Advanced Query Techniques

Vicky Wall and Lenna Wenke

Welcome!

At the end of this course users will:
- Be able to add criteria to a left outer join.
- Utilize subqueries to effectively search for query data information.
- Effectively apply aggregate functions to query results.
- Utilize the Having tab to add criteria to fields using aggregate functions.
- Be able to create expressions to allow complex refining of query results.
- Understand BIND records and how they affect query development.

Standard Joins

Standard joins only display rows where there is a match between BOTH records, meaning it is possible not all the rows from Record A will be displayed.
Outer Join

This join type will always display all the rows in Record A. It will simply display BLANK where there is no match.

Example: Normal Outer Join

- Suppose we wanted to have a list of all our enrolled GRAD students for Fall 2016, and we want to know if they have a mobile phone number stored in our system.
- Start with your base query: Add the STDNT_CAR_TERM record and define your enrollment criteria.

Next, look up the PERSONAL_PHONE record. We are going to outer join this record to STDNT_CAR_TERM. When you click the “Join Record” link, select the outer join option on the next screen:
You should now notice a couple of differences in your query. The first is on your Query tab: the phone record has additional information next to its name, indicating that it is outer joined and which record it is joined to:

The second is on your Criteria tab. Any criteria for outer joins has an additional column filled in: "Belongs to".

This new criteria column will become important when we narrow our results. Go back to your Query tab and add new criteria on the PHONE_TYPE field in the phone record. Since this is an outer join, you will have to change the "belongs to" field at the bottom to the same letter as the record (in our case, since we are adding criteria to record B, the criteria should belong to outer join B). Click "OK".
Correct outer join

Criteria have been erroneously placed on the WHERE clause.

SubQueries

Criteria page when you are using a subquery
Query page when you are creating a subquery
2 common types of criteria conditions for sub-queries:
1. Using in list
2. Using exists which does not require using a field
The 2 look very different.

Expressions are calculations that PeopleSoft Query performs as part of a query. Use them when you must calculate a value that PeopleSoft Query does not provide by default—for example, to add the values from two fields together or to multiply a field value by a constant.

Use Expressions in two ways:
- As comparison values in selection criteria.
- As columns (fields) in the query output.
**Expression Operator Basics**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition operator</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction operator</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication operator</td>
</tr>
<tr>
<td>/</td>
<td>Division operator</td>
</tr>
<tr>
<td>(</td>
<td>Expression list delimiter</td>
</tr>
<tr>
<td>)</td>
<td>Expression list end delimiter</td>
</tr>
<tr>
<td>[</td>
<td>Concatenation operator</td>
</tr>
<tr>
<td>]</td>
<td>Concatenation operator</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Relational operator (not equals)</td>
</tr>
<tr>
<td>&lt;</td>
<td>Relational operator (less than)</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Relational operator (greater than or equal)</td>
</tr>
<tr>
<td>&gt;</td>
<td>Relational operator (greater than)</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Relational operator (less than or equal)</td>
</tr>
<tr>
<td>=</td>
<td>Relational operator (equals)</td>
</tr>
</tbody>
</table>

**Expression Example**

**Example: Translating Values**

Sometimes, all you want to know is “Does this person have something in this field or not?” or “What does this funny combination of numbers and letters mean in plain English?” You don’t care about the details that the system stores; you just want something that you can glance at and easily make sense of. The following will allow you to create an expression that will “translate” the system values into something that you, the query writer, define.

Suppose we wanted to look at `STDNT_CAR_TERM` and count the number of students enrolled in each class standing (freshman, sophomore, junior, etc.) for Fall 2016. Create your new query with `STDNT_CAR_TERM` and add your fields.

Click the “Edit” button next to the `EMPLID` field and apply the count aggregate function.
Next, add your criteria:

If you run your query now, the academic level will read "10", "20", "30", and other values that may not make much sense unless you know what each value translates to. Our job is to translate these values for the user, so that they don’t have to do it themselves. Doing this can make the query easier to read at a glance.

For most queries, you will want to remove the field that you are translating from the query. There’s no need to have both the old, hard to read stuff and your new stuff. For the sake of the example, however, keep it in if you want to see the “before and after” contrast.

Head over to the Expressions tab and add a new expression. The easiest way to translate our system values is through a CASE statement expression. They are written like this:

```
CASE
  WHEN A.FIELDNAME = 'system value 1'
  THEN 'your value'
  WHEN A.FIELDNAME = 'system value 2'
  THEN 'your value'
  ELSE 'default value'
END
```

Use field as a column in the result set
Using Expressions in Criteria

Case statements:
When there is a possibility of not meeting your specified condition(s), multiple rows can be returned for the same entry. To get around this you can use Aggregates such as minimum or maximum to specify which value you want returned.

Expression Gotchas
Any time you are working with number there is a potential for data distortion. You can get around this by using a sum or count expression but that may also eliminate rows that you may need.

Advanced Functions

<table>
<thead>
<tr>
<th>Version</th>
<th>Code</th>
<th>Description</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC1</td>
<td></td>
<td>Check if student is still in school when NoT爾 returns 0.</td>
<td>False if NoTő returns 0; True otherwise.</td>
</tr>
<tr>
<td>NC2</td>
<td></td>
<td>Check if student is still in school when NoTő returns 0.</td>
<td>False if NoTő returns 0; True otherwise.</td>
</tr>
<tr>
<td>ROWNUM</td>
<td></td>
<td>Returns a number indicating the row number not returned to a query.</td>
<td>ROWNUM</td>
</tr>
<tr>
<td>ROWNUM</td>
<td></td>
<td>Returns the number of rows returned to a SQL query.</td>
<td>ROWNUM</td>
</tr>
<tr>
<td>ROWNUM</td>
<td></td>
<td>Returns the number of rows returned to a SQL query.</td>
<td>ROWNUM</td>
</tr>
</tbody>
</table>

This is not an absolute value, as if you specify a ROWNUM not greater than 200 you may, for example, get 210 rows.
Pop Select Query Protocol

Pop Select Queries have the following protocol:
1. Identify the Query/Batch Process relationship. If unsure what the relationship is look at other queries that your run control can see to determine the correct BIND record.
2. All key fields in the BIND record must be displayed in the Query results.
3. Only display the BIND record key fields in the Query results. In some batch processes if any other column is displayed besides a BIND record key field it will cause the process to fail. If it is desired to use the Pop Select Query for data values in addition to its use in the batch process it is recommended to create two separate Queries.
4. Describe the Query as a Pop Select Query in the description.
5. Provide a full Definition, including the batch process the Query is tied to.
Having

Criteria Tab

Create Your Own Query!