In Supabase, the relationship between tables follows the principles of relational databases (similar to PostgreSQL, which is what Supabase uses under the hood). Here's a simple summary of how relations work between tables:

1. **Tables**: Each table represents an entity, like `users`, `courses`, or `payments`. A table is made up of columns (fields) and rows (records).

2. **Primary Key**: Each table usually has a **primary key**, which is a unique identifier for each record in the table (often named `id`). For example, in a `users` table, `user_id` might be the primary key.

3. **Foreign Key**: A **foreign key** is a column in one table that links to a **primary key** in another table, establishing a relationship between the two tables. For example, in a `courses` table, you might have a `teacher_id` column that refers to the `id` column in a `users` table (the teacher). This way, the `courses` table knows which teacher is associated with each course.

4. **One-to-Many Relationship**: This is the most common type of relationship. One record in a table can be linked to many records in another table. For example, one `user` can be linked to multiple `orders` in an `orders` table through a foreign key.

5. **Many-to-Many Relationship**: Sometimes, you have relationships where many records in one table can relate to many records in another table. For example, a `courses` table and a `students` table might have a many-to-many relationship, where each student can enroll in many courses, and each course can have many students. This is usually handled by creating a **join table** like `enrollments`, with foreign keys pointing to both the `courses` and `students` tables.

6. **Cascade Behavior**: When defining foreign key relationships, you can specify what should happen if a related record is deleted or updated. For example, you can set a rule to **cascade delete** related records (e.g., if a user is deleted, all their related orders get deleted too).

In Supabase, you can manage these relationships through its visual interface or SQL queries, and it follows the same principles as PostgreSQL, providing powerful relational data management.