PLANSOURCE

Best Practices for Improving Report Performance

Business Intelligence

Performance Guidelines

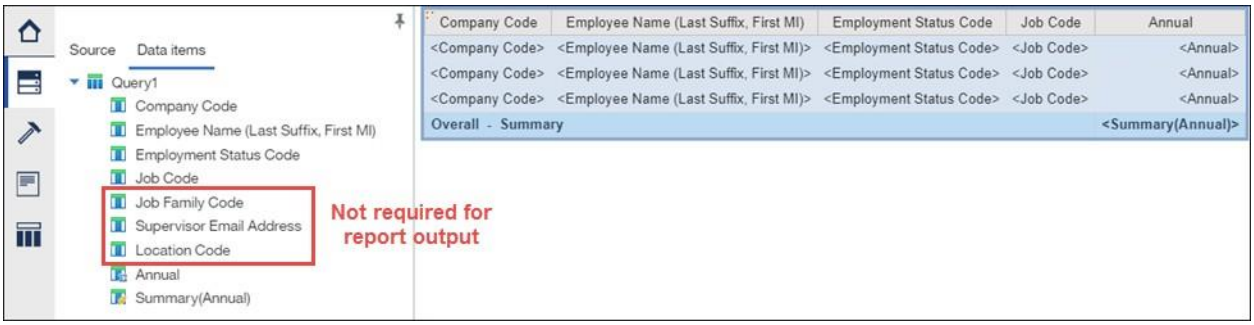
As you build more complex Business Intelligence reports, you may notice that some reports take longer than others to run. Often, simple design changes can result in report execution spanning a few minutes instead of several hours.

Follow recommended performance guidelines to ensure that your reports run quickly and work efficiently.

Data Items and Queries

Eliminate data items from the query that are not required for report output or filtering.

Unnecessary data items extend report run time.



In addition, remove queries that are not required for report execution.

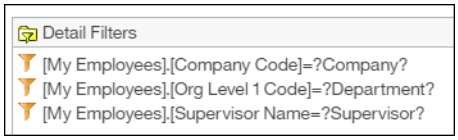
Filters

Filters narrow the scope of your report to include only specific records. Filters can affect report performance significantly.

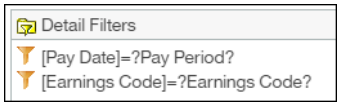
1. Order filters effectively.

   Order filters based on the amount of data filtered, from highest to lowest. The filter that would eliminate the most records should be applied first.

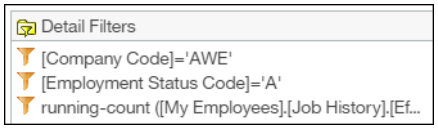
   Example 1: An executive of a 5,000 employee organization wants to view employees by supervisor. The report author should filter first by the data item that will eliminate the largest number of employees from the query; for example, by company. Then, filtering by department eliminates the next largest number of employees from the query. Finally, filtering by supervisor produces the desired results.



   Example 2: You want to report on Earnings Codes used in the past year; however, PlanSource includes ten years of payroll data. Filter on Pay Period to reduce ten years of data to one year. Next, filter for Earnings Codes.

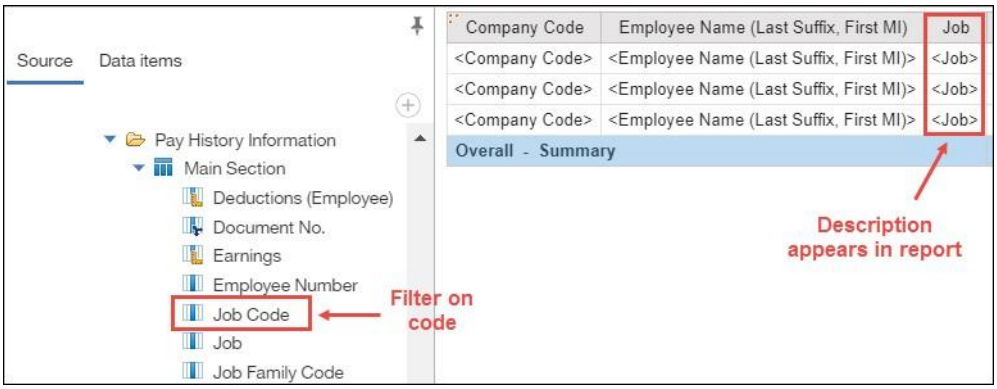


2. Position filters based on fields first and calculations last. In this example, filters for codes appear first and a filter for a calculation appears last.



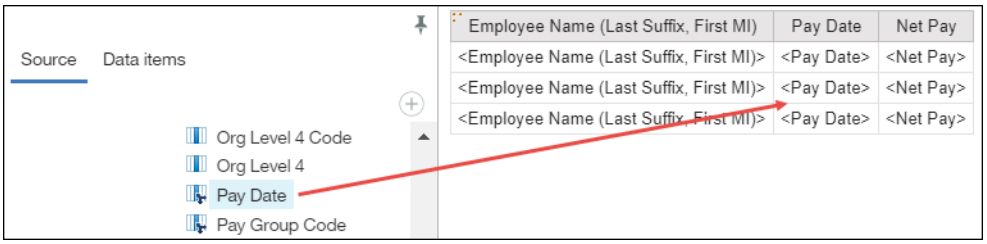
3. Filter on codes and not on descriptions.

   It is possible for two data items to have different codes but identical descriptions. Additionally, descriptions can change over time. Filtering on a code is the only way to ensure that the correct data item is retrieved.



4. When a data item is not required to appear in report output, narrow the scope of your report by creating a filter directly in the query.

   In the example, the Pay Date data item is added to the report specification, and then a filter is applied to Pay Date. This can result in long processing time because all pay history is pulled into the report before the filter is applied.



   A more efficient method is to apply a filter directly in the query.

5. When filtering for a single value, use the "=" (equal sign) instead of "IN" in the filter expression to improve performance.

   For example, use the following:

   - [Employment Status Code] = 'A'

     Do not use:

   - [Employment Status Code] IN ('A')

Joins

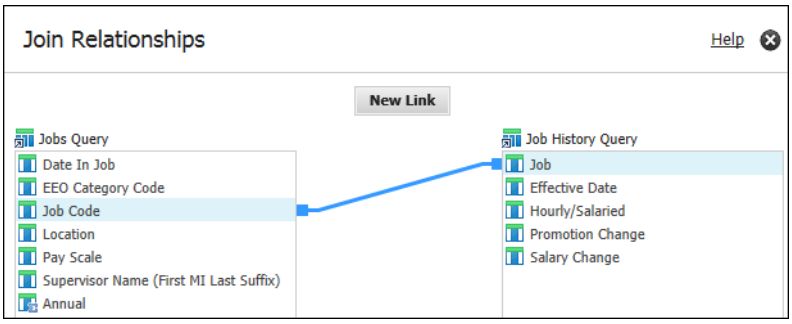
There might be times when a Join is required to combine multiple data sets. Joins must be created properly for optimal performance.

1. Check the links required when building Joins.

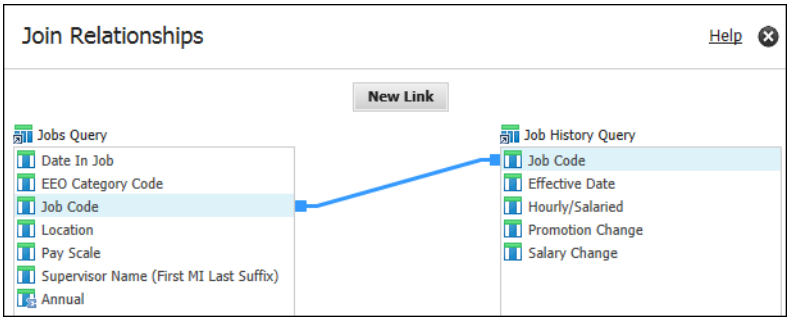
   When linking queries, if a query does not include the correct data items, then the queries cannot be correctly joined.

   In this example, Job is a description, but Job Code is a code. The output results in no matches being found.

   - Incorrect data items joined (Job Code joined to Job):



   - Correct data items joined (Job Code joined to Job Code):



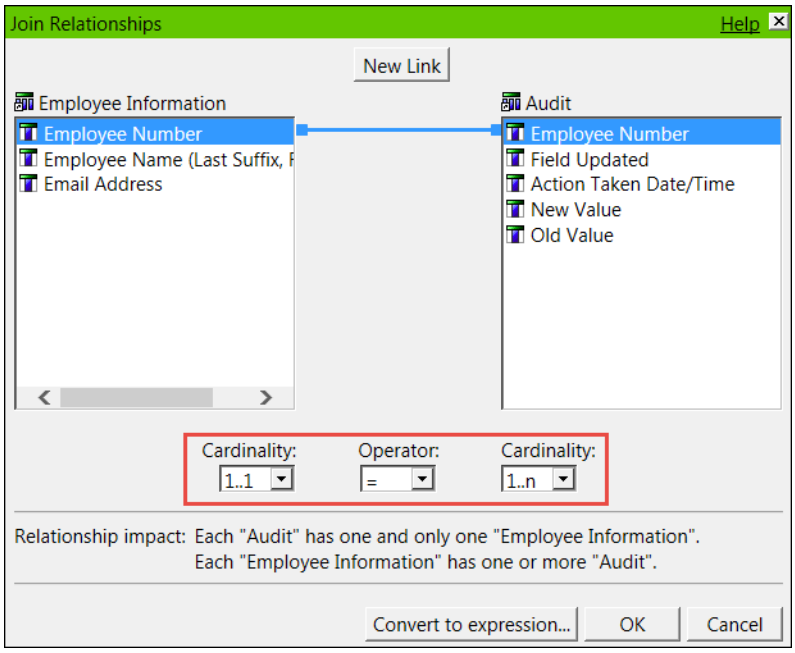
2. Use the correct Join cardinality.

   Cardinality defines the relationship of one query to another. Incorrect cardinality can increase report run time from minutes to hours.

   Use one of the following cardinalities:

   - (1,1) = (1,n)

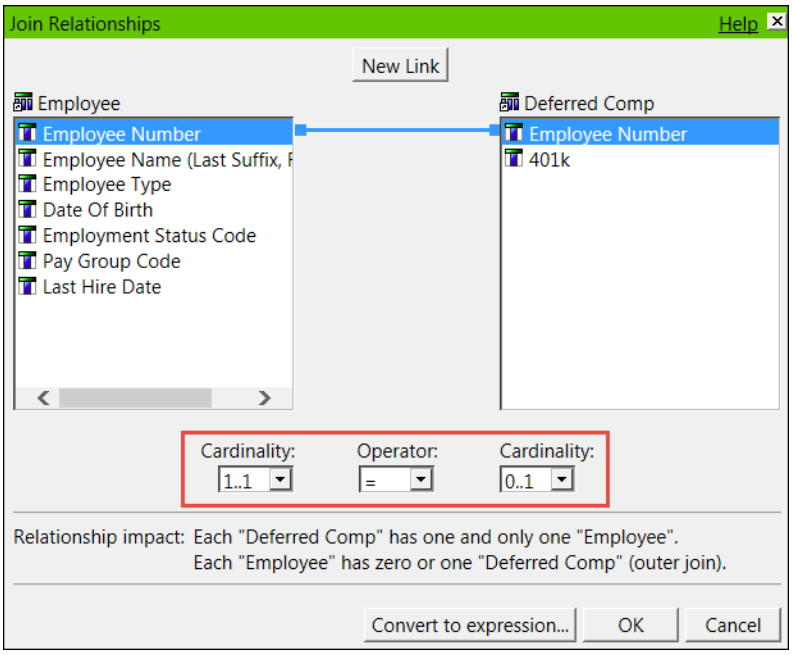
     This cardinality returns results when data exists in both queries. Use this cardinality when a record in Query 1 corresponds to one or more records in Query 2. In this example, employee names are listed in report output for those that have both employee information data and audit data.



- (1,1) = (0,1)

  This cardinality returns broader results. Use this cardinality when a record in Query 1 corresponds to zero or one record in Query 2. For example, employees (Employee query) may or may not have a 401(k) account (Deferred Comp query). A common use is in non-discrimination testing.

  In general, use this cardinality to identify employees who may or may not have something, for example, a benefit for which they are eligible. This cardinality can be used in many scenarios not limited to deductions.



Avoid using the following cardinalities:

- (0,n) = (0,n)
- (0,1) = (0,1)
- (1,n) = (1,n)

3. Avoid joining queries with historical data on both sides of the query because thousands of unnecessary records are read, resulting in extended runtime.

   Example:

   If there are 1,000 records in Query 1 and 1,000 records in Query 2, then 1,000,000 records are returned (1,000 x 1,000).

   - Job Code from My Employees > Jobs > Job History > Job Code
   - Job Code from My Employees > Pay > Pay History Information > Main Section > Job Code

   Filters should be used effectively to narrow data sets.

   In the rare case when this type of historical reporting is required, contact Consulting Services.