PRODUCT  SET unit\_price = unit\_price + 0.01\*max\_unit\_price  WHERE units\_in\_stock > 60; |  |
|  | UPDATE PRODUCT  SET units\_in\_stock = units\_in\_stock – 60  WHERE units\_in\_stock > 60; |
|  | COMMIT; |
| UPDATE PRODUCT  SET unit\_price = unit\_price + 0.02\*max\_unit\_price  WHERE units\_in\_stock <= 60; |  |
| COMMIT; |  |

Some product prices are increased two times because the units\_in\_stock value is decreased after the price is raised, and then raised again due to the low stock level.



SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

DECLARE

max\_unit\_price NUMBER(9);

current\_unit\_stock NUMBER(9);

BEGIN

SELECT NVL(MAX(unit\_price), 0)

INTO max\_unit\_price

FROM PRODUCT;

SELECT units\_in\_stock

INTO current\_unit\_stock

FROM PRODUCT;

UPDATE PRODUCT

SET unit\_price = unit\_price + 0.01\*max\_unit\_price

WHERE current\_unit\_stock > 60;

UPDATE PRODUCT

SET unit\_price = unit\_price + 0.02\*max\_unit\_price

WHERE current\_unit\_stock <= 60;

COMMIT;

END;

/

|  |  |
| --- | --- |
| Transaction 1 | Transaction 2 |
| SELECT NVL(MAX(unit\_price), 0)  INTO max\_unit\_price  FROM PRODUCT; |  |
| SELECT units\_in\_stock  INTO current\_unit\_stock  FROM PRODUCT; |  |
| UPDATE PRODUCT  SET unit\_price = unit\_price + 0.01\*max\_unit\_price  WHERE current\_unit\_stock > 60; |  |
|  | UPDATE PRODUCT  SET units\_in\_stock = units\_in\_stock – 60  WHERE units\_in\_stock > 60; |
|  | COMMIT; |
| UPDATE PRODUCT  SET unit\_price = unit\_price + 0.02\*max\_unit\_price  WHERE current\_unit\_stock <= 60; |  |
| COMMIT; |  |

Value is read into a temporary value one time, so that each statement is consistently updated regardless of external values. Schedule is now conflict serializable, meaning no issues in transaction order will occur.