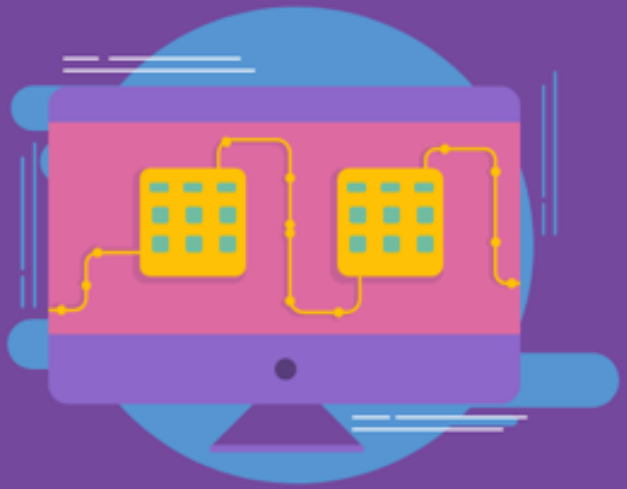
WHAT IS A RELATIONSHIP?



A relationship is an association, or connection, between two tables.

When you relate two tables, you save time because you don't need to enter the same info over and over in separate tables.

Projects have multiple tasks, a company could have multiple locations, or several books can be written by the same author.

Got it

Next

Table-to-table relationships

WHY USE A RELATIONSHIP?

For example:

When entering a **new company**, it would be nice to enter all the contacts who work there. Or, when entering a **new contact**, to also pick the company they're with.

You can use a **relationship** to do this. Instead of creating new fields for company info on the contacts form, you can simply connect info from a Companies table right into the Contacts form.

And you can **summarize all the info** from Contacts to display in Companies, for instance, the number of Contacts.

Almost every Quick Base app has several relationships.

Got it | Back | Next

Table-to-table relationships

BENEFITS OF RELATIONSHIPS

When you relate two tables you:

Save time because you don't need to enter the same data over and over in separate tables.

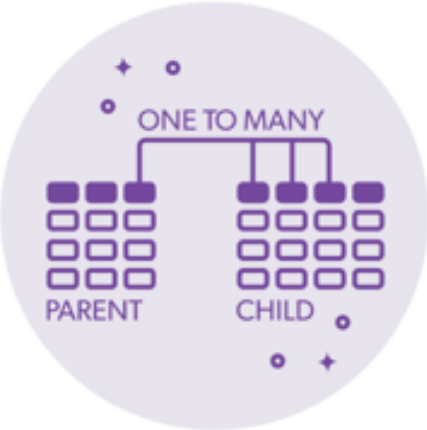
Reduce the risk of data-entry **errors**.

Can **summarize data** from a related table.

Got it | Back | Next

Table-to-table relationships

WHAT HAPPENS ONCE YOU CREATE A RELATIONSHIP



When you create a relationship, you're telling Quick Base to **connect a single record** in one table (called the parent table) **to many records** in the other table (called the child table).

This is called a one-to-many relationship.

In a relationship in Quick Base, the table on the "one" side is called the **parent table** and the table on the "many" side is called the **child table**.

Got it | Back | Next

Table-to-table relationships

ONE-TO-MANY

It's easy to think of real-world examples of one-to-many relationships:

- A business has many locations.
- A project has many tasks.
- One cat may have many kittens.

You get the idea. These are all examples of one-to-many relationships.

Got it | Back | Next

Table-to-table relationships

FOR EXAMPLE

If there is a **one-to-many** relationship between a Companies table and a Contacts table, when your employee enters a new contact, the new contact info can be related to an existing customer.

The company name and related info **come directly** from the Companies table.

For each additional contact, there's **never a need to re-enter** company info. And if the company info changes, it's **automatically updated** in all the related contact records.

Got it | Back | Next

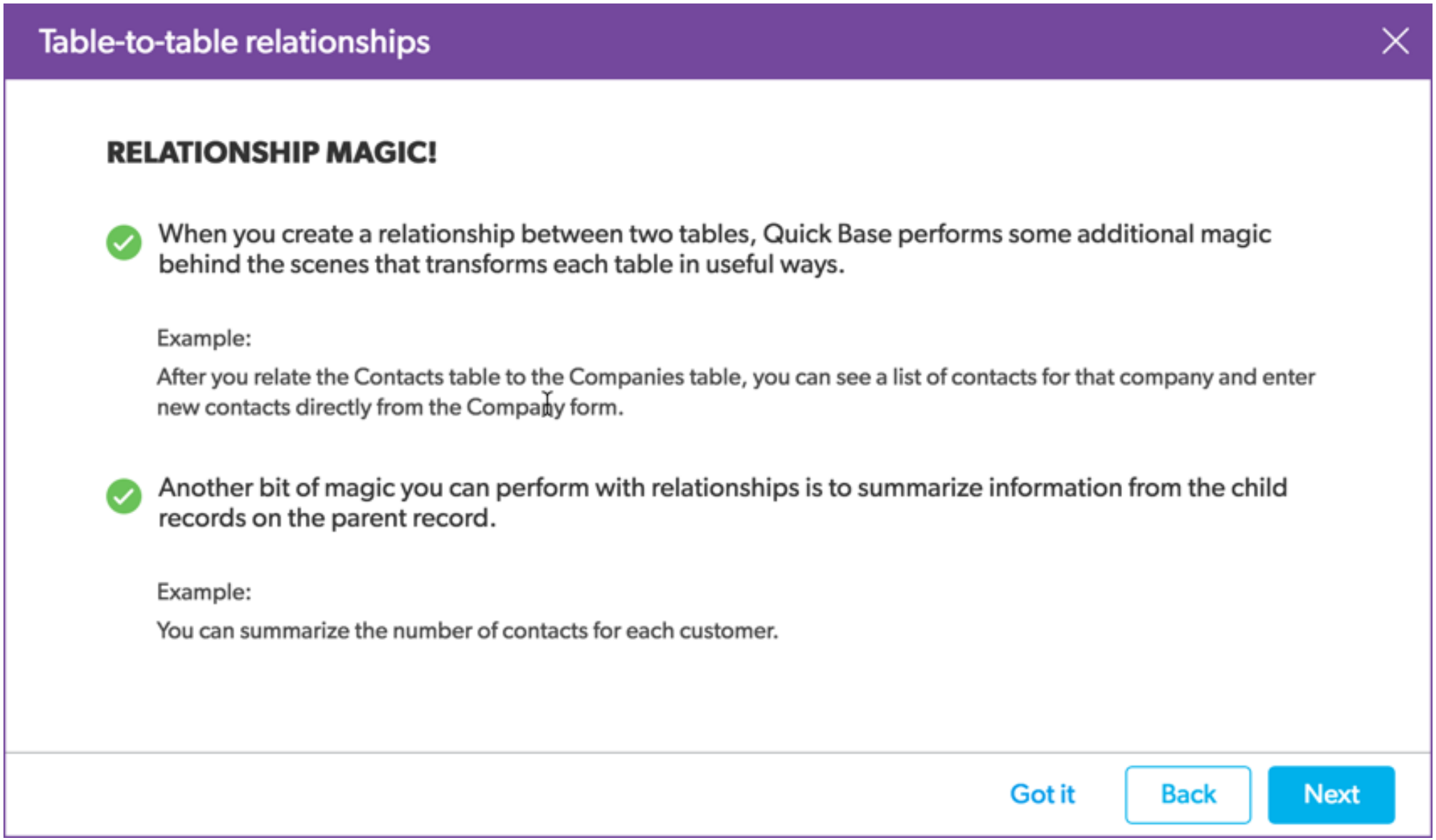


Table-to-table relationships
RELATIONSHIP MAGIC!
When you create a relationship between two tables, Quick Base performs some additional magic behind the scenes that transforms each table in useful ways.
Example:
After you relate the Contacts table to the Companies table, you can see a list of contacts for that company and enter new contacts directly from the Company form.
Another bit of magic you can perform with relationships is to summarize information from the child records on the parent record.
Example:
You can summarize the number of contacts for each customer.
Got it
Back
Next

Table-to-table relationships

AWESOME POWER!

Let's dig a little deeper to show the awesome power of related tables.

- Keep in mind that while a relationship is created between two tables, it's actually the **records in the tables** that form the relationship. So, for example, each Company record has many Contact records.
- Also note that a parent table can have **more than one child** table. A company can have many contacts, but also many activities, and many documents.
- A table can be on **both sides** of a relationship. For example, a country can be the parent to many states, and a state can be the parent to many cities. In this case, a States table is both a parent table and a child table.

Got it | Back | Next

Table-to-table relationships

ARE YOU FEELING THE POWER?

As you learn more about table-to-table relationships, you'll start to see the remarkable power of this concept and how it can help you create apps that can **facilitate complex workflows and business logic**.

Back Got it