

Performance Optimization

Dmitrii Naumov
ISV Solutions Architect
Acumatica

Agenda

Intro

- Optimization as an Equilibrium
- Detecting the Bottleneck

Optimization of Reading Data

- Data View Select Delegates
- Implicit Joins and Subselects
- Reading Huge Amounts of Data

Optimization of Data Processing

- Reusing and Recycling
- Redesigning of Heavy Processing
- Saving Data



Summary

```

    .AddScreenConfigurationFor(screen =>
    {
        screen
        .StateIdentifierIs<status>()
        .AddDefaultFlow(flow =>
        {
            flow
            .WithFlowStates(fss =>
            {
                fss.Add(initialState, flowState => flowState);
                fss.Add<State.hold>(flowState => ...);
                fss.Add<State.open>(flowState => ...);
                fss.Add<State.confirmed>(flowState => ...);
                fss.Add<State.partiallyInvoiced>(flowState => ...);
                fss.Add<State.invoiced>(flowState => ...);
                fss.Add<State.completed>(flowState => ...);
            })
            .WithTransitions(transitions =>
            {
                transitions.AddGroupFrom(initialState, ts =>
                {
                    transitions.AddGroupFrom<State.hold>(ts =>
                    {
                        ts.Add(t => t.To<State.hold>().IsTriggered);
                        ts.Add(t => t.To<State.confirmed>().IsTriggered);
                    });
                    transitions.AddGroupFrom<State.confirmed>(ts =>
                    {
                        ts.Add(t => t.To<State.open>().IsTriggered);
                        ts.Add(t => t.To<State.invoiced>().IsTriggered);
                        ts.Add(t => t.To<State.partiallyInvoiced>().IsTriggered);
                    });
                    transitions.AddGroupFrom<State.partiallyInvoiced>(ts =>
                    {
                        transitions.AddGroupFrom<State.invoiced>(ts =>
                        {
                            transitions.AddGroupFrom<State.completed>(ts =>
                            {
                                // ...
                            })
                        })
                    })
                })
            })
        })
    })
}

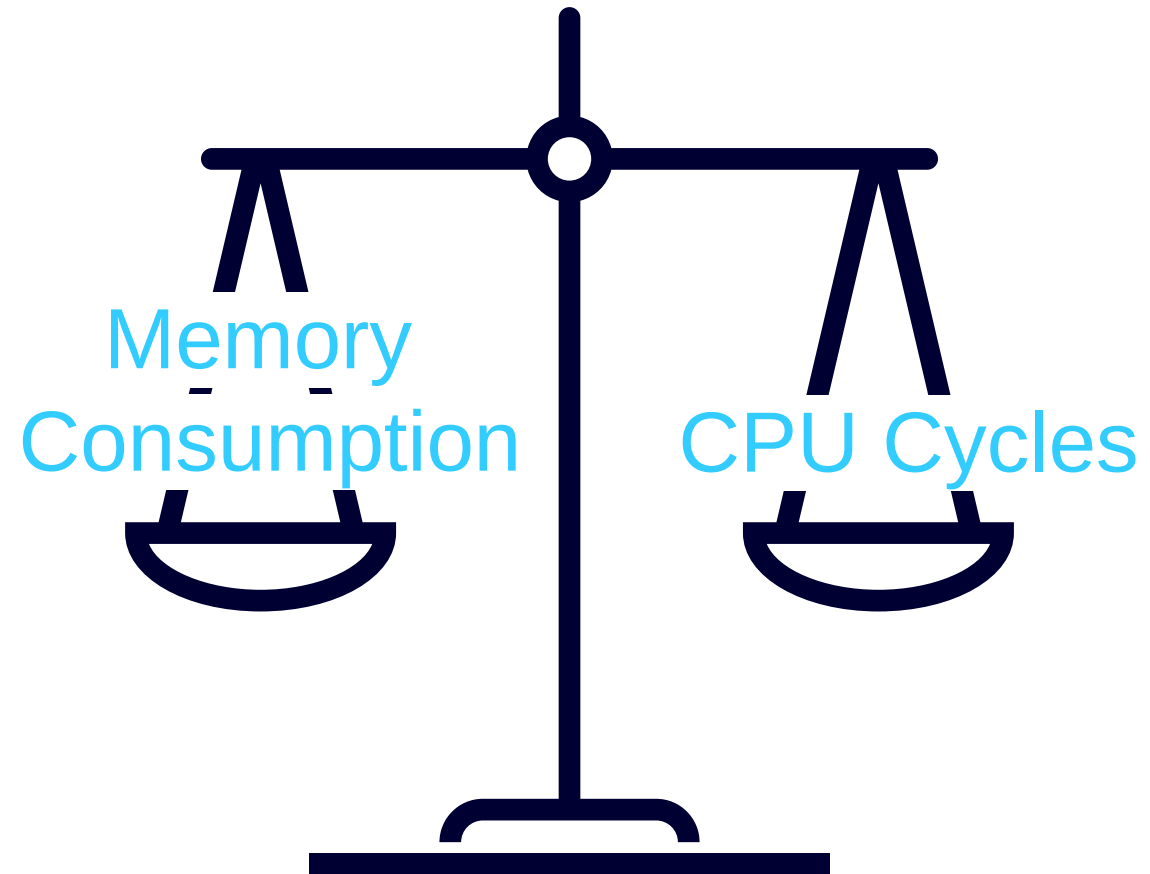
```

Intro: Optimization as an Equilibrium

*Premature optimization is
the root of all evil.*

Donald Knuth

Equilibrium



Equilibrium



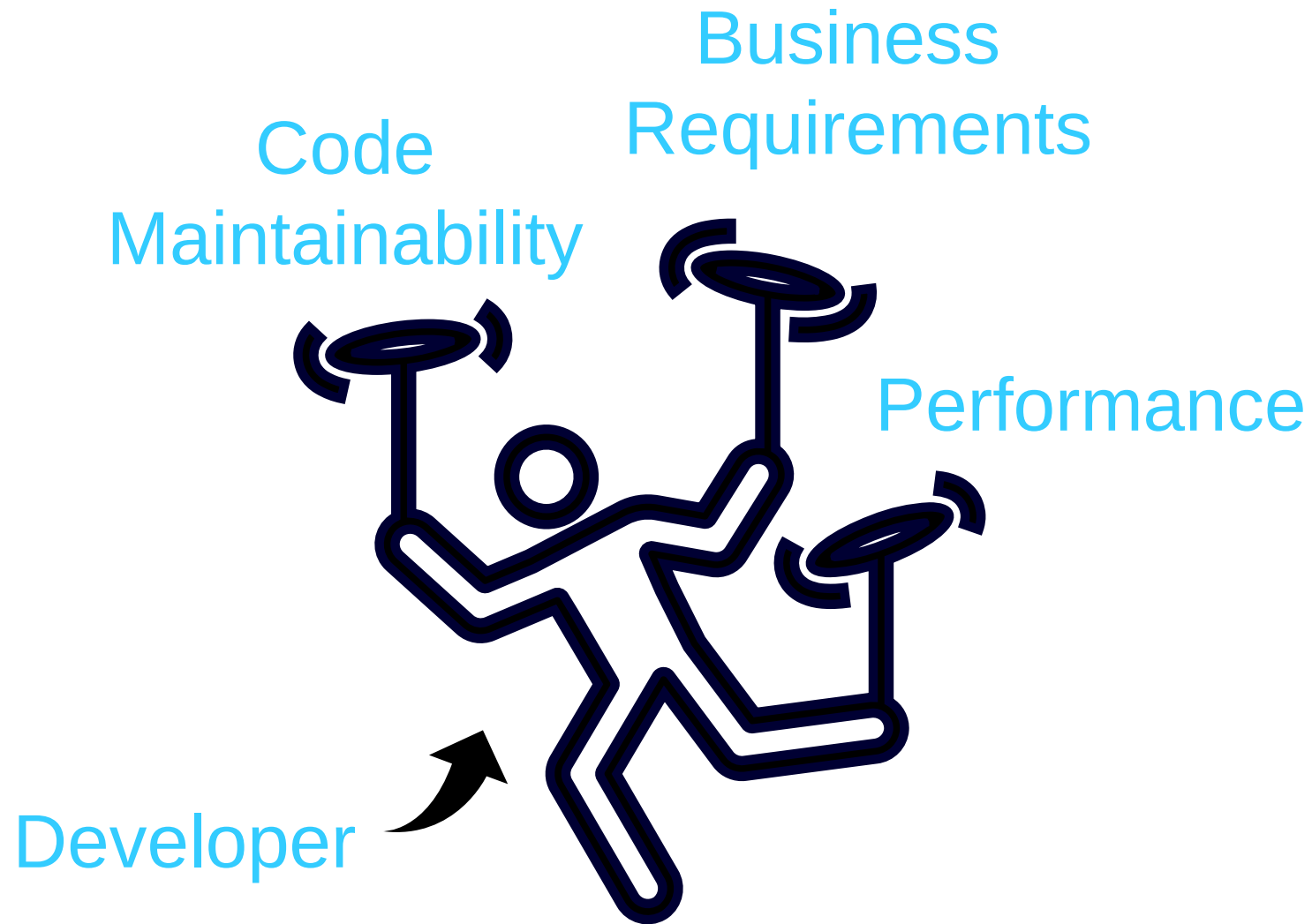
How Optimization Affects Maintainability

- Obscure code
- Hard to catch bugs
- “Leaking” abstraction layers

- “Hello World!”
 - Assembly version

```
1 ; -----
2 ; Writes "Hello, World" to the console using only system calls.
3 ; To assemble and run:
4 ;
5 ;     nasm -f elf hello.asm & ld -m elf_i386 hello.o -o hello
6 ; -----
7
8         global    _start
9
10        section    .text
11 _start:  mov     eax, 4          ; system call number for write
12         mov     ebx, 1          ; file handle 1 is stdout
13         mov     ecx, message    ; address of string to output
14         mov     edx, 13         ; number of bytes
15         int     80h             ; request an interrupt on lib
16 exit:    mov     eax, 1          ; syscam call number for exit
17         mov     ebx, 0          ; return 0 status on exit - 'l
18         int     80h             ; request an interrupt on lib
19
20        section    .data
21 message: db      "Hello, World", 0Ah    ; note the newline at the end
22
```

Equilibrium



Know Your Enemy

DB SLOWNESS

WRONG
EXPECTATIONS

APPLICATION
SLOWNESS

Know Your Enemy

DB Slowness

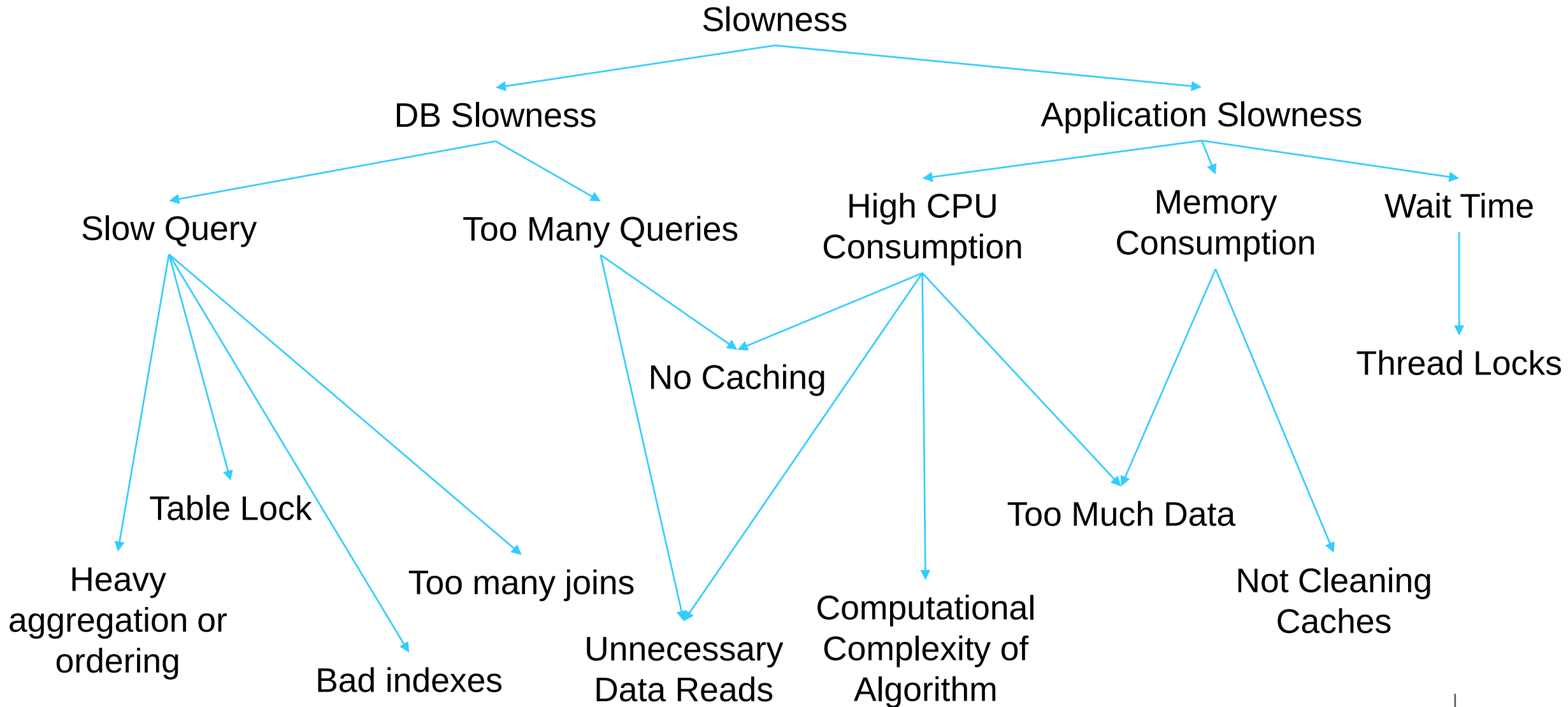
Application Slowness

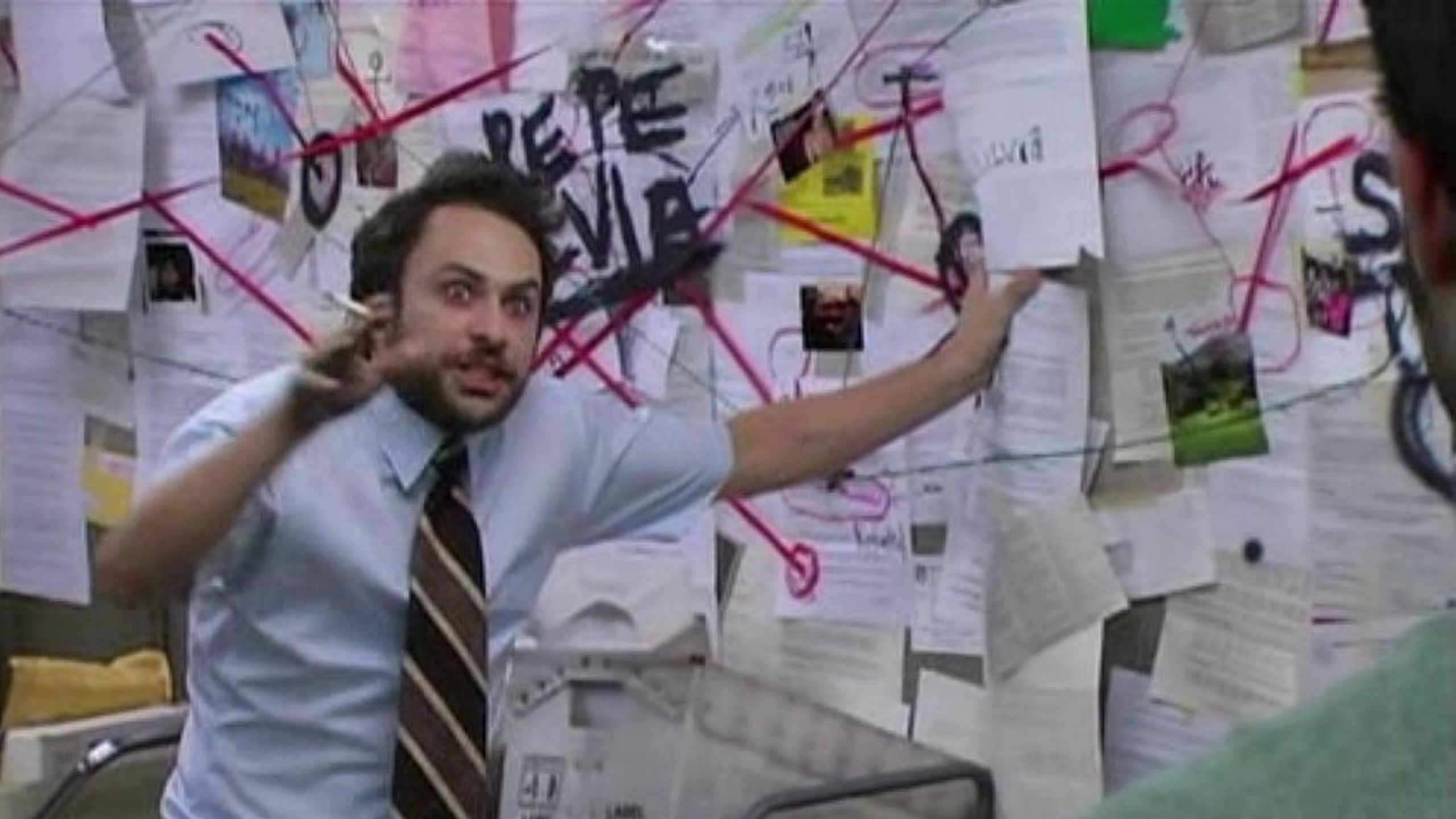
Data Read
Slowness



Data Processing
Slowness

Know Your Enemy





Steps to Identify the Root of the Slowness

Step 1: Expectations (+ License, Hardware)

Step 2: Request Profiler to Categorize the Slowness

Step 3: Additional Measurements or Tests to detect the bottleneck

Step 4: Fix

Step 5: Test

Demo: Profiling Tools

Request Profiler

Request Profiler

CUSTOMIZATION TOOLS ▾

REFRESH RESULTS CLEAR LOG EXPORT IMPORT

☒ Default Logging (Expensive Requests and Requests with Important Exceptions)

REQUEST LOGGING

☒ Log Requests (Apply Filter)

Server Time Threshold:

URL:

SQL Count Threshold:

Username:

SQL LOGGING

☒ Log SQL (Apply Filter)

Row Count Threshold:

Executed by Method:

SQL Time Threshold:

☐ Include Cached SQL Results

EXCEPTION LOGGING

☐ Log Exceptions

EVENT LOGGING

☐ Log Events (Apply Filter)

Log Level:

Warning ▾

Category:

▾

REQUESTS

SQL

EXCEPTIONS

EVENT LOG

↺

+

×

VIEW SQL

VIEW EVENT LOG

OPEN URL

PIN/UNPIN

↔

🗒

All Records ▾

🔍

	Request Start Time	Username	URL	Screen	Request Type	Stat	Command Target	Command Name	Clie Tim	Server Time, ms	SQL Time, ms	Server CPU, ms	SQL Cot	Log SQL Cot	SQL Rov	Exc Cot	Log Exc Cot	Eve Cot	Log Eve Cot	Managu Memor	Mar Mer Byt	Pea Mer Byt	Wa Tim
>	16 Jun 11:16:1	admin	~/pages/ar/ar...	AR402000	Screen		ctl00\$phF\$form	Save		162.55	36.80	125.00	24	24	27	0	0	3	0	789.31	78931	28448	0.7

 **Acumatica**
The Cloud ERP

15

SQL Time

Request Profiler

REFRESH RESULTS CLEAR LOG EXPORT IMPORT

☒ Default Logging (Expensive Requests and Requests with Import)

REQUEST LOGGING

☒ Log Requests (Apply Filter) Server Time Threshold:

SQL Count Threshold:

SQL LOGGING

☒ Log SQL (Apply Filter) Row Count Threshold:

SQL Time Threshold:

REQUESTS SQL EXCEPTIONS EVENT LOG

Request Start Time	Username	URL	Screen
>	16 Jun 11:16:11 admin	~/pages/ar/ar...	AR402000

View SQL

🔄 + ✎ × |<> ⌂

Tables: Note,NoteDoc,ARRegister,ARInvoice,ARPayment

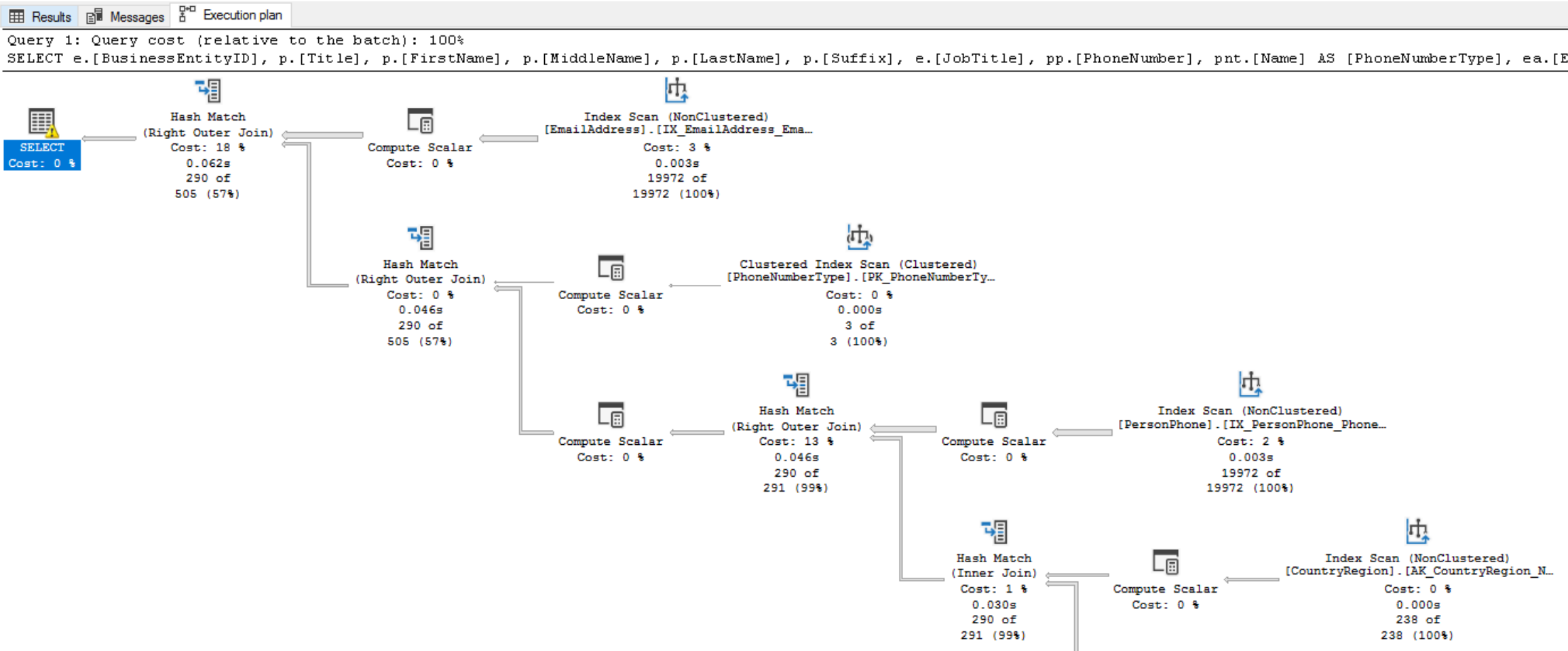
☒ Show Stack Trace

SQL:

```
DECLARE @P1 AS INT = 7017, @P2 AS INT = 16;

SELECT TOP (14) [ARDocumentResult_ARRegister].[DocType] AS [DocType],
[ARDocumentResult_ARRegister].[RefNbr] AS [RefNbr],
[ARDocumentResult_ARInvoice].[InstallmentCntnr] AS [InstallmentCntnr],
[ARDocumentResult_ARRegister].[BranchID] AS [BranchID],
[ARDocumentResult_ARRegister].[CustomerID] AS [CustomerID],
[ARDocumentResult_ARRegister].[DocDate] AS [DocDate],
[ARDocumentResult_ARRegister].[DocDesc] AS [DocDesc],
[ARDocumentResult_ARRegister].[DueDate] AS [DueDate],
[ARDocumentResult_ARRegister].[CuryID] AS [CuryID],
[ARDocumentResult_ARRegister].[CuryInfoID] AS [CuryInfoID],
[ARDocumentResult_ARRegister].[OrigDocType] AS [OrigDocType],
[ARDocumentResult_ARRegister].[OrigRefNbr] AS [OrigRefNbr],
[ARDocumentResult_ARRegister].[Status] AS [Status],
[ARDocumentResult_ARRegister].[TranPeriodID] AS [TranPeriodID],
[ARDocumentResult_ARRegister].[FinPeriodID] AS [FinPeriodID],
[ARDocumentResult_ARRegister].[ClosedFinPeriodID] AS [ClosedFinPeriodID],
[ARDocumentResult_ARRegister].[ClosedTranPeriodID] AS [ClosedTranPeriodID],
[ARDocumentResult_ARRegister].[NoteID] AS [NoteID],
(SELECT TOP (1) [Note].[NoteText]
FROM [Note] AS [Note]
WHERE ([Note].[CompanyID] IN (1, 2)
AND 8 = SUBSTRING([Note].[CompanyMask], 1, 1) & 8)
AND ([Note].[CompanyID] IN (1, 2)
AND 8 = SUBSTRING([Note].[CompanyMask], 1, 1) & 8)
AND [Note].[NoteID] = [ARDocumentResult_ARRegister].[NoteID]) AS [NoteText],
(SELECT TOP (1) COUNT(*)
FROM [NoteDoc] AS [NoteDoc]
WHERE ([NoteDoc].[CompanyID] IN (1, 2)
AND 8 = SUBSTRING([NoteDoc].[CompanyMask], 1, 1) & 8)
AND ([NoteDoc].[CompanyID] IN (1, 2)
AND 8 = SUBSTRING([NoteDoc].[CompanyMask], 1, 1) & 8)) AS [Count]
```


DB Query Optimization



Application Time

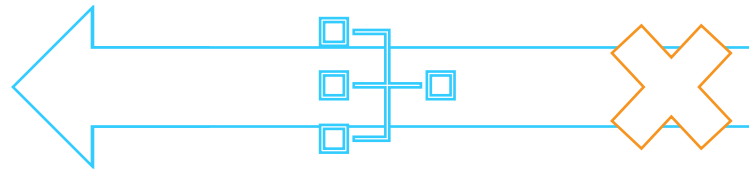
View SQL									
↺ + ✎ × ⏮ ⏭									
	Order	Statement ID	Tables	Row Count	Start Time	SQL Time, ms	Parameters	From Cache	
	1	15283370	@@DBTS, IDENT_CURRENT('WatchDog') O...		1.65	0.32		☐	
>	1	15283370	@@DBTS, IDENT_CURRENT('WatchDog') O...		5.97	0.35		☐	
	1	15283370	@@DBTS, IDENT_CURRENT('WatchDog') O...		6.71	0.26		☐	
	2	13473288	@@DBTS, IDENT_CURRENT('WatchDog') O...		18.60	0.30		☐	
	3	18576284	Users		19.34	0.30	'b5344897-037e-4d58-b5c3-1bdfd0f4...	☐	
	2	13473288	@@DBTS, IDENT_CURRENT('WatchDog') O...		27.75	0.33		☐	
	3	18576284	Users		28.52	0.30	'b5344897-037e-4d58-b5c3-1bdfd0f4...	☐	
	4	15257443	ARRegister,ARInvoice,ARPayment	1	47.43	12.79	7017, 16, 'INV', 'AR010699'	☐	
	5	63461856	MUIFavoriteScreen		62.57	0.32	'69a932ad-b141-4a0c-b5f0-b605a72e...	☐	
	6	19520057	ARRegister,ARInvoice,ARPayment	1	71.29	3.66	7017, 16	☐	
	7	10846385	ARSetup	1	81.94	0.38		☐	
	8	15279478	Company	1	83.00	0.26		☐	
	2	13473288	@@DBTS, IDENT_CURRENT('WatchDog') O...		84.03	0.25		☐	
	9	16144216	BAccount,Organization,Branch	1	87.84	0.37	16	☐	
	10	13297781	BAccount	1	90.52	0.33	'PRODWHOLE '	☐	
	11	14894822	FinPeriod	1	92.17	0.56	5, '2022-06-16T00:00:00.000'	☐	
	12	88529536	FeaturesSet	1	98.09	0.45		☐	
	13	62304400	Organization	1	99.37	0.30		☐	
	14	26484744	CuryARHistory,BAccount	1	101.51	1.43	4899, 4899	☐	

JetBrains DotTrace

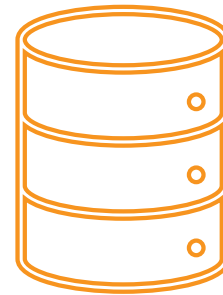
The screenshot displays the JetBrains DotTrace Performance Viewer interface. The top menu bar includes File, Edit, View, and Help. Below the menu, the 'All Calls' tab is active, showing a call stack for 'Thread #23' with a total duration of 7,001 ms. The call stack is a tree view where each node represents a method call, with its percentage of total time, duration, and the calling method. The root of the stack is 'ProcessRequestNotificationHelper' (10.78%, 755 ms), which calls 'ProcessRequestNotification' (10.69%, 748 ms). This method then calls 'ProcessRequest' (10.23%, 716 ms), which in turn calls 'ProcessRequestInternal' (5.31%, 371 ms). The stack continues with 'RenderControlsData' (5.31%, 371 ms), 'GetCallbackResult' (5.31%, 371 ms), 'PerformSelect' (5.12%, 359 ms), 'Select' (3.76%, 263 ms), 'ExecuteSelect' (3.76%, 263 ms), 'InvokeDelegate' (3.76%, 263 ms), 'documents' (3.76%, 263 ms), 'SelectDetails' (3.57%, 250 ms), 'GetResult' (3.30%, 231 ms), 'MoveNext' (0.17%, 12 ms), 'CreateResult' (0.10%, 7 ms), 'PXFieldScope..ctor' (0.19%, 13 ms), 'OnDataSourceViewSelect' (1.19%, 83 ms), 'SynchronizeColsState' (0.18%, 12 ms), 'GetClientData' (0.18%, 13 ms), 'OnInit' (2.09%, 146 ms), 'OnPreLoad' (1.24%, 87 ms), 'Page_InitComplete' (0.77%, 54 ms), 'Page_Init' (0.46%, 32 ms), 'Page_Load' (0.18%, 13 ms), 'FrameworkInitialize' (0.10%, 7 ms), 'OnPage_PreLoad' (0.09%, 6 ms), 'SetAndReleaseItemExclusive' (0.13%, 9 ms), 'BeginFinalWork' (0.11%, 8 ms), and 'BeginEvent' (0.10%, 7 ms). The bottom-most call is 'ProcessRequestNotificationPrivate' (0.09%, 7 ms). The interface also includes a 'Threads (user/all): 11/25' filter and a 'Show system threads' checkbox.

Data View Select Delegates

Data View Delegates



Data View Delegate



UI Pagination, Ordering and Filtering

Journal Transactions

CUSTOMIZATION ▾ TOO

Module: All ▾Status: All ▾Ledger: All ▾Post Period: All ▾

Y

...

	Module	Batch Number	Status	Ledger	Transaction Date	Post Period	Description	Control Total	Currency	Created On
>	GL	GL001849	Balanced	ACTUAL	5/17/2022	05-2022	fgtdfd	0.00	USD	5/17/2022
	GL	GL001848	Balanced	ACTUAL	5/17/2022	05-2022	sgsdfg	0.00	USD	5/17/2022
	GL	GL001847	Balanced	ACTUAL	3/29/2022	03-2022	gdgdf	10.00	USD	3/29/2022
	CM	CM000280	Posted	ACTUAL	3/1/2022	03-2022	Revalue Euro AR February 2022	0.00	EUR	2/3/2022
	GL	GL001846	Posted	EUACTUAL	2/28/2022	02-2022	Consolidation created from 'European Entity'	1,482,306.93	USD	2/4/2022
	AR	AR010809	Posted	ACTUAL	2/28/2022	02-2022	Contract Renewal CT00000683: Software ...	2,800.00	USD	2/4/2022
	AR	AR010808	Posted	ACTUAL	2/28/2022	02-2022	Contract Renewal 63ACTSTAFF: Active St...	7,000.00	USD	2/4/2022
	AR	AR010807	Posted	ACTUAL	2/28/2022	02-2022	Contract Renewal 40SOFTSAAS: SaaS co...	7,840.00	USD	2/4/2022
	AR	AR010806	Posted	ACTUAL	2/28/2022	02-2022	Contract Renewal 30SOFTSAAS: SaaS co...	7,000.00	USD	2/4/2022
	AR	AR010805	Posted	ACTUAL	2/28/2022	02-2022	Contract Renewal 10SOFTLIC : Software c...	2,800.00	USD	2/4/2022
	DR	GL001845	Posted	ACTUAL	2/28/2022	02-2022		35,224.94	USD	2/3/2022
	FA	FA000128	Posted	ACTUAL	2/28/2022	02-2022		9,266.66	USD	2/3/2022
	CA	CA000241	Posted	ACTUAL	2/28/2022	02-2022	Interest Earned February 2022	33,000.00	USD	2/3/2022
	GL	GL001844	Posted	ACTUAL	2/28/2022	02-2022	Allocate salary expenses by headcount	362,902.66	USD	2/3/2022
	GL	GL001843	Posted	ACTUAL	2/28/2022	02-2022	Allocate supply expenses by headcount	7,452.74	USD	2/3/2022
	GL	GL001842	Posted	ACTUAL	2/28/2022	02-2022	Allocate utility expenses by headcount	21,203.02	USD	2/3/2022
	GL	GL001841	Posted	ACTUAL	2/28/2022	02-2022	Rental expense to branches	44,330.00	USD	2/3/2022
	GL	GL001839	Posted	ACTUAL	2/28/2022	02-2022	Record monthly advertising expense	114,000.00	USD	2/3/2022
	GL	GL001837	Posted	ACTUAL	2/28/2022	02-2022	Monthly payroll expense - 2022	243,900.00	USD	2/3/2022
	GL	GL001835	Scheduled	ACTUAL	2/28/2022	02-2022	Record monthly advertising expense	114,000.00	USD	2/3/2022
	AR	AR010801	Posted	ACTUAL	2/28/2022	02-2022	Overdue Charge	14,407.06	USD	2/3/2022

1-21 of 25717 records

1

of 1225 pages

Data View Select Delegates

```
protected virtual IEnumerable ardocumentlist()
{
    PXSelectBase<BalancedARDocument> cmd =
        new PXSelectJoinGroupBy<BalancedARDocument,
            ...
            OrderBy<Asc<BalancedARDocument.docType, //Set necessary sorting fields: use the key fields
            Asc<BalancedARDocument.refNbr>>>> //Set necessary sorting fields: use the key fields
            (this);

    PXDelegateResult delegResult = new PXDelegateResult
    {
        IsResultFiltered = true,
        IsResultTruncated = true,
        IsResultSorted = true
    }

    foreach (PXResult<BalancedARDocument> res_record in cmd.SelectWithContext())
    {
        // add the code to process res_record
        delegResult.Add(res_record);
    }
    return delegResult;
}
```

Implicit Joins & Subselects

BQL to SQL

PXSelectJoin<

ARInvoice,

InnerJoin<

Customer,

On<ARInvoice.customerID

Equal<Customer.bAccto

Where<ARInvoice.lastFinCh

LessEqual<Cu

SELECT FROM

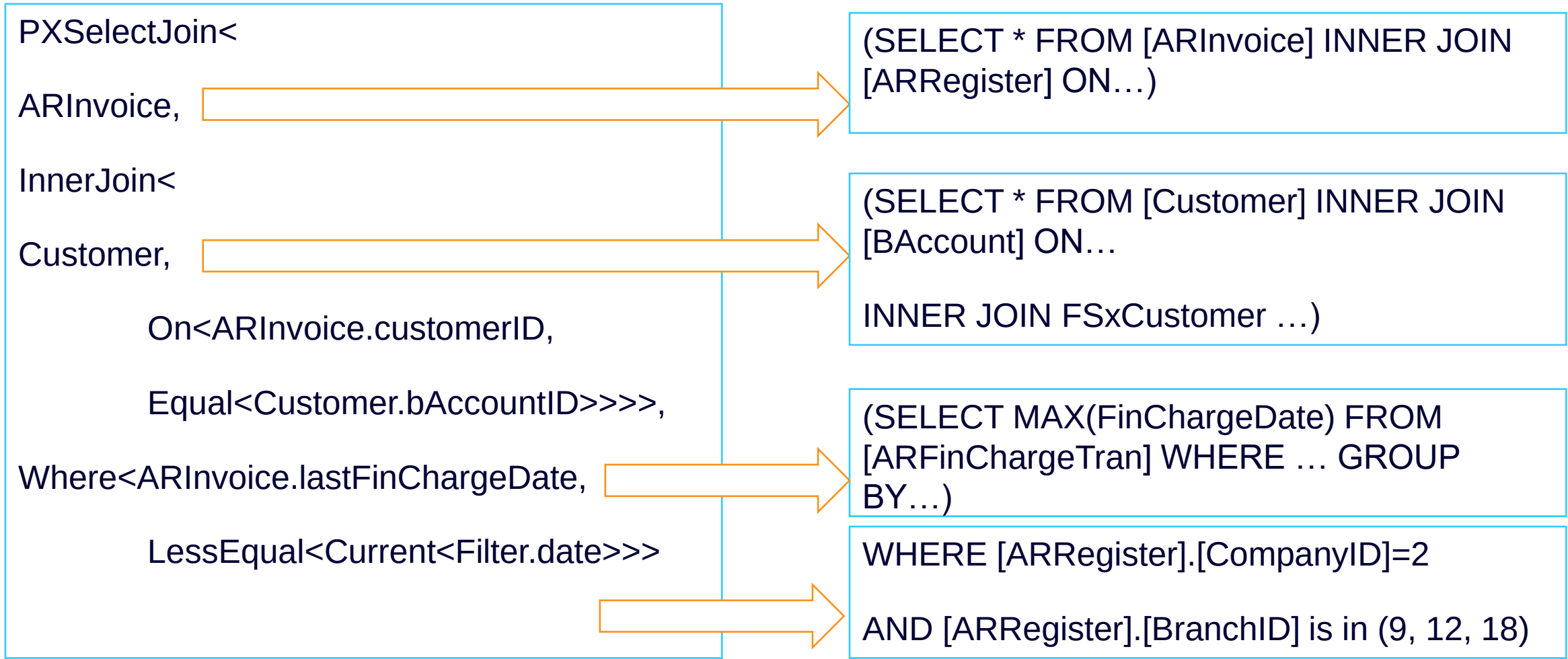
ON [ARInvoice].[CustomerID]

= [Customer].[BAccountID]

[ARInvoice].[LastFinChargeDate]

WRONG!

BQL to SQL



Using JoinSingleTable

PXSelectJoin<ARRegister,
InnerJoin<Customer, On<...>, ...>



SELECT * FROM ARRegister
INNER JOIN

(SELECT * FROM [Customer] INNER JOIN
[BAccount] ON...) ON ...

PXSelectJoin<ARInvoice,
InnerJoinSingleTable<Customer, On<...>, ...>



SELECT * FROM ARRegister
INNER JOIN [Customer] ON ...

Avoiding Heavy attributes

Field Attributes:

- PXSelectorAttribute
- PXDBScalarAttribute
- PXNote Attribute

DAC Attributes:

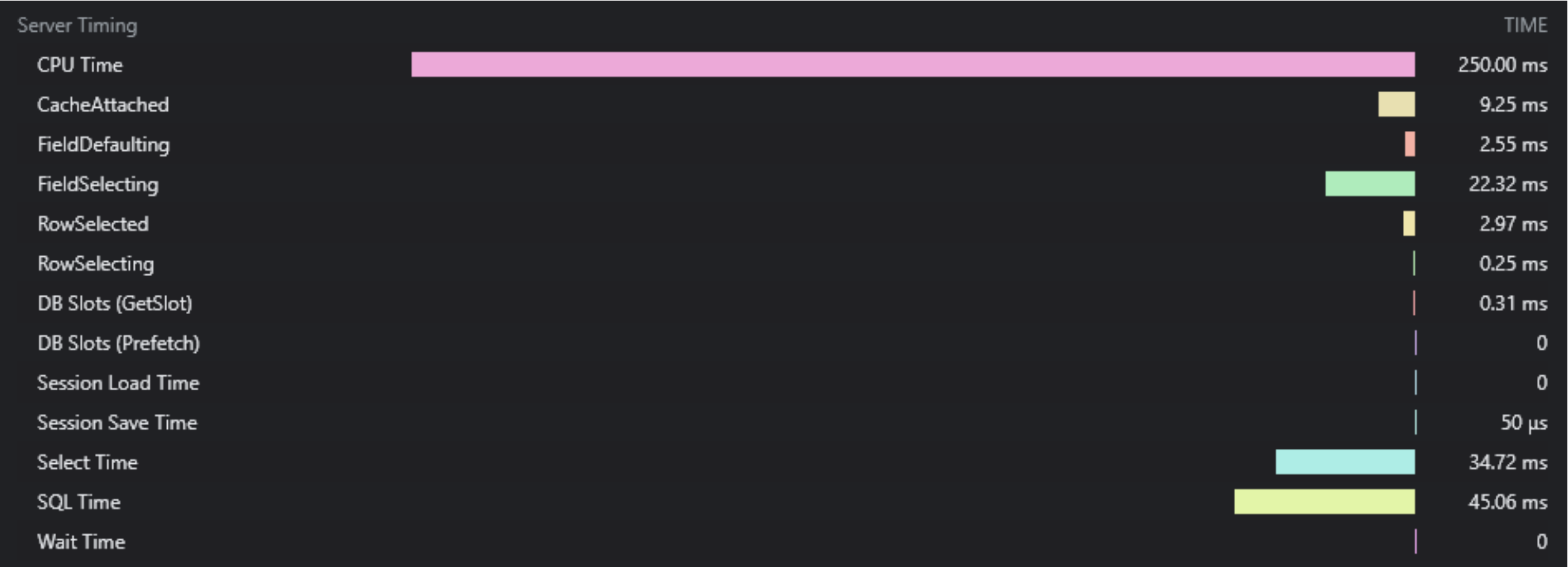
- PXProjection Attribute
- PXTable Attribute on Derived DAC or Cache Extensions



PRO Tip: Use CacheAttached to remove some heavy attributes that you don't need in a graph

Reading Huge Amounts of Data

RowSelecting and FieldSelecting Events

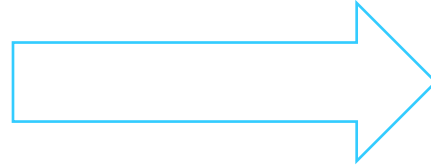


Restricting Number of Fields

- Using LINQ2Bql
- Using PXFieldScope
- Using Light Versions of DACs

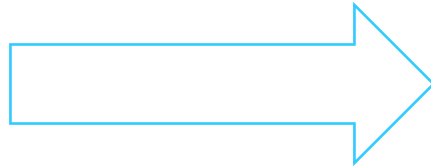
Linq2Bql

```
PXSelect<ARRegister>.Select(this).  
Where(_ => _.DocType="INV");
```



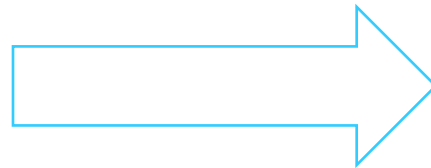
```
SELECT * FROM ARRegister  
WHERE DocType = 'INV'
```

```
PXSelect<ARRegister>.Select(this).  
FirstOrDefault();
```



```
SELECT TOP(1) * FROM ARRegister
```

```
PXSelect<ARRegister>.Select(this).  
Select(_ => _.RefNbr);
```

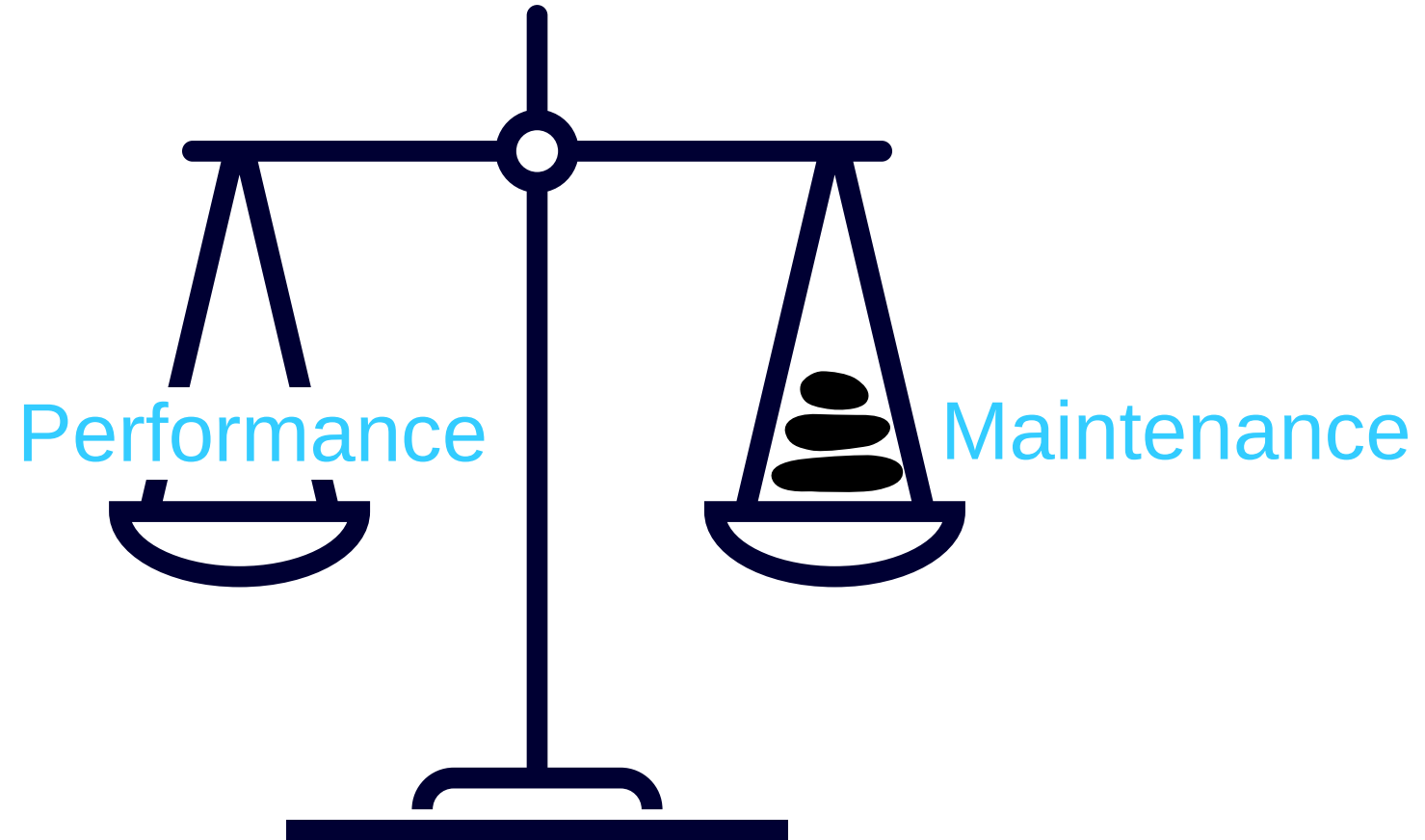


```
SELECT RefNbr FROM ARRegister
```


Using PXFieldScope

```
using (new PXFieldScope(_ARInvoice.View,  
typeof(ARInvoice.docType), typeof(ARInvoice.refNbr), typeof(ARInvoice.docDate)...))  
{  
    foreach (ARInvoice ardoc in _ARInvoice.Select(cust.BAccountID))  
    {  
        ...  
    }  
}
```

Using Light Versions of DACs



Using Light Versions of DACs

```
namespace PX.Objects.AR
```

```
public class Customer : Baccout
```

```
{
```

```
    Field1
```

```
    Field2
```

```
    ...
```

```
    Field238
```

```
}
```

```
namespace PX.Objects.AR.Light
```

```
public class Customer : IBqlTable
```

```
{
```

```
    Field1
```

```
    Field2
```

```
    Field3
```

```
}
```

Optimization of Data Processing: Reusing and Recycling

Reusing and Recycling

```
private static void ProcessDoc(List<Document> list)
{
    foreach(var doc in list)
    {
        var processingGraph = PXGraph.CreateInstance<ProcessingGraph>();
        var newDoc = new ResultDocument();
        newDoc.Field1 = doc.Field1;
        newDoc.Field2 = doc.Field2;
        ...
        processingGraph.Document.Insert(newDoc);
        processingGraph.Save.Press()
    }
}
```

Reusing and Recycling

```
private static void ProcessDoc(List<Document> list)
{
    var processingGraph = PXGraph.CreateInstance<ProcessingGraph>();
    foreach(var doc in list)
    {
        processingGraph.Clear();
        var newDoc = new ResultDocument();
        newDoc.Field1 = doc.Field1;
        newDoc.Field2 = doc.Field2;
        ...
        processingGraph.Document.Insert(newDoc);
        processingGraph.Save.Press()
    }
}
```

Reusing and Recycling

```
private static void ProcessDoc(List<Document> list)
{
    var processingGraph = PXGraph.CreateInstance<ProcessingGraph>();
    foreach(var doc in list)
    {
        processingGraph.Clear();
        var newDoc = new ResultDocument();
        newDoc.Field1 = doc.Field1;
        newDoc.Field2 = doc.Field2;
        ...
        processingGraph.Document.Insert(newDoc);

        foreach(var doc in doc.Details)
        {
            //insert document lines
        }
        processingGraph.Save.Press()
    }
}
```

Reusing and Recycling

```
private static void ProcessDoc(List<Document> list)
{
    var processingGraph = PXGraph.CreateInstance<ProcessingGraph>();
    foreach(var doc in list)
    {
        processingGraph.Clear();
        var newDoc = new ResultDocument();
        newDoc.Field1 = doc.Field1;
        newDoc.Field2 = doc.Field2;
        ...
        processingGraph.Document.Insert(newDoc);

        foreach(var doc in doc.Details)
        {
            //insert document lines
            processingGraph.Save.Press() // do for every 1000
        }
    }
}
```


Optimization of Data Processing: Redesign of Heavy Processing

Redesign of Heavy Processing

Examples of a Heavy Process:

- Data recalculation – aggregating historical data – e.g. account balances
- Data recalculation – quadratic complexity – e.g. discounts

Redesign of Heavy Processing

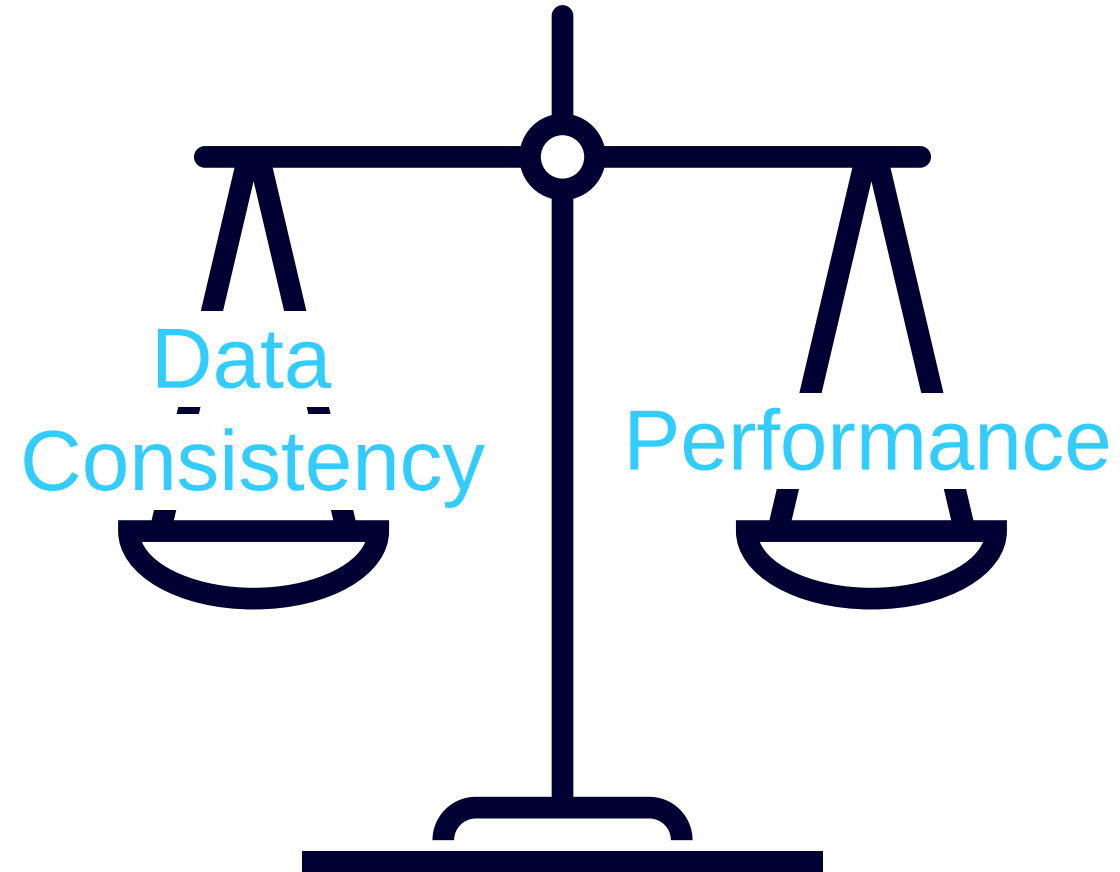


Optimization of Data Processing: Saving Data

Big Transaction Scopes

- Consume all available memory of SQL Server
- Lock a lot of tables and prevent parallel processes

Big Transaction Scopes



Summary

Optimization

1. Do not optimize what you don't need to optimize
2. Identify the areas that you need to optimize using available tools
3. Understand why the slowness appears and fix the reason
Use Best Practices for Data reading and Data Processing

A banner image for the Acumatica Virtual DevCon 2022. It features a hand pointing at a screen with code, overlaid with a blue wireframe grid and colorful bokeh lights. The text 'Acumatica Virtual DevCon 2022' is in the top left, and 'June 15-16' is below it.

Acumatica
Virtual DevCon
2022

June 15-16



Thank You!

www.acumatica.com/developers