Mapping Database Objects

Torben Jensen
Developer/Cloud Architect

Overview

SQL Server Programability

Mapping queries to views

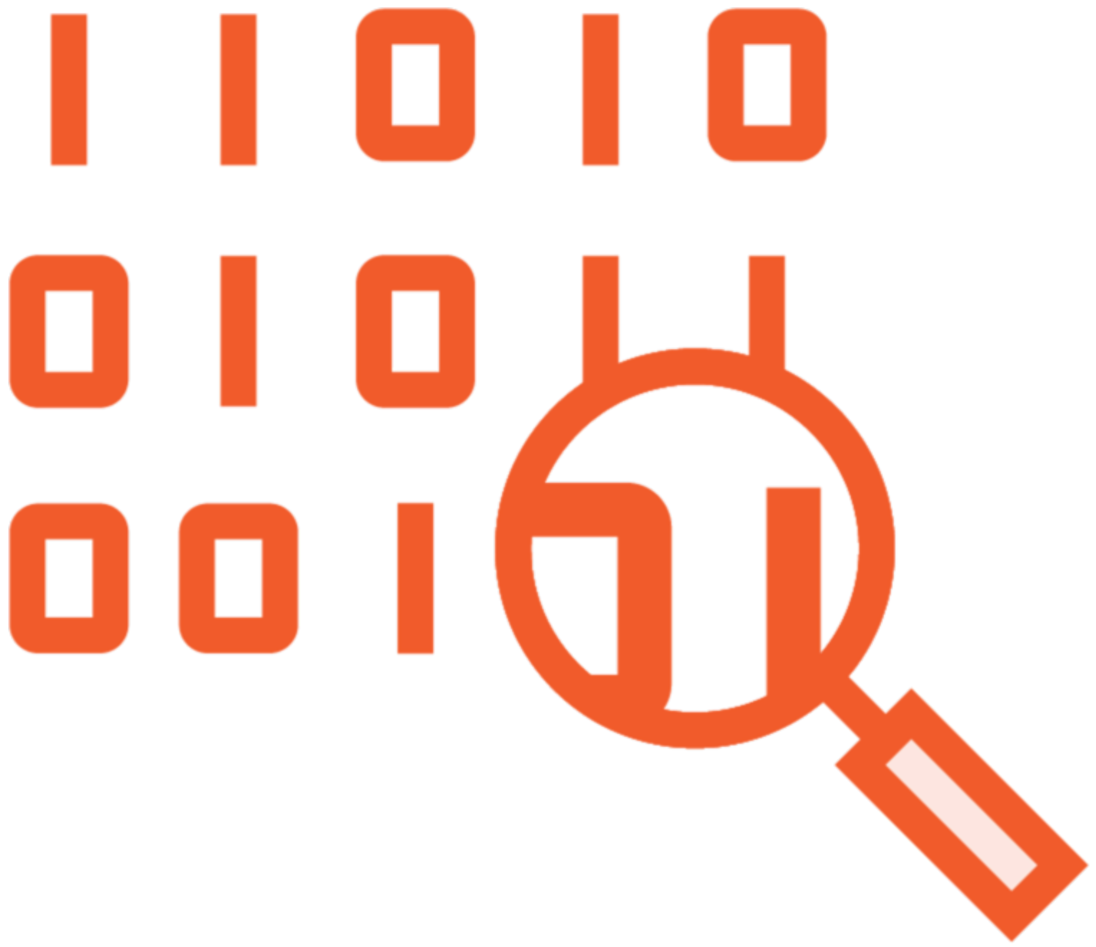
Adding indexes to tables

Know the Fundamentals

Entity Framework Core 6 Fundamentals

Julie Lerman

Stored Procedures



Consist of one or more SQL statements

Can be reused

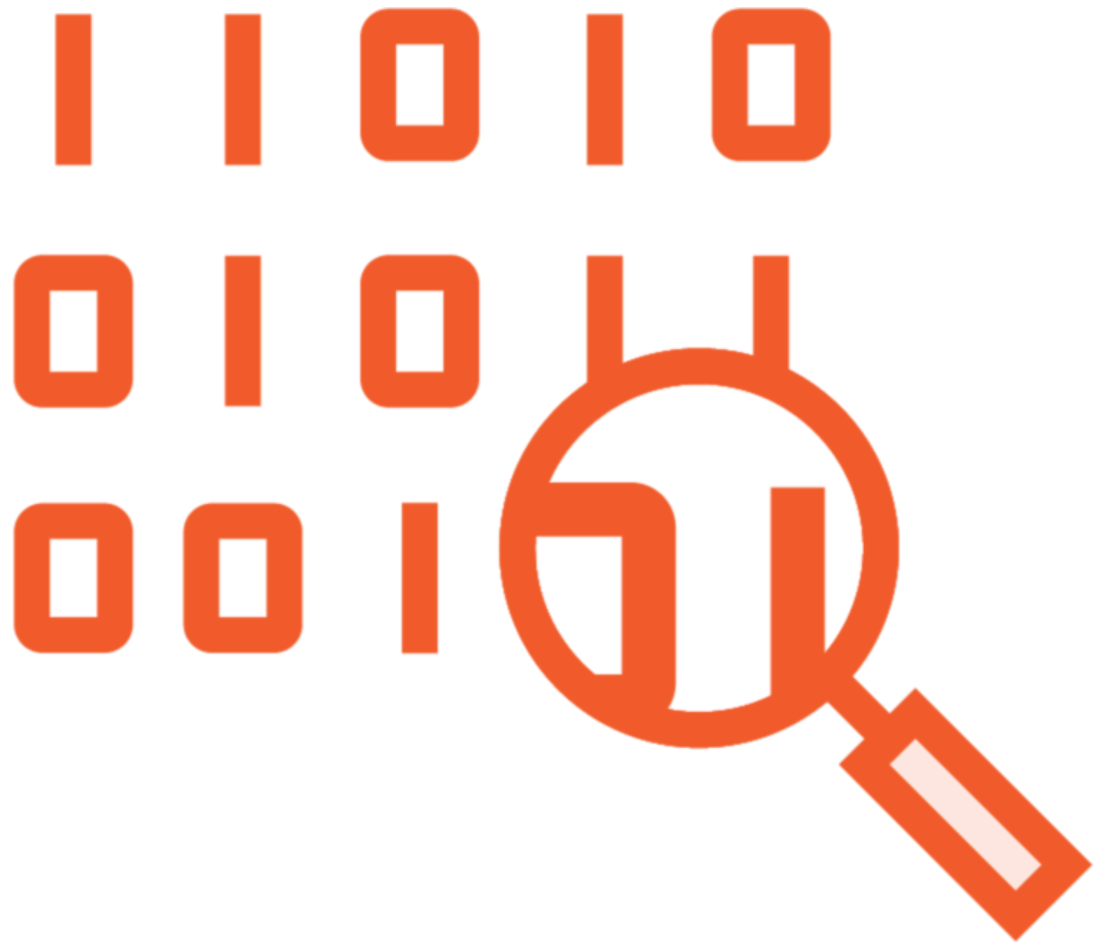
Similar to C# methods

Accept input parameters

Can return several output parameters

Can call other stored procedures

Stored Procedures



Are used for several reasons

Often used because of policy

Stored Procedures

Reduce network traffic

Avoid SQL injection

Control permissions

Code reuse

Customer Orders Query

```sql
SELECT * FROM Orders
```

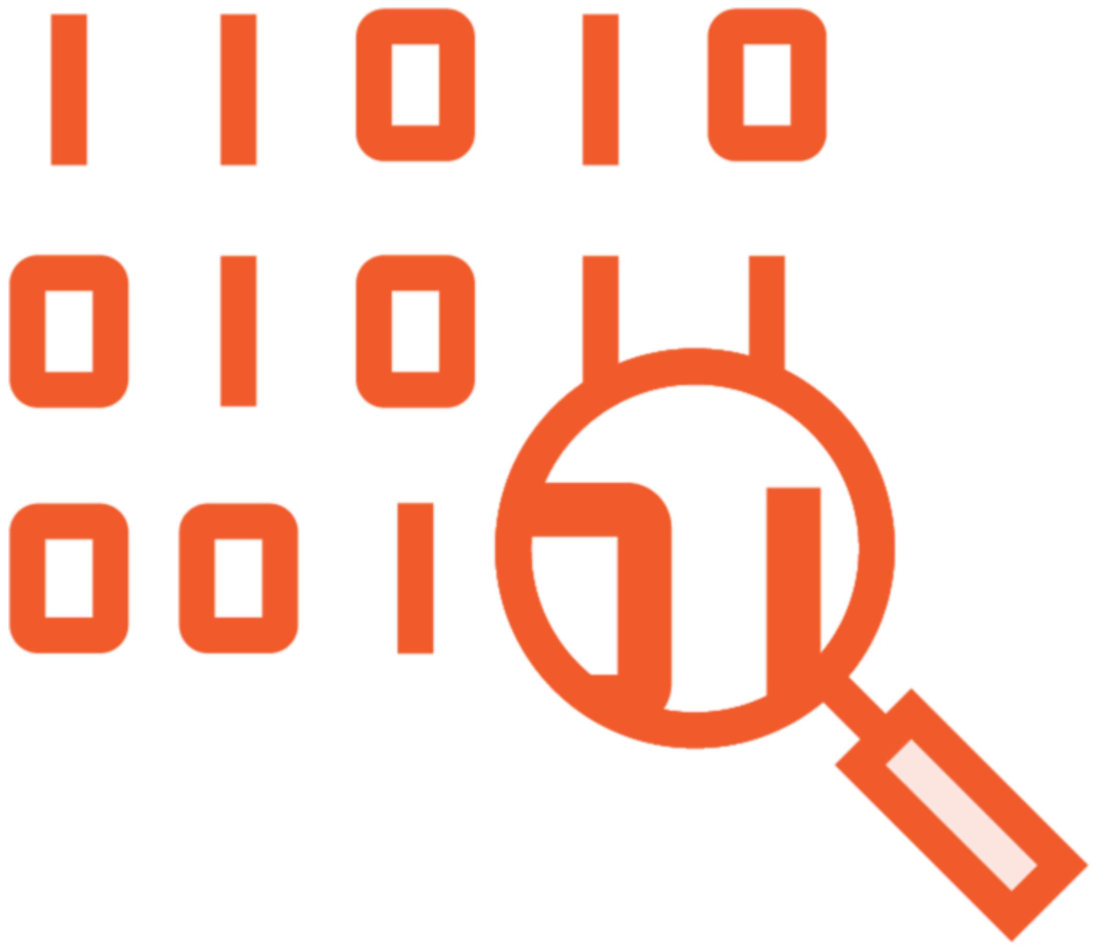
Demo

Implement stored procedure

FromSqlInterpolated

Add parameters to procedure

What Did We Just Do?



Maping stored procedure

Using code first migrations

Execute using FromSqlInterpolated

Views

Virtual tables based on the result of a SQL query

Can be used for some of the same things as stored procedures

Views

Represents query as a single, virtual table

Can be used as an ordinary table within other queries

Query limitations

Customer Orders Query

```sql
SELECT
     Orders.Id AS OrderId,
     Orders.Name AS OrderName,
     Customers.FirstName AS CustomerId,
     CONCAT(Customers.FirstName,' ', Customers.LastName) AS CustomerName,
     Items.Description AS ItemDescription,
     Items.UnitPrice,
     Items.UnitPriceAfterVAT
FROM ItemOrder
JOIN Items ON ItemOrder.ItemsId = Items.Id
JOIN Orders ON ItemOrder.OrdersId = Orders.Id
JOIN Customers ON Orders.CustomerId = Customers.Id
```

Demo

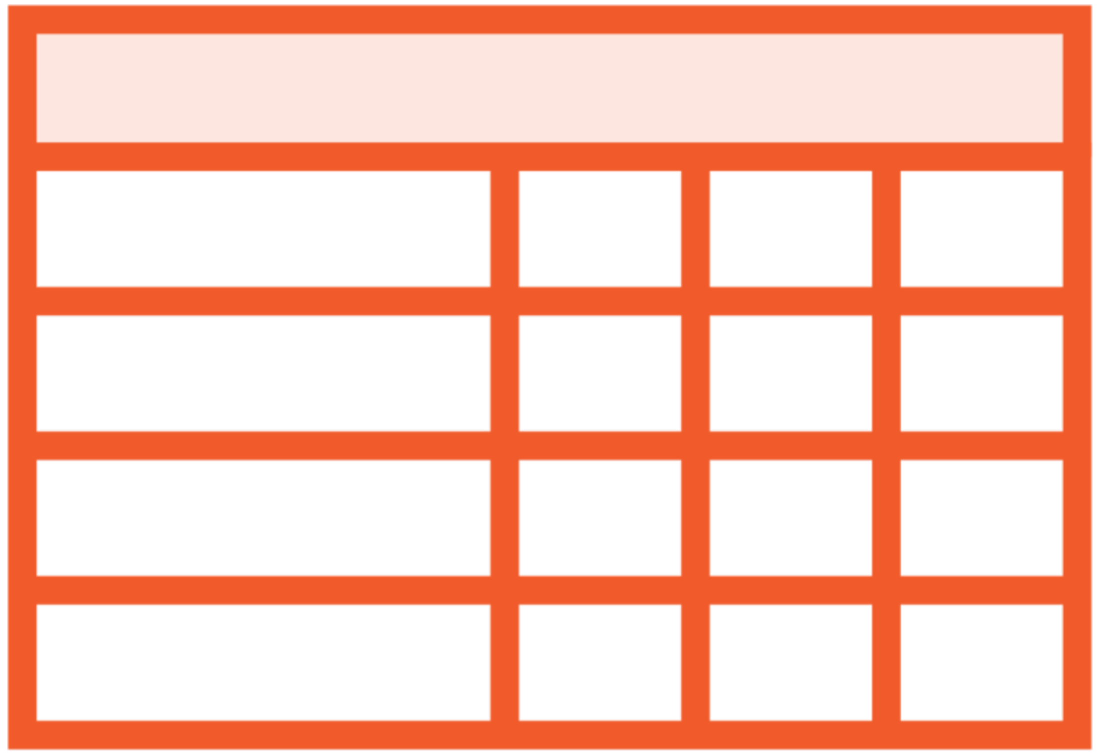
Create database view

Create code first migration

Create entity for view

Configure in database context

What Did We Just Do?



Maping views

ToView

Compose further queries on view

User-Defined Functions

Accept parameters

Perform calculations

Retrieve data

Scalar and Table-Valued Functions

Scalar

Always return a single value

Are like parameterized views

Table-valued

Return table with result of query

Inline and multi-statement

Inline TVFs are like parameterized views

Multi-statement TVFs are like tables

Inline and Multi-Statement TVFs

Scalar	Table-valued
Return query resultset	Can be more flexible
Implicitly defined by a query	We must explicitly define a table structure
	Can consist of one or more statements

Advantages of User-Defined Functions

Can be called inside queries

Facilitate code reuse

Similar to methods in C#

Limitations Compared to Stored Procedures

Always return a value

Scalar or result set

Can only return one value

Cannot modify database state

Limited error handling

Demo

Explore user-defined functions

Execute using Server Explorer

Inline table-valued function

Multi-statement table-valued function

Scalar-valued function

Demo

Mapping user-defined functions

Piecing the Puzzle

Define fields in database context for retrieving data

IQueryables of entities for table-valued functions

Method returning int for scalar function

Map in OnModelCreating

Using HasDbFunction method

Caution

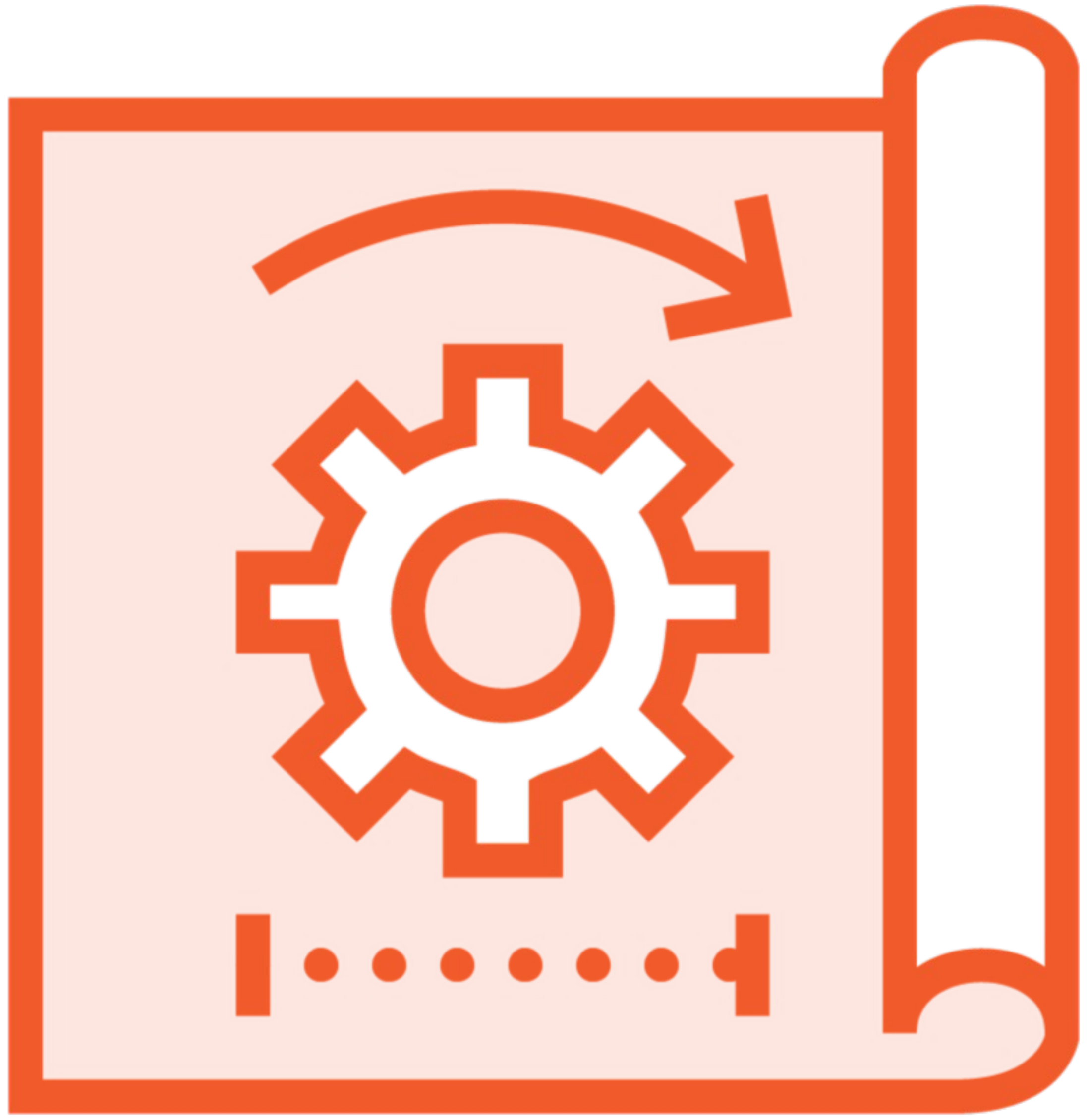
Considder performance

Multi-statement TVFs are not optimized in relation to surrounding query

Incur context switching costs

Scalar functions now perform much better thanks to inline expansion

What Did We Just Do?



Maping user-defined functions

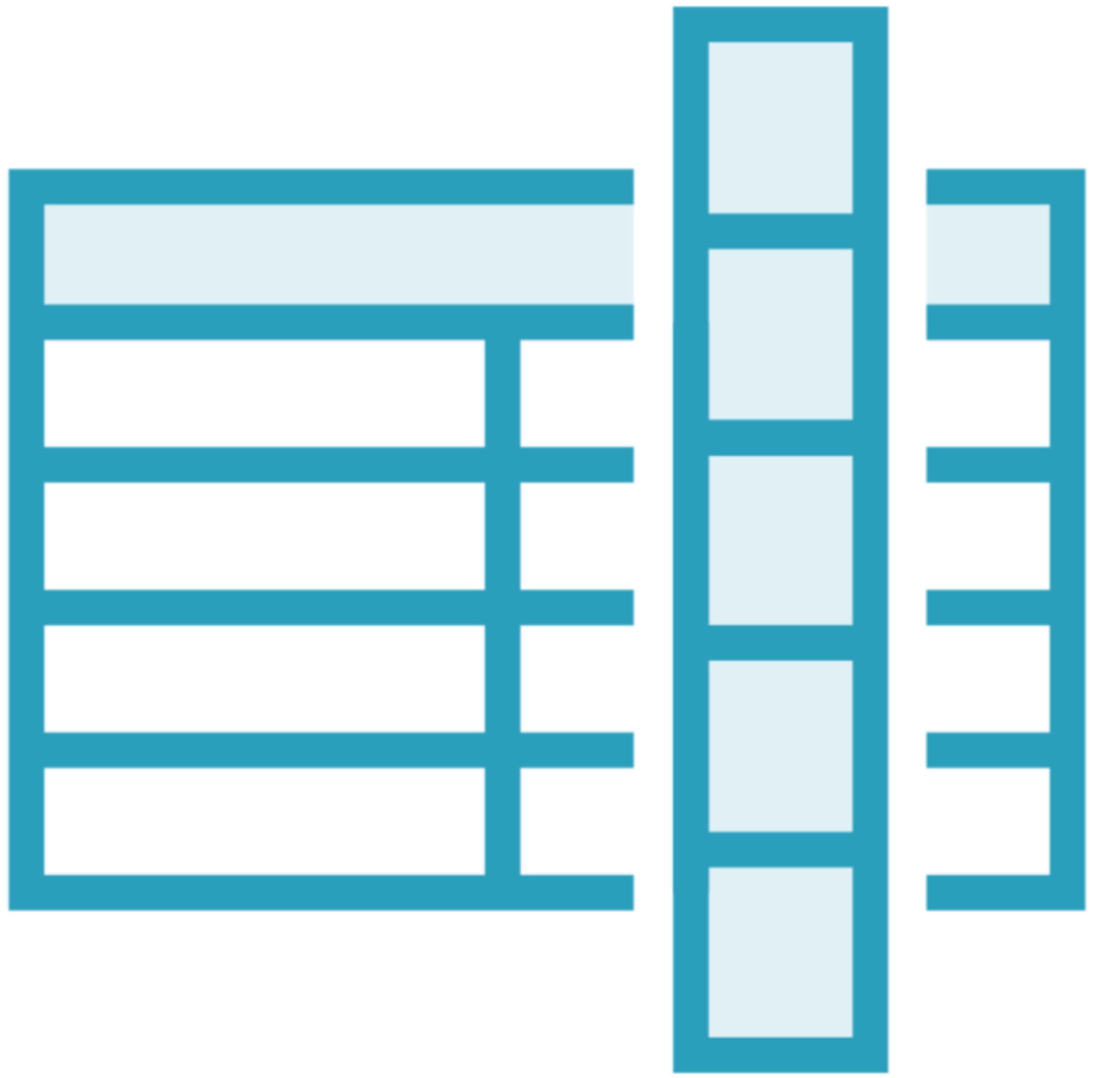
Inline table-valued functions

Multi-statement table-valued functions

Scalar-valued functions

HasDbFunction

Indexes



Make column-based data lookup more efficient

Indexes on primary and foreign keys

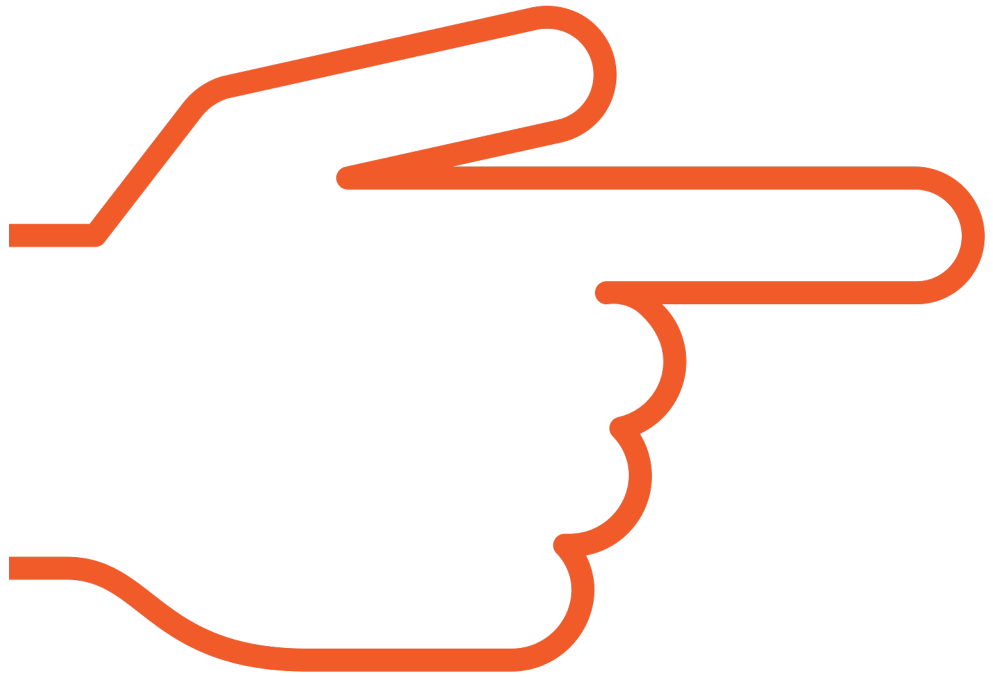
Demo

Mapping indexes

Create indexes on Order entity

Create composite index on Customer entity

What Did We Just Do?



Maping indexes

HasIndex

Composite index

Summary

Mapping database objects

Integrate into EF Core datamodel

Might require reverse engineering