**Labsheet 12 - Views**

For this lab we will use a new set of database tables that stores information about products and sales. The statements to create the appropriate tables and to insert some data in them are in the onlinestore.sql file on Moodle. Download the file and save it in your folder.

Connect to Oracle using SQLDeveloper. Open this file (onlinestore.sql) and run the code.

Ensure all the tables have been created properly and they have been populated with data.

The tables created have the following schemas (with the primary keys underlined):

MyProduct (BarCode, PName, Price, QuantityInStock)

Create view MyProduct

As SELECT p.Barcode,p.pName,p.Price,QuantityInStock

FROM Products p;

MySale (SaleID, DeliveryAddress, CreditCard)

Create view mySales(Sales ID,Delivery Address, Credit Card)

As Select SaleID, DeliveryAddress, CreditCard

FROM sales;

MySaleItem (SaleID, BarCode, Quantity)

Create view mySaleItem

As SELECT SaleID, BarCode, Quantity

FROM saleItems;

**Exercise 1**

List all products, with barcode and name, and the sale id and quantity for the sales containing that product, if any (products that were never sold should still be listed in your result). Hint: use a left join

SELECT p.barcode,p.pName,sa.SaleID,ss.Quantity

FROM products p left join SaleItems ss On p.barcode = ss.barcode left join Sales sa on sa.SaleID = ss.SaleID

Order by p.pName;

**Exercise 2**

Write a SQL statement to create a view called AllProductsSales based on the query in Exercise 1. Run the SQL statement.

Create view AllProductsSales

As SELECT p.barcode,p.pName,sa.SaleID,ss.Quantity

FROM products p left join SalesItems ss On p.barcode = ss.barcode left join Sales sa on sa.SaleID = ss.SaleID

ORDER BY p.pName;

**Exercise 3**

A virtual table or view can be used as if it is a table in the database.  Now write a SQL query against the AllProductsSales view as if it was a table: select everything from the AllProductsSales, ordered by BarCode and SaleID.

SELECT \*

FROM AllProductsSales

ORDER BY BarCode, SaleID;

**Exercise 4**

**IMPORTANT:** The view **does not store any data**. The data is stored just in the tables used in the definition of a view. When a query that uses AllPorductsSales is executed, the system first evaluates the query that defines AllProductsSales, and then performs further evaluation of the query that uses the view.

 To demonstrate this, do the following:

1. Insert a row with value for PName = ‘Chips’ in Products table (use INSERT INTO … VALUES …). (You can change the product name to fit your store).

INSERT INTO Products VALUES(32, 'Chips', 0.2,32);

1. Execute the query from exercise 3 again. Did the results change?  Why?

Yes. It is because you entered a value in the base table which caused the value in the view table to increase in its values.

**Exercise 5**

Create a view called ProductProfit to show the barcode, the name of the product, and total\_profit. total\_profit represents the total profit for that product, which is based on the fraction of the product price that represents the profit margin (assume a fixed 10%), and the quantity of products sold (found in the SaleItems table).

Create view ProductProfit

SELECT p.BarCode "Barcode", p.pName "Name",(((p.price/100) \* 10) \* ss.Quantity) "Total\_Profit"

FROM Products p, SaleItems ss, Sales sa

WHERE p.BarCode = ss. BarCode AND sa.SaleID = ss.SaleID

ORDER BY p.pName;

**Exercise 6**

Write the SQL query on the ProductProfit view to show the list of products based on total profit in order of highest profit to lowest profit.

SELECT Name, Total\_Profit

FROM ProductProfit

ORDER BY Total\_Profit DESC;

**Exercise 7**

Write the SQL statement to update the total\_profit for ‘Bread roll’ to be 10,000, using the view. Use the UPDATE...SET...WHERE clause.

Run the query. Does the query succeed? Why?

Upadate ProductProfit

Set Total\_Profit = 10000

WHERE Name = ‘Bread roll’;