Whitireia New Zealand
Faculty of Business
Programme: Postgraduate Diploma in IT

IT7x29: Data Models and Databases

Assignment 2

Application Development

Author:
Dmitry Shiltsov
21402582

Supervisor:
Adrian Hargreaves

June 1, 2015

Contents

1 Problem definition

The primary task is to create an application, which generates a report based on the data in the AdventureWorks2012 database for year and month, provided by user. The report should include the following information for all orders placed at the given date: PurchaseOrderID, ProductNumber, OrderQty, UnitPrice, LineTotal, the total for the order, the total for all orders. All the data should be extracted using stored procedure based on cursors.

2 Solution

2.1 Data Migration

Based on the script, generated by Microsoft SQLServer, a proper file with DDL and data inserts for PostgreSQL was created.

2.2 Backend

All the data required may be returned by a single stored procedure call. The logic behind the procedure consists of the following nested loops:

Get the matching order headers.

This is achieved by selecting all the entries from the **PurchaseOrderHeader** table, where month and year extracted from the **OrderDate** column match the ones sent by user. We take only the value of **PurchaseOrderID** column, which is required in the report and will be used during the next step.

Get the corresponding order detail.

For each order header, we get all the entries from **PurchaseOrderDetail** table matching the **PurchaseOrderID**. Apart from required columns **OrderQty, UnitPrice, LineTotal** we also select **ProductID** column, which is being used later.

Get product number.

Here we select **ProductNumber** from **Product** based on the **ProductID** from the previous part.

Submit resulting row.

At this stage we have all the data we need for current product from the purchasing order. We combine it in the right order and return as a single row.

2.3 Frontend

To make a frontend application, we are using the Dancer2, lightweight Perl web framework. The required year and month are being submitted through a simple text form. The stored procedure is called via standard DBI (Database Interface) module, using PostgreSQL plugin. The returned array is aggregated to calculate totals and simplify resulting table creation. The output page (Figure 1) is generated using standard template toolkit module.

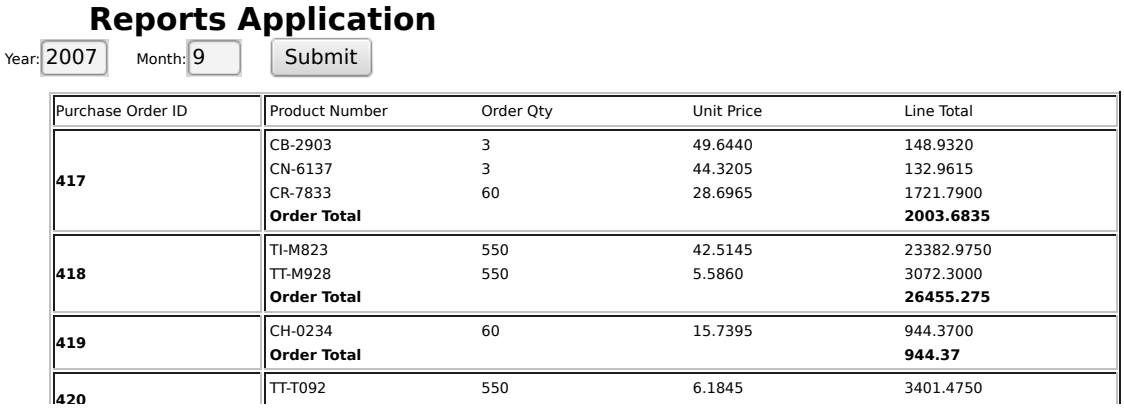

Figure 1: Web application interface

3 Performance considerations

The idea to use nested cursors just to extract the data contradicts the main ideology of relational databases. To prove the inefficiency of such approach, a simple experiment was conducted, where the stored procedure and the following simple SQL query were called 10000 times.

```
\set year 2007
\set month 9
select PurchaseOrderID, productNumber, OrderQty, UnitPrice, LineTotal
    FROM Purchasing_PurchaseOrderHeader
        INNER JOIN Purchasing_PurchaseOrderDetail USING (PurchaseOrderID)
        INNER JOIN production_product USING (productid)
            WHERE EXTRACT(YEAR FROM OrderDate)=:year
                AND EXTRACT(MONTH FROM OrderDate)=:month;
```

The results are presented in the Table 1. The performance of the simple unoptimized query with joins is 5 times better than of partially optimized stored procedure. Preparing the query in advance reduces time by 10%. Further steps, like cleaning the query, adding indexes, tweaking PostgreSQL engine (this DB is known to be suboptimal out of the box) might make the difference even larger.

Stored procedure	248 s
Unprepared SQL query	51 s
Prepared SQL query	46 s

Table 1: Time comparison for 10000 queries.

4 Stored procedure code

```
DROP TABLE IF EXISTS PurchaseReport CASCADE;
CREATE TABLE PurchaseReport (
    PurchaseOrderID int,
    productNumber varchar(25),
    OrderQty smallint,
    UnitPrice numeric(19,4),
    LineTotal numeric(19,4)
);

-- Usage example: SELECT * FROM spCreateReport(2007,9);
DROP FUNCTION IF EXISTS spCreateReport(integer, integer);
CREATE FUNCTION spCreateReport (integer, integer) RETURNS SETOF PurchaseReport AS $$
    use strict;
    my ($year, $month) = @_;

    #########     Queries    ##########

    my $orderheader_querry = "SELECT purchaseorderid
        FROM purchasing_purchaseorderheader
        WHERE EXTRACT(YEAR FROM orderdate) = $year
            AND EXTRACT(MONTH FROM orderdate) = $month";

    my $order_detail_querry = 'SELECT orderqty, unitprice, linetotal, productid
        FROM purchasing_purchaseorderdetail
        WHERE PurchaseOrderID = $1';

    my $order_detail_plan = spi_prepare($order_detail_querry, 'integer');

    my $product_querry = 'SELECT productnumber
        FROM production_product
        WHERE productid = $1';

    my $product_plan = spi_prepare($product_querry, 'integer');


    #########      Actual code     #########

    my $orderheader_results = spi_query($orderheader_querry);

    while (defined (my $orderheader = spi_fetchrow($orderheader_results))) {
        my $purchase_order_id = $orderheader->{purchaseorderid};
        my $order_detail_results = spi_query_prepared($order_detail_plan, $purchase_order_id);

        while (defined (my $orderdetail = spi_fetchrow($order_detail_results))) {
            my $orderqty  = $orderdetail->{orderqty};
            my $unitprice = $orderdetail->{unitprice};
            my $linetotal = $orderdetail->{linetotal};
            my $productid = $orderdetail->{productid};

            my $product_results = spi_exec_prepared($product_plan, $productid);
            my $product_number = $product_results->{rows}->[0]->{productnumber};
            return_next {
                purchaseorderid => $purchase_order_id,
                productnumber   => $product_number,
                orderqty        => $orderqty,
                unitprice       => $unitprice,
                linetotal       => $linetotal
            };
        }
    }

    return undef;
$$ LANGUAGE plperl;
```