**Understanding Joins in Ad Hoc Reports**

What is a join?

When you use more than one data table in the Ad Hoc report writer, you need to join the tables together using a common value in both sets of data. You are able to connect the data from these tables in different ways, and the ways you can connect them are called joins.

Why are joins important to my report?

The data that you are able to pull into your report will depend on the type of join you select. If you join two tables of data in an incompatible way, you may get inaccurate results or no results at all.

What types of joins exist?

Accela has five types of joins available in Ad Hoc: the inner (direct) join, the cross join, the left (first exists) join, the right join, and the full join.

How do I know which join to use?

Being familiar with the different types of joins, as well as being familiar with your data, will allow you to choose to the best possible join type. Read about the different types of joins and contact our helpdesk if you have any questions.

**Terminology**

The terminology we use when training on the ad hoc report writer is difficult to explain succinctly, but it is very important to being able to understand joins. To explain the terminology used in this training document, we will use an example of a more simplified database:

A **database** is where all of the information in a software program is stored. The information in the database is grouped into tables. The tables in our example are:

Customers table – would contain information recorded by staff related to customers

Products type table – would contain information about the products for sale

Purchases table – would contain information on all of the purchases the business makes

Employees table – would contain information related to the employees.

When an employee of this business records a sale, they are entering the customer’s name and phone number into the system and that data is saved in the data table called the customers table. Similarly, when an employee is doing inventory, they are entering data into the system about the products in stock, which is stored in the products type table. So a **table** is a set of data that has been grouped together based on relevance.

The tables of data are built with the information that is entered into various fields on forms or applications. When an employee enters data about a customer into the system, they are entering the information into fields, such as the name field, the phone number field, or the address field. So a **field** is a place on an application or a form where information is collected. The information collected in these fields is what you are copying from the data table to put into your report. So if you wanted a report on all of your customer’s names and numbers so you could contact them about a recall, you would build a report that uses the customers table to pull their name and contact information into your report.

Let’s take a closer look at the tables. What does a table look like in the database?

The employees table has been broken into groups of data that correspond to the fields on an application. These are called columns. So when employee information is entered into the software, it is stored in columns, which are titled to correspond with the field name. So a **column** is where information is grouped based on which field in which it was entered. The information within a column, such as John Smith’s name, is called a **value**.

To summarize, the **database** is made up of **tables** of related information, which contain **columns** of data, which were built using **values**, which are populated from information entered into **fields** on an application or form.

When you build a report, let’s say from the employees table, you will be pulling columns of data into your report by selecting fields, and then you will typically filter that data based on your report needs. A **filter** is exactly what it sounds like: it filters out data that you do not want to see in your report. If you wanted to see a report on the employee information for everyone with a salary range higher than 25, you would select the fields for name, employee id, and salary range. Before applying the filter, your report would contain information on all three employees shown in the diagram above. But if you apply a filter to your report that says something like “Salary Range is greater than 25,” then your report would contain the fields related to John Smith and Jane Smith only.

This hypothetical database works just like the database you build ad hoc reports from. If you built a report in ePermitting and pulled in the field for permit number from the record table, your report would contain every permit number that exists in your agency, unless you apply a filter, such as one for permits submitted in the last month.

When we talk about **joins**, we are talking about building a report that contains fields from two different tables. How you join two tables together will impact the data that can be displayed in the report.

**How to Join Tables**

First, select the two tables you want to use in your report. The join view, called DB View Relationships, will automatically pop up when a second table is checked. Here is a screen shot of what DB View Relationships will look like after you fill in the cells using the dropdown menus:



For this example, the V\_OSM\_FEE\_PAYMENT\_HISTORY table has automatically populated in the top left field, because it was the first table selected in the checkboxes. The field directly below it is the first cell, where you use a dropdown menu to select a data table to create the join. The second table that was checked, V\_OSM\_BUILDING\_RECORD in the example, is the table you will want to select for that first cell.

In the second cell, you will select a value from the second data source. The value you select here needs to be able to be mapped to the column you will select in the fifth cell. We will explain this more as we cover joins below. In the example, we have selected RECORD\_ID.

The dropdown menu in the second cell

The third cell is a read-only operator that contains the equal sign.

The fourth cell should automatically populate with the name of the first data table you selected, V\_OSM\_FEE\_PAYMENT\_HISTORY in our screen shot example. The fifth cell is for selecting a column from the first data source that you are mapping data to from the second data source. In our example, a column in both of the chosen data tables (V\_OSM\_FEE\_PAYMENT\_HISTORY and V\_OSM\_BUILDING\_RECORD) is RECORD\_ID. RECORD\_ID is the column that contains the record numbers assigned to permits, planning applications, ect. Because that column is the same in both tables, the tables can joined through this column.

The sixth cell is where you will identify how you want to join the tables. You can join the data in specific ways to either include or exclude certain data from your report.

The join type dropdown in the sixth cell

**Types of Joins**

The Venn diagram is a visual representation of what is happening between the two data tables, while the data flowchart is a more concrete visual showing how the data is interacting.

**Inner (Direct) Join -**

The rows that are returned from an inner join will be the rows from the two tables when the value in a row for the first table also appears in a row for the second table. So the data you get using the inner join will exclude **values** that do not exist in both of your data sources. In the Venn diagram, the information that would be returned is represented by the dark green overlap of the two circles. In the flowchart, you can see that when there is no RECORD\_ID match in the columns, there is no result returned. What that means for your data is if you select the table V\_OSM\_FEE\_PAYMENT\_HISTORY and join it to V\_OSM\_BUILDING\_RECORDS using an inner join via RECORD\_ID, then you will not be able to include data in your report for records that have had no payments made on them.

RECORD\_ID RECORD\_ID

(from V\_OSM\_BUILDING\_RECORD) (from V\_ OSM\_FEE\_PAYMENT\_HISTORY)

1. 119-15-0123 a. 119-15-0111 No match found, not returned
2. 119-15-0135 b. 119-15-0123 Match found, result returned
3. 119-15-0249 c. 119-15-0150 No match found, not returned
4. 119-15-0300 d. 119-15-0300 Match found, result returned
5. 119-15-0410 e. 119-15-0410 Match found, result returned

From this join, the data that is returned will include:

All payment and record values for 119-15-0123, 119-15-0300, and 119-15-0410

No payment or record values for 119-15-0111, 119-15-0135, 119-15-0249, and 119-15-0150

What does this mean for your report? Well, the first step after selecting your join is to choose fields for the report. If you choose the field for record id, your report would display only the returned values, so it would have 119-15-0123, 119-15-0300, and 119-15-0410. If you selected the field for date the record was opened, or the field for type of record, it would display the dates corresponding to those records. It might look something like this:

RECORD\_ID DATE\_OPENED RECORD\_TYPE AMOUNT\_DUE

119-15-0123 12/20/2015 Res Mechanical $80.00

119-15-0300 1/3/2016 Com Electrical $50.65

119-15-0410 1/7/2016 Res Structural $1100.30

**Left (First Exists) -**

A left join (left outer join) returns all rows from the left table, whether the joined columns have a match or not. Remember that the first table you select will always be your left table. A field will display a null value if the corresponding table does not contain a matching row. Since the left outer join returns all rows from the left table as well as the matching rows from the right table, if the field from the right table has no match, it will not be returned. This can be helpful in identifying when information is missing from records in the right table.

Left (First Exists) Join:

RECORD\_ID RECORD\_ID

(from V\_OSM\_BUILDING\_RECORD) (from V\_OSM\_FEE\_PAYMENT\_HISTORY)

1. 119-15-0123 a. 119-15-0111 No match found, not returned
2. 119-15-0135 b. 119-15-0123 Match found, result returned
3. 119-15-0249 c. 119-15-0150 No match found, not returned
4. 119-15-0300 d. 119-15-0300 Match found, result returned
5. 119-15-0410 e. 119-15-0410 Match found, result returned

 All of these results returned

What would a report with the left first exists join look like?

RECORD\_ID DATE\_OPENED RECORD\_TYPE AMOUNT\_DUE

119-15-0123 12/20/2015 Res Mechanical $80.00

119-15-0135 12/24/2015 Res Plumbing NULL

119-15-0249 12/28/2015 Com Mechanical NULL

119-15-0300 1/3/2016 Com Electrical $50.65

119-15-0410 1/7/2016 Res Structural $1100.30

Even though there was no payment information for records 119-15-0135 and 119-15-0249, they still were returned in the results for this report because this type of join includes every record in the left table, which in Accela is the first table selected in datasource. The amount due for those two records displayed a null value since there was no information in the database for them. In the inner direct join, records 119-15-0135 and 119-15-0249 would not have been in the data in this report. Carefully considering the data that is relevant to your report will help you make the decision of which join to use.

**Full Join -**

A full join (also known as a full outer join) returns all rows from both the right outer and left outer joins. A field in a result row displays a null if the corresponding table does not contain a matching row. In a full join, the full results from both tables will be returned, and the values within the records that do not match, will result in a null value.

Full Join:

RECORD\_ID RECORD\_ID

(from V\_OSM\_BUILDING\_RECORD) (from V\_OSM\_FEE\_PAYMENT\_HISTORY)

1. 119-15-0123 a. 119-15-0111
2. 119-15-0135 b. 119-15-0123
3. 119-15-0249 c. 119-15-0150
4. 119-15-0300 d. 119-15-0300
5. 119-15-0410 e. 119-15-0410

All Results Returned All Results Returned

In our example, the values returned in your report would look very similar to the ones returned in the left (first exists) join, but it would also include results from the second table that are not part of the first table. If we add a field to our example report for AMOUNT\_PAID, we would also see payments that are not associated to building records. Our example report only shows some of the records in the list above so that the join type’s impact to the data is simpler to visualize.

RECORD\_ID DATE\_OPENED RECORD\_TYPE AMOUNT\_DUE AMOUNT\_PAID

119-15-0123 12/20/2015 Res Mechanical $80.00 NULL

119-15-0135 12/24/2015 Res Plumbing NULL $120.00

119-15-0249 12/28/2015 Com Mechanical NULL $80.00

119-15-0300 1/3/2016 Com Electrical $50.65 NULL

119-15-0410 1/7/2016 Res Structural $1100.30 $350.00

00239\_POS 1/11/2016 Point of Sale Map NULL $1.50

Records 119-15-0135 and 119-15-0249 were displayed in the left (first exists) join report on the previous page even though there was a null field under AMOUNT\_DUE, but it excluded the point of sale payment information. This report that uses the full join includes the point of sale payment in the data, because data that exists in the right table, the V\_OSM\_FEE\_PAYMENT\_HISTORY table in our example, but does not in the left table, V\_OSM\_BUILDING\_RECORD, is still returned. This would be helpful for a report that is intended to include the department’s incoming payments, but the left (first exists) join would be a better choice for a report that is intended to show incoming payment from building permits.

**Right Join -**

A right join (also known as a right outer join) returns all rows from the right table in the right outer join clause, whether the joined columns match or not. A field in a result row displays a null if the corresponding table does not contain a matching row. If you use the right join (right outer join), the results that will be returned will be everything from the second data table, and the values that match from the first data table.

Right Join:

RECORD\_ID RECORD\_ID

(from V\_OSM\_BUILDING\_RECORD) (from V\_OSM\_FEE\_PAYMENT\_HISTORY)

1. 119-15-0123 a. 119-15-0111 No match found, not returned
2. 119-15-0135 b. 119-15-0123 Match found, result returned
3. 119-15-0249 c. 119-15-0150 No match found, not returned
4. 119-15-0300 d. 119-15-0300 Match found, result returned
5. 119-15-0410 e. 119-15-0410 Match found, result returned

All Results Returned

This type of join has excluded the building records that had no fees paid on them, but it has included all data on payments made, such as the point of sale map and copy fee. This is because those payment records are part of the data from the right table, which has all of it’s data returned when you use the right join.

RECORD\_ID DATE\_OPENED RECORD\_TYPE AMOUNT\_PAID

119-15-0123 1/3/2016 Com Mechanical $80.96

119-15-0111 1/4/2016 POS\_Map $1.50

119-15-0150 1/8/2016 POS\_Copy $.25

119-15-0300 1/11/2016 Com Electrical $80.96

119-15-0410 1/7/2016 Res Structural $350.00

**Cross Join** –

A cross join returns a result table where each row from the first table combines with each row from the second table. In the Venn diagram, both circles are dark in color since everything from both tables will be returned.

RECORD\_ID RECORD\_ID

(from V\_OSM\_BUILDING\_RECORD) (from V\_OSM\_FEE\_PAYMENT\_HISTORY)

1. 119-15-0123 a. 119-15-0111 Result is combo of all options
2. 119-15-0135 b. 119-15-0123 Result is combo of all options

Result is a combo of all options Result is a combo of all options

It will be rare that you will find a use for the cross join. A cross join will create a result from every possible combination of values, so it would be including multiple results in the report wherever data is in both tables. This typically returns more values than any report would need. If you think the cross join is the best join for your report, contact our helpdesk.