# List of Precomputed Tables and Indexes

**states\_precomputed: Table that contains state name, state id, and the total amount of money spent in that state.**

CREATE TABLE states\_precomputed AS

SELECT st.name, st.id AS sid, SUM(s.quantity \* s.price) AS price

FROM states st

LEFT OUTER JOIN users u

ON st.id = u.state\_id

LEFT OUTER JOIN orders s

ON s.user\_id=u.id

GROUP BY st.id, st.name

ORDER BY price DESC nulls last;

**products\_precomputed: Table that contains product name, product id, product category, and the total amount of money spent on that product.**

CREATE TABLE products\_precomputed AS

SELECT p.name, p.id AS pid, p.category\_id, COALESCE( SUM(s.quantity \* s.price), 0 ) AS price

FROM products p

JOIN categories c ON p.category\_id = c.id

LEFT OUTER JOIN orders s ON s.product\_id = p.id

GROUP BY p.id, p.name

ORDER BY price DESC nulls last;

**cell\_precomputed: Table that contains state id, product id, and the total amount of money that state spent on that product.**

CREATE TABLE cell\_precomputed AS

SELECT u.state\_id, s.product\_id, SUM(s.quantity \* s.price)

FROM users u, orders s

WHERE s.user\_id = u.id

GROUP BY u.state\_id, s.product\_id;

**Indexes:**

CREATE INDEX cell\_product\_index ON cell\_precomputed(product\_id)

CREATE INDEX products\_id\_index ON products\_precomputed(pid)

CREATE INDEX products\_catid\_index ON products\_precomputed(category\_id)

CREATE INDEX states\_id\_index ON states\_precomputed(sid)

# Precomputation code

String query1 = "drop table if exists products\_precomputed;";

stmt.executeUpdate(query1);

String query2 = "drop table if exists states\_precomputed;";

stmt.executeUpdate(query2);

String query3 = "drop table if exists cell\_precomputed;";

stmt.executeUpdate(query3);

String index2 = "create index bb on orders(product\_id)";

stmt.executeUpdate(index2);

String queryS = "CREATE TABLE states\_precomputed AS "

+ " SELECT st.name, st.id AS sid, SUM(s.quantity\*s.price) AS price "

+ " FROM states st "

+ " LEFT OUTER JOIN users u "

+ " ON st.id = u.state\_id"

+ " LEFT OUTER JOIN orders s "

+ " ON s.user\_id=u.id"

+ " GROUP BY st.id, st.name"

+ " ORDER BY price DESC nulls last";

stmt.executeUpdate(queryS);

String queryP = "CREATE TABLE products\_precomputed AS "

+ " SELECT p.name, p.id AS pid, p.category\_id, COALESCE( SUM(s.quantity \* s.price), 0 ) AS price "

+ " FROM products p "

+ " JOIN categories c"

+ " ON p.category\_id = c.id"

+ " LEFT OUTER JOIN orders s "

+ " ON s.product\_id = p.id "

+ " GROUP BY p.id, p.name "

+ " ORDER BY price DESC nulls last";

stmt.executeUpdate(queryP);

String queryC = "CREATE TABLE cell\_precomputed AS"

+ " SELECT u.state\_id, s.product\_id, SUM(s.quantity \* s.price) "

+ " FROM users u, orders s "

+ " WHERE s.user\_id = u.id "

+ " GROUP BY u.state\_id, s.product\_id";

stmt.executeUpdate(queryC);

String indexDrop2 = "drop index bb ";

stmt.executeUpdate(indexDrop2);

# Code that takes care of the buying

//get the latest order id

stmt = conn.createStatement();

rs = stmt.executeQuery("SELECT id FROM orders ORDER BY id DESC LIMIT 1");

rs.next();

**int** latestID = rs.getInt("id");

System.out.println(latestID);

//Insert the specified number of queries

**int** queries\_num = Integer.parseInt(request.getParameter("queries\_num"));

Random rand = **new** Random();

**int** random\_num = rand.nextInt(30) + 1;

**if** (queries\_num < random\_num)

random\_num = queries\_num;

stmt = conn.createStatement();

stmt.executeQuery("SELECT proc\_insert\_orders(" + queries\_num + "," + random\_num + ")");

out.println("<script>alert('" + queries\_num + " orders are inserted!');</script>");

stmt = conn.createStatement();

//Query to get all orders that have an order ID higher than the latest id

ResultSet neworders = stmt.executeQuery("SELECT o.id, s.id, o.product\_id, (o.quantity \* o.price) as amount "

+ " FROM states s, orders o, users u "

+ " WHERE u.state\_id = s.id AND o.id > " + latestID + " AND o.user\_id = u.id");

// Insert every new order into the log table

**while**(neworders.next()) {

String insertquery = "INSERT INTO log\_table(order\_id, state\_id, product\_id, amount)"

+ " VALUES ("+neworders.getInt(1)+", "+neworders.getInt(2)+", "+neworders.getInt(3)+", "+neworders.getInt(4)+");";

stmt = conn.createStatement();

stmt.execute(insertquery);

}

# Code that executes upon Run

//Filling in statesList

query = "SELECT \* FROM states\_precomputed"

+ " ORDER BY price DESC NULLS LAST"

+ " LIMIT 50";

rs = stmt.executeQuery(query);

**while**(rs.next()) {

String stateName = rs.getString("name");

Integer stateId = rs.getInt("sid");

**double** total = rs.getDouble("price");

statesList.add(**new** StatesRows(stateName, stateId, total));

}

query = "SELECT \* FROM products\_precomputed p"

+ " WHERE " + categoryFilter

+ " ORDER BY price DESC NULLS LAST LIMIT 50";

rs = stmt.executeQuery(query);

**while** (rs.next()) {

String productName = rs.getString(1);

Integer productId = rs.getInt(2);

Integer categoryId = rs.getInt(3);

**double** total = rs.getDouble(4);

productsList.add(**new** ProductColumns(productName, productId, categoryId, total));

}

query = "SELECT \* FROM cell\_precomputed x "

+ " WHERE x.state\_id IN (SELECT sid FROM states\_precomputed ORDER BY price DESC NULLS LAST LIMIT 50)"

+ " AND x.product\_id IN (SELECT pid FROM products\_precomputed ORDER BY price DESC NULLS LAST LIMIT 50)";

rs = stmt.executeQuery(query);

**while** (rs.next()) {

Integer stateId = rs.getInt(1);

Integer productId = rs.getInt(2);

Integer total = rs.getInt(3);

hashmap.put(**new** StateProductIdPair(stateId, productId, **true**),total);

}

%>

<table class=*"table table-striped"* align=*"center"*>

<thead>

<tr align=*"center"*>

<th> States </th>

<%

**for**(ProductColumns pr : productsList)

{

%>

<th>

<%=pr.productName%>

<div id = <%="pid"+ Integer.toString(pr.productId) %> >

<%=pr.total%></div>

</th>

<%

}

%>

<tr>

</thead>

<tbody>

<%

**int** rows = 0; // rows

**while**(rows < productsList.size())

{

%>

<tr>

<td>

<b> <%= statesList.get(rows).stateName %></b>

<b><div id = <%="sid" + Integer.toString(statesList.get(rows).stateId) %>>

<%= statesList.get(rows).total %>

</div></b>

</td>

<%

**for**(ProductColumns pcIter : productsList)

{

StateProductIdPair getPair = **new** StateProductIdPair(statesList.get(rows).stateId, pcIter.productId, **true**);

**if**(hashmap.get(getPair) == **null**){

%>

<td id = <%="sid" + Integer.toString(statesList.get(rows).stateId) + "pid" + Integer.toString(pcIter.productId) %>>

<%=0%>

</td>

<%

}

**else**{

%>

<td id = <%="sid" + Integer.toString(statesList.get(rows).stateId) + "pid" + Integer.toString(pcIter.productId) %>>

<%=hashmap.get(getPair)%></td>

<%

}

}

rows++;

}

%>

</tbody>

</table>

# Code that executes upon Refresh

**orders.jsp**

**if** (action.equals("request")){

<script> makeRequest(latestIDthisScope) </script><%

}

**function** makeRequest(lastId)

{

$.ajax(

{

type: 'POST',

url: "/cse135-for-project3/ajax.jsp?lastId="+lastId,

dataType:'json',

beforeSend:**function**(){

//Update Stats

$('#status').html('Request Sent');

},

success:**function**(response){

**var** response = String(data);

**var** array = eval("[" + response + "]");

updateTable(array);

},

error:**function**(){

// Failed request

$('#status').html('Oops! Error.');

}

});

}

**function** updateTable(array)

{

**for**(**var** i = 0; i < array.length; i= i+4)

{

**var** order\_id = array[i];

**var** state\_id = array[i + 1];

**var** product\_id = array[i + 2];

**var** amount = array[i + 3];

**var** cellResult = document.getElementById("sid" + eval(sid) + "pid" + eval(pid) );

**if**(cellResult != **null**)

{

cellResult.innerHTML = eval(eval(cellResult.innerHTML) + amount);

cellResult.style.color = "red";

}

**var** stateResult = document.getElementById("sid" + eval(sid));

**if**(stateResult != **null**)

{

stateResult.innerHTML = "("+eval(eval(stateResult.innerHTML) + price) + ")";

stateResult.style.color = "red";

}

**var** productResult = document.getElementById("pid" + eval(pid));

**if**(productResult != **null**)

{

productResult.innerHTML = "("+eval(eval(productResult.innerHTML) + price) +")";

productResult.style.color = "red";

}

}

}

**ajax.jsp**

<%

Connection conn = **null**;

**try** {

Class.forName("org.postgresql.Driver");

String url = "jdbc:postgresql://localhost:5433/shopping";

String admin = "postgres";

String password = "Asdf!23";

conn = DriverManager.getConnection(url, admin, password);

}

**catch** (Exception e) {}

**int** lastId = Integer.parseInt(request.getParameter("lastId"));

PreparedStatement query = conn.prepareStatement("SELECT \* FROM log\_table WHERE order\_id > ? ;");

query.setInt(1, lastId);

ResultSet rs = query.executeQuery();

JSONArray resultArray = **new** JSONArray();

**while**(rs.next())

{

JSONObject resultObj = **new** JSONObject();

resultObj.put("order\_id", rs.getInt(1));

resultObj.put("state\_id", rs.getInt(2));

resultObj.put("product\_id", rs.getInt(3));

resultObj.put("amount", rs.getInt(4));

resultArray.add(resultObj);

}

out.print(resultArray);

out.flush();

%>