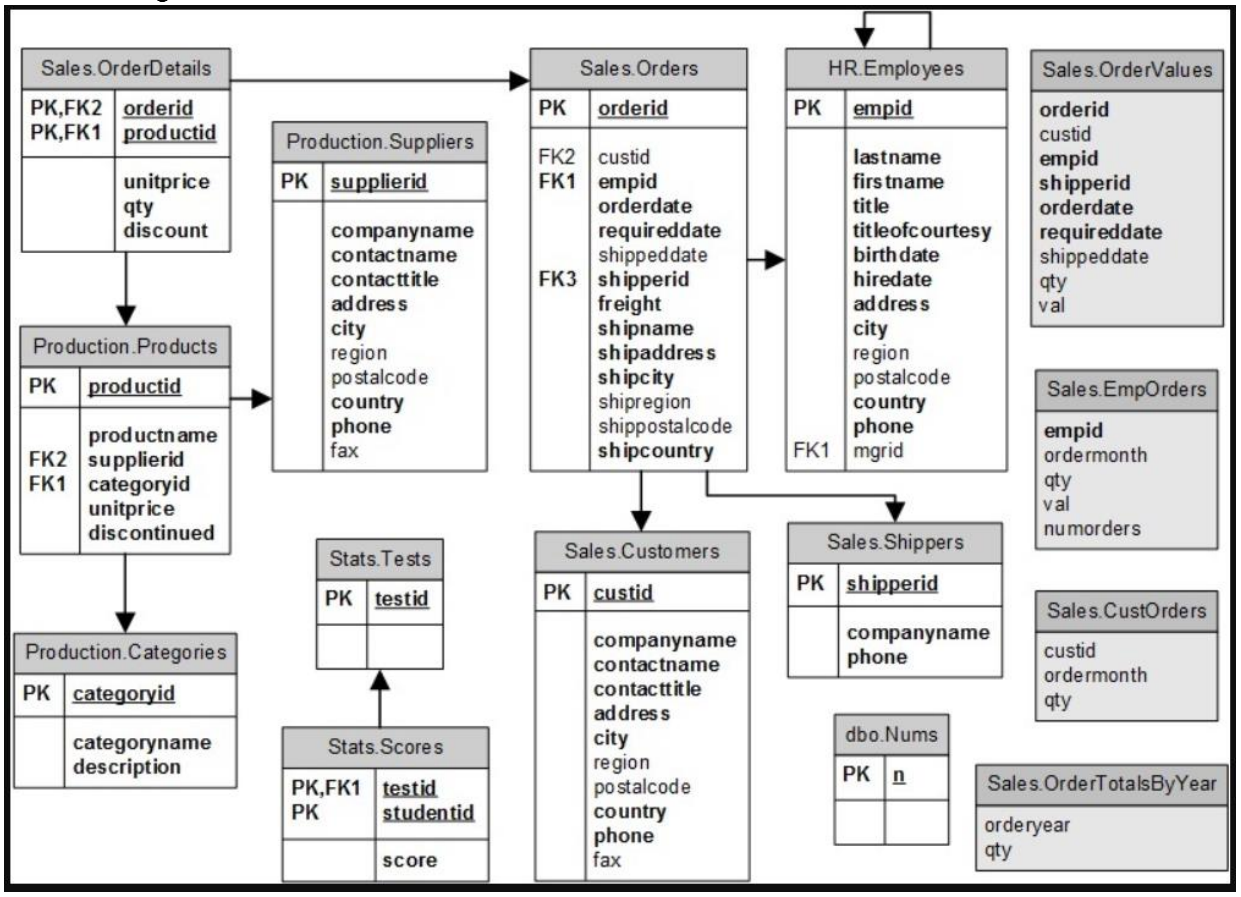
```
/*
Neba Nfonsang
Project 7: Queries that use transactions and
different isolation levels.

*/
USE TSQLV4
```



```
/*
Query # 1: A transaction that records information
about new orders
*/

-- Start a new transaction
BEGIN TRAN;

  -- Declare a variable
  DECLARE @neworderid AS INT;

  -- Insert a new order into the Sales.Orders table
  INSERT INTO Sales.Orders
      (custid, empid, orderdate, requireddate,
shippeddate,
       shipperid, freight, shipname, shipaddress,
shipcity,
       shippostalcode, shipcountry)
    VALUES
      (85, 5, '20090212', '20090301', '20090216',
       3, 32.38, N'Ship to 85-B', N'6789 rue de
l''Abbaye', N'Reims',
       N'10345', N'France');

  -- Save the new order ID in a variable
  SET @neworderid = SCOPE_IDENTITY();

  -- Return the new order ID
  SELECT @neworderid AS neworderid;

  -- Insert order lines for the new order into
Sales.OrderDetails
  INSERT INTO Sales.OrderDetails
      (orderid, productid, unitprice, qty, discount)
    VALUES(@neworderid, 11, 14.00, 12, 0.000),
          (@neworderid, 42, 9.80, 10, 0.000),
          (@neworderid, 72, 34.80, 5, 0.000);

-- Commit the transaction
COMMIT TRAN;


/*
Query # 2: The READ UNCOMMITTED Isolation Level
*/
-- Connection 1: update unitprice of product with
productid=2
BEGIN TRAN;

  UPDATE Production.Products
    SET unitprice += 1.00
  WHERE productid = 2;

  SELECT productid, unitprice
  FROM Production.Products
  WHERE productid = 2;

-- Connection 2: set isolation level to read
uncommitted changes
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

SELECT productid, unitprice
FROM Production.Products
WHERE productid = 2;

-- Connection 1: included ROLLBACK TRAN at the end
of connection 1,
-- transactions will be rolled back if there is a
failure
ROLLBACK TRAN;
```

```
/*
Query # 3:Set transaction isolation level to READ
COMMITTED;
*/
```

```
-- Connection 1: update the unit price of product
with productid=2
-- this locks the row for product 2 explicitly

BEGIN TRAN;

  UPDATE Production.Products
    SET unitprice += 1.00
  WHERE productid = 2;

  SELECT productid, unitprice
  FROM Production.Products
  WHERE productid = 2;

-- Connection 2:set session to read committed and
query product2 id.
-- This is the default isolation level, so set it if
it was changed

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

SELECT productid, unitprice
FROM Production.Products
WHERE productid = 2;
```

```
-- Connection 1: Run COMMIT TRAN  at the end of
connection 1 to so that
-- connection 2 can be able to read committed
changes
COMMIT TRAN;

-- Cleanup
UPDATE Production.Products
  SET unitprice = 19.00
WHERE productid = 2;

/*
Query # 4: Set transaction isolation level to
REPEATABLE READ;
*/

-- Connection 1: sets the transaction isolation to
REPEATABLE READ
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

BEGIN TRAN;

  SELECT productid, unitprice
  FROM Production.Products
  WHERE productid = 2;

-- Connection 2: tries to update same row being
-- read by transaction 1. This would not work
UPDATE Production.Products
  SET unitprice += 1.00
WHERE productid = 2;

-- Connection 1: the results of the second read
would be the
```

```
-- same as the first read in connection 1
  SELECT productid, unitprice
  FROM Production.Products
  WHERE productid = 2;

COMMIT TRAN;

-- Cleanup
UPDATE Production.Products
  SET unitprice = 19.00
WHERE productid = 2;
```

```
/*
Query # 5: The SERIALIZABLE isolation level
*/
-- Connection 1: prevent phantom read or other
transactions with future
-- rows not yet queried in connection 1
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

BEGIN TRAN

  SELECT productid, productname, categoryid,
unitprice
  FROM Production.Products
  WHERE categoryid = 1;

-- Connection 2: is blocked from performing
transaction with other rows
-- until connection 1 commits its transaction
INSERT INTO Production.Products
    (productname, supplierid, categoryid,
     unitprice, discontinued)
  VALUES('Product ABCDE', 1, 1, 20.00, 0);
```

```
-- Connection 1
  SELECT productid, productname, categoryid,
unitprice
  FROM Production.Products
  WHERE categoryid = 1;

COMMIT TRAN;

/*
Query # 6: The SNAPSHOT isolation level
*/

ALTER DATABASE TSQLV4 SET ALLOW_SNAPSHOT_ISOLATION
ON;

-- Connection 1
BEGIN TRAN;

  UPDATE Production.Products
    SET unitprice += 1.00
  WHERE productid = 2;

  SELECT productid, unitprice
  FROM Production.Products
  WHERE productid = 2;

-- Connection 2
SET TRANSACTION ISOLATION LEVEL SNAPSHOT;

BEGIN TRAN;

  SELECT productid, unitprice
  FROM Production.Products
  WHERE productid = 2;

-- Connection 1

COMMIT TRAN;

-- Connection 2

  SELECT productid, unitprice
  FROM Production.Products
  WHERE productid = 2;

COMMIT TRAN;

-- Connection 2
BEGIN TRAN

  SELECT productid, unitprice
  FROM Production.Products
  WHERE productid = 2;

COMMIT TRAN;

-- Cleanup
UPDATE Production.Products
  SET unitprice = 19.00
WHERE productid = 2;
```

```
/*
Query # 7:
*/

-- Turn on READ_COMMITTED_SNAPSHOT
ALTER DATABASE TSQLV4 SET READ_COMMITTED_SNAPSHOT
ON;

-- Connection 1
USE TSQLV4;

BEGIN TRAN;

  UPDATE Production.Products
    SET unitprice += 1.00
  WHERE productid = 2;

  SELECT productid, unitprice
  FROM Production.Products
  WHERE productid = 2;

-- Connection 2
BEGIN TRAN;

  SELECT productid, unitprice
  FROM Production.Products
  WHERE productid = 2;

-- Connection 1

COMMIT TRAN;

-- Connection 2
```

```
  SELECT productid, unitprice
  FROM Production.Products
  WHERE productid = 2;

COMMIT TRAN;

-- Cleanup
UPDATE Production.Products
  SET unitprice = 19.00
WHERE productid = 2;


/*
Query # 8: Simple Deadlock Example
*/
-- Connection 1
USE TSQLV4;

BEGIN TRAN;

  UPDATE Production.Products
    SET unitprice += 1.00
  WHERE productid = 2;

-- Connection 2
BEGIN TRAN;

  UPDATE Sales.OrderDetails
    SET unitprice += 1.00
  WHERE productid = 2;

-- Connection 1

  SELECT orderid, productid, unitprice
  FROM Sales.OrderDetails
  WHERE productid = 2;

COMMIT TRAN;

-- Connection 2

  SELECT productid, unitprice
  FROM Production.Products
  WHERE productid = 2;

COMMIT TRAN;

-- Cleanup
UPDATE Production.Products
  SET unitprice = 19.00
WHERE productid = 2;

UPDATE Sales.OrderDetails
  SET unitprice = 19.00
WHERE productid = 2
  AND orderid >= 10500;

UPDATE Sales.OrderDetails
  SET unitprice = 15.20
WHERE productid = 2
  AND orderid < 10500;

/*
Query # 9:
*/

BEGIN TRAN;
```

```
  UPDATE Sales.OrderDetails
    SET discount += 0.05
  WHERE orderid = 10249;

  SELECT orderid, productid, unitprice, qty,
discount
  FROM Sales.OrderDetails
  WHERE orderid = 10249;

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

SELECT orderid, productid, unitprice, qty, discount
FROM Sales.OrderDetails
WHERE orderid = 10249;

ROLLBACK TRAN;

/*
Query # 10: set the isolation level to READ
COMMITTED and query Sales.OrderDetails:
*/

BEGIN TRAN;

  UPDATE Sales.OrderDetails
    SET discount += 0.05
  WHERE orderid = 10249;

  SELECT orderid, productid, unitprice, qty,
discount
  FROM Sales.OrderDetails
  WHERE orderid = 10249;

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
```

```
BEGIN TRAN;
SELECT orderid, productid, unitprice, qty, discount
FROM Sales.OrderDetails
WHERE orderid = 10249;

COMMIT TRAN;

/*
Query # 11:
*/

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

BEGIN TRAN;

  SELECT orderid, productid, unitprice, qty,
discount
  FROM Sales.OrderDetails
  WHERE orderid = 10249;

  UPDATE Sales.OrderDetails
  SET discount += 0.05
WHERE orderid = 10249;

INSERT INTO Sales.OrderDetails
    (orderid, productid, unitprice, qty, discount)
  VALUES(10249, 2, 19.00, 10, 0.00);

SELECT orderid, productid, unitprice, qty, discount
  FROM Sales.OrderDetails
  WHERE orderid = 10249;

COMMIT TRAN;
```