SISENSE

Whitepaper

Joins, Aggregations and Many-to-Many Relationships

The Data Mash-Up Cheat Sheet

www.sisense.com

Intro

Mashing up multiple data sources to generate a single source of truth is an integral part of data analysis. It allows you to compare and cross-reference records stored in different formats and locations, and to perform queries and calculations.

This guide is intended to run you through some basic concepts in data analysis that you should become familiar with when joining data stored in multiple tables, and to give you a clear picture of the types of problems you might encounter and how to resolve them. Additionally it will provide some practical tips for evaluating the mash-up capabilities of different Business Intelligence software tools.

CONTENTS

Click on the title to skip to the relevant section

Understanding Joins

Inner and Outer Joins

What's a Join? And isn't that a verb?

In the context of SQL and database management, a join is a way of combining records from multiple tables. A join requires common fields between the two tables in order to form a logical connection, and is the basis for combining different data sources and an integral part of data analysis.

Most organizations don't manage to store all of their data in one single spreadsheet, but in rather do so multiple (often dozens or hundreds) of separate tables. But when it comes to analyzing this data, it often needs to be combined in one central locations in order to perform queries and calculations.

To connect two different tables there must be some way of forming a logical connection between the datasets – in effect, this means there must be a common field between the two. Here are the main types of joins you should be familiar with:

Inner Joins

Used for connecting identical fields

An inner join is used to connect two or more tables that contain fields with identical records. For example, if we look at these two tables:

Product	Price
Apples	2
Pears	2
Oranges	3

Product	Stock
Apples	5
Pears	8
Oranges	10

These two tables would be combined via an inner join based on the common field – Product. The result would be one table which contains data from the previous two:

Product	Price	Stock
Apples	2	5
Pears	2	8
Oranges	3	10

If one of the tables contained a record that does not appear in the other (e.g. if the Product-Stock table would contain a fourth row with details of banana stock), that data would be disregarded.

Outer Joins

Used for connecting common, but not identical fields

Outer joins are divided into *left, right and full joins*. To understand the difference between the two, let's once again look at two tables:

Product	Price
Apples	2
Pears	2
Oranges	3
Bananas	5

Product	Stock
Apples	5
Pears	8
Oranges	10
Avocados	12

As you can see, there is no column with completely matching records between these two tables. There are three main ways we can go about combining this data:

- **Left join**: Generates a table that contains all the records from the lefthand table, along with any matching records found on the righthand one. In our example:

Product	Price	Stock
Apples	2	10
Pears	2	8
Oranges	3	5
Bananas	5	NULL

- **Right join**: Same as left join, but will contain all records from the righthand table.
- **Full outer join**: Will essentially perform both a left and right join, combining the two tables despite any records not matching, e.g.:

Product	Price	Stock
Apples	2	10
Pears	2	8
Oranges	3	5
Bananas	5	NULL
Avocados	Null	12

These are the main types of joins you're likely to encounter. Each might be used in different scenarios, depending on the analyses you will want to perform on the combined data.

Relationships

Avoiding the Dreaded Many-to-Many Relationship

Relationship Status: It's complicated

A relationship specifies the logic used to combine data from one or more tables. You create relationships by connecting fields between two or more tables, and this determines what data is eventually presented.

There can be 3 types of relationships:

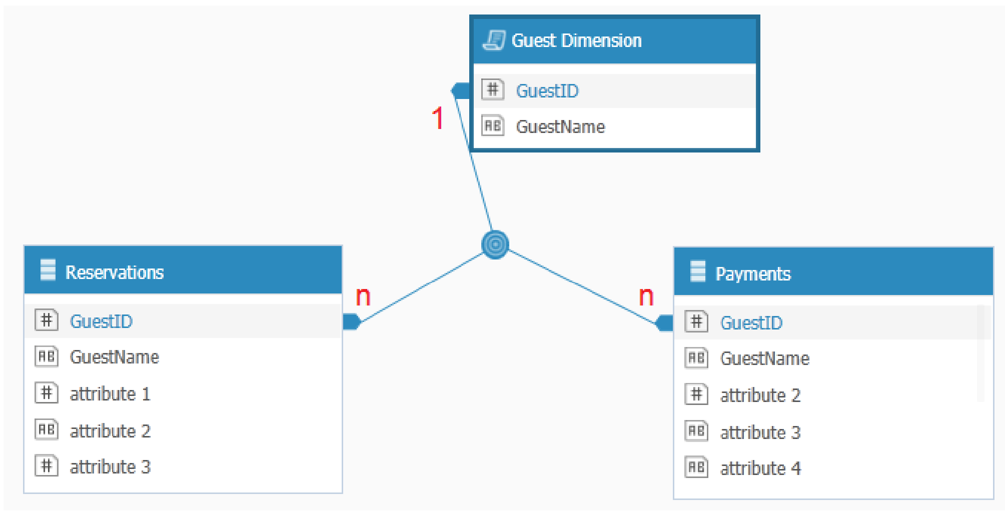
- **One-to-One relationship.** In this scenario, the field used to connect both tables only has unique values in every row.
- **One-to-Many relationship.** One table holds unique values for every row, but the other table holds duplicate values for any or all of the corresponding values in the first table.
- **Many-to-Many relationship.** In this scenario, on both sides of the table there are duplicated values, causing excessive calculations for every query running against it

While the first two logical relationships pose no problem, the third should be avoided at all costs as it is certain to cause issues later on.

Many-to-Many Relationships Defined

When a field from two or more tables contains the same value, and these values are duplicated in both tables, a connection is made based on this field and a many-to-many relationship is created.

We know, it's confusing, so here's a business case example: a hotel can have a data table with reservation data and a table with payment data. In both tables the name of the guest is stored. A guest can have multiple reservations under their name, as well as multiple payments for their stay recorded in their name. If a relationship between the reservation and payment table was created based on the guest's name, a many-to-many relationship would be created (as the guest's name appears multiple times in each table).



Negative Consequences: The problem with this kind of relationship is that it can create complex data sets which either: do not return the correct results, or use excessive computing resources and don't return any results. Both scenarios lead to data havoc such as creating duplicates, incorrect results, and performance lags, which is why you should try to avoid many-to-many relationships all together.

Best Ways to Resolve M2M Relationships

There are many ways to resolve M2M relationships, but here are the common approaches based on the number of M2M relationships that exist as well as the number of tables involved. Your best options would usually be to:

1. Break this relationship into two separate one-to-many relationships
2. Use the Lookup function to copy a value from one table and import it into another based on a logical test.
3. Combine the two tables into one.

The most important aspect of understanding a many-to-many relationship is to know that they need to be avoided in order to ensure you have accurate data, no duplicate values, and your performance does not lag—a negative consequence of an M2M relationship that will ruin the user experience and discourage queries.

Further Reading:

5 simple rules for building relationships

SISENSE

Mashing up Data with BI Software

3 Questions to Ask

What you should look out for

Business Intelligence software is meant, among other things, to allow you to easily connect different data sources and join tables for centralized analysis. Answer these 3 questions to determine your BI tool's mash-up capabilities.

1. Can you access the raw data?

A BI tool with a strong analytics engine should enable you to connect directly to your data sources, whether these are in the form of flat files, servers or cloud apps. If you need to invest heavily (in time, IT or DBA resources) in order to prepare your data before you can start to combine and integrate it, this could be a sign that your BI tool is not up to speed in terms of data preparation.

2. Can you perform row level joins?

A row level join means that after two data sets are combined, the original records are still kept within the data model and accessible. This, once again, is a question of the strength and sophistication of the engine that powers your analytics tool: A robust one should be "smart" enough to understand the data and model it in such a way that will allow you to drill down into the full granularity of your data, as it was before the tables were joined.

In contrast, BI software that relies on a less powerful back-end will instead create a *view* of the data by *aggregating* it, thus compromising its granularity. The guiding line should be this: *can you easily access the original, row-level details of your data after the join is performed?* If the answer is no, your BI software might be lacking in terms of its mash-up capabilities.

3. Will you need to know scripting or coding?

While (we hope) the information in this guide will come in handy, the fact of the matter is you shouldn't actually need to use it in your day-to-day work with data. BI software has come a long way towards self-service, and modern tools should be able to perform most mash-ups automatically, without the end-user needing to write SQL or otherwise manually defining the parameters in which the join is performed.

While there is no software currently available that can offer a truly "zero IT" solution for scenarios that involve many complex and unstructured datasets, simple joins should definitely be a matter that can be handled by the software itself, especially when it comes to the more “mainstream” data sources – Excel, CSV files, Salesforce, etc. If joining these types of sources requires you to take a crash course in SQL, this could definitely be indicative of a weak business intelligence platform.

PRACTICE MASHING UP YOUR OWN DATA!

DOWNLOAD A FREE TRIAL OF SISENSE