





PickedQuantity

LastEditečdBv

LastEditedWhe

PickingCompletedW

 $F = \{ EF Core, Dapper, NHibernate \}$ $W = \{Q1, Q2, Q3\}$ z_{Q1} = 1 $z_{Q2} = 1$ MEM = 153,600 kB (15 MB)N = 1



ILP Solver

Runtime (ms)	Dapper	EFCore	NHibernate
Q1	30	21	21
Q2	748	745	743
Q3	182	557	2,474
SUM	960	1,323	3,238

Memory (kB)	Dapper	EFCore	NHibernate
Q1	989	3,048	819
Q2	1,169	3,472	979
Q3	40,924	144,094	175,866
SUM	43,082	150,614	177,664



runtime: 960 ms Dapper (C#) memory: 43,082 kB public class Customer { public int CustomerID { get: set: } public required string CustomerName { get; set; } public class CustomerTransaction { public int CustomerTransactionID { get; set; } public int CustomerID { get; set; }

public class OrderLines { public int OrderLineID { get; set; } public int OrderID { get; set; }

public int StockItemID { get; set; } public required string Description { get; set; } public int PackageTypeID { get; set; } public int Quantity { get; set; }

public decimal? UnitPrice { get; set; } public decimal TaxRate { get; set; } public int PickedQuantity { get; set; }

public DateTime? PickingCompletedWhen { get; set; } public int LastEditedBy { get; set; } public DateTime LastEditedWhen { get; set; }

var from = **new** DateTime(2014, 12, 20): var to = new DateTime(2014, 12, 31):

var orderLines = connection.Query<OrderLine>(

var customers = new Dictionary<int, Customer>();

connection. Query < Customer, Customer Transaction, Customer > ("""SELECT c.*, ct.* FROM Customers c LEFT JOIN CustomerTransactions ct

ORDER BY c.CustomerID""" (customer, transaction) => { if (!customers.TryGetValue(customer.CustomerID, out var c)) {

customers.Add(customer.CustomerID, c); } if (transaction != null) {

existing.Transactions.Add(transaction); } return existing; },

return customers. Values. ToList():