2017-12-5

Used Furniture Store

Team members:

Tianrou Chang – txc172430

Xiaoyu Zhang – xxz173130

Yidan Sheng – yxs173130

CS6360.003 Database Design

Table of Contents

Requirements

The used furniture company includes three branch stores and four warehouses.
Each branch store can be restocked from different warehouses, and each warehouse can deliver the furniture to different branch store.
Every branch store can check all the furniture information in the warehouses including which warehouse they stock in and which branch store they could be deliver and sold. So customers can buy the furniture they want at different branch store or get the information about which branch store they can get this furniture.
Each branch can get furniture from 1 ~ 4 warehouses (this number depends on the location).

Replenish stock process

1. Supplier

The used furniture company will buy used furniture from suppliers, and the company should keep track of information:

- Supplier type: personal, company;
- Supplier name;
- Supplier contact information;

Each supplier should have a unique supplier number to distinguish them.

2. Supply-deal Record

When the used furniture company get furniture from the supplier, every deal should generate a supply record to save these information:

- Record number;
- Transaction date;
- Buying price;
- Furniture number;
- Stock number;
- Supplier number;

According to this record, the staff can check the initial information of every furniture. And if the deal has some problems, the company can contact supplier to solve them in time.

Furniture

Every furniture stored in the warehouse needs to record the following information:

- Furniture number;
- Furniture type: chair, table, bed, cabinet, light etc.
- Furniture name;
- Stock number;
- Whether it is available;

According to this record, the staff can check the latest information of every furniture. The customer can look for the furniture they want by use the furniture type.

Deal process

1. Customer

The used furniture company will sell used furniture to customers, and the company should keep track of information:

- Customer name;
- Customer contact information;

2. Deal record

When the used furniture company sell furniture to the customer, every deal should generate a deal record to save these information:

- Deal number;
- Transaction date;
- Selling price;
- Furniture number;
- Cashier Ssn;
- Customer name;
- Customer contact information;

According to this record, the staff can check the selling information of each furniture. And if there are some problems with the deal, the staff can find the cashier who sell the furniture and the customer who buy this furniture at once.
When one deal completes successfully, the system should update the information of the furniture so that its status is marked as sold.

Stock

For these 4 warehouses, the company should keep track of information:

- Stock number;
- Address;
- Quantity;
- Capacity;
- Manager Ssn;

When the company buy the furniture from the suppliers, the system should check whether there is enough space in the warehouse to store the furniture, and if not, change to another warehouse for storage.
When the storage is successful, the number of the furniture quantity in the warehouse in which it store should be increased by one.
When the company sell the furniture to the customers, the number of the furniture quantity in the warehouse in which the furniture store should be decreased by one.

Employee system

1. Employee

Employees are divided into three types according to their work content:
(1)Cashier;
(2)Stock manager;
(3)Deliverer;
The cashiers work in the branch stores.
The Stock managers work in the warehouses.
The deliverers are responsible for deliver furniture from warehouses to branch stores and shipping furniture to customers.

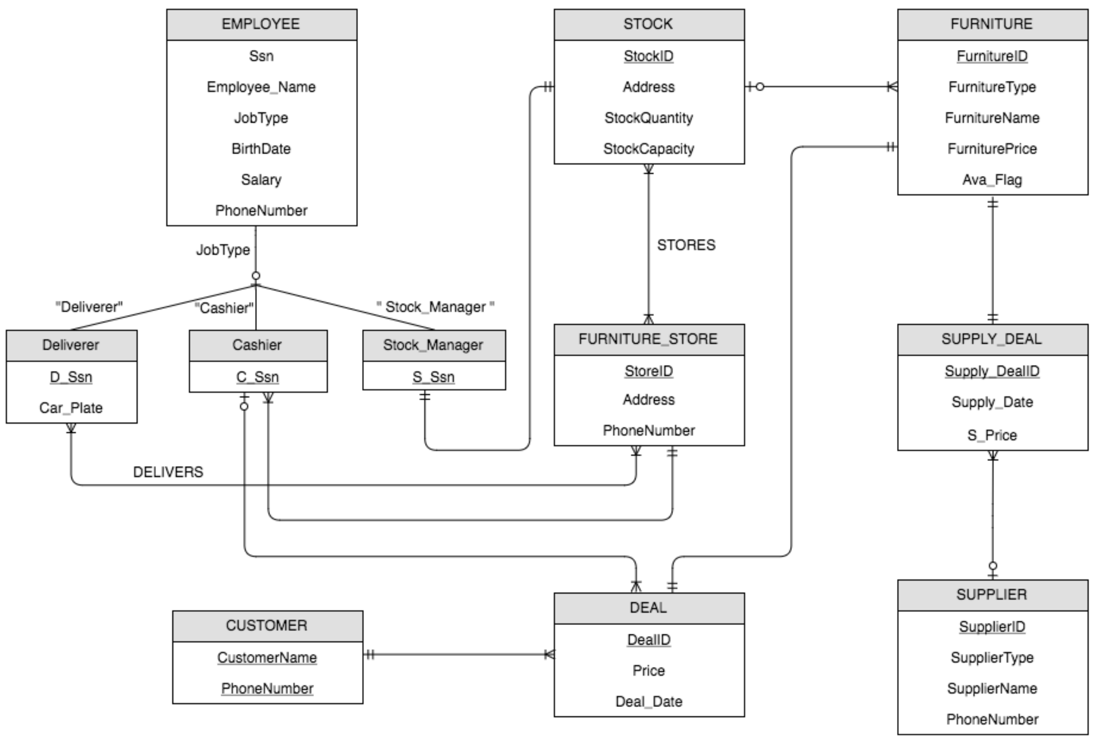
2. Bonus

When a cashier sells two or more pieces of furniture in one season, the company gives the cashier a bonus. The amount of the bonus was 20 percent of the cashier's original salary.

Profit

According to the supply deal record and the deal record, the company can calculate the profit of each piece of furniture.

Modeling of Requirements as ER-Diagram:



1. A stock stores 0 (if empty) or more used furniture, a furniture is stored from 1 stock (N:1).
2. A stock offers furniture for 1 or more used furniture stores, a furniture store can be provided from 1 or more stock (M:N).
3. A supplier can generate 0 or more deal record, a deal record can be generated by 1 supplier (N:1).
4. A furniture information can be recorded at 1 supplying deal, a supply deal describes 1 used furniture supplying detail (1:1).
5. A customer may carry out 1 or more deals from furniture store, a deal can be carried out by 1 customer (1:N).
6. A furniture information can be recorded at 1 transaction, a deal is associated with and describes 1 furniture (1:1).
7. Employees are classified under deliverer, cashier and stock manager according to the job type.
8. A stock manager manages 1 stock, a stock must be managed by 1 stock manager (1:1).
9. A cashier is in charge of 1 or more transactions, a deal is under 1 cashier's responsibility (N:1).
10. A cashier works for 1 furniture store, a furniture store employs 1 or more cashiers (1:N).
11. A deliverer delivers needed furniture to 1 or more furniture store, a furniture store employs 1 or more deliverers (M:N).

Mapping of ERD in Relational Schema

1. FURNITURE_STORE

StoreID	Address	PhoneNumber

- Primary Key: StoreID
- Foreign Keys: None

2. STOCK

StockID	Address	StockQuantity	StockCapacity

- Primary Key: StockID
- Foreign Keys: None

3. FURNITURE

FurnitureID	FurnitureType	FurnitureName	StockID	Ava_Flag

- Primary Key: FurnitureID
- Foreign Keys: StockID

4. DEAL

DealID	Price	Deal_Date	FurnitureID	Cashier_Ssn	CustomerName	C_PhoneNumber

- Primary Key: DealID
- Foreign Keys : FurnitureID, CustomerName, C_PhoneNumber ,Cashier_Ssn

5. SUPPLIER

SupplierID	SupplierType	SupplierName	PhoneNumber

- Primary Key: SupplierID
- Foreign Keys: None

6. CUSTOMER

CustomerName	PhoneNumber

- Primary Key: CustomerName, PhoneNumber
- Foreign Keys: None

7. SUPPLY_DEAL

Supply_DealID	Supply_Deal_Date	SupplierID	StockID	FurnitureID	S_Price

- Primary Key: Supply_DealID
- Foreign Keys: SupplierID, StockID, FurnitureID

8. EMPLOYEE

Ssn	JobType	Employee_ Name	BirthDate	Salary	PhoneNumber

- Primary Key: Ssn
- Foreign Keys: None

9. CASHIER

C_Ssn	StoreID

- Primary Key: C_Ssn
- Foreign Keys: C_Ssn , StoreID

10. DELIVERER

D_Ssn	Car_plate

- Primary Key: D_Ssn
- Foreign Keys: D_Ssn

11. STOCK_MANAGER

S_Ssn	StockID

- Primary Key: S_Ssn
- Foreign Keys: S_Ssn ,StockID

12. DELIVERS

D_Ssn	StoreID

- Primary Key: D_Ssn, StoreID
- Foreign Keys: D_Ssn, StoreID

13. STORES

StoreID	StockID

- Primary Key: StoreID ,StockID
- Foreign Keys: StoreID ,StockID

SQL Statements to create Relations in DB and Add Constraints

```
CREATE TABLE FURNITURE_STORE(
    STOREID       INTEGER,
    ADDRESS       VARCHAR(50),
    PHONENUMBER      CHAR(10),
    CONSTRAINT FURNITURE_STORE_PK PRIMARY KEY (STOREID)
);

CREATE TABLE STOCK(
    STOCKID       INTEGER,
    ADDRESS       VARCHAR(50),
    STOCKQUANTITY      INTEGER,
    STOCKCAPACITY      INTEGER NOT NULL,
    MANAGERSSN      CHAR(9),
    CONSTRAINT STOCK_PK PRIMARY KEY (STOCKID)
);

CREATE TABLE FURNITURE(
    FURNITUREID       INTEGER,
    FURNITURETYPE       VARCHAR(20) NOT NULL,
    FURNITURENAME       VARCHAR(30),
    STOCKID      INTEGER,
    AVA_FLAG CHAR(1) NOT NULL,
    CONSTRAINT FURNITURE_PK PRIMARY KEY (FURNITUREID)
);


CREATE TABLE DEAL(
    DEALID       INTEGER,
    PRICE       INTEGER,
    DEAL_DATE      DATE DEFAULT TRUNC(SYSDATE),
    FURNITUREID      INTEGER,
    CASHIER_SSN      CHAR(9),
    CUSTOMERNAME    VARCHAR(30),
    C_PHONENUMBER CHAR(10),
    CONSTRAINT DEAL_PK PRIMARY KEY (DEALID)
);

CREATE TABLE SUPPLIER(
    SUPPLIERID       INTEGER,
    SUPPLIERTYPE        VARCHAR(20),
    SUPPLIERNAME        VARCHAR(30) DEFAULT 'UNKNOWN',
    PHONENUMBER     CHAR(10),
    CONSTRAINT SUPPLIER_PK PRIMARY KEY (SUPPLIERID)
);

CREATE TABLE CUSTOMER(
    CUSTOMERNAME        VARCHAR(30),
    PHONENUMBER     CHAR(10),
    CONSTRAINT CUSTOMER_PK PRIMARY KEY (CUSTOMERNAME,PHONENUMBER)
);

CREATE TABLE SUPPLY_DEAL(
    SUPPLY_DEALID       INTEGER,
    SUPPLY_DEAL_DATE    DATE DEFAULT TRUNC(SYSDATE),
    SUPPLIERID      INTEGER,
    STOCKID     INTEGER,
    FURNITUREID INTEGER,
    S_PRICE INTEGER NOT NULL,
    CONSTRAINT SUPPLY_DEAL_PK PRIMARY KEY (SUPPLIERID)
);

CREATE TABLE EMPLOYEE(
SSN CHAR(9),
JOBTYPE VARCHAR(20) DEFAULT 'CASHIER',
EMPLOYEE_NAME VARCHAR(30),
BIRTHDATE DATE,
SALARY INTEGER DEFAULT 10000,
PHONENUMBER CHAR(10) NOT NULL,
CONSTRAINT EMPLOYEE_PK PRIMARY KEY (SSN));


CREATE TABLE CASHIER(
C_SSN CHAR(9),
STOREID INTEGER,
CONSTRAINT CASHIER_PK PRIMARY KEY (C_SSN));

CREATE TABLE DELIVERER(
D_SSN CHAR(9),
CAR_PLATE VARCHAR(10) NOT NULL,
CONSTRAINT DELIVERER_PK PRIMARY KEY (D_SSN));

CREATE TABLE STOCK_MANAGER
(
S_SSN CHAR(9),
STOCKID INTEGER,
CONSTRAINT STOCK_MANAGER_PK PRIMARY KEY (S_SSN));

CREATE TABLE DELIVERS
(
D_SSN CHAR(9),
STOREID INTEGER,
CONSTRAINT DELIVERS_PK PRIMARY KEY (D_SSN,STOREID));

CREATE TABLE STORES
(
STOREID INTEGER,
STOCKID INTEGER,
CONSTRAINT STORES_PK PRIMARY KEY (STOREID,STOCKID));

ALTER TABLE STOCK ADD CONSTRAINT STOCK_FK FOREIGN KEY(MANAGERSSN)
REFERENCES STOCK_MANAGER(S_SSN) ON DELETE SET NULL;
ALTER TABLE FURNITURE ADD CONSTRAINT FURNITURE_FK FOREIGN
KEY(STOCKID) REFERENCES STOCK(STOCKID) ON DELETE SET NULL;
ALTER TABLE DEAL ADD CONSTRAINT DEAL_FKA FOREIGN KEY(FURNITUREID)
REFERENCES FURNITURE(FURNITUREID)ON DELETE SET NULL;
ALTER TABLE DEAL ADD CONSTRAINT DEAL_FKB FOREIGN
KEY(CUSTOMERNAME,C_PHONENUMBER) REFERENCES
CUSTOMER(CUSTOMERNAME, PHONENUMBER)ON DELETE SET NULL;
ALTER TABLE DEAL ADD CONSTRAINT DEAL_FKD FOREIGN KEY(CASHIER_SSN)
REFERENCES CASHIER(C_SSN)ON DELETE SET NULL;
ALTER TABLE SUPPLY_DEAL ADD CONSTRAINT SUPPLY_DEAL_FKA FOREIGN
KEY(SUPPLIERID) REFERENCES SUPPLIER(SUPPLIERID)ON DELETE SET NULL;
ALTER TABLE SUPPLY_DEAL ADD CONSTRAINT SUPPLY_DEAL_FKB FOREIGN
KEY(STOCKID) REFERENCES STOCK(STOCKID)ON DELETE SET NULL;
ALTER TABLE SUPPLY_DEAL ADD CONSTRAINT SUPPLY_DEAL_FKC FOREIGN
KEY(FURNITUREID) REFERENCES FURNITURE(FURNITUREID)ON DELETE SET
NULL;
ALTER TABLE CASHIER ADD CONSTRAINT CASHIER_FK FOREIGN KEY(STOREID)
REFERENCES FURNITURE_STORE(STOREID)ON DELETE SET NULL;
ALTER TABLE STOCK_MANAGER ADD CONSTRAINT STOCK_MANAGER_FK
FOREIGN KEY(STOCKID) REFERENCES STOCK(STOCKID)ON DELETE SET NULL;
```

Normalization of Relational Schema

The following Functional Dependencies exists in the relational schema –

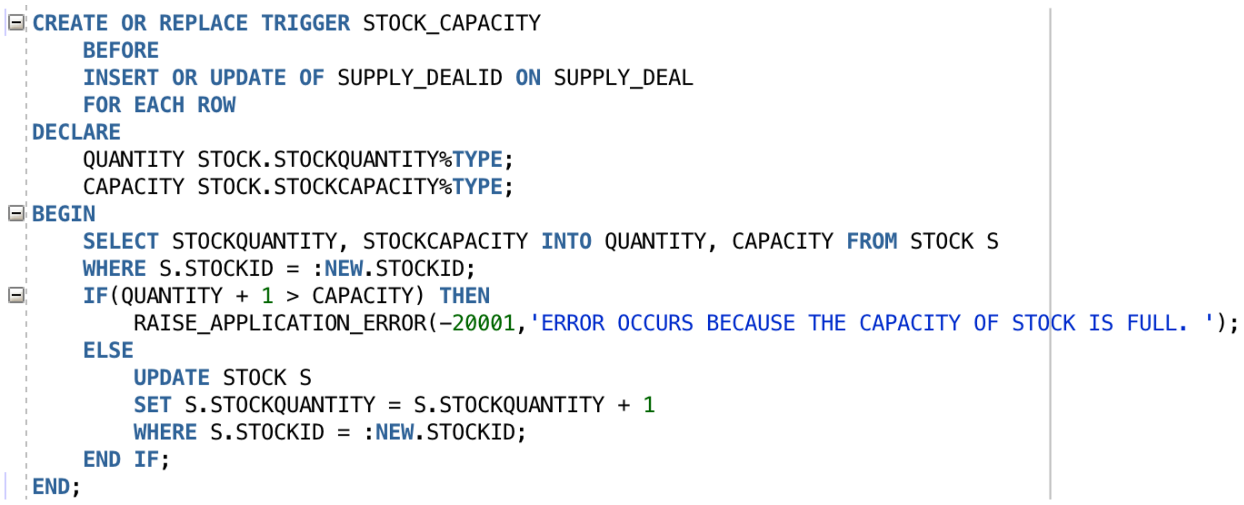
1. FURNITURE_STORE { StoreID -> Address, PhoneNumber}
2. STOCK { StockID -> Address, StockQuantity, StockCapacity }
3. FURNITURE { FurnitureID -> FurnitureType, FurnitureName, StockID ,Ava_Flag}
4. DEAL {DealID -> Price, Deal_Date, FurnitureID, Cashier_Ssn, CustomerName, C_PhoneNumber }
5. SUPPLIER { SupplierID -> SupplierType, SupplierName, PhoneNumber}
6. CUSTOMER { CustomerName , PhoneNumber }
7. SUPPLY_DEAL { Supply_DealID -> Supply_Deal_Date, SupplierID, StockID , FurnitureID , S_Price }
8. EMPLOYEE { Ssn -> JobType , Employee_ Name , BirthDate , Salary , PhoneNumber }
9. CASHIER { C_Ssn -> StoreID }
10. DELIVERER { D_Ssn -> Car_plate }
11. STOCK_MANAGER{ S_Ssn -> StockID }
12. DELIVERS { D_Ssn , StoreID }
13. DELIVERER { StoreID ,StockID }

The above functional dependencies cause the schema to be in third normal form.

PL/SQL – Triggers

Trigger-I Stock Capacity

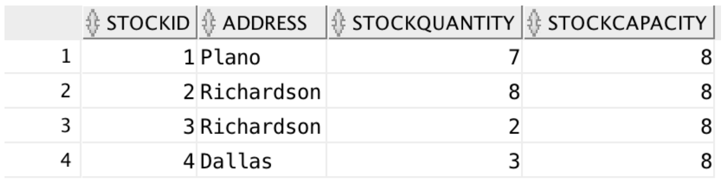
When the company buys the furniture from the supplier, this furniture will store at the certain stock. If that stock capacity is not full in warehouse, the number of the furniture quantity in the warehouse in which it store should be increased by one. Otherwise, the system raises an error.



```
CREATE OR REPLACE TRIGGER STOCK_CAPACITY
    BEFORE
    INSERT OR UPDATE OF SUPPLY_DEALID ON SUPPLY_DEAL
    FOR EACH ROW
DECLARE
    QUANTITY STOCK.STOCKQUANTITY%TYPE;
    CAPACITY STOCK.STOCKCAPACITY%TYPE;
BEGIN
    SELECT STOCKQUANTITY, STOCKCAPACITY INTO QUANTITY, CAPACITY FROM STOCK S
    WHERE S.STOCKID = :NEW.STOCKID;
    IF(QUANTITY + 1 > CAPACITY) THEN
        RAISE_APPLICATION_ERROR(-20001,'ERROR OCCURS BECAUSE THE CAPACITY OF STOCK IS FULL. ');
    ELSE
        UPDATE STOCK S
        SET S.STOCKQUANTITY = S.STOCKQUANTITY + 1
        WHERE S.STOCKID = :NEW.STOCKID;
    END IF;
END;
```

1. Positive Test Case:

Before insert, stock Table has shown,

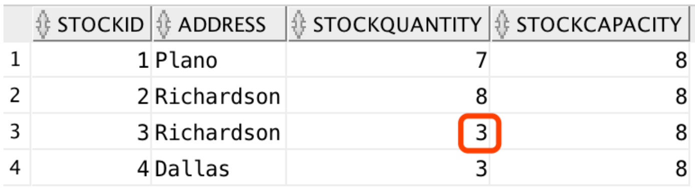


	STOCKID	ADDRESS	STOCKQUANTITY	STOCKCAPACITY
1	1	Plano	7	8
2	2	Richardson	8	8
3	3	Richardson	2	8
4	4	Dallas	3	8

Test by SQL:

INSERT INTO
SUPPLY_DEAL(Supply_DealID,SUPPLY_DEAL_DATE,SUPPLIERID,STOCKID,
FURNITUREID,S_Price)
VALUES (23,to_date ('03/20/1995' , 'MM/DD/YYYY'),2,3,23,60);

Positive Test Case Output:
The No.3 Stock is not full, then the quantity of this stock increases by 1.



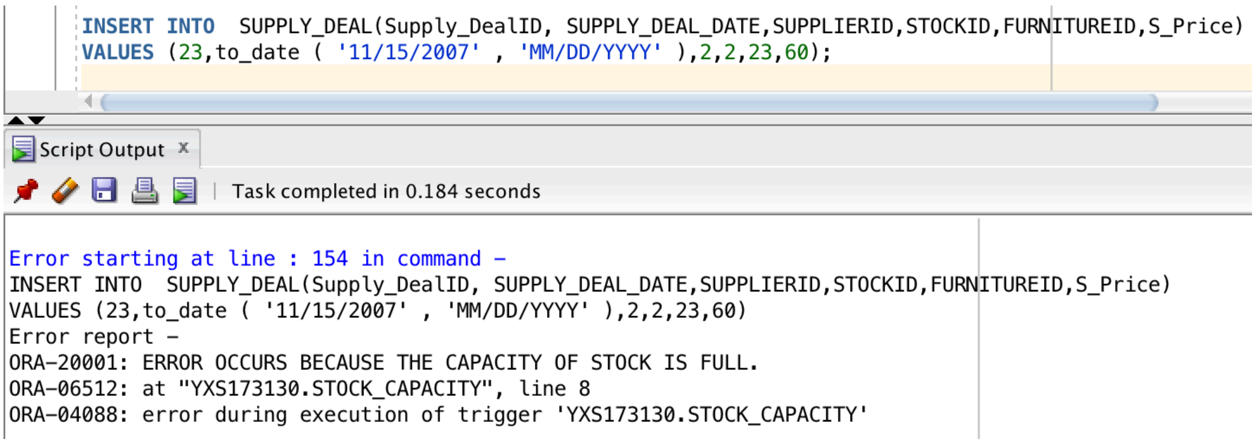
	STOCKID	ADDRESS	STOCKQUANTITY	STOCKCAPACITY
1	1	Plano	7	8
2	2	Richardson	8	8
3	3	Richardson	3	8
4	4	Dallas	3	8

2. Negative Test Case:

Test by SQL:

INSERT INTO
SUPPLY_DEAL(Supply_DealID,SUPPLY_DEAL_DATE,SUPPLIERID,STOCKID,
FURNITUREID,S_Price)
VALUES (23,to_date ('11/15/2007' , 'MM/DD/YYYY'),2,2,23,60);

Negative Test Output:
The No.2 Stock is not full, the information can't be inserted and raises a self-defined error.

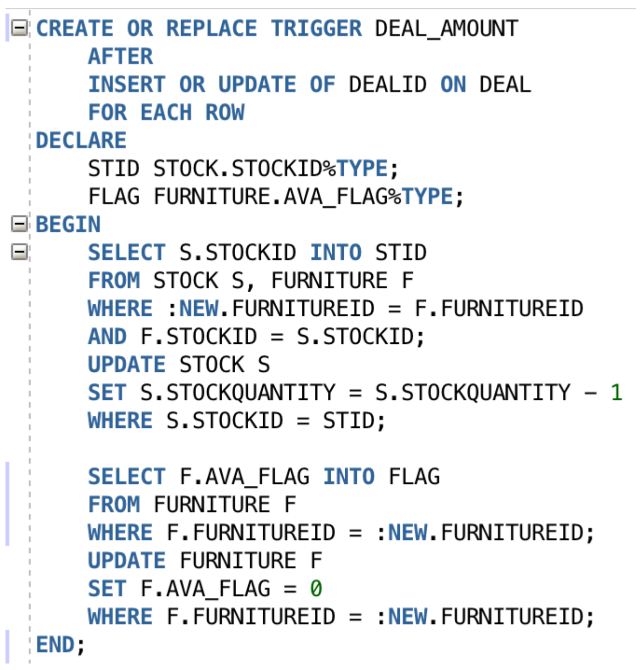


```
INSERT INTO  SUPPLY_DEAL(Supply_DealID, SUPPLY_DEAL_DATE,SUPPLIERID,STOCKID,FURNITUREID,S_Price)
VALUES (23,to_date ( '11/15/2007' , 'MM/DD/YYYY' ),2,2,23,60);
```

```
Error starting at line : 154 in command -
INSERT INTO  SUPPLY_DEAL(Supply_DealID, SUPPLY_DEAL_DATE,SUPPLIERID,STOCKID,FURNITUREID,S_Price)
VALUES (23,to_date ( '11/15/2007' , 'MM/DD/YYYY' ),2,2,23,60)
Error report -
ORA-20001: ERROR OCCURS BECAUSE THE CAPACITY OF STOCK IS FULL.
ORA-06512: at "YXS173130.STOCK_CAPACITY", line 8
ORA-04088: error during execution of trigger 'YXS173130.STOCK_CAPACITY'
```

Trigger-II Deal Amount

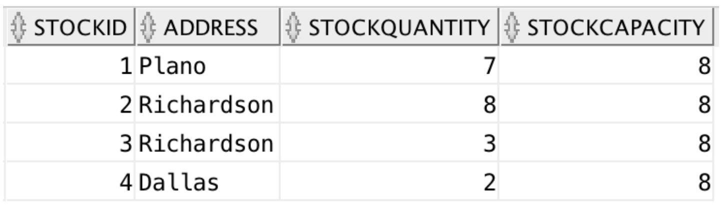
When one deal completes successfully, the number of the furniture quantity in the warehouse in which the furniture store should be decreased by one, the system should update the information of the furniture so that its status is marked as sold.



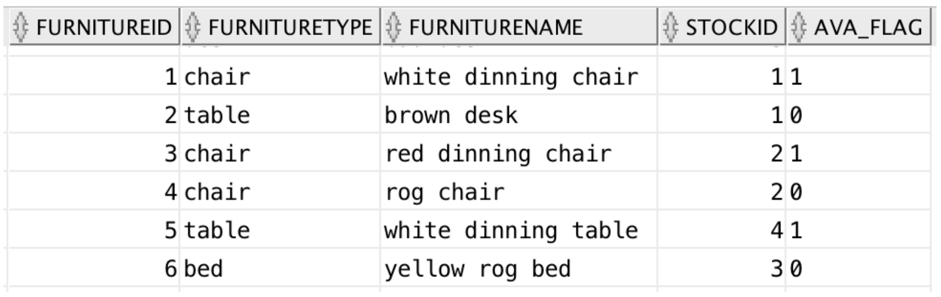
```
CREATE OR REPLACE TRIGGER DEAL_AMOUNT
    AFTER
    INSERT OR UPDATE OF DEALID ON DEAL
    FOR EACH ROW
DECLARE
    STID STOCK.STOCKID%TYPE;
    FLAG FURNITURE.AVA_FLAG%TYPE;
BEGIN
    SELECT S.STOCKID INTO STID
    FROM STOCK S, FURNITURE F
    WHERE :NEW.FURNITUREID = F.FURNITUREID
    AND F.STOCKID = S.STOCKID;
    UPDATE STOCK S
    SET S.STOCKQUANTITY = S.STOCKQUANTITY - 1
    WHERE S.STOCKID = STID;

    SELECT F.AVA_FLAG INTO FLAG
    FROM FURNITURE F
    WHERE F.FURNITUREID = :NEW.FURNITUREID;
    UPDATE FURNITURE F
    SET F.AVA_FLAG = 0
    WHERE F.FURNITUREID = :NEW.FURNITUREID;
END;
```

Test Case:

Furniture Table and Stock Table has shown,



STOCKID	ADDRESS	STOCKQUANTITY	STOCKCAPACITY
1	Plano	7	8
2	Richardson	8	8
3	Richardson	3	8
4	Dallas	2	8

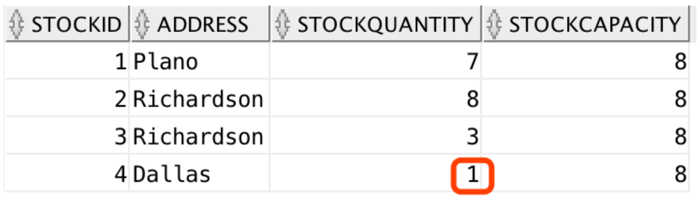


FURNITUREID	FURNITURETYPE	FURNITURENAME	STOCKID	AVA_FLAG
1	chair	white dinning chair	1	1
2	table	brown desk	1	0
3	chair	red dinning chair	2	1
4	chair	rog chair	2	0
5	table	white dinning table	4	1
6	bed	yellow rog bed	3	0

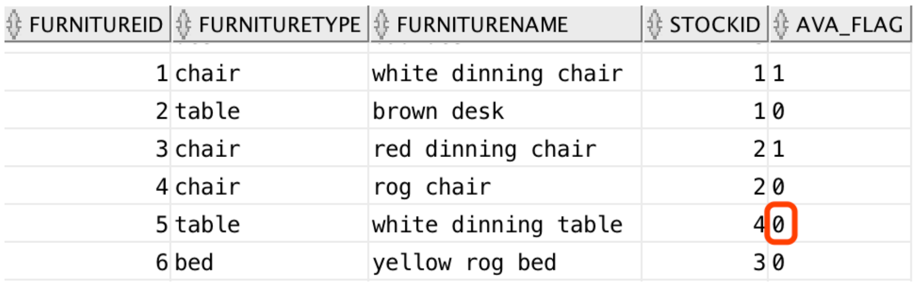
Test by SQL:

```
INSERT INTO DEAL(DEALID, PRICE, DEAL_DATE, FURNITUREID,
CASHIER_SSN, CUSTOMERNAME, C_PHONENUMBER)
VALUES(10,110,to_date ( '03/27/2017' ,
'MM/DD/YYYY' ),5,333333333,'Tom',4693343687);
```

Test Output:
When the deal is done, the furniture which was sold should be marked as not available. Then the associated stock quantity will decrease by 1.



STOCKID	ADDRESS	STOCKQUANTITY	STOCKCAPACITY
1	Plano	7	8
2	Richardson	8	8
3	Richardson	3	8
4	Dallas	$\boxed{1}$	8

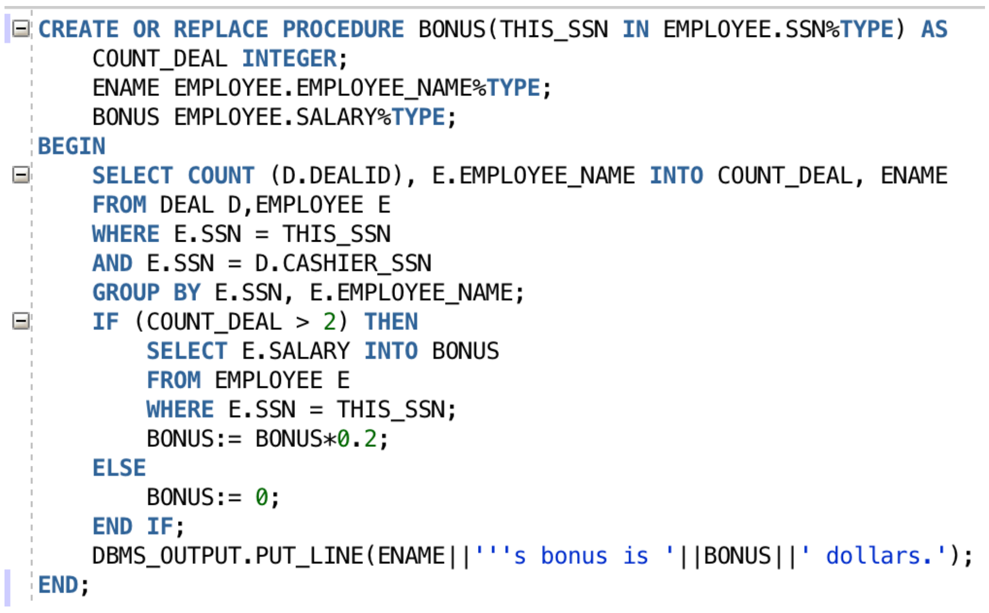


FURNITUREID	FURNITURETYPE	FURNITURENAME	STOCKID	AVA_FLAG
1	chair	white dinning chair	1	1
2	table	brown desk	1	0
3	chair	red dinning chair	2	1
4	chair	rog chair	2	0
5	table	white dinning table	4	$\boxed{0}$
6	bed	yellow rog bed	3	0

PL/SQL- Procedures

Procedure-I Calculating Bonus

When a sell man sells two or more furniture in one season, the company gives this employee the extra bonus. The amount of the bonus was 20 percent of this employee's original salary.



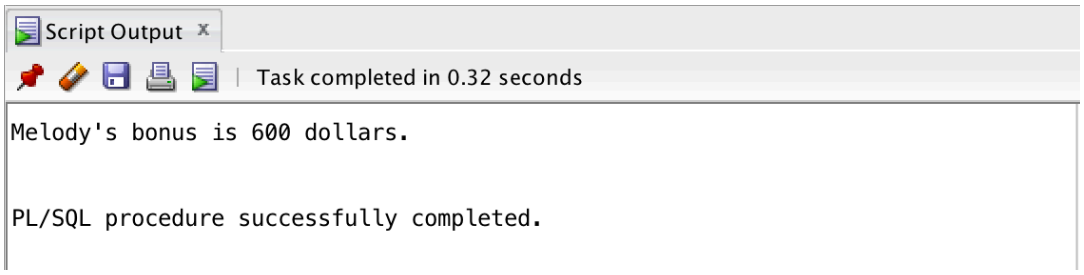
```
CREATE OR REPLACE PROCEDURE BONUS(THIS_SSN IN EMPLOYEE.SSN%TYPE) AS
    COUNT_DEAL INTEGER;
    ENAME EMPLOYEE.EMPLOYEE_NAME%TYPE;
    BONUS EMPLOYEE.SALARY%TYPE;
BEGIN
    SELECT COUNT (D.DEALID), E.EMPLOYEE_NAME INTO COUNT_DEAL, ENAME
    FROM DEAL D,EMPLOYEE E
    WHERE E.SSN = THIS_SSN
    AND E.SSN = D.CASHIER_SSN
    GROUP BY E.SSN, E.EMPLOYEE_NAME;
    IF (COUNT_DEAL > 2) THEN
        SELECT E.SALARY INTO BONUS
        FROM EMPLOYEE E
        WHERE E.SSN = THIS_SSN;
        BONUS:= BONUS*0.2;
    ELSE
        BONUS:= 0;
    END IF;
    DBMS_OUTPUT.PUT_LINE(ENAME||'''s bonus is '||BONUS||' dollars.');
END;
```

Arg: (Employee SSN IN)

Test by SQL:

```
DECLARE
  THIS_SSN CHAR(9);
BEGIN
  THIS_SSN := '222222222';
  BONUS(
    THIS_SSN => THIS_SSN
  );
END;
```

Test Case Output:



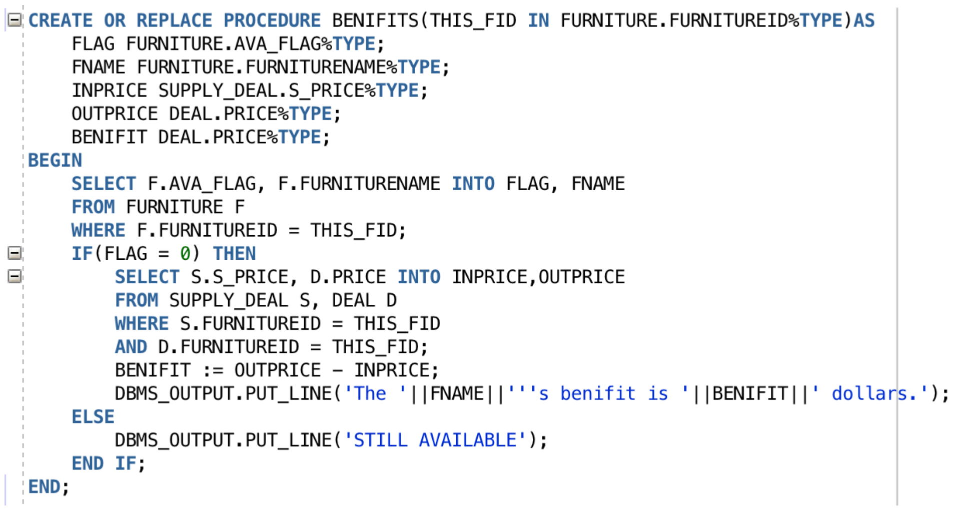
Script Output

Task completed in 0.32 seconds

```
Melody's bonus is 600 dollars.


PL/SQL procedure successfully completed.
```

Procedure-II Calculating Benefit for Each Deal

According to the supply deal record and the deal record, the company can calculate the profit(buying price – selling price) of each furniture.



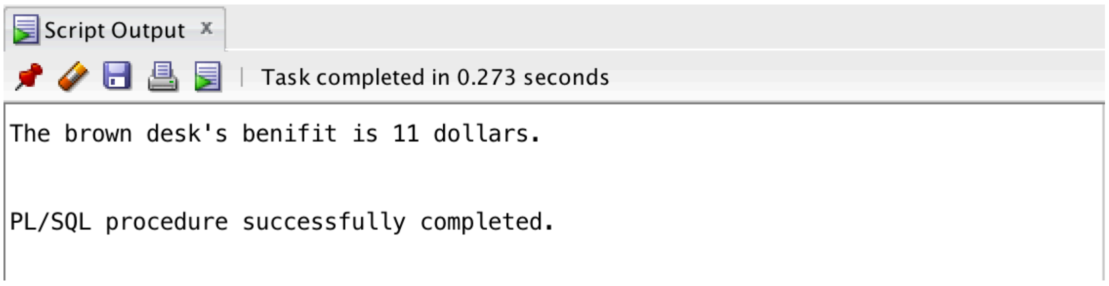
```
CREATE OR REPLACE PROCEDURE BENIFITS(THIS_FID IN FURNITURE.FURNITUREID%TYPE)AS
    FLAG FURNITURE.AVA_FLAG%TYPE;
    FNAME FURNITURE.FURNITURENAME%TYPE;
    INPRICE SUPPLY_DEAL.S_PRICE%TYPE;
    OUTPRICE DEAL.PRICE%TYPE;
    BENIFIT DEAL.PRICE%TYPE;
BEGIN
    SELECT F.AVA_FLAG, F.FURNITURENAME INTO FLAG, FNAME
    FROM FURNITURE F
    WHERE F.FURNITUREID = THIS_FID;
    IF(FLAG = 0) THEN
        SELECT S.S_PRICE, D.PRICE INTO INPRICE,OUTPRICE
        FROM SUPPLY_DEAL S, DEAL D
        WHERE S.FURNITUREID = THIS_FID
        AND D.FURNITUREID = THIS_FID;
        BENIFIT := OUTPRICE - INPRICE;
        DBMS_OUTPUT.PUT_LINE('The '||FNAME||'''s benifit is '||BENIFIT||' dollars.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('STILL AVAILABLE');
    END IF;
END;
```

Arg: (Furniture FurnitureID IN)

Test Case SQL:

```
DECLARE
  THIS_FID INTEGER;
BEGIN
  THIS_FID := 2;
  BENIFITS(
    THIS_FID => THIS_FID
  );
END;
```

Test Case Output:



```
The brown desk's benifit is 11 dollars.


PL/SQL procedure successfully completed.
```