Epicor Kinetic - Classic User Interface
Business Activity Queries
Course
11.1

Disclaimer

ED872905

90521-10-0103-583101001

11.1

Revision: March 08, 2021 6:02 p.m.

Total pages: 77

course.ditaval

Contents

Business Activity Queries Course

Business Activity Queries Course

This course introduces the Business Activity Query (BAQ) Designer data extraction tool. It discusses data location concepts and provides an overview of query building techniques using the Query Builder, the Criteria Wizard, and the Calculated Field Wizard.

This course provides techniques for creating static business activity queries (BAQs) as well as updatable BAQs. Both queries can be used as the foundation for reports and dashboards, or to review specific details of your day-to-day business.

The course begins with a discussion on which tools you can use in Epicor ERP Version 10 to locate the table and field, and understand table relations. It continues with an introduction to BAQ Designer, then covers techniques you can use to create simple queries, also called FLAT queries. These BAQs are comprised of one TopLevel SubQuery. The next part of the course explains how to create more complex BAQs, comprised of several SubQueries of different types, to obtain the desired information from the application database.

The course continues with discussion of the updatable BAQ functionality - a powerful mechanism to share and update data details when used as the foundation of a dashboard. The updatable BAQ creates a powerful tool for team collaboration displayed on the standard Epicor application, or modified for display on a mobile device such as IPhone®.

The course concludes by exploring BAQ Utilities - additional capabilities such as BAQ export, or changing an author.

Upon successful completion of this course, you will be able to:

- Use the field level technical help or the data dictionary viewer to locate the table and field criteria for use in a BAQ.
- Create a simple BAQ using the Query Builder and its query-building views.
- Construct an advanced BAQ comprised of several SubQueries.
- Use the Criteria Wizard to filter the data returned by the BAQ.
- Add calculated and aggregated values to a query using the Calculated Field Wizard.
- Design an updatable BAQ to increase team collaboration.
- Import or export queries between companies, or to a .xml or ASCII file type.

Before You Begin

Read this topic for information you should know in order to successfully complete this course.

Audience

Specific audiences will benefit from this course.

- CFO/Controller
- COO/Operations Manager
- Production Manager
- Account Manager
- Sales Representative
- System Administrator
- IT/Technical Staff

Prerequisites

To complete the workshops in this course, the necessary modules must be licensed and operating in your training environment. For more information on the modules available, contact your Epicor Customer Account Manager. It is also important you understand the prerequisite knowledge contained in other valuable courses.

- **Foundations Agenda Courses on Epicor Learning Center** - These courses describe logging in to Epicor ERP, using menus and toolbars, working with Tree view and sheets. They give you a quick overview how to enter data in Epicor ERP, use searches to find data and work with grids. The courses in this agenda teach you to personalize your application, print forms and reports and use trackers to view information.
- **Database Concepts Course** - This course reviews the table and field name identification process using Field Help, Customization Tools, and the Data Dictionary Viewer functionality. It also describes table linking procedures and requirements as well as join type definitions and specifications.
- **Recommended Industry Knowledge:**
  - Fundamental knowledge of relational database concepts such as table relationships, records, and field types.
  - An understanding of the functionality of the current release of the Epicor application.

Environment Setup

The environment setup steps and potential workshop constraints must be reviewed in order to successfully complete the workshops in this course.

Your Epicor training environment, in which the Epicor demonstration database is found, enables you to experience Epicor functionality in action but does not affect data in your live, production environment.

The following steps must be taken to successfully complete the workshops in this course.

1. Verify the following or ask your system administrator to verify for you:
   - **Your Epicor training icon (or web address if you are using Epicor Web Access) points to your Epicor training environment with the Epicor demonstration database installed.** Do not complete the course workshops in your live, production environment.

     Note It is recommended that multiple Epicor demonstration databases are installed. Contact Support or Systems Consulting for billable assistance.

   - **The Epicor demonstration database is at the same version as the Epicor application.** The demonstration database is installed from the Epicor Administration Console using the "Add Demo Database" command under Database Server. See Epicor ERP installation guides for details. If you are an Epicor Cloud ERP customer (and have licensed embedded education), the demonstration database is installed for you.
   - **Your system administrator restored (refreshed) the Epicor demonstration database prior to starting this course.** The Epicor demonstration database comes standard with parts, customers, sales orders, and so on, already defined. If the Epicor demonstration database is shared with multiple users (that is, the database is located on a server and users access the same data, much like your live, production environment) and is not periodically refreshed, unexpected results can occur. For example, if a course workshop requires you to ship a sales order that came standard in the Epicor demonstration database, but a different user already completed this workshop and the Epicor demonstration database was not restored (refreshed), then you will not be able to ship the sales order. If you are an Epicor Cloud ERP customer see section below.

2. Log in to the training environment using the credentials **epicor/epicor**.

   If you are asked to log into your training environment with another User ID, for example, 'nancy' or 'bhoward', on the initial logon, a message appears to inform you the password has expired. Perform the following steps to set a new password:

   1. To the **Password Expired** message, click **Yes**.
   2. In the **Current password** field, enter the User ID of the user you are asked to log in as, for example, 'nancy'.
   3. In the **New password** field, enter a new password, for example '**Train123**'.

      Important In **Epicor ERP Cloud** environment, the password must not contain user ID, must be longer than 7 characters and include at least one uppercase letter.

   4. In the **Confirm new password** field, re-enter the same password.
   5. Click **OK**. The Change Password window closes and you are logged on with the new user ID.

Note Record the new password. This is important as this will be the password everyone uses when they log on with this User ID, until the database is refreshed.

3. From the Main menu, select the company **Epicor Education (EPIC06)**.
4. From the Main menu, select the **Main** site.

Kinetic Interface Specific Information

All the workshops in this course can be completed using the **Classic** layout. To ensure that the user interfaces used in this course display in the Classic layout, use the steps below:

1. Navigate to **Kinetic Application Maintenance**.

   Menu Path: System Setup > Security Maintenance > Kinetic Application Maintenance

2. From the **Menu** list, select the **Epicor Education** company.

   The **Company Epic Corporation** pane displays.

   Note If you need to work in a multi-company environment you will have to disable the Kinetic layout for all the companies used in the course.

3. Inactivate the **Enable Kinetic UI** ON/OFF button.
4. Exit Kinetic Application Maintenance.
5. On the standard toolbar, select **Options > Preferences**.
6. In the window that displays, in the **Form To Use** field, select **Classic**.
7. Select **OK** to confirm.
8. Relaunch Epicor ERP.

Epicor Cloud ERP Specific Information

Note If you are an Epicor Cloud ERP customer, then note the following about your Epicor-hosted education company. All logins referenced in the course (such as manager, or epicor) should be changed to be the *<site ID>*-. For example, if your site ID is 98315, then wherever you are instructed to use the login **epicor**, instead use **98315-epicor**. The password is 'Train18!'.

Note To refresh your Epicor training data, enter a support ticket in **EpicCare** and include your site ID.

Business Activity Queries (BAQs) Overview

Use the **Business Activity Query Designer** (BAQ) to create personalized queries. You can use custom queries and custom updatable queries as the foundation for dashboards, quick searches, and BAQ reports. The exporting capabilities makes the queries available to view and edit in third party applications.

The BAQ Designer has several components you can use to compile a query:

- Use the **Query Builder** to drag and drop tables onto the design area. The table properties display on the left of the design canvas area. If necessary, select the connection link and draw a connection between tables. Use the filter to quickly access the table name. The Query Builder is composed of the following sheets:
  - The **Phrase Build** sheet is where you select the tables and fields you wish to include in the query. Use this sheet and its sub-sheets to set up everything from basic queries with a single table to complex joins between multiple tables or SubQueries.
  - The **Display Fields** sheets define which columns display and in what order they display in the query. You can also create and display a special calculated field you need within the current business activity query.
  - The **SubQuery Options** sheet is where define SubQuery parameters. When you construct a BAQ, use this sheet control what data displays in the SQL output by selecting an appropriate SubQuery type. You also have the ability to control the SQL results set. For example, you can construct an SQL text to only display top 50% of rows from the retrieved results set.
  - The **SubQuery List** sheet displays all SubQueries you create within a BAQ. On this sheet, all SubQueries you create are ordered by the sequence number. You have the ability to create and remove SubQueries, change their sequence value and view how data displays in the results set.

  Six sheets are available at the bottom of the Phrase Build sheet. Use these sheets to indicate how tables are linked together, define the relation between the tables, and specify the selection criteria for the query.
  - Use the **Table List** sheet to view the list of tables that make up your query and to change their order.
  - Use the **Table Relations** sheet to display and modify the fields that make up the joins between tables and subqueries. You can add, modify and delete the join fields within the current query.
  - Use the **Table Criteria** sheet to enter criteria for a table. You can apply one or more filter criteria to any table in the query.
  - Use the **SubQuery Criteria** to enter criteria for the whole SubQuery. These criteria are used to construct the WHERE and HAVING clauses in a SELECT statement.
  - Use the **Function Call Parameters** sheet to specify values for the parameters in a Table-Valued Function. A Table-Valued Function (TVF) is a user-defined function that returns a table. This option is only available in External Business Activity Query.
  - Use the **Pivot SubQuery FOR Clause** sheet to build the PIVOT clause within a query to transform data from row-level to columnar data.

You can use queries in various areas including the following:

- Dashboards
- Business Process Management (BPM)
- Epicor Portal Views
- Information Worker
- Enterprise Search
- Service Connect
- SharePoint web parts

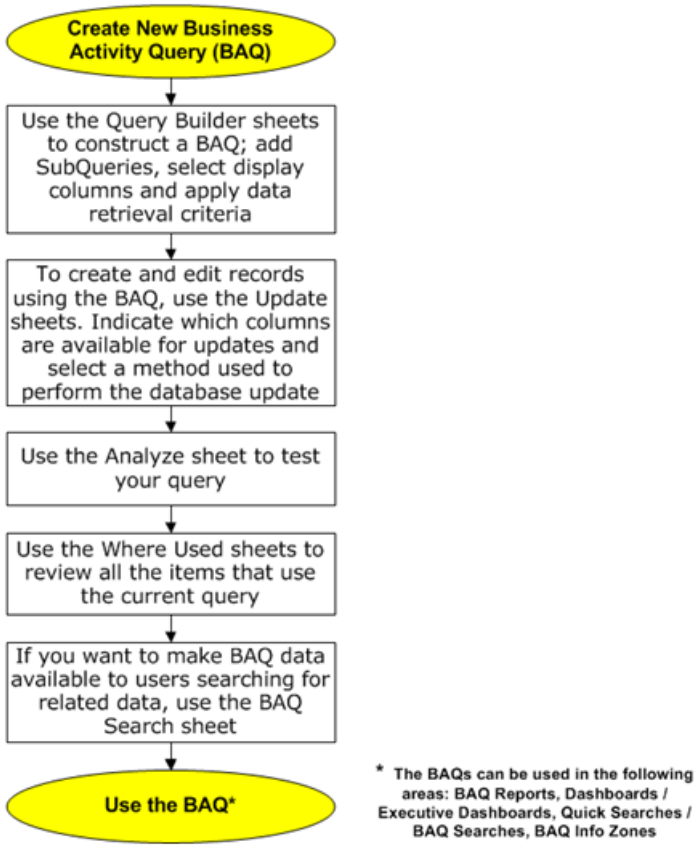
- BAQ Zones
- External queries using ODBC connections

Menu Path

Navigate to this program from the Main Menu:

- Executive Analysis > Business Activity Management > Setup > Business Activity Query

Important This program is not available in Epicor Web Access.



Workshop - Tour the BAQ Designer

This workshop provides a tour of the Business Activity Query (BAQ) Designer using a system query as the foundation.

Navigate to the **Business Activity Query Designer**.

Menu Path: Executive Analysis > Business Activity Management > Setup > Business Activity Query

Important This program is not available in Epicor Web Access.

Locate an Existing Query

A number of existing BAQs are shipped with the Epicor ERP application. Those on the list beginning with **z** are system queries which are used by the application to build reports, dashboards, and so on.

1. Click **Query ID** and click **Search**.
   Shared queries display in the query search. If the query is not identified as shared, it is only accessible by the author of the query.

2. In the **Query ID Starts With** field, enter **z**.

3. Search for and select one of the system queries installed with the Epicor ERP application - **zCustContacts**.
   To make your selection, double-click the query, or select the query and click the **OK** button.

   Note zQueries are write protected and you cannot modify them. If one of the application queries provides the information you need, copy the query and then modify as necessary. Notice the **System BAQ** check box and the message indicate you are working with the system query.

4. View the **Query Phrase** section.
   This section is an English translation of the Phrase that was used to build this query. It is presented in an SQL statement.

Review the Query Builder

Use the **Query Builder** sheets to design a Business Activity Query. You first select what data you want to display. You can use a simple table to design your BAQ or you can drag multiple tables and SubQueries onto the canvas and link them. You then select what columns you want to display in your BAQ and specify other selection criteria.

1. Click the **Query Builder** sheet.
   Verify the **Query Builder > Phrase Build** sheet is in focus.

2. View the **Active SubQuery** field displays **SubQuery1**.

   Note Each Business Activity Query can contain one main SubQuery, where Type=TopLevel and several SubQueries of different types that determine how they correlate with each other.

   The BAQ you selected - zCustContact, is an example of a flat query as it constructed using the single main SubQuery.

Use the Phrase Build sheet to design a query using the visual representation of the current SubQuery. A SubQuery in general can be made of tables as well as different SubQueries.

On this sheet, you select tables and other SubQueries you wish to include in the current SubQuery. You can set up everything from basic queries with a single table to complex joins between multiple tables or SubQueries.

Initially, application database tables display in the table palette in the left pane. To include a table in your query, select a table and drag it on the canvas in the center pane. You can perform the same action by double-clicking a table. It is also possible to add multiple tables at the same time. The tables, of which the query is composed, display as diagram blocks.

Six sheets are available at the bottom of the Phrase Build sheet. Use these sheets to indicate how tables or SubQueries are linked together, define the relation, specify the selection criteria for the resulting data set and build the PIVOT clause.

3. Navigate to the **Display Fields > Column Select** sheet.

   Use the Column Select sheet to select the specific columns (fields) you wish to display in the current SubQuery.

   The sheet lists the table fields available to select in the SubQuery. You can also use it to rename the column labels (not the field names).

4. On the **Column Select** sheet, select the **Calculated Field Editor** button.

   The **Calculated Field SQL Editor** window displays. It includes a display of all available fields by table for use in side calculations. On the right, it displays pre-defined functions and mathematical operators available, as well as a small calculator to check your values.

   As you create calculations, they display inside the **Editor** section of the screen. Use the **Check Syntax** button in the lower right to validate the calculated statement.

5. Exit the **Calculated Field SQL Editor** window.

6. On this sheet, you can also access the **Advanced Group By Clause Editor**, where you can build advanced data summarizing expressions.

7. Lastly, you can use the **Field Attributes Editor** to add, edit or remove attributes of the fields that display on your query.

8. Navigate to the **Display Fields > Sort Order** sheet.

   Use the Sort Order sheet to arrange the sorting order for fields. You can move fields up or down and select a descending order of fields, if you do not wish to use the default ascending order.

9. Navigate to **Query Builder > SubQuery Options** sheet.

   Use this sheet to define SubQuery parameters. A subquery is a query that is nested inside a SELECT statement, or inside another subquery. A subquery can be used anywhere an expression is allowed.

   When you create a new BAQ, it is automatically defined as Top Level SubQuery1.

10. Navigate to **Query Builder > SubQuery List** sheet.

    Use this sheet to view and manage all SubQueries you use to build the BAQ. When you create multiple SubQueries, they become ordered by the sequence number.

    In this example, a single SubQuery displays in the grid.

    More information on SubQuery functionality is found later in the course.

Updatable BAQ Properties

Updatable BAQs are queries you can edit to populate application data. This exercise is a brief tour of the sheets; more details are provided later in the course.

1. Navigate to the **Update > General Properties** sheet.

   Use this sheet to define the base rules for an updatable BAQ. All selected columns display in the list and can be defined as updatable.

2. Navigate to the **Update > Update Processing** sheet.

   An updatable BAQ requires further definition to tell the application how to handle the updated criteria. You use the controls on the Update Processing sheet to select and configure a processing method used to update the target database.

Analyze a Query

After a query is built, use the Analyze sheet to check the syntax (Analyze) and to execute the query to view results.

1. Navigate to the **Analyze** sheet.
   Notice the buttons on this screen.

   The **Analyze** button runs a final syntax check of the entire query statement. The **Test** button executes the query so you can test the accuracy of the results. If the results are different than expected, return to the other areas of the Query Builder to make corrections.

   In the right corner, there are other buttons specific to updatable BAQs. These are used instead of the Analyze and Test buttons and are discussed later in the course.

2. Click the **Analyze** button and verify the **Syntax is OK** message displays.

3. Click the **Test** button to execute the **zCustContacts** query.
   By default, the query is allowed to return maximum of 10000 records. This limitation is only applied on retrieving BAQ results while testing their execution.

   Tip When you design a query, test its results and the execution takes too long, you can interrupt testing by clicking the X button. On BAQ execution, this button displays to the right of the Clear Grid button.

4. View the **Query Results** grid that displays the query output.

   Tip If you want to limit the number of rows returned by the BAQ, select a predefined value from the **Rows to Return** drop-down list or enter a value of your choice.

Review the Where Used Sheets

Use the **Where Used** sheets to review all the items that use the current query. The information on these sheets helps you decide if you should re-design the current query, delete the query, or create a new query.

1. Navigate to the **Where Used > Dashboard List** sheet.
   Use this sheet to see what dashboards incorporate the current query.

2. The remaining **Where Used** sheets to display the following feature-specific information.
   - **Quick Search List** - Displays all quick searches that use the current query.
   - **BAQ Report List** - Displays the BAQ reports that use this query.
   - **Report List** - Displays BAQ reports that use this query as a data source.
   - **Business Activity Query List** - You can link a business activity query as a parameter to another query. This sheet displays all business activity queries that reference the BAQ in focus.

Review the BAQ Search Sheet

Use the **BAQ Search** sheet to select data in your query that you want to make available to users searching for related data.

1. Navigate to the **BAQ Search** sheet.

A BAQ search is a feature that allows custom BAQs to become a base search criteria for a specific field. It requires the use of **Like** columns, which link the BAQ to the specific field used for the search. This is a very powerful tool discussed later in the course.

2. On the **Standard** toolbar, click **Clear**.

Application Setup

The first step to create a query is to find the appropriate data. The Epicor application offers tools you can use to locate the table and field data for the query.

This section covers the process of table and field name identification, use of the Online Technical Field Help, the Data Dictionary Viewer functionality, and the Table Relations from Dictionary Tracker embedded within the BAQ Designer. Also included is a discussion of table relationships and joining styles.

Table Linking

When you use two or more tables in a query, you must identify a link between the tables to pull records from both tables.

Example

If you use the Quote table and the Customer table, connect the tables so that each quote is matched up with the customer on the quote.

Table Relationship

If the query pulls data from multiple tables, you must create relationships between tables through a joining process. Use indexed fields to establish a relationship or join. In general, an index points to the data location. It is a named path designed within the database using common table fields to quickly locate, store, or retrieve data. Use the Data Dictionary to identify the index name, path, or index order.

Note

Always use the **Company ID** field as the first indexed field, since it is at the top of the Epicor application table hierarchy. The application uses the Company ID field to store the data for each company separately from the data for other companies. Since all tables use the Company ID field, select it as the first field to use in a join.

Joining Types

Following are the four basic types of joins you can select in BAQ Designer:

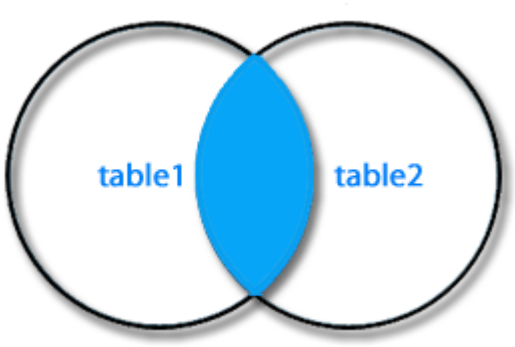
- Inner Join
- Left Outer Join
- Right Outer Join
- Full Join

Important

It is crucial a user explicitly determines the correct order of tables in the query to retrieve the expected results.

Inner Joins

An inner join, also known as an equal join, is the standard type of join. The report output from an inner join includes all the records in which the linked field value in both tables is an exact match. Records from either table that do not have a match in the other table are excluded from the report.



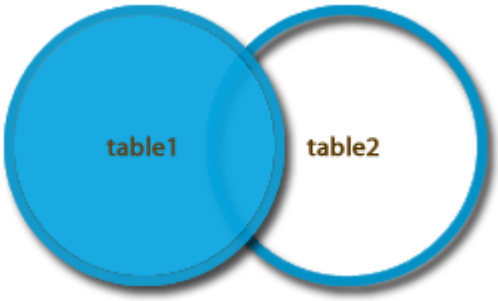
Example

The join between the **Customer** table and the **OrderDtl** table creates a view that displays customers and orders placed. In this case, the view includes only customers who have placed an order. Records for any customers who have not placed an order are excluded.

Use this join to display only matching records between the primary table and the lookup table.

Left Outer Joins

A left outer join retains all of the rows of the "left" table, regardless of whether there is a row that matches on the "right" table. The "left" table is simply the table that comes first in the join statement, therefore, it appears to the left of the keyword "join".

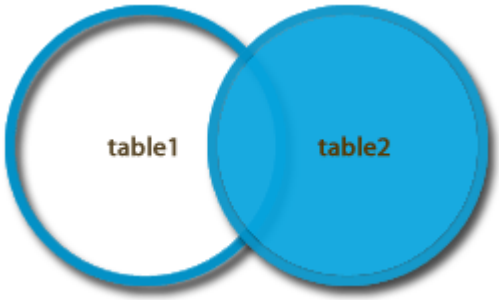


Example

Use a left outer join to view all customers and the orders for these customers, and to retrieve a row for every customer who has not placed any orders. Fields that otherwise hold the order information display blank for these customers.

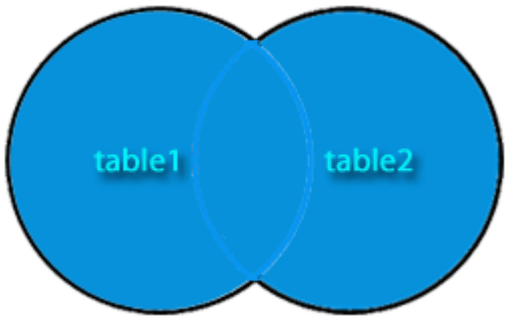
Right Outer Joins

A right outer join is similar to left outer join, except that all the rows from the right table are displayed in the result set, regardless of whether or not they have matching values in the left table.



Full Join

In SQL the Full Outer Join combines the results of both left and right outer joins and returns all (matched or unmatched) rows from the tables on both sides of the join clause.



Field Help Technical Details

The Field Help feature is a quick reference tool that provides a brief field description and the technical property reference for selected fields.

To enable Field Help, from the **Help** menu, select **Field Help** and click a field in the interface.

The Field Help sheet contains two menu items:

- **Field Level Help** - The Field Level Help is a text description of the field from the Application Help. You can use Field Help as a learning tool, as it allows you to access documentation for each field.
- **Technical Details** - The technical details include the data dictionary information for the field. You use the technical specifications for a field when building business activity queries (BAQs), using Business Process Management (BPM) methods, and other advanced functions of the Epicor ERP application.

  Note To view technical details, you must have permission to access the corresponding business object. This permission is set in **Process Security Maintenance** for the **bo.DataDict** business object.

  The technical details of the Field Help are valuable in understanding the table structure of the database. Keep in mind that data can reside in multiple tables. The query topic displays where the data comes from.

  You can also use the Customization feature to locate tables and fields.

Workshop - Use the Field Help

In this workshop, use the Field Help to find table and field names.

The Field Help provides a Technical Details sheet to display the data dictionary information for a selected field. The properties that display are helpful when you create a Business Activity Query and discuss the Data Dictionary Viewer.

- **Field Name** - Displays the field name as defined by the Epicor application. This name is used in all expressions.
- **EpiBinding** - Displays the tables (epiDataViews) stored in memory on the client workstation. The syntax is always <Tablename>.<Fieldname>. For example, the field displays Part.PartNum, which means the table is the Part table and the field to pull into the query is the Part Number field.
- **DB Field** - Displays the database field property which also displays as <Tablename>.<Fieldname>. When you create or modify reports, the DB Field property defines the field in the report layout. While usually a direct connection exists between the DB Field and the EpiBinding, in some cases fields in the epiDataView are not found in the corresponding data table.
- **Format** - Describes the database format for this field and the number of characters to which that field is limited.
- **Like** - Use this field to validate a BAQ search and generate Foreign Key Views to indicate the common field between two tables or datasets.

1. Navigate to **Part Maintenance**.

   Menu Path: Material Management > Inventory Management > Setup > Part

2. From the **Help** menu, select **Field Help**.

   In the tree view pane, the **Field Help** sheet displays. It automatically opens in an undocked position.

3. Place the cursor in the **Field Help** sheet header and click the **push pin** icon to dock the **Field Help** window.

4. Place the cursor in the **Description** field.

5. In the **Field Help** sheet, click **Technical Details** and view the information.

   You can adjust the Field Help width to see the fields properly.

6. Navigate to the **Part > Sites > Warehouses > Primary Bin** sheet and place the cursor in the **Bin** field.

   The properties populate with the data in this field. The table and field combination display in the EpiBinding field of the Technical Details in the format of <table>.<field>.

7. Navigate to the **Part > Sites > Warehouses > Bin Information > Detail** sheet and place the cursor in the **Bin** field.

   Notice the database field is different.

   Note Data is often stored in multiple places, and table selection depends on the purpose of the query. This example presented a part's listing with a part bin location.

8. Exit Part Maintenance.

Data Dictionary Viewer

Use the **Data Dictionary Viewer** to find and review details of each field and table within the database. It helps you better understand the purpose and data values of each field.

This utility may help you identify the fields and tables to use for a customization, BAQ, or custom report. The Data Dictionary Viewer is also helpful during upgrades, as you can view the current database structure and compare it against a previous database version.

Tip You can print a hard copy of the data dictionary and use this report as a reference during upgrades and customization.

To access this report, from the **Actions** menu select **Print Field Definitions**. Define report parameters in the **Data Dictionary Field Report** window.

Tables Sheet

The Data Dictionary is organized by tables. A table is a set of fields that contains related information. Use the **Tables** sheet to find and select the database table to review.

Example Use the **Customer** table to store all your customer records. The **OrderHed** table records your Order Header records.

Data is often stored in multiple tables, each with a specific purpose. Use the **Search** functionality to display tables in a grid format. The grid includes a brief overview of the tables and helps you determine which table you should use when a field is located in more than one table.

Fields Sheet

The **Fields > Detail** sheet displays all values for a selected field, such as format, label, and description.

Example The customer table includes the identifying code for the customer (CustNum), name (Name), and other specific customer details.

For information regarding the Data Dictionary Viewer properties, refer to the Technical Details topic in the Field Help.

Workshop - Use the Data Dictionary Viewer

This workshop demonstrates how to use the Data Dictionary Viewer to identify fields and tables.

Navigate to the **Data Dictionary Viewer**.

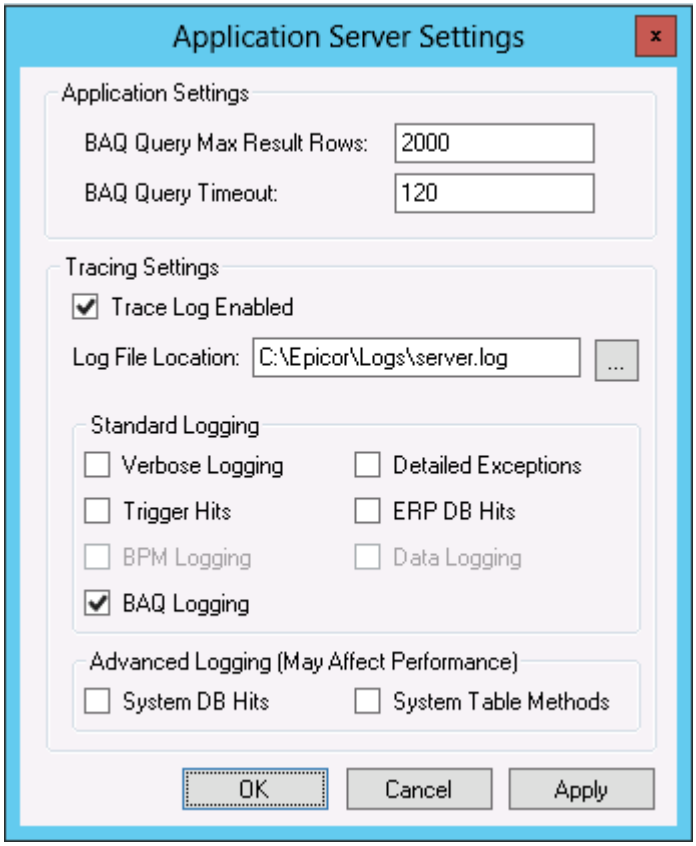
Menu Path: System Setup > System Maintenance > Data Dictionary Viewer

1. For the **Table Schema Type**, select **Product**.
2. In the **Table** field, enter **OrderHed** and press **Tab**.
   Information populates in the **Tables** sheet, the **Fields** sheet, and the tree view.
3. In the **Description** field, review the brief description about the purpose of the table.
4. In the **Indexes** grid, view the **IndexFields** column. This column displays the index name used by the database to access a specific field. Each field is displayed using its database schema name. For example, IX_OrderHed_CustPO.

5. The **Fields** column displays which column(s) of a database table are associated with a particular index as well as the order in which columns are listed in the index definition.
6. In the **Display Format** pane, select **Alphabetic**.
   This option sorts fields alphabetically.
7. Navigate to the **Fields > Detail** sheet.
8. In the **Field Name** field, from the drop-down list, select **DiscountPercent**.
9. Review the information about the selected field.
   This way, you can understand the purpose and data values of every field you want to include in your query.
10. Exit the Data Dictionary Viewer.

BAQ Application Server Options

Several BAQ related options are found within **Application Server Settings** accessible from the **Epicor Administration Console**.



Using the **BAQ Query Max Result Rows** field, you can limit the number of rows returned by each Business Activity Query. This prevents the query from pulling in an unlimited number of records, restricting situations where a runaway BAQ consumes too many system resources to generate query results.

For the **BAQ Query Timeout** field, you can enter how many seconds can elapse before the application server stops the query. By entering a value in this field, you define how long each BAQ is allowed to run. When a query

attempts to generate results and reaches this time limit, the application server stops the query and sends the user a time out message.

To record BAQ database calls within the application log, select the **BAQ Logging** check box. Each time user activity activates a BAQ, the application server log records which query was called and how long it took this BAQ to gather the data results.

BAQ Design

This section describes the process for creating queries using the Business Activity Query Designer.

The Query Builder

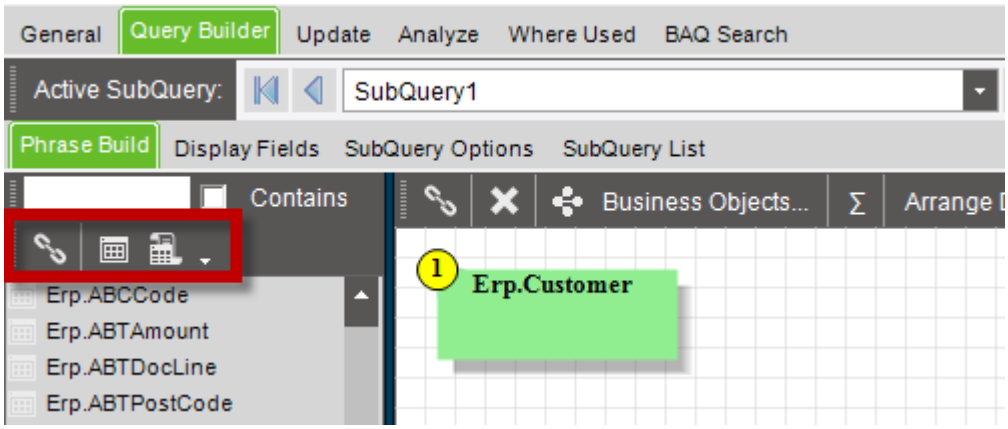
Use the **Query Builder** sheets to design a Business Activity Query.

The **Phrase Build** sheet is where you design the current SubQuery.

In the left pane, the list of all Epicor ERP tables displays. You can use the buttons above tables to display:

- **Connected Only Tables** - Only the tables that have relations, described in system tables, with the table selected on the canvas.
- **SubQueries** - Another SubQuery created within the BAQ referenced in the current SubQuery.
- **Table-valued functions** - These items are user-defined functions created in the SQL database that return a table. You click and drag the table valued functions from a panel just like an ordinary table. They usually require specifying input parameter values you define on the **Function Call Parameters** sheet. This functionality is only available for use with External BAQs.

Example



To include a table, SubQuery, or a Table-valued function in your query, select and drag it on the canvas in the center pane. You can perform the same action by double-clicking on the item. You can also select multiple items by holding the Ctrl button as you click.

To easily locate the table you want, in the **Filtering** field, enter a value.

Six sheets are available at the bottom of the **Phrase Build** sheet. Use these sheets to do the following:

- View the list of tables and SubQueries used in the current SubQuery
- Define the relation between the tables and SubQueries
- Specify the selection criteria for the tables and SubQueries
- Specify values for the parameters in a Table-Valued Function
- Apply the pivot operator within a query to transform data from row-level to columnar data

Example To create a query that displays part and bin information, double-click the Erp.Part table and select the **Connected only** check box to filter the list of available tables that contain related information to the Part table. This helps narrow the list of available related tables from which you can select.

Tip Instead of selecting tables individually, you may use the **Business Objects** button to search for and load table(s) within an entity, for example, Erp.Part. The links between loaded tables display automatically.

The **Display Fields** sheets define which columns display and in what order they display in the query. You can also create and display a special calculated field you need within the current business activity query.

The **SubQuery Options** sheet is where you define SubQuery parameters. When you construct a BAQ, use this sheet to control what data displays in the SQL output by selecting an appropriate subquery type.

The **SubQuery List** sheet displays all SubQueries you create within a BAQ. This sheet orders all the SubQueries you create by the sequence number. You have the ability to create and remove subqueries, change their sequence value, and view how data displays in the results set. You can also use parenthesis to group SubQueries.

Table List

The **Query Builder > Phrase Build > Table List** displays the list of tables and SubQueries you place on the canvas for the SubQuery in focus.

If your BAQ incorporates multiple SubQueries, switch between them using the **Active SubQuery** navigational toolbar found above the Phrase Build tab.

Table Relations

Use the Table Relations sheet to display and modify the fields that make up the joins between tables. You can add, modify and delete the join fields within the current query.

Important By default, queries are set up to display Inner Joins, which means that data from the first table only displays if it is linked to data within the second table. The query output from an inner join includes all the records in which the table relations values in both tables are an exact match. Records from either table that do not have a match in the other table are not included in the query results.

Example

Use an inner join to view all customers and the orders they have placed. You will not get a match for any customer who has not placed orders. Most queries you create only need this type of join.

The following options are available when joining tables:

- **INNER JOIN** - only rows satisfying join criteria from both joined tables are selected.
- **LEFT OUTER JOIN** - rows satisfying join criteria from both joined tables are selected as well as all remaining rows from left joined table are being kept along with Nulls instead of actual right joined table values.
- **RIGHT OUTER JOIN** - rows satisfying join criteria from both joined tables are selected as well as all remaining rows from right joined table are being kept along with Nulls instead of actual left joined table values.
- **FULL OUTER JOIN** - rows satisfying join criteria from both joined tables are selected as well as all remaining rows both from left joined table and right joined table are being kept along with Nulls instead of values from other table.

You can use the **Dictionary** button to view and select one of the predefined relations between tables. When two tables are placed on the canvas, and relation between tables is described in the dictionary, the relation is drawn by the BAQ automatically. As tables are placed on the canvas subsequently, the newly added table always checks relation with the previous table within the table order. It does not check relations to all possible tables within the query. This button is enabled when some predefined relation exists in the dictionary for the selected join connection, and its title contains number of predefined dictionary relations in the parentheses. The **Table**

Relations from Dictionary window that displays when you click the button list all relations with their fields. The result relation expression displays in the **Expression** textbox at the bottom. You can select one of the relations and press **Replace In Query** button. As a result, fields from Dictionary form will replace fields used in current BAQ relation.

Display

Once you identify the query tables in the Phrase Build, define the remaining query parameters. These parameters include which fields display as columns, define column headings labels, and sort the data. Additional sheets analyze the query for syntax, test the query, and identify where the query is used in the Epicor application.

The **Display Fields** sheet consists of two sub sheets:

- Use the **Display Fields > Column Select** sheet to define the columns that display on your query.

  You can use this sheet to set up the order in which the selected columns display in BAQ results and consequently, in dashboards or reports that will use the BAQ as the datasource. You can also configure the display names for each column and create a calculation for a selected field. The display names display on the query instead of the default column name.

  Example The default column name taken from the database is **Customer.CustID**. You can configure the display name for this column as **Customer ID**.

- The **Display > Sort Order** sheet displays the selected tables in the order they were selected. One or more table and field combinations can be used to define the order of the query data display.

  By default, the fields in each table display in schema order. Use the A to Z icon to sort all tables and fields alphabetically.

Workshop - Manually Join Tables

This workshop demonstrates how you manually join tables in a BAQ.

In many instances when multiple tables are placed on the work surface they join automatically. This occurs because there is a predefined join in the system for the selected tables. On occasion there are multiple joins predefined and you can use the Dictionary button select among joins. In this example, we are trying to join two tables manually because the desired join does not appear manually nor is it selectable via the dictionary.

Our query joins the OrderHed table with the Customer table. We want to display information about the order and about the customer to whom the order was shipped.

Define the Query Tables

Maximize **Business Activity Query Designer**.

1. Click **New** to create a new query.
2. In the **Query ID** field, enter **XXX_ShipToOrders** (where XXX are your initials) and press **Tab**.

   Important The Query ID can contain any value that does not exceed 30 characters in length and must not contain invalid symbols such as (/:*?<> or space.

3. In the **Description** field, enter **Orders and ShipTos**.
4. Select the **Shared** check box.
   This value indicates that other users within the current company can use this query.
   Notice the remaining BAQ options:

Query Type	Description
All Companies	Indicates whether the BAQ definition is visible across companies. The BAQ can only be updated from the original company displayed in the Owning Company field, but is available for execution and review in other companies. To modify the BAQ definition in other companies, create a copy of the BAQ.
Updatable	Indicates whether the current query will be used for data entry. Selecting this check box activates the sheets under the **Update** tab.
Cross-Company	Indicates whether the selected query brings back results from multiple companies.

5. Navigate to the **Query Builder > Phrase build** sheet.
   View the list of all available database tables you can use to construct the BAQ.

6. In the **Filtering** field that displays above the list of tables, enter **Custom**.
   Notice all tables beginning with **Custom** display in the table window.

   Note Epicor tables available in the BAQ designer come from two different schemas. Schemas are identified by the prefix before the table name. They either belong to the product schema (**Erp**) or system schema (**Ice**).

7. From the **tables** listing click and drag the **Erp.Customer** table onto the design canvas (grid) area.

8. In the **Filtering** field, enter **Ord**.

9. From the tables listing, select and drag the **Erp.orderHed** table onto the workspace (grid) area.
   Notice that the two tables automatically join with the **OrderHed.BTCustNum** joining to the **Customer.CustNum**. Notice further that the Dictionary button displays a 2 indicating 2 possible joins.

10. Click the **Dictionary button**.
    Notice there is a predefined link for both **BTcustnum** and **Custnum**. You want to define another join, this one with the **ShiptoCustnum**.

Notice the selected tables are not directly linked.

Set Table Relations

1. Close the Dictionary View.

2. Click on the AZ sort icon just above the **Table Criteria** tab, then click the **Table Relations** tab and go to the second row and change the **OrderHed.BTCustNum** field to **OrderHed.ShiptoCustNum**.

3. Click **Save**.

4. Remain in the **Business Activity Query Designer**.

The table join is now defined and columns can be selected for the returned Dataset.

Select the Columns and Test

After you establish the table relations, you next select the fields from the tables and then test the query.

1. Navigate to the **Display Fields > Column Select** sheet.
   The selected tables display in the Available Columns area to allow field selection.

2. In the **Available Columns** area, expand the **Customer** table.

3. Double-click on the **CustID** and the **Name** fields.
   The fields move into the Display Column(s) area in the order they were selected.

   Tip You can change the order of columns using the up and down arrows. If you need, you can adjust column names using the **Label** field or change the **Format** of columns you want to display.

4. Collapse the **Customer** table and expand the **OrderHed** table.

5. Press and hold **Ctrl** and highlight the following columns
   - **OrderNum**
   - **Orderdate**
   - **ShipViaCode**
   - **DocOrderAmt**

6. Click the **right arrow** to move the selected columns to the **Display Columns** list.

7. Click **Save**.
   You are now ready to test the query.

8. Navigate to the **Analyze** sheet.

9. Click the **Analyze** button.
   Verify the **Syntax is OK** message displays.

10. Click the **Test** button.
    The query is executed and results display in the **Query Results** grid.

11. Click **Save** and remain in the Business Activity Query Designer.

Workshop - Create the My Parts BAQ

This workshop shows you how to create a BAQ that retrieves part and part bin information. Tables used in this workshop are directly linked by the application.

The BAQ you design displays part number details, warehouse locations, bin locations, and quantities on-hand.

Add the BAQ

1. In the **Business Activity Query Designer**, click **New** to create a new query.
2. In the **Query ID** field, enter **XXX_PartStatus** (where XXX are your initials).
3. In the **Description** field, enter **XXX Part and Bin Information**.
4. Select the **Shared** check box.
   This check box indicates that this query is available to all users.

Select the Tables

1. Navigate to the **Query Builder > Phrase Build** sheet.
2. Above the list of tables, in the filtering field, enter **p**.
3. From the list, select the **Erp.Part** table and drag it on the canvas in the center pane.
4. Click the **Connected Only Tables** button found above the list of tables.
   Recall when this option is selected, the palette only displays the tables that have relations, described in system tables, with the table selected on the canvas.
5. Scroll down and double-click the **Erp.PartBin** table to place the table in the center pane.
   Notice the linking line automatically displays between the two tables.
6. Click the line between the tables. This causes the **Table Relations** to display.
7. Click the **Dictionary** button to launch **Table Relations from Dictionary** utility.
   Use this utility to view the relationships between tables as defined in the system schema dictionary.
8. Notice these tables are sharing the **Company** and **PartNum** fields.
9. Click **Close** to exit the utility.

Select the Columns

1. Navigate to the **Display Fields > Column Select** sheet.
2. In the **Available Columns** list, verify the **Sort Columns Alphabetically** control is inactive.
3. Expand the **Part** node and select the **PartNum** column.

4. To select multiple columns at once, hold **Ctrl** and select the following columns:
   - **PartDescription**
   - **TypeCode**
5. Click the right arrow button to move the columns to the **Display Column(s)** list.
6. Collapse the **Part** node.
7. In the **Available Columns** list, expand the **PartBin** node.
8. In the **Available Columns** list, click **WarehouseCode**.
9. Press and hold **Ctrl** to select the following columns:
   - **BinNum**
   - **OnHandQty**
   - **LotNum**
   - **DimCode**
10. Click the right arrow button to move the columns to the **Display Column(s)** list.

Enter the Sort Order

1. Navigate to the **Display Fields > Sort Order** sheet.
2. In the **Available Columns** list, expand the **Part** node.
3. Click the **PartNum** column.
4. Click the right arrow button to move the column to the **Sort By** list.
5. Click **Save**.

Defining a sort order ensures the data returned by the query displays in a specific way. In this example, the results display in part number sequence.

Analyze and Test the BAQ

The Analyze function checks the query code to make sure the syntax is correct.

The Test function executes the query and displays the results in the Analyze window. Review the results to verify the query returned the expected data and to trouble shoot any discrepancies in the data.

1. Navigate to the **Analyze** sheet.
2. Click the **Analyze** button.
   Similar to a compile, Analyze reviews the query to check for syntax errors and potential run-time errors. The results of this check display in the **Query Execution Messages** pane.
3. Click the **Test** button to execute the BAQ.
   Data returned by the query displays in the grid. To further analyze the BAQ results, right-click anywhere in the BAQ Results grid.

You can use the context menu to do the following:

- Group data together through specific column(s) that you select
- Summarize data
- Copy data from a grid and paste its contents to a third-party application for further analysis, for example to Microsoft® Excel®

4. Click **Clear** and remain in the BAQ Designer.

Selection Criteria

You can control query detail selection using the **Table Criteria** and **SubQuery Criteria** sheets.

Note Table criteria are associated with a table, so to see table's criteria you need to select a table on design surface first. SubQuery criteria are associated with a SubQuery and applied to query result.

Selection Field Details

You can limit query data by defining criteria. To create a combined criteria, you link **And** or **Or** statements. The columns include the details as explained below.

- **And Or** - Allows you to select logical operations with the conditions. Three options are available:
  - **And** - The **And** criteria tells the application to select the record only when the two criteria are true.
  - **Or** - The **Or** criteria tells the application to select the record whether criteria one or criteria two is true.
  - **<Null>** - Adds no clause to the criteria string. Specify this option for the first row of the criteria.
- **LeftP** or **RightP** - Use these columns to enter left or right parenthesis. Use parentheses to nest selection criteria.
- **Neg** - Use the Neg value to indicate a negative value.
- **Field Name** - The fields of the selected table display in this listing for filtering.
- **Compare Operator** - Several comparison are available, such as equals, not equal to, and so on.

Important The Epicor database requires all table cells have values, so it does not allow DB NULLS. The only exclusion is for Date type fields. If you want to find rows with an EMPTY string value, use a comparison with an empty constant.

To specify an empty string, for the **Filter Value**, select specified constant and leave the **Value** field **empty**.

If you want to find rows with unassigned fields, use the ISNULL operation. This operation is useful in external queries which execute against non-Epicor databases.

The Filter Value Column

The Filter Value Column provides the ability to conditionally limit the data coming into the BAQ. Once you select a field name, the columns allows you to specify criteria.

The expressions that populate in this column are similar to the directives used in Business Process Management. The expression has a linked field which, when you select it, provides a value statement. Available expressions:

- **specified table field value** - Select this filter value to call out a specific table and field combination. Select the word **specified** to open a select table pane which presents the table in the query. You can use any table.field combination as a filter value to limit the query results.
- **specified constant** - Use this filter to limit the query data to a specific constant value, such as a State. Select the word **specified** to open the Specify a value window. The value must be valid in the database.
- **specified expression** - Use this filter to limit the query results using a custom expression. Click the word **specified** to launch the expression editor, where you can compose an expression that evaluates against the comparison. The expression can contain literal values, BAQ Constants, and functions that can perform calculations and data type conversions.
- **specified parameter** - You can define parameters and use them as filter values. Select the word **specified** to display the Select Parameter window. Use this window to select or define parameters.
- **BAQ special constants** - Several constants are readily available to use a special characters. Such as CurrentDate, SessionName, FirstDayofMonth, and so on. These constants are recognized by the application and populate with the session object details. Select the word **special** to open the constant listing.

- **current date + specified interval** - Use this option to specify date intervals in days, weeks, months, and years. Select the word **specified** to open the Select a date window.
- **selected value(s) of field from specified subquery** - Use this option to compare a field against ANY or ALL fields retrieved from the selected subquery. Select the word **selected** to switch between ANY or ALL subquery field values. Select **field from specified** to select SubQuery field you want to use in the criterion.

Workshop - Apply Filter and Modify Display Parameters

This workshop explains how to use table criteria to limit BAQ results and change parameters of display columns.

The BAQ you create in this workshop displays all open sales order detail lines. The OrderDtl table used in this workshop has a boolean field for open orders. You will use the OrderDtl.OpenLine field to limit the dataset to open sales orders.

When users enter a new line in Sales Order Entry and select an existing part, the Order Line Description field (OrderDtl.LineDesc) automatically defaults to part description found in part master table (Part.Description). In this BAQ, you want to display records where part and order line descriptions are different.

Define the Part Description Compare Query

The Part Master table contains part specific details including the part number, description, weight, and volume. The Sales Order Entry program contains the order information with line details stored in the Order Detail table. Use both tables to construct a BAQ.

1. In the **Business Activity Query Designer**, click **New** to create a new query.
2. In the **Query ID** field, enter **XXX_PartCompare** (where XXX are your initials).
3. In the **Description** field, enter **Part Description Compare Query**.
4. Select the **Shared** check box.

Use Table Criteria

1. Navigate to the **Query Builder > Phrase Build** sheet.
2. In the **Filtering** field enter **ord** then drag the **Erp.OrderDtl** table onto the design canvas (grid).
3. Below the filtering field, select the **Connected Only Tables** icon.
4. In the **Filtering** field enter **par** then drag the **Erp.Part** table on to the design canvas (grid).
   The tables automatically link.
5. First, create a filter to only retrieve open sales order lines:
   a. On the design canvas area, select the **Erp.OrderDtl** table.
   b. At the bottom of the screen, verify the **Table Criteria** sheet displays.
   c. From the toolbar, click the **Add Row** button.
      The new line displays.
   d. From the toolbar above the **Criteria** sheet header, select the **A-Z** icon.
      This sorts the fields in alphabetical order for easy access.
   e. In the **Field** column, select **OpenLine**.
      Once you select the field name, set up the condition using the compare operator.
   f. In the **Operation** column, verify the equal sign **(=)** defaults.
      This states that the field information has to match the selected filter criterion.

g. From the **Filter Value** field, select **specified constant**.
The word **specified** displays in blue to indicate a variable or value is required.

h. Click the word **specified**.
The **Specify Value** window displays.

i. In the **Value** field, enter **0** to get OrderDtl record in a "closed" status or **1** to get Orderdtl record in a "open" status.

Important When you build criteria, you do not have to use quotes when specifying a filter value.

j. In the **Specify Value** window, click **OK**.
This limits the query selection to open orders.

Tip When you add a criteria on table or a SubQuery, the word **+criteria** displays on a BAQ item. Use this visual identifier to quickly allocate filters applied in your BAQ. This may be useful when troubleshooting a BAQ for potential errors.

6. Now create a filter to display records where order line descriptions and part descriptions differ.

a. From the toolbar, click **Add Row** to add another criteria.

b. In the **Field** column, select **LineDesc**.

c. In the **Operation** column, select **not equal (<>)**.

d. From the **Filter Value** field, select **specified table field value**.

e. Click the word **specified**.

f. From the **Table** drop-down, select **Part**.

g. From the field list, select **Part Description** and click **OK**.

7. **Save** the query.

8. Remain in the Business Activity Query Designer.

Define the Columns and Test

1. Navigate to the **Display Fields > Column Select** sheet.

2. From the toolbar in the **Available Columns**, click the **Sort columns alphabetically** (A to Z icon).
Tables and fields always sort schematically by default. The Sort icon places all the details in alphabetical order.

3. In the **Available Columns** area, expand the **OrderDtl** table.

4. From the field listing, add the following columns to the Display Column(s) area:

Field Name
LineDesc
OrderLine
OrderNum
PartNum
Request Date

5. In the **Available Columns** area, expand the **Part** table.
6. From the field listing, add the following columns to the Display Column(s) area:

Field Name
NetVolume
NetWeight
PartDescription

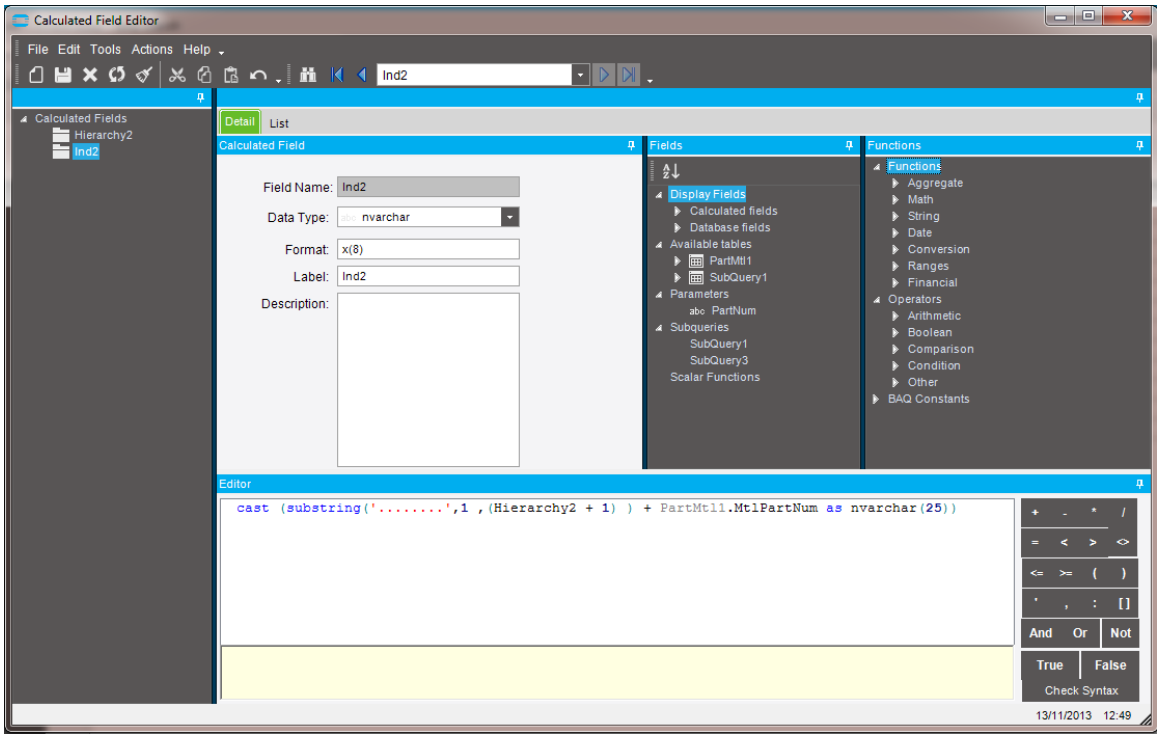
7. You can change the field names to save confusion on labels. In the **Label** field of the **OrderDtl_LineDesc** field, enter **Order Line Description**.
8. In the **Label** field of the **Part_PartDescription** field, enter **Part Number Description**.
9. Click **Save**.
   You are now ready to test the query.
10. Navigate to the **Analyze** sheet and click the **Test** button.
    The query is executed and results display in the **Query Results** grid.
    Verify Order Line Description and Part Number Description fields are different. The BAQ should return open orders. To test, right-click any Order number from the list and from the context menu, select **Open With > Sales Order Tracker**. Verify the order in focus is not closed. Once complete, exit Sales Order Tracker.
11. **Save** the query and **Clear** the current record.
12. Remain in the BAQ Designer.

Calculated Field Editor

Use the **Calculated Field Editor** to create user-defined columns based on a calculation of selected fields within the query. You can either create new fields or modify existing ones that came over from a copied query.

Example

The following is an example of the Calculated Field Editor.



The **Detail** sheet is divided into several sections:

- **Calculated Field** - This section contains the calculated field definition.
- **Fields** - In the tree view, you can select any of the current Display Fields and existing Calculated fields you want to add to the calculation. You can also pull in any fields contained within the Available tables included with the current query. If you have created Parameters and Scalar Functions to use with the BAQ, these options are available for selection as well.
- **Functions** - Use the tree view to find and select the Functions, Operators, and BAQ constants you will use in the current calculated field.

  A function calculates a value through specified parameters. An operator is a mathematical expression used to evaluate a function. BAQ operators and functions are consistent with SQL naming methodology.

  A BAQ constant is a query that pulls in specific data contained within the Constants table, like current plant, current date, and so on.

  Tip For a complete list of available functions, see the Calculated Field topics within the application help.

- **Editor** - The Editor pane displays the calculation you create. A warning displays if a table or field is selected that is not in the query.

  To the right side of the editor box, use the buttons to quickly enter operators. This section also contains additional symbols and the **Check Syntax** button.

The Check Syntax button verifies the correct syntax of the calculated field.

The functions and operators you drop in the editor box or enter through the right side buttons have trailing spaces. If the character field placed before the functions or operators on the screen does not include a space, a leading space is inserted to assist in using the correct syntax.

Workshop - Create a Calculated Field

A Calculated field can be a mathematical field or a field that is defined using other fields in the database to provide query specific data.

This workshop shows how to create a calculated field to display total labor and burden hours for each shop employee. The Employee Basic (EmpBasic) table from the Shop Employee Maintenance provides the employee details. Join this table to the Labor Details (LaborDtl) table. The LaborDtl table contains time records from Time and Expense Entry. Filter results against a specific date and create calculated fields to generate the total labor and burden hours.

Create the Labor Summary BAQ

1. In the **Business Activity Query Designer**, click **New** to create a new query.
2. In the **Query ID** field, enter **XXX_EmpHours** (where XXX are your initials).
3. In the **Description** field, enter **Employee Labor Summary**.
4. Select the **Shared** check box.

Use a Date Filter

1. Navigate to the **Query Builder > Phrase Build** sheet.
2. In the **Filtering** field enter **Em**, then drag the **Erp.EmpBasic** table onto the design canvas (grid).
3. Click the **Connected Only Tables** button.
4. From the table listing, drag the **Erp.LaborDtl** table on to the design canvas (grid).
   The tables automatically link. The data is to be filtered based on a cutoff payroll date.
5. On the design canvas area, select the **Erp.LaborDtl** table.
6. At the bottom of the screen, verify the **Table Criteria** sheet displays.
7. From the toolbar above the **Table Criteria** sheet header, click the **Add Row** button.
   The new line displays.
8. From the **Field** column, select **PayrollDate**.
9. In the **Operation** column, select **Greater Than or Equal to (>=)**.
10. From the **Filter Value** field, select **specified constant**.
11. Click the word **specified**.
12. In the **Value** field, enter **01/01/2008**.

13. In the **Specify a Value** window, click **OK.**
14. From the toolbar above the **Criteria** sheet header, click the **Add Row** button.
15. In the **AndOr** column, verify **And** displays.
16. From the **Field** column, select **PayrollDate**.
17. In the **Operation** column, select **Less Than or Equal to (<=)**.
18. From the **Filter Value** field, select **specified constant**.
19. Click the word **specified**.
20. In the **Value** field, enter **01/01/2016**.
21. In the **Specify a Value** window, click **OK**.
    This limits the query details to the selected pay period.

Select Columns and Define the LaborHrs Field

1. Navigate to the **Display Fields > ColumnSelect** sheet.
2. Verify the **Sort Columns Alphabetically** option is selected.
   If not, select this option.
3. From the **Available Columns**, expand the **EmpBasic** node and move **Name** into the Display Column(s) area.
4. In the **Display Columns** grid, for the **EmpBasic_Name** column, select the **Group By** check box.
5. Click the **Calculator** icon.
   The **Calculated field editor** window displays.
6. Click **New** and enter these field values:

Data Field	Data Value
Field Name	SumOfLaborHrs
Data type	Decimal
Label	LaborHrs

   Create a calculated field to total each employee's labor hours using the Aggregate function.

7. In the **Functions** area, expand **Aggregate**.
   This expands a listing of aggregations available. Aggregate functions add all values found by the query in a specified column.
8. From the Aggregate listing, double-click **Sum(x)**.
   Selecting the function displays the function syntax in the Editor field. In this query, the field calculates the total labor hour values linked to each shop employee record.

9. In the **Fields** section, expand the **Available tables > LaborDtl** node and double-click **LaborHrs**.
   Verify the Editor displays the following calculation:
   ```
   sum( LaborDtl.LaborHrs )
   ```

10. Click the **Check Syntax** button.

11. To the **Syntax is OK** message, click **OK**.

12. Repeat steps 6 through 11 to create the calculated field **SumOfBurdenHrs**. For Label, enter BurderHrs. Use the Sum(x) function again and select LaborDtl.BurdenHrs.
    Verify the Editor displays the following calculation:
    ```
    sum( LaborDtl.BurdenHrs )
    ```

13. Click **Save** and exit the Calculated Field Editor.

14. Remain in the Business Activity Query Designer.

The calculated field displays in the Display Column(s) section. The label can be modified if desired.

Analyze and Test the Query

1. Navigate to the **Analyze** sheet.

2. Click the **Test** button to execute the query and return the records.
   The query returns the employee listing and displays the total labor and burden hours for each employee.

3. Click **Save**.

4. Click **Clear** and remain in the BAQ Designer.

Updatable BAQs

You can design Business Activity Queries (BAQs) which update the database. Through this feature set, you can create updatable smart client, mobile, or SharePoint Publisher dashboards.

Updatable BAQ Overview

These updatable queries have business object methods connected to them to allow you to create, edit, and delete records to update the database. The BAQ begins as a regular query with tables, filter criteria, display columns, and so on. You define which fields are updatable and then ensure the business object methods are correctly linked to the updatable fields on a BAQ.

To complete this functionality, you can use the entered data to call a Business Process Management (BPM) directive. As users enter data through an updatable BAQ, you create methods that validate whether the data being entered is correct, send email alerts, or cause other processes to run.

You can place these BAQs on a smart client dashboard and/or use then on a mobile device, such as an iPhone© or a Blackberry©. Users enter data through either the dashboard or the mobile device and the new data updates records within the main database. You can create Business Process Management (BPM) directives to monitor the data entered through an updatable BAQ. Based on the conditions defined in the BPM directive, various actions run automatically. For example, you can use this functionality to verify the data is being entered correctly.

Security Requirements

Updating BAQs is considered an advanced operation and requires the user id to have the security permissions.

Important To create updatable BAQs, you must have **BAQ Advanced User** rights granted in User Account Maintenance. In on-premise installations, these rights are granted by your system administrator. Epicor Cloud ERP customers are asked to attend an Updatable BAQ training prior to obtaining these rights. For more information, send your request to **EpicorCloudERPOperations@epicor.com**.

Users having **BPM Advanced User** rights are allowed to use the entire BPM toolset, when designing uBAQ method directive.

Furthermore, in order to use BPM Update Processing via UpdateExt and Advanced BPM Processing (design, import/export and copy), the **BusinessProcessManagement** module must be **licensed** in your Epicor ERP installation. The BPM license is not required for database updates through Epicor Service Connect workflows.

Important **Advanced BPM User** rights are not available to **Epicor Cloud ERP - Multi Tenant Users**, as freeform C# code may introduce a security risk for access of data outside of the tenancy. Use of C# in BPMs is regulated, and requires advanced approval from Cloud Operations to ensure conformity with our security and operational standards. Most common business requirements for reading and writing data within the tenancy restrictions can be accomplished by using the built in interface. To request a feature requiring **BPM Advanced User** rights, contact Epicor Professional Services or an authorized Epicor partner. You can also submit a request to **CloudCodeReview@epicor.com** for deployment.

In order to complete the following workshops, having **BAQ Advanced User** rights is sufficient. For more information, contact your system administrator.

BAQ Update Processing

Updatable BAQs interact with business object methods, and in turn these methods interact with a database. A business object (also called a BO) houses the methods used to enter, view, and calculate data for a specific

function within the application. Each action a business object can take is called a method; by default each updatable BAQ contains at least the following methods:

- **GetList** - This method pulls in the existing records from the database table from which the business object interacts. These records then display within the query results.
- **GetNew** - This method generates a new record and adds it to the database table. If your updatable BAQ allows new records, when users create new methods they activate this method.
- **UpdateExt** - This update method governs the whole data transaction process between the updatable BAQ and the related business object.

You can monitor each of these methods through Business Process Management (BPM) directives. These directives can evaluate the data being passed into or out of the database, interrupting the processing when certain conditions are met. Various actions can then automatically run in response to the data.

Workshop - Create a Sales Order Updatable BAQ

A Business Activity Query (BAQ) can be updated as a query, or added to a dashboard to create an updatable dashboard. They are frequently used on dashboards.

Example The order entry department is looking to update the sales orders as quickly as possible. You are asked to create an updatable BAQ so users can add new sales orders within the grid.

Order information is stored by the sales order in several different tables. Create the query using the OrderHed and OrderDtl tables to bring in the sales order and order line details. Each table is explained below:

- **OrderHed** - The OrderHed table holds the sales order header information such as Bill to and Ship to customer numbers, purchase orders, shipping terms and so on. Create the first query to allow entry of a sales order with no lines.
- **OrderDtl** - The OrderDtl table holds all the line item details, but not releases. Create a second query to allow line details to update an existing sales order.

Important The following workshops require a user is provided with **BAQ Advanced Developer** permissions. Verify these permissions have been granted using the **User Account Maintenance** program.

Define Query Details

Identify the query as updatable and select DB fields you want to update.

1. In the **Business Activity Query Designer**, click **New** to create a new query.
2. In the **Query ID** field, enter **XXX_Orders** (where XXX are your initials) and press **Tab**.
3. In the **Description** field, enter **Sales Order Updatable BAQ**.
4. Select the **Shared** check box.
5. Select the **Updatable** check box.
   This indicates that users can update the BAQ and activates the sheets under the Update sheet.
6. Navigate to the **Query Builder > Phrase Build** sheet.
7. In the **Filtering** field, enter **Ord**.
8. Click and drag the **OrderHed** table to the grid area.

9. Navigate to the **Display Fields > Column Select** sheet.
10. Expand the **OrderHed** table to display the fields.
11. Move the following fields into the **Display Column(s)** section and change the field labels as noted:

Tip For easier field search, sort columns alphabetically.

Field Value	Change Label Value
OrderHed_OpenOrder	Open
OrderHed_Company	Company
OrderHed_OrderNum	Order Number
OrderHed_CustNum	Customer Number
OrderHed_PONum	PO Number
OrderHed_OrderHeld	Hold Order
OrderHed_RequestDate	Request Date
OrderHed_FOB	FOB
OrderHed_ShipViaCode	Ship Via
OrderHed_TermsCode	Customer Terms
OrderHed_BTCustNum	Bill To Customer Number
OrderHed_ShipToCustNum	Ship To Customer Number

Create Basic Processing Rules

This workshop defines basic processing rules for the updatable query.

1. Navigate to the **Update > General Properties** sheet.
2. Verify the **Allow New Record** check box is selected.

   Selecting this field allows you to add new customer records through this BAQ. If it is not selected, only updates to existing records are allowed.

3. Select the **Allow Multiple Row Updatable** check box.

   Selecting this check box allows updating of multiple customer records on the same BAQ.

4. Click the check box in the **Updatable** column header to indicate you want to update all fields through this BAQ.
5. Clear the **Updatable** check box for the **OrderHed_OrderNum** field.
6. Navigate to the **Update > Update Processing** sheet.
7. Verify the **BPM Update** option is selected by default.

   For regular uBAQs, use the default BPM Update method to perform database updates using the special Business Object method UpdateExt that simulates the standard way of how application manipulates data within business objects.

8. Click the **Business Object** button.
   The Select Business Object window displays.
   The suggested business objects window displays objects that relate to the data in the BAQ, in this case the Sales Order related business object displays.

9. Select **Erp.Sales Order** and click **OK**.

10. In the **Tables to update** pane, verify **OrderHed** is selected.

11. Click **Save**.

Use an Expression to Map Fields

In this example, you want Customer Number, Bill to Customer Number, and Ship To Customer number be the same. Use an expression to populate each field with a single entry.

1. On the **Update > Update Processing** sheet, click the **Expression Editor** button.
   The **Business Object Update C# Expression** window displays.

2. From the **Dataset fields**, select **OrderHed.CustNum**.
   The Editor pane displays ttResult.OrderHed_CustNum.

3. In the **Editor** pane, highlight the **ttResult.OrderHed_CustNum** and copy it by pressing **Ctrl+C**.

4. From the **Dataset fields**, select **OrderHed.BTCustNum**.

5. In the Editor pane, highlight **ttResults.OrderHed_BTCustNum** and then replace it with **ttResult.OrderHed_CustNum**, using **Ctrl+V**.
   The new expression maps the BTCustNum to the OrderHed_CustNum.

6. Repeat step 5 for **OrderHed.ShipToCustNum**.

7. Click **Save** and exit the Business Object Update Expression window.

8. Navigate to the **Update > General Properties** sheet.

9. Clear the **Updatable** field for the **OrderHed_BTCustNum** and the **OrderHed_ShipToCustNum** fields.

10. Click **Save**.

This expression maps the two numbers so that one entry on the CustNum field populates both. This expressions assumes the two numbers are always the same and may not apply in your environment.

Test the Sales Order BAQ

Use the Get List method to test the update fields of the Orders BAQ. Use the Sales Order Tracker to view the updated information in the database.

1. Navigate to the **Analyze** sheet.

2. Click the **Get List** button.

3. To the warning message, respond **Yes**.

4. Click the **Order Number** column header to sort the results by this column. Double-click the row containing order number **5198**.
   The Fields window displays.

5. In the **OrderHed_PONum** field, delete the existing entry and enter **any number**.

6. Click **OK**.
   The line you modified becomes highlighted.

7. Click the **Update** button.

8. To the warning message, respond **Yes**.

9. Verify in the **Query Execution Messages** pane, the following message displays:
   ```
   Query returned 1 row(s). Updatable query action fulfilled and reported no e
   rror.
   ```

10. Remain in the Business Activity Query Designer window.

Create a New Sales Order

You defined the Sales Order updatable BAQ to update and to add new records. This workshop provides the steps to enter a new order.

It is important to know the minimum requirements of an order. Each required field is validated before the record is added. The required fields of a sales order include:

- Customer Number
- Bill to Number
- Order Date
- Terms Code
- Ship To Number

1. Verify the **Analyze** sheet is opened.

2. Click **Get New** button.

3. To the warning message, respond **Yes**.

4. Double-click the **new row** at the bottom of the **Query Results** window.

5. Enter the following details, press **Tab** to move between fields.

Data Field	Data Value
OrderHed_OpenOrder	Select
OrderHed_Company	EPIC06
OrderHed_CustNum	24
OrderHed_PONum	Enter any number

Data Field	Data Value
OrderHed_OrderHeld	Clear
OrderHed_RequestDate	Today's Date
OrderHed_FOB	Factory
OrderHed_ShipViaCode	UPS
OrderHed_TermsCode	2/10

6. Click **OK** and click **Update**.
7. To the warning message, respond **OK.**
8. Click **Save** and refresh the information by clicking the **Get List** button.
9. To the warning message, respond **Yes**.
   When the sales order is validated against the database, the refreshed list provides the new sales order number at the bottom of the list.
10. Click the **Order Number** column heading to sort orders in descending order. Locate and write down the newly created sales order number _____.
11. Minimize the Business Activity Query Designer.

Verify the Updated Orders

Use the Sales Order Tracker to verify the changes have been updated to the database.

Navigate to the **Order Tracker**.

Menu Path: Sales Management > Order Management > General Operations > Order Tracker

1. Click in the **Sales Order** field, enter **5198**, and press **Tab**.
   On the **Header** section, notice the **PO** number change made.

2. Click in the **Sales Order** field, enter the number for the sales order you previously created, and press **Tab**.
   Verify the new order and ensure accuracy in the data. Notice this sales order just contains header information.

3. Minimize the **Sales Order Tracker**.

4. Maximize the **Business Activity Query Designer**.

Workshop - Create the Sales Order Detail Updatable BAQ

An updatable BAQ can update more than one table as long as the two tables are within the business object. Order Header and Order Lines are both within the Sales Order business object. In addition, you can update other business objects by adding a BPM directive to call a Service Connect workflow to perform additional work.

This workshop provides the sales order line details for the sales order added during the workshop **Workshop - Create a Sales Order Updatable BAQ** using a new updatable query.

Create the Sales Order Detail Query

1. In the **Business Activity Query Designer**, click **New** to create a new query.
2. In the **Query ID** field, enter **XXX_OrderLines** (where XXX are your initials) and press **Tab**.
3. In the **Description** field, enter **Sales Order Lines Updatable BAQ**.
4. Select the **Shared** and **Updatable** check boxes.
5. Navigate to the **Query Builder > Phrase Build** sheet.
6. In the **Filtering** field, enter **Ord**.
7. Hold **Ctrl**, select **Erp.OrderHed** and **Erp.OrderDtl** tables and move them on the designer canvas.
8. Navigate to the **Display Fields > Column Select** sheet.
9. Expand the **OrderHed** table to display the fields.
10. Move the following fields into the **Display Column(s)** section and change the field labels as follows:

Field Value	New Label Value
Company	Company ID
OrderNum	Order Number

11. Expand the **OrderDtl** table to display the fields.
12. Move the following fields into the **Display Column(s)** section and change the field labels as follows:

Field Value	New Label Value
Company	Company ID
OrderNum	Order Number
OrderLine	Order Line
PartNum	Part Number
LineDesc	Line Description
IUM	Unit of Measure
DocUnitPrice	Unit Price
SellingQuantity	Quantity

Define Basic Processing Details

1. Navigate to the **Update > General Properties** sheet.
2. Verify the **Allow New Record** check box is selected.
3. Select the **Allow Multiple Row Update** check box.
4. Allow data updates for all fields by clicking the **Updatable** column header check box.
5. Navigate to the **Update > Update Processing** sheet.
6. Verify the **BPM Update** option is selected.
7. Click the **Business Object** button.
   The Select Business Object window displays.
8. In the **Suggested business objects** section, select the **Erp.SalesOrder** object and click **OK**.
9. In the **Tables to update** pane, ensure both **OrderHed** and **OrderDtl** are selected. Both tables are part of the same business object so they can be updated both at once.
10. Click **Save.**

Create an Expression to Link Details

OrderDtl is a child table of the OrderHed table. Map the OrderHed.Company and OrderHed.OrderNum fields to the OrderDtl to pull data by the order number.

1. Click the **Expression Editor** button.
   The **Business Object Update C# Expression** window displays.
2. From the **Dataset fields**, select **OrderHed.Company**.
3. In the **Editor** window, delete the **ttResult.OrderHed_Company** entry.
4. In the **Fields** pane, double-click **OrderDtl_Company**.
   This step maps the OrderHed.Company field to the OrderDtl.Company field.
5. From the **Dataset fields**, select **OrderHed.OrderNum**.
6. In the **Editor** window, delete the **ttResult.OrderHed_OrderNum** entry.
7. In the **Fields** pane, double-click **OrderDtl_OrderNum**.
   This step maps the OrderHed.OrderNum field to the OrderDtl.OrderNum field.
8. Click **Save**.
9. Exit the **Business Object Update Expression** window.
10. Navigate to the **Update > General Properties** sheet.

11. Clear the **Updatable** check box for **OrderHed_Company** and **OrderHed_OrderNum**.
    After mapping these two fields, the fields populate with the OrderDtl data and do not need to be updated.
12. Click **Save**.
13. Navigate to the **Analyze** sheet.
14. Click **Get List** to refresh.
15. To the warning message, click **Yes**.

Enter a New Order Line

This workshop creates a sales order line for the sales order created in the Workshop - Create a Sales Order Updatable BAQ.

1. Click **Get New** button and respond **Yes** to the warning message.
2. At the bottom of the **Query Results** window, double-click the **new row**.
3. Enter the following details, press **Tab** to move between fields.

Data Field	Data Value
OrderDtl.Company	EPIC06
OrderDtl.OrderNum	Your Order Number
OrderDtl.Orderline	1
OrderDtl.PartNum	DCD-800-CDL
OrderDtl.LineDesc	Cargo Door Latch
OrderDtl.IUM	EA
OrderDtl.DocUnitPrice	33.00
OrderDtl.SellingQuantity	3

4. Click **OK**, and then click **Update**.
5. If necessary, respond **Yes** to the save query message.
   Refresh the list using Get List to retrieve the new record and see the order number.
6. Click **Save** and then click **Get List**.
7. Remain in the BAQ Designer.

When the list refreshes, the new order number displays with a single order line. By default, the sales order also has one release.

Test the Order Line BAQ

Use the Sales Order Tracker to verify the new order has order line details.

1. On the **Analyze** sheet, in the **Query Results** grid, right-click the **Order Number** you created.

2. From the Context menu, select **Open With ... > Sales Order Tracker**.
   The tracker opens the record for the sales order you indicated.

3. Navigate to the **Lines > Detail** sheet.
   Verify the sales order displays the order line values.

4. Exit Sales Order Tracker.

Note Multiple tables can be used in an updatable BAQ as long as the tables all belong to the same business object.

Using SubQueries

A SubQuery is a query that is nested inside a SELECT statement, or inside another SubQuery. You can use a SubQuery anywhere an expression is allowed. Each Business Activity Query can contain one main SubQuery, where Type=TopLevel and several SubQueries of different type to indicate how they correlate with each other.

SubQueries give different advantages for query design, including:

- Support of widely used query criteria as IN ({inner subquery}), EXISTS ({inner subquery})
- Using common table expressions (CTE), including CTE with recursion and UNION ALL for querying hierarchical data
- Using CTE and inner SubQuery in the FROM clause and joins
- Using set operations, like UNION, UNION ALL, EXCEPT, INTERCEPT

By default, a simple query contains only one SubQuery with the following attributes:

Field	Value
Name	SubQuery1
Type	TopLevel
Results Row Set	All

When designing a new SubQuery, the following types are available for selection:

Type	Description
TopLevel	Default value for simple query; each BAQ can contain one main TopLevel SubQuery.
Union	The UNION command is used to select related information from two tables, much like the JOIN command. However when using the UNION command, all selected columns need to be of the same data type. With Union, only distinct values are selected.
UnionAll	The UNION ALL command is equal to the UNION command, except that UNION ALL selects all values. The difference between Union and Union all is that Union all does not eliminate duplicate rows, instead it just pulls all rows from all tables fitting your query specifics and combines them into a table.
Intersect	Use the INTERSECT command to return the results of two or more select queries. However, it only returns the rows selected by all queries. If a record exists in one query and not in the other, it is omitted from the results.

Type	Description
Except	Returns any distinct values from the query to the left of the EXCEPT operand that are not also returned from the right query.
CTE	A common table expression (CTE) can be thought of as a temporary result set that is defined within the execution scope of a single SELECT, INSERT, UPDATE, DELETE, or CREATE VIEW statement. A CTE is similar to a derived table in that it is not stored as an object and lasts only for the duration of the query. Using a CTE offers the advantages of improved readability and ease in maintenance of complex queries. You can divide the query into separate, simple, logical building blocks. These simple blocks can then be used to build more complex, interim CTEs until the final result set is generated.
InnerSubQuery	InnerSubQuery is usually added in the WHERE Clause of the SQL statement as a nested query inside another query. Most of the time, a subquery is used when you know how to search for a value using a SELECT statement, but do not know the exact value.

Important

In BAQ Designer, SubQueries are concatenated in the sequential order, so one or more SubQueries of Union, UnionAll, Except, Intercept types can go after TopLevel, or CTE SubQueries.

When you combine one or more SubQueries using the above set operators, their fields, specified on the Display Fields > Columns Select sheet must conform to the following rules:

- The number and the order of the columns must be the same in all SubQueries.
- The data types must be compatible, through implicit conversion.
- Field aliases of the result record set are taken from the first SubQuery.

You also have the ability to control the SQL result set by modifying the SELECT keyword. The following options are available for selection in the **Result Set Row** field:

Keyword	Description
All	Default value; the SELECT ALL command returns all data without restriction.
Distinct	SELECT DISTINCT specifies that only unique rows can display in the result set.
Top	SELECT TOP clause specifies the number of rows or percent of rows to return. You can enter either a number or a percent of rows that the result set displays. WITH TIES specifies that the query result set includes any additional rows that match the values in the ORDER BY column or columns in the last row returned. This may cause more rows to be returned. TOP...WITH TIES can be specified only in SELECT statements, and only if an ORDER BY clause is specified. **Important** TOP as well as DISTINCT TOP cannot be combined with OFFSET and FETCH in the same query expression.
DistinctTop	SELECT DISTINCT TOP corresponds to using DISTINCT and TOP clauses simultaneously. The query results set contains top unique rows.

The **Offset** and **Fetch** fields provide you with an option to fetch only a window or page of results from the result set.

Argument	Description
Offset	Specifies the number of rows to skip, before starting to return rows from the query expression. **Example** This BAQ skips first five rows from the sorted result set of all employees and returns the remaining rows. `select [EmpBasic].[EmpID] as [EmpBasic_EmpID], [EmpBasic].[FirstName] as [EmpBasic_FirstName], [EmpBasic].[LastName] as [EmpBasic_LastName] from Erp.EmpBasic as EmpBasic order by EmpBasic.EmpID OFFSET 5 ROWS`
Fetch	Specifies the number of rows to return, after processing the OFFSET clause. **Example** This BAQ skips first five rows from the sorted result set and returns next ten rows from the employees table. `select [EmpBasic].[EmpID] as [EmpBasic_EmpID], [EmpBasic].[FirstName] as [EmpBasic_FirstName], [EmpBasic].[LastName] as [EmpBasic_LastName] from Erp.EmpBasic as EmpBasic order by EmpBasic.EmpID OFFSET 5 ROWS FETCH NEXT 10 ROWS ONLY`

The OFFSET/FETCH rowcount expression can be any arithmetic, constant, or parameter expression that returns an integer value.

Please be aware of the following limitations:

- Using OFFSET/FETCH requires MS SQL Server 2012 or later.
- ORDER BY clause is mandatory to use OFFSET and FETCH clause.
- OFFSET clause is mandatory with FETCH. You can never use, ORDER BY … FETCH.
- TOP/DISTINCT TOP cannot be combined with OFFSET and FETCH in the same query expression.
- OFFSET and FETCH cannot be combined with TOP/DISTINCT TOP keywords in the same query expression.

Workshop - Use Inner SubQueries

In this workshop, create the BAQ that counts open, closed, and total amount of orders per customer. Create two SubQueries that will count open and closed orders and group them together by customer. The information obtained by the Inner SubQueries is presented by the TopLevel SubQuery you create at the end of this workshop.

Create Open Orders SubQuery

The first SubQuery you create counts open orders by customer.

1. In the **Business Activity Query Designer**, click **New** to create a new query.
2. For the **Query ID**, enter **XXX_OrderCount**, where XXX are your initials.
3. In the **Description** field, enter **Order Count per Customer**.
4. Select the **Shared** check box.
5. Navigate to the **Query Builder > Phrase Build** sheet.

6. In the filtering field, enter **ord**.
7. From the list, select the **Erp.OrderHed** table and drag it on the canvas in the center pane.
8. Verify the **Table Criteria** sheet at the bottom is in focus.
9. From the toolbar, click the **Add Row** to add new line.
10. In the **Field** column, select **OpenOrder**.
11. In the **Operation** column, verify the equal sign (**=**) defaults.
12. From the **Filter Value** field, select **specified constant**.
13. Click the word **specified**.
    The Specify Value window displays.
14. In the **Value** field, enter **True** and click **OK**.
    This limits the query selection to open orders.

Select Columns

1. Navigate to the **Display Fields > Column Select** sheet.
2. Expand the **OrderHed** table.
3. Double-click **CustNum** to add the **OrderHed_CustNum** field to the list of Display Column(s).
4. Select the **Group By** check box to indicate you want to group open orders by customer.
5. Click the **Calculator** icon.
   The **Calculated Field SQL Editor** window displays.
6. For the **Field Name**, enter **CountOpen** and press **Tab**.
7. In the **Data Type** field, select **int** (interger).
8. In the **Label** field, verify **CountOpen** displays.
9. In the **Functions** area, expand **Aggregate**.
10. From the Aggregate listing, double-click **Count(x)**.
    Selecting the function displays the function syntax in the Editor field. In this query, the field counts the amount of open orders.
11. To indicate you want to count all open order records, inside the brackets, enter ***** (asterisk).
12. Verify the **Editor** pane displays the following:
    ```
    count( * )
    ```
13. Click **Save** and exit the Calculated Field SQL Editor.

14. Navigate to the **Query Builder > SubQuery Options** sheet.
15. For SubQuery 1 you just created, in the **Type** field, select **InnerSubQuery**.
    This indicates this SubQuery will become a nested query inside the TopLevel BAQ you will create later.

Create Closed Orders SubQuery

The Second SubQuery you create counts closed orders by customer.

1. In the **Active SubQuery** toolbar, click **Add Subquery**.
2. On the **Query Builder > SubQuery Options** sheet, accept the following defaults:

Name	Type
SubQuery2	InnerSubQuery

3. Navigate to the **Query Builder > Phrase Build** sheet.
4. Above the list of tables, in the filtering field, enter **ord**.
5. From the list, select the **Erp.OrderHed** table and drag it on the canvas in the center pane.
6. In the **Specify Table Alias** window, click **OK**.

   Note The BAQ Designer must create the table alias for the OrderHed table since it is already used in the previous SubQuery.

7. Verify the **Table Criteria** sheet at the bottom is in focus.
8. From the toolbar, click the **Add Row** to add new line.
9. In the **Field** column, select **OpenOrder**.
10. In the **Operation** column, verify the equal sign (**=**) defaults.
11. From the **Filter Value** field, select **specified constant**.
12. Click the word **specified**.
    The Specify Value window displays.
13. In the **Value** field, enter **False** and click **OK**.
    This limits the query selection to closed orders.

Select Columns

1. Navigate to the **Display Fields > Column Select** sheet.
2. Expand the **OrderHed1** table.
3. Add **CustNum** to add the to the list of Display Column(s).

4. Select the **Group By** check box to indicate you want to group closed orders by customer.
5. Click the **Calculator** icon.
6. For the **Field Name**, enter **CountClosed** and press **Tab**.
7. In the **Data Type** field, select **int** (interger).
8. In the **Label** field, verify **CountClosed** displays.
9. In the Editor pane, manually add the calculation that counts all closed orders.
   ```
   count( * )
   ```
10. Click **Save** and exit the Calculated Field Editor.

Create TopLevel SubQuery

Create the main SubQuery that displays the BAQ results using both inner SubQueries.

1. In the **Active SubQuery** toolbar, click **Add Subquery**.
2. Navigate to the **Query Builder > SubQuery Options** sheet.
3. For **SubQuery3**, in the **Type** field, select **TopLevel**.
4. Navigate to the **Query Builder > Phrase Build** sheet.
5. Search for and add the **Erp.Customer** table on the canvas in the center pane.
6. Above the table listing, click the **SubQueries** button to displays the list of available SubQueries.
7. Drag both **SubQuery1** and **SubQuery2** to the center pane.
8. From the toolbar above the canvas, select the **Add Connection** icon (chain links).
   The cursor turns to a cross-hair.
9. Click on the **Erp.Customer** table and drag a line to the **SubQuery1** table, then release.
10. Select the **square** icon in the middle of the connection line.
11. For **Join Type**, select **Left Join**.
    This type of join ensures customers having open order but no closed orders and vice versa also display on the list of results.
12. Verify the **Table Relations** sheet at the bottom is selected.
13. Click the **Add Row** button.
14. From the **Customer field or any expression** field, select **CustNum**.
15. From the **SubQuery1 field or any expression** field, select **OrderHed_CustNum**.
16. Repeat steps **8 - 15** to create connection between **Erp.Customer** and **SubQuery2** tables.

17. Click **Save**.

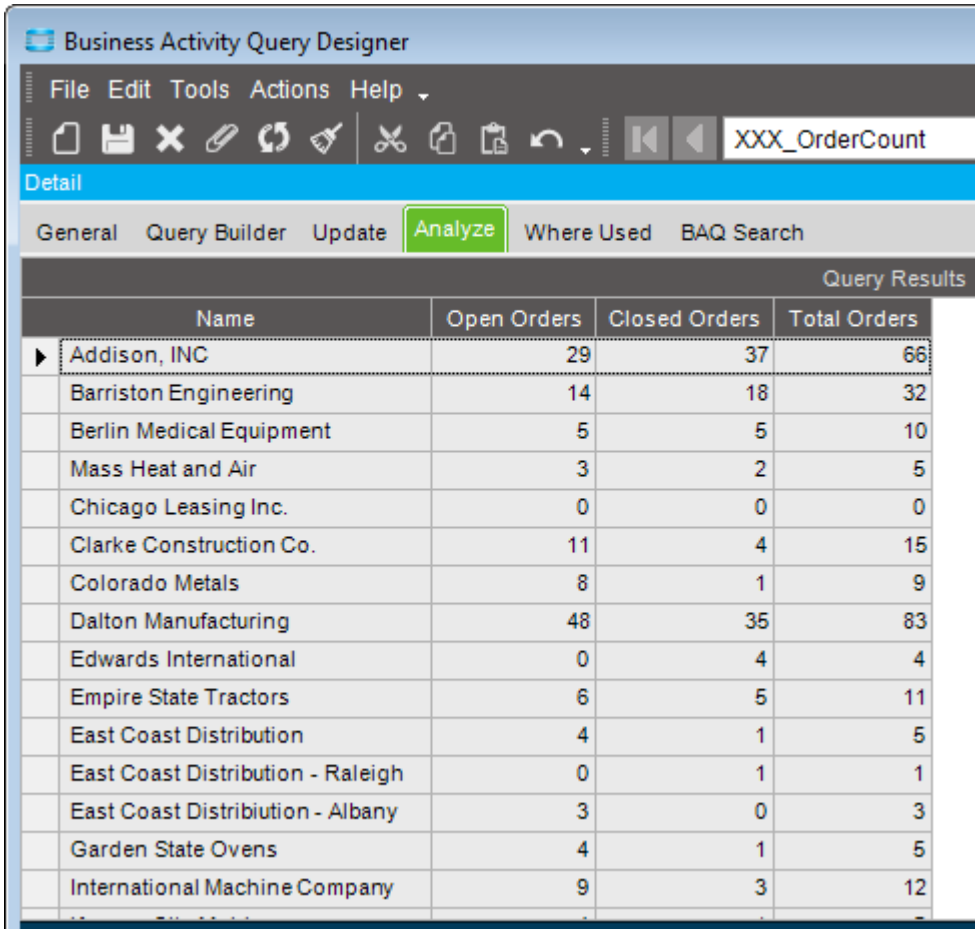
Select BAQ Display Columns

1. Navigate to the **Display Fields > Column Select** sheet.
2. Expand the **Customer** table and move the **Name** field to the list of **Display Column(s)**.
3. Click the **Calculator** icon.
4. For the **Field Name**, enter **OpenOrders** and press **Tab**.
5. In the **Data Type** field, select **int** (interger).
6. In the **Label** field, enter **Open Orders**.
7. In the **Editor** pane, enter **IsNull(Calculated_CountOpen,0)**.
   This function either returns the actual value of open orders, or if a null value is returned, the query will display a zero.
8. Click **Save** and click **New** to create another calculated field.
9. For the **Field Name**, enter **ClosedOrders** and press **Tab**.
10. In the **Data Type** field, select **int** (interger).
11. In the **Label** field, enter **Closed Orders**.
12. In the **Editor** pane, enter **IsNull(Calculated_CountClosed,0)**.
13. Click **Save** and click **New**.
14. For the **Field Name**, enter **Total** and press **Tab**.
15. In the **Data Type** field, select **int** (interger).
16. In the **Label** field, enter **Total Orders**.
17. In the **Editor** pane, enter **OpenOrders** + **ClosedOrders**.
18. Click **Save** and exit the Calculated Field Editor.

Test the BAQ

Test the BAQ and verify it displays open, closed and total amount of orders per customer.

1. Navigate to the **Analyze** sheet.
2. Click **Test**.
3. Verify the BAQ result looks similar to the following:



Name	Open Orders	Closed Orders	Total Orders
Addison, INC	29	37	66
Barriston Engineering	14	18	32
Berlin Medical Equipment	5	5	10
Mass Heat and Air	3	2	5
Chicago Leasing Inc.	0	0	0
Clarke Construction Co.	11	4	15
Colorado Metals	8	1	9
Dalton Manufacturing	48	35	83
Edwards International	0	4	4
Empire State Tractors	6	5	11
East Coast Distribution	4	1	5
East Coast Distribution - Raleigh	0	1	1
East Coast Distribiution - Albany	3	0	3
Garden State Ovens	4	1	5
International Machine Company	9	3	12

4. Remain in the Business Activity Query Designer.

Workshop - Combine Result Sets

In this workshop, build a BAQ that displays total value breakdown of sales for a given day.

This BAQ will provide information on total values of:

- Quotes entered in the sales pipeline.
- Orders booked in the system.
- Invoices sent to customers.

You accomplish this task by creating a BAQ comprised of three SubQueries and combine their results into one dataset.

Create TopLevel BAQ

1. In the **Business Activity Query Designer**, click **New** to create a new query.
2. In the **Query ID** field, enter **XXX_SalesValue** (where XXX are your initials) and press **Tab**.
3. In the **Description** field, enter **Total Value of Quotes, Orders, Invoices by Date**.
4. Select the **Shared** check box.

5. Navigate to the **Query Builder > Phrase Build** sheet.
6. Place the following tables on the designer canvas:
   - **Erp.QuoteHed**
   - **Erp.Customer**
7. Navigate to the **Display Fields > Column Select** sheet.
8. Move the following columns to the **Display Column(s)** list.
   - **QuoteHed_Company**
   - **Customer_Name**
   - **QuoteHed_QuoteNum**
   - **QuoteHed_EntryDate**
   - **QuoteHed_QuoteAmt**
9. Change the existing Field Labels as follows:

Field	Label
QuoteHed_QuoteNum	DocNumber
QuoteHed_EntryDate	Date
QuoteHed_QuoteAmt	Total Value

10. Click the **Calculator** icon to access the **Calculated Field SQL Editor** window.
11. Click **New** and enter these field values:

Data Field	Data Value
FieldName	SubQueryName
Data Type	nvarchar
Label	Document Type
Editor pane	'Quotes'

12. Click **Save** and exit the Calculated Field Editor.
13. Move the **Calculated_SubQueryName** column up and make it the first column in the list.
14. Navigate to the **Analyze** sheet and click **Test**.

The grid displays the list of quotes and their total amounts. Notice the Document Type column indicates all records belong to the Quotes category. In the following tasks you will add two more SubQueries that retrieve similar data for Orders and Invoices and have all results display in one grid.

Create Order View SubQuery

1. Navigate to the **Query Builder > SubQuery Options** sheet.
   You will first change the name of the TopLevel SubQuery in focus.

2. In the **Name** field, enter **QuoteHed** and click **Save**.
   Now you will create a new SubQuery.
3. In the **Active SubQuery** toolbar, click **Add Subquery**.
4. In the **Name** field, enter **OrderHed**.
5. In the **Type** field, select **Union**.
6. Navigate to the **Query Builder > Phrase Build** sheet.
7. Place the following tables on the designer canvas:
   - **Erp.OrderHed**
   - **Erp.Customer**
8. Since the Customer table is already used in the BAQ definition, accept the proposed table alias.

   Example By default, the table name defaults to **Customer1**.
9. Navigate to the **Display Fields > Column Select** sheet.
10. Move the following columns to the **Display Column(s)** list.
    - **OrderHed_Company**
    - **Customer1_Name**
    - **OrderHed_OrderNum**
    - **OrderHed_OrderDate**
    - **OrderHed_OrderAmt**
11. Click the **Calculator** icon to access the **Calculated Field Editor** window.
12. Click **New** and enter these field values:

Data Field	Data Value
FieldName	SubQueryName1
Data Type	nvarchar
Label	Document Type
Editor pane	'Orders'

13. Click **Save** and exit the Calculated Field Editor.
14. Move the **Calculated_SubQueryName1** column up and make it the first column in the list.

    Note Recall the number and the order of the columns is the same as specified in the TopLevel SubQuery.
15. Click **Save**.

Create Invoice View SubQuery

1. Navigate to the **Query Builder > SubQuery Options** sheet.
2. In the **Active SubQuery** toolbar, click **Add Subquery**.
3. In the **Name** field, enter **InvcHed**.
4. In the **Type** field, select **Union**.
5. Navigate to the **Query Builder > Phrase Build** sheet.
6. Place the following tables on the designer canvas:
   - **Erp.InvcHead**
   - **Erp.Customer**
7. Accept the proposed Customer table alias.
8. Navigate to the **Display Fields > Column Select** sheet.
9. Move the following columns to the **Display Column(s)** list.
   - **InvcHead_Company**
   - **Customer2_Name**
   - **InvcHead_InvoiceNum**
   - **InvcHead_InvoiceDate**
   - **InvcHead_InvoiceAmt**
10. Click the **Calculator** icon to access the **Calculated Field Editor** window.
11. Click **New** and enter these field values:

Data Field	Data Value
FieldName	SubQueryName2
Data Type	nvarchar
Label	Document Type
Editor pane	'Invoices'

12. Click **Save** and exit the Calculated Field Editor.
13. Move the **Calculated_SubQueryName2** column up and make it the first column in the list.
14. Click **Save**.

Create Query Parameter

Create a Query Parameter to filter the data using an alternative parameter value you define. In this example, you want to filter results by date. When you launch the BAQ, the parameter value displays for input.

1. From the **Actions** menu, select **Define Parameters**.
   The **Query Parameters** window displays.
2. In the **Parameter Name** field, enter **Date**.
3. In the **Date Type** field, select **date**.
4. Accept the default values and click **Save**.
5. Exit the Query Parameters window.
   You now define which BAQ fields you want filter by the date parameter.
6. Navigate to the **Query Builder > Phrase Build** sheet.
7. With **InvcHead** SubQuery in focus, in the design canvas, click on the **Erp.InvcHead** table.
8. At the bottom of the screen, verify the **Table Criteria** sheet displays.
9. From the toolbar, click the **Add Row** button.
10. In the **Field** column, select **InvoiceDate**.
11. In the **Operation** column, verify the equal sign (**=**) defaults.
12. From the **Filter** Value field, select **specified parameter**.
13. Click the word **specified** to launch the **Select Parameter** window.
14. Verify the **Date** parameter you created is highlighted and click **Select**.
15. Repeat steps 7 - 14 to apply the same filter on **OrderHed** and **QuoteHed** SubQueries. To switch between SubQueries, use the **Active SubQuery** toolbar.
    Use the below table for reference:

Table	Field	Operator	Filter Value
Erp.OrderHed	OrderDate	=	@Date parameter
Erp.QuoteHed	EntryDate	=	@Date parameter

16. Once finished, click **Save**.
    You can now test the BAQ.

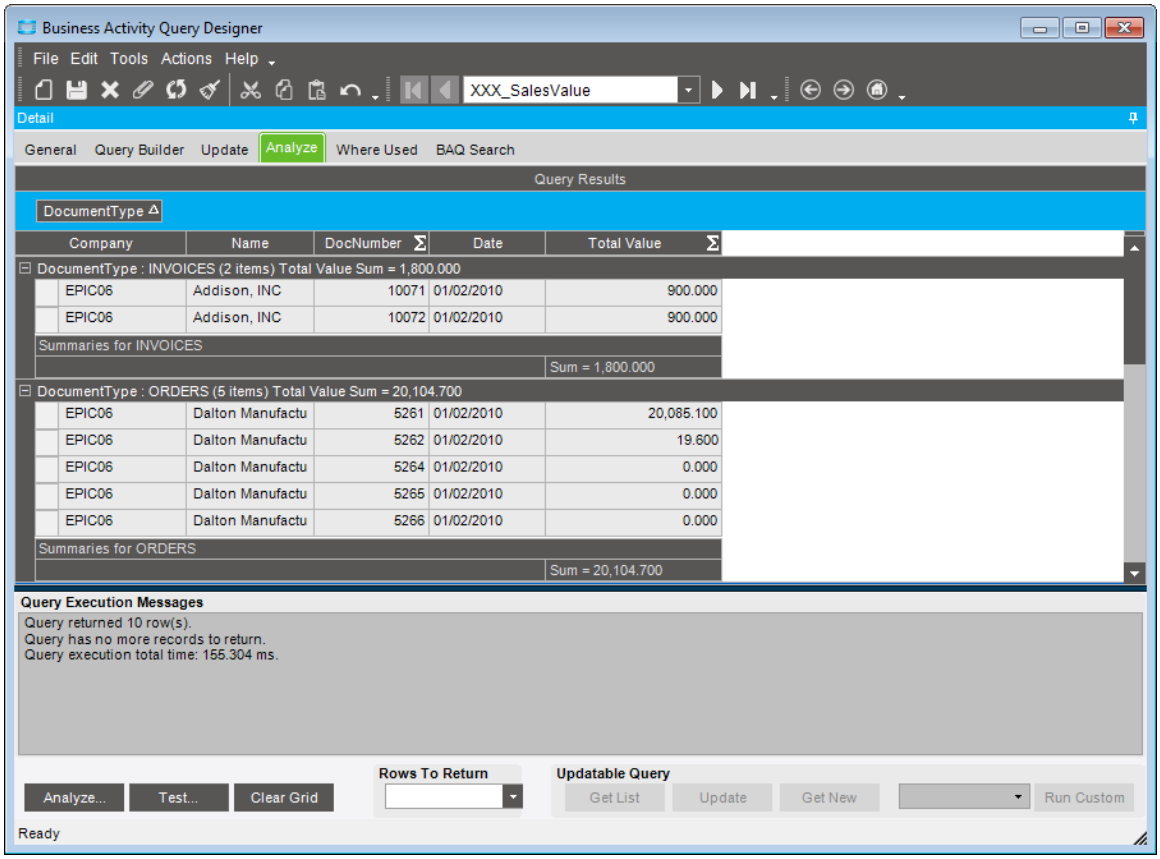
Test the BAQ

1. Navigate to the **Analyze** sheet.
2. Click the **Test** to invoke the **Parameters** window.
3. To test the BAQ, you can use the following dates that have sales records in the Demonstration Database.
   - **03/25/2019 (March 25)**
   - **03/27/2019 (March 27)**

- **04/05/2019 (April 5)**

4. Right-click anywhere in the grid and select **Show Group By** and **Show Summaries**.
5. Drag the **Document Type** column to the pane above the grid.
6. In the **Total Value** header, click the **Σ** (Sigma) icon.
7. In the **Select Summaries** window, select **Sum** and click **OK**.
8. This breaks down the BAQ results by Quotes, Orders, and Invoices with their total values for a given day.

Example



9. Remain in the Business Activity Query Designer.

BAQ Search

Use the **BAQ Search** sheet to select data in your query that you want to make available to users searching for related data.

Every Search form has a sheet called BAQ Search. This feature provides an opportunity to create your own type of search criteria.

Example Your BAQ for parts uses two columns from the Part table: **Part_PartNum** and **Part_PartDescription**. These are the columns you selected on the **Display** sheet. Both columns are listed in the **Available Like Columns** section of the BAQ Search sheet.

Your own BAQ becomes an option for a part number search. If you select the **Shared** check box on the **Detail** sheet of the query, all users can use the BAQ as a unique search mechanism.

Since there are no searches based on part description as the record key, selecting the **Part_PartDescription** column for the BAQ Search Like Column is not useful.

Note You can not select application queries on the BAQ Search sheet. Application queries come loaded with the Epicor application and always start with the prefix **z**. For example, zAPCheckDtl.

Workshop - Use BAQ Search

This workshop explains how to create a BAQ search using the query created in the previous workshop. Enable the query to be become a search mechanism throughout the Epicor application.

Create BAQ Search

1. In **Business Activity Query Designer**, navigate to the **General** sheet.
2. Click **Query ID**, and search for and select the **XXX_Parts** query (where XXX are your initials).
3. Navigate to the **BAQ Search** sheet.
4. In the **Available "Like" Columns** list, select **Part_PartNum**.
5. Click the right arrow button to move the selected column to the **BAQ Search "Like" Columns** list.
6. Click **Save**.
7. Click **Clear** and minimize BAQ Designer.

View BAQ Search

Navigate to **Part Maintenance**.

Menu Path: Material Management > Inventory Management > Setup > Part

1. Click the **Part** button.
   The **Part Search** window displays.

2. In the **Part Search** window, navigate to the **BAQ** sheet.

3. Select the **XXX_Parts** (where XXX are your initials) query and click **Search**.

   The **Search Results** pane in the search window displays the query information you can use to search for a part.

   Since you selected the **Part Number** field as the **Like** column for searching, any program where you can search for a part has this BAQ search available.

4. In the **Part Search** window, click **Cancel**.

5. Exit Part Maintenance.

Cross-Company Queries

Use this feature to create and run Business Activity Queries (BAQ) against multiple companies within a single Epicor ERP installation.

Use this functionality to run multi-company dashboards with a single BAQ call.

To enable a Cross-Company query creation for a user, in **User Account Maintenance**, select the **Allow Creation of Cross Company BAQ** check box. This check box only controls the ability to create and maintain Cross-Company BAQs.

Users can only run Cross-Company BAQs against companies they can access. You define which companies each user can access through User Account Security Maintenance. Likewise because Field Security settings for each user can be different in each company, the Cross-Company BAQ uses the field security defined within each company.

Example If the user has no access to AbcCode.CountFreq field in EPIC02 company and has no access to AbcCode.PcntTolerance field in EPIC03, then all rows returned for AbcCodes from EPIC02 company contains empty CountFreq field and empty PcntTolerance field for the EPIC03 company.

External Business Activity Queries

The **External Business Activity Query** related programs provide an extension to the standard Business Activity Query capability. External BAQs provide the mechanism for retrieving, updating, and displaying information from an external database using the SQL language.

The process of creating an external BAQ involves the following:

- Establish a connection to an external datasource using the ADO .NET DB provider available on the server and compose a connection string to access the external database management system.
- Enable the external datasource you create in the current company.
- Set up security restrictions to schema and data coming from an external datasource as required.
- Modify external datasource metadata specified in the source database as required.
- Build an external Business Activity Query and use it as a datasource for reports, dashboards, trackers, and so on.
- Expose the data to users using the Epicor Everywhere™ Framework tools.

BAQ Zones

BAQ (Business Activity Query) zone is an embedded query you can link to a specific field on a program interface. When you activate a BAQ zone, it displays as a linked tool tip window. The data that populates this window depends on both the business activity query and current value, if any, within the linked field.

You create and modify business activity queries within the BAQ Designer. Use this custom query tool to select and join tables. You then define what columns from the selected tables display in the results. Through this functionality, you can also create calculated fields that run an expression against the query results to return unique values. These custom queries display the data you want; you can then link these queries as BAQ zones.

After you create or modify the BAQ you will use for the BAQ zone, you then link the BAQ to a specific field by either using **Extended Property Maintenance**, **Context Menu Maintenance** or embedding the BAQ zone in a customization. When a BAQ zone is linked to a field, a zone indicator displays on a program interface during Rune Mode. These zone indicators display as arrow buttons next to the field.

For more information on BAQ Zones, review the **Advanced Embedded Customization** course and the **Application Help**.

BAQ Utilities

The Actions menu of the Business Activity Query (BAQ) Designer contains several functions. These functions provide additional capabilities that pertain to the query currently in focus on the screen. This section reviews some of these options for managing BAQs.

Export and Import the BAQ

By default, a Business Activity Query is linked to the company in which the query is created. As long as the database structures are the same, you can export the same query from one company and import it into another company.

Export the BAQ

Use the Export BAQ option to export the current Business Activity Query (BAQ). You can save the query within a personal archive directory to provide multiple versions. You can rename the Query ID and identify the file name.

The Export BAQ option only exports the query definition. To export the data generated through the selected query, use **Business Activity Query Export Process**. You can use this program to export query data through either the **.xml** or **ASCII** file formats that display on an .ASP page.

You import the query back into the Epicor application using the Import BAQ option.

Import BAQ

Use the **Import BAQ** option to import a query that was previously exported.

Example You often use the export and import BAQ functionality to reuse a query in another company. You can also use the import functionality to pull in BAQs created in previous Epicor ERP versions. While the BAQ is imported, its definition is uplifted so you can use the imported BAQ in the current version, utilizing the updated ICE framework.

When you import the BAQ, you can change the query's identifier. You can also indicate that this query must immediately display within the Business Activity Query Designer. Within the Designer, you can check the imported query works properly and perform necessary adjustments in BAQ design as required.

Revert to Saved

Use the **Revert to Saved** option to restore the saved version of the current Business Activity Query. All changes that you made to the current query are removed, and the query reverts back to its previously saved version.

Workshop - Export and Import BAQs

This workshop demonstrates how to export the PartBin query created in an earlier workshop and then import it into the Epicor application with a new name.

Export BAQ

1. In **Business Activity Query Designer**, navigate to the **General** sheet.
2. Click **Query ID**, search for and select any BAQ you created in this course.

3. From the **Actions** menu, select **Export BAQ**.
   The **Export Business Activity Query** window displays.
4. Click **Select File**.
5. Select **Desktop**.
6. In the **File name** field, enter **XXX_Export** (where XXX are your initials).
7. In the **Export Business Activity Query** window, click **Save**.
8. Click **Export**.
   A message displays in the Export processes message window indicating whether the export was successful.
9. Once the process completes, click **Close.**
10. Click **Clear** and remain in the Business Activity Query Designer.
11. View your Desktop to verify the file exported correctly.

Import BAQ

1. In **Business Activity Query Designer**, from the **Actions** menu, select **Import BAQ**.
2. In the **Import Business Activity Query** window, click the **Select File** button.
   The Import BAQ window displays.
3. Navigate to the **Desktop** folder.
4. Click the **XXX_Export** (where XXX are your initials) file.
5. In the **Import Business Activity Query** window, click **Select File**.
6. In the **New Query ID** field, enter **XXX_Import** (where XXX are your initials).
7. Select the **Show in Designer** check box.
8. In the **Import Business Activity Query** window, click **Import**.
   Notice the query displays within the Business Activity Query Designer with a new name.
9. Exit the **Import BAQ Window**.
10. Click **Clear** and remain in the Business Activity Query Designer.

Generate ASP

Use the **Generate ASP** option to convert a query to an .asp page. You can display this page on your website, updated with current information from your database.

Use this program to both select which query fields you want to display and define the search options you use with this data. You may also change the name of field columns. To finish generating the page, name the ASP file and save it within your web deployment directory.

File Types

Once you click the **Generate ASP** button, the Epicor application creates two files within the selected directory:

- **.asp** – The file generated by the program. This file is referenced by your company's website.
- **.xsl** – The file that sorts and displays data results on the .asp page.

By default, these files are generated in `C:\EpicorData\WebDeployment` folder on your application server. You only need to create these files once.

You must also copy two other files - filterdata.xsl and exports.css - to the same WebDeployment directory. These files are standard style sheets provided by Epicor:

- **exports.css** - This file is a Cascading Style Sheet document. Developers use the Cascading Style Sheet to control the style and layout of multiple Web pages all at once. As a Web developer, you can define a style for each HTML element and apply it to as many Web pages as you want. To make a global change, change the style and all elements in the Web are updated automatically.
- **filterdata.xsl** - This file is an XSL Style Sheet. This file is used by the ASP page to create the filters as defined in the generate ASP program. You can use XSL to display information in your XML document.

In Epicor ERP installation, these files are found in the `Server\ud` directory.

The last step is that you need to generate the active **.xml** data that will be displayed within the .asp page. You generate this data through the **Business Activity Query Export Process** program. Because this is a process program, you can automatically update the BAQ data through a scheduled interval. This will refresh the information periodically, letting your customers, suppliers, and other users see your current data. You must set up this process to generate the .xml file into the same WebDeployment directory. For more information on how to export query data, see the following topic **Export Query Data**.

Important The directory path \\<ServerName>\EpicorData\WebDeployment needs to be defined as a Virtual Directory before the asp page will worked properly.

Export Query Data

Use the **Business Activity Query Export Process** to export query data through either the .xml or ASCII file formats.

This data can then be used on an ASP page, or imported to Microsoft® Excel® for analysis and publishing.

First, select the Query ID and select the output format. This can be either

- XML - The data source file used by the .asp page.
- CSV - For use with Microsoft Excel.

Next, enter the Output FileName for the exported query. By default, this file will be automatically saved on the server within the EpicorData\Processes\<UserName> directory, for example in `C:\EpicorData\Companies\EPIC06\Processes\EPICOR`. The location of this data directory is specified in System Agent Maintenance.

Note For Epicor Cloud ERP users, the path is `...\Companies\siteID\siteID-<UserName>`.

You can override the default output location by exporting the file to a custom location you define. It is allowed to use:

- Relative paths (root folder level only), for example, temp/filename.csv
- Network paths, for example \\server\share\folder\filename.xml

For CSV output, you can define how you will separate units in the exported file (Text Delimiter) and choose to export field labels with the query data (Output Labels).

From this list, select the schedule option during which you would like the report to run. Options include Now, Startup Task Schedule, and any other user-defined schedules created for your company. You can indicate that the report should be run on a repeating basis by selecting the Recurring checkbox. This check box is available only if a schedule other than Now is selected.

If you don't have direct access to the server file system, such as with SaaS and Early Access, use the **Server File Download** utility. It transfers one file at a time from one of three directories on the server: Company, User, or Reports to the client. Specify the file type, select the file, and then browse for a path for the download. This program is found in **System Management > Schedule Processes**.

Copy Query

Use the **Copy Query** option to duplicate the current Business Activity Query. This allows you to use an existing query as a base for a new query that you need.

Note System queries (with a z prefix in their IDs) are the default queries required for the Epicor ERP application to run. You cannot edit system queries, however, the Copy Query option allows you to duplicate the query. You can edit and customize this duplicate query as you need.

Workshop - Copy a Query

This workshop demonstrates how to copy a system query.

1. In the **Business Activity Query Designer**, verify the **General** sheet displays.
2. Click the **Query ID** button, and search for and select **zCustomerShipments**.
3. From the **Actions** menu, select **Copy Query**.
   In the **Copy Query** window, the **Copy From** section displays the current **Query ID** and **Description**.
4. In the **Copy To** section, in the **Query ID** field, enter **XXX_CustShip** (where XXX are your initials).
5. Confirm by clicking **OK**.
6. The copied query displays within the Business Activity Query Designer.
7. Remain in the Business Activity Query Designer with XXX_CustShip BAQ in focus.

Change Author

Use the **Change Author** option from the Actions menu command to change the author name of the current Business Activity Query. By default, queries can only be updated by the user who created the query, known as the author. You can use the Change Author option to reassign the author of a query to a different user.

Once you change the author, the Epicor application uses the new author identifier immediately. The new author is allowed to update the query as needed.

The Change Author functionality is beneficial when a user leaves the organization or changes roles. It can also be helpful when a more experienced user within an organization helps a less experienced user with formulas, filtering, and so on.

Workshop - Change Author

This workshop shows how to use the change author functionality.

1. Verify the **XXX_CustShip** (where XXX are your initials) query is open.
2. From the **Actions** menu, select **Change Author**.
   The **Change Author** window displays.
3. Click the **Author** button, and search for and select **Steven Tyler (STyler)**.
4. Click **OK**.
   View the red **Other Owner. No Save** message that displays.
   This message indicates you cannot make changes to the query as it is assigned to a different author. To change the current query or to select a different author, you must log in as **STyler**.
5. Exit Business Activity Query Designer.

Conclusion

Congratulations! You have completed the Business Activity Queries course.

Additional information is available at the Education and Documentation areas of the EPICweb Customer Portal. To access this site, you need a Site ID and an EPICweb account. To create an account, go to http://support.epicor.com.