NAV techdays 2012

a mibuso.com conference

WHEN YOU ARE PASSIONATE ABOUT MICROSOFT DYNAMICS NAV...

27 & 28 Sept 2012, Antwerp (Belgium)

NAV techdays 2012

a mibuso.com conference

C/AL coding for performance in NAV 2013

Jesper Falkebo, Lars Hammer, Bardur Knudsen

C/AL Coding for Performance

A main theme for NAV 2013 was to improve performance significantly. To reach this goal we had to rewrite the data stack completely! Join us in this session to hear about how you can adopt the changes and achieve even better performance. We will go through new extensions to the record API, how/when to use the Query object.

Contents

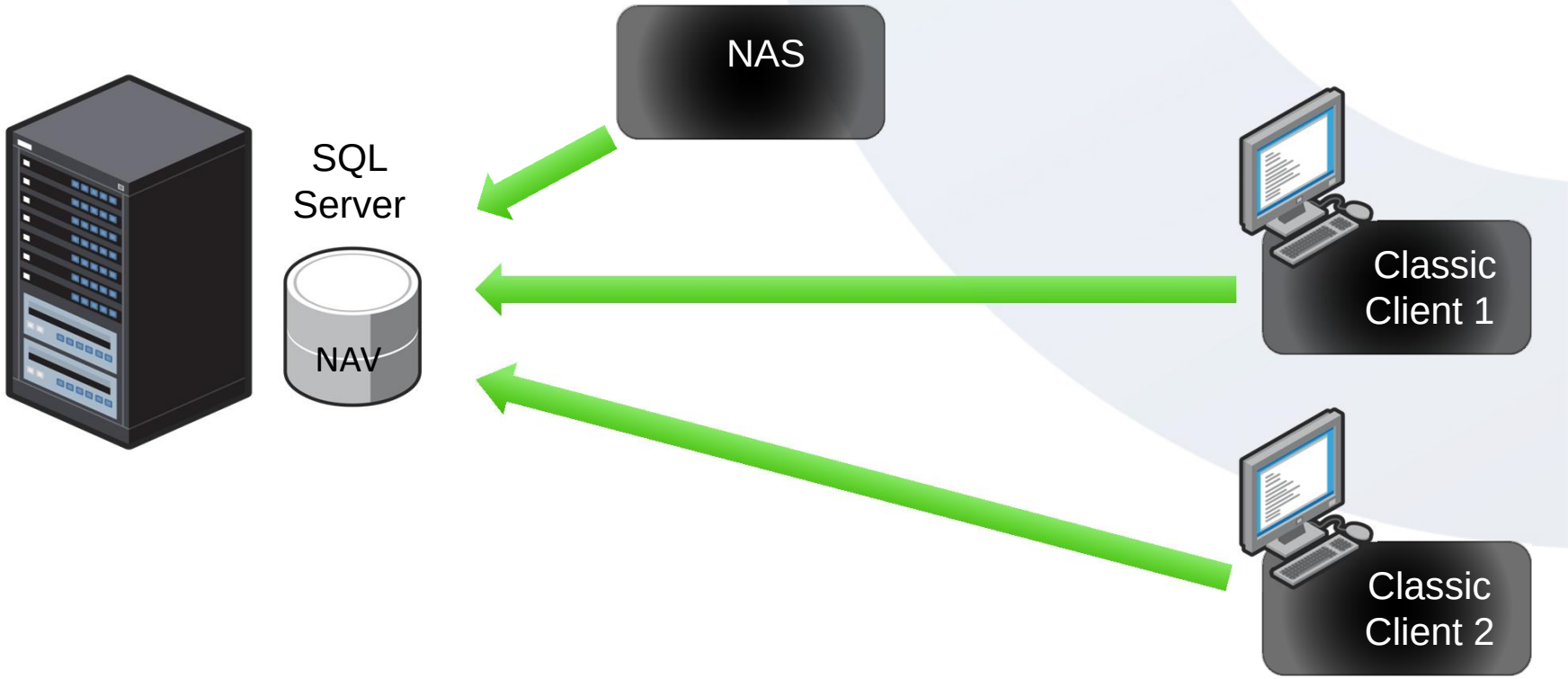
1. Lars: Architectural overview
2. Bardur: C/AL Coding for performance
3. Jesper: Data Stack, improvements

C/AL coding for performance

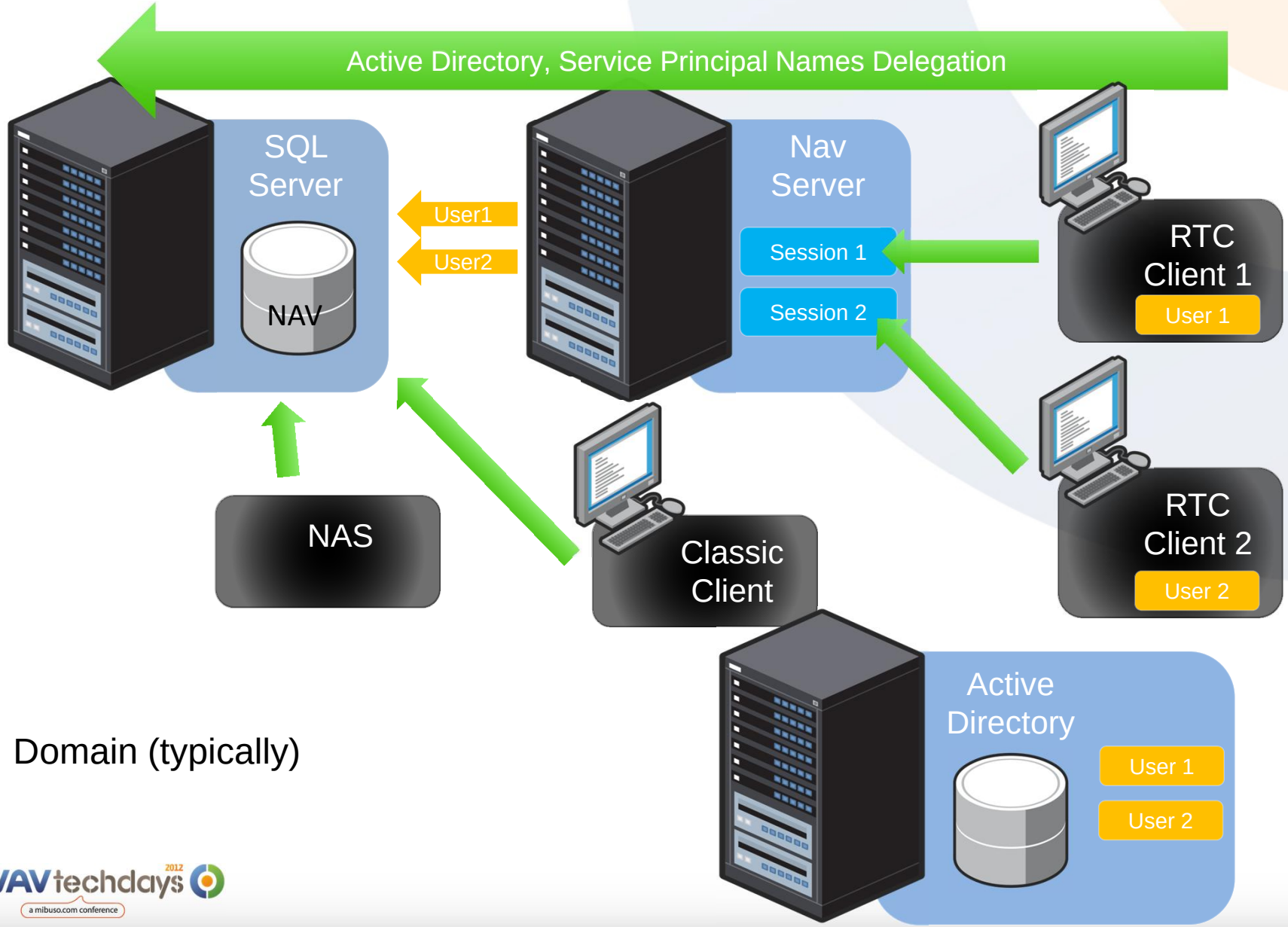
Relevant High Level NAV 2013 Changes

- Server Deployment
- Classic Runtime Discontinued
- Server Features, Optimizations and Architecture
- Query – the new Application Object Type
- C/AL Programming Constructs

NAV 5.0 'Classic' 2-tier

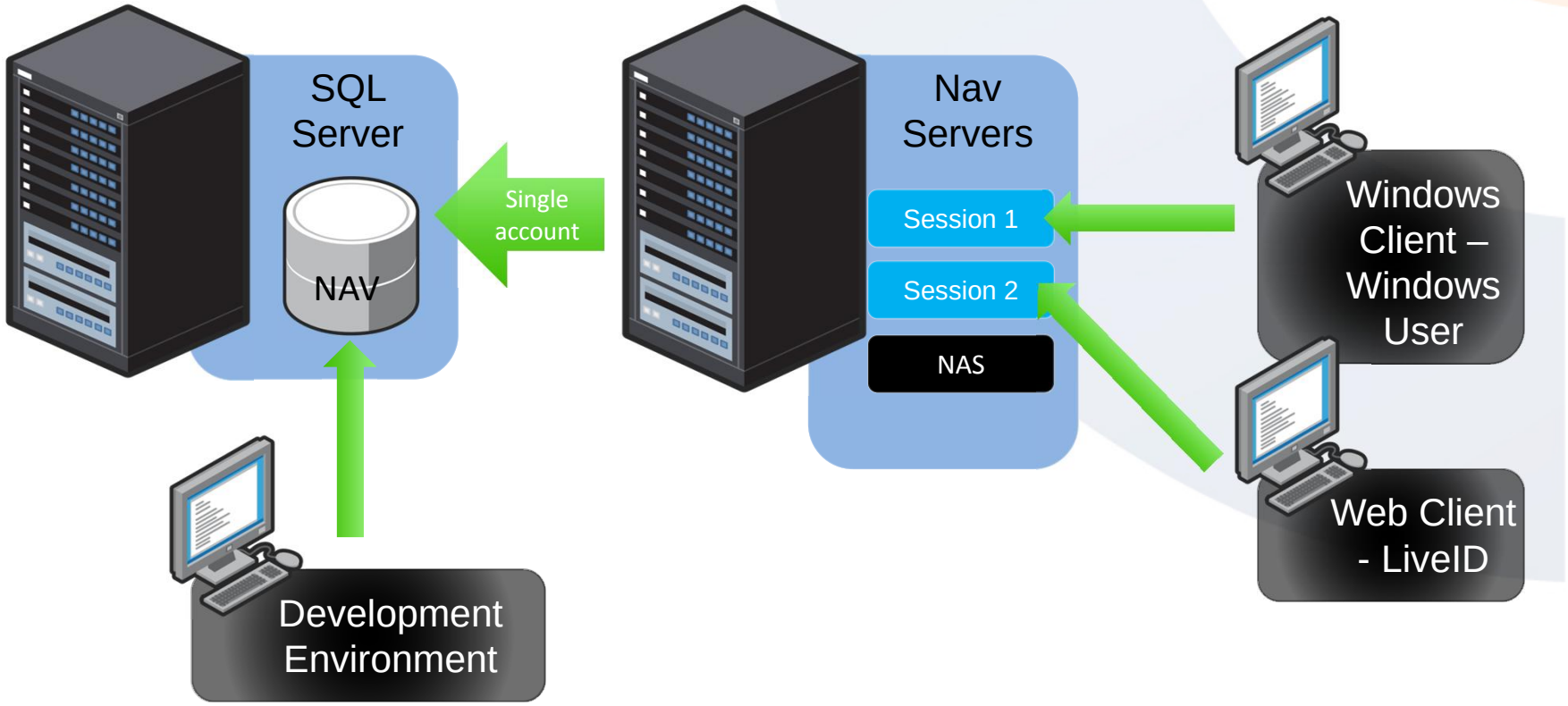


NAV 2009 3-tier (mixed mode)



Domain (typically)

NAV 2013 – back to Simple J



(Many other authentication options…)

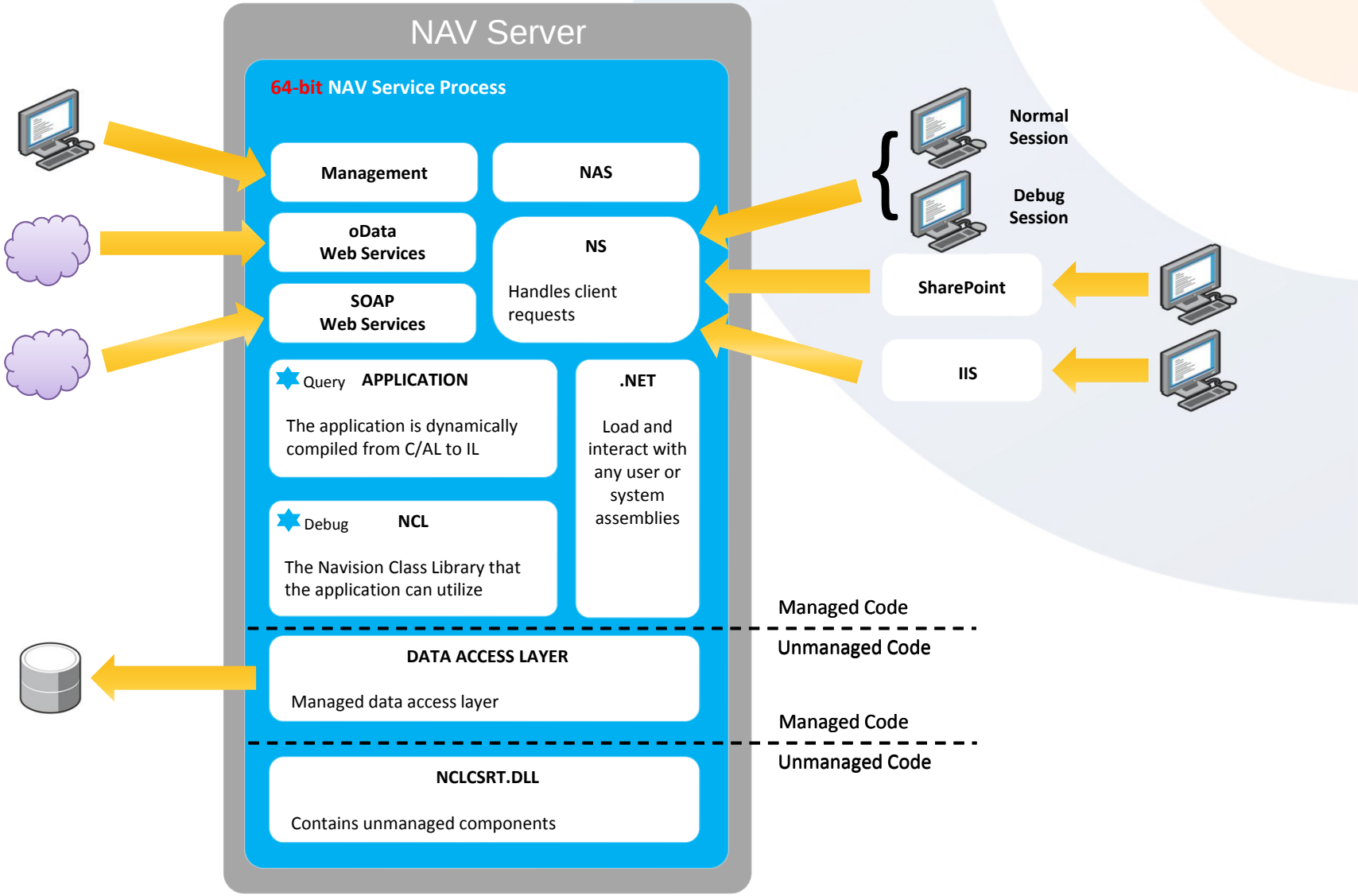
Classic Runtime Discontinued

- Only SQL Server
- No Classic Reports
- No Forms
- No C/AL execution in the Development Environment

Server Features and Optimizations

- SQL Server only
- Managed Data Access
  - Optimized for SQL server
  - Improved caching
  - Connection pooling
- Query Application Object
- Background Sessions

Server Architecture



Query – the new Application Object Type

- Read-only
- Joining Tables
- Grouping
- Totaling
- Single SELECT

- Ambition: *The* way of reading data in NAV

C/AL Programming Constructs

- Query
  - OPEN, READ, CLOSE
- Background Sessions, NAS replacement
  - STARTSESSION, STOPSESSION

C/AL Coding

Example from Upgrade – "OR" condition

```
Loop1()
WITH Item DO BEGIN
  SETFILTER("Roll-up Kit Pricing",'<>%1',0);
  MODIFYALL("Roll-up Kit Pricing",0);

  SETFILTER("Components on Sales Orders",'<>%1',0);
  MODIFYALL("Components on Sales Orders",0);

  SETFILTER("Components on Invoices",'<>%1',0);
  MODIFYALL("Components on Invoices",0);
END;
ERROR('hi');
```

// filters are cleared
// in the real code

~1 sec.

```
Loop2()
WITH Item DO
  IF FIND('-') THEN
    REPEAT
      IF ("Roll-up Kit Pricing" <> "Roll-up Kit Pricing"::Never) OR
         ("Components on Sales Orders" <> 0) OR
         ("Components on Invoices" <> 0)
      THEN BEGIN
        "Roll-up Kit Pricing" := 0;
        "Components on Sales Orders" := 0;
        "Components on Invoices" := 0;
        MODIFY;
      END;
    UNTIL NEXT = 0;
ERROR('hi');
```

~12 sec.

```
Loop3()
WITH Item DO BEGIN
  MODIFYALL("Roll-up Kit Pricing",0);
  MODIFYALL("Components on Sales Orders",0);
  MODIFYALL("Components on Invoices",0);
END;
ERROR('hi');
```

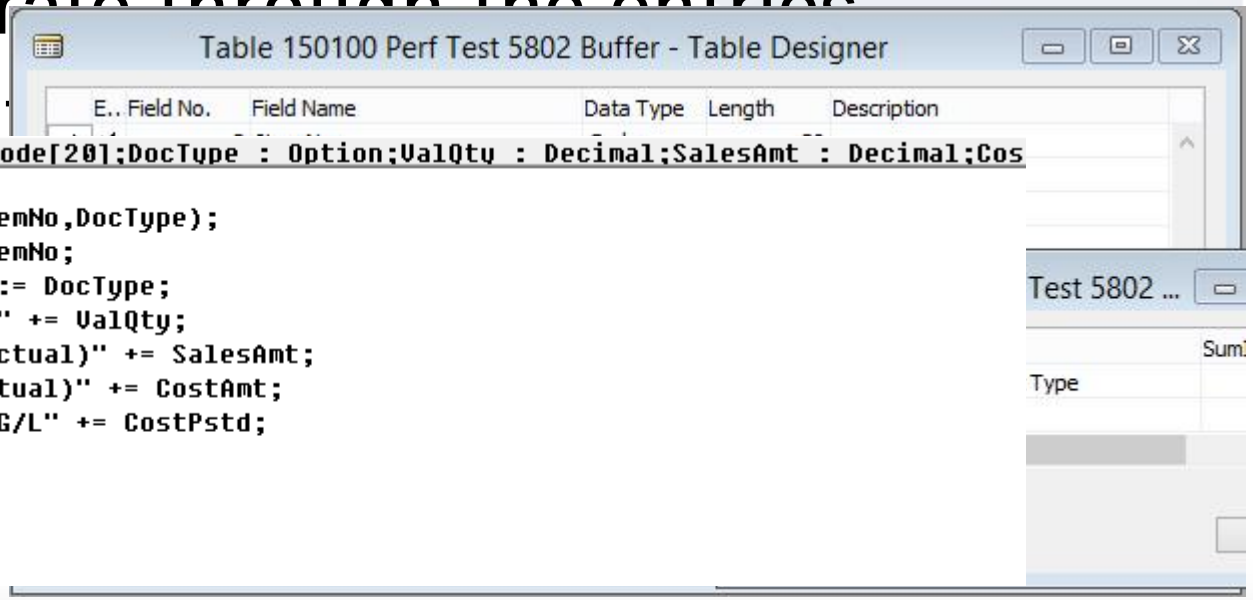
~5 sec.

Iterating a set

Example:

We want to summarize some Value Entries (5802) by ”Item No.” and ”Document Type”:

1. We create a (temp) table to hold values while we iterate through the entries
2. Then we wri




```
AddVE(ItemNo : Code[20];DocType : Option;ValQty : Decimal;SalesAmt : Decimal;Cos
INIT;
Exists := GET(ItemNo,DocType);
"Item No." := ItemNo;
"Document Type" := DocType;
"Valued Quantity" += ValQty;
"Sales Amount (Actual)" += SalesAmt;
"Cost Amount (Actual)" += CostAmt;
"Cost Posted to G/L" += CostPstd;
IF Exists THEN
  MODIFY
ELSE
  INSERT;
```

Which is faster?

```
// function 1
WITH ValueEntry DO BEGIN
  SETFILTER("Item No.",'7*');
  IF FINDSET THEN
    REPEAT
      VEBuffer.AddVE("Item No.","Document Type",…);
    UNTIL NEXT = 0;
END;
```

~74s

~212s

Item ‘7*’ -> ~0.86 mill. entries out of ~6 mill.

Item ‘1*’ -> ~2.4 mill. entries out of ~6 mill.

```
// function 2
WITH ValueEntry DO BEGIN
  SETCURRENTKEY("Item No.","Posting Date","Item Ledger Entry Type”,…);
  SETFILTER("Item No.",'7*');
  IF FINDSET THEN
    REPEAT
      VEBuffer.AddVE("Item No.","Document Type",…);
    UNTIL NEXT = 0;
END;
```

~412s

~1015s

How about using Query?

```
WITH ValueEntryQry DO BEGIN
  SETFILTER(Item_No,'7*');
  IF OPEN THEN
    WHILE READ DO
      VEBuffer.AddVE(Item_No,Document_Type,…
```

~0.9 mill. entries out of ~6 mill. ~53s

~2.4 mill. entries out of ~6 mill. ~56s

Calculate Sums – no SIFT

```
CalcSum1()
PerfTestTable.SETRANGE(Weekday,PerfTestTable.Weekday::Thursday);
IF PerfTestTable.FIND('-') THEN
  REPEAT
    Total := Total + PerfTestTable.Amount;
  UNTIL PerfTestTable.NEXT = 0;
```

~350ms

```
CalcSum2()
PerfTestTable.SETCURRENTKEY(Weekday);
PerfTestTable.SETRANGE(Weekday,PerfTestTable.Weekday::Thursday);
PerfTestTable.CALCSUMS(Amount);
```

~12ms

Size Matters...

A simple iteration over the Item Ledger Entry table (~6 mill. records) with three queries with different number of fields and one full-record loop:

Description	Average Duration (s)
Query on Table 32, first 6 fields	61
Query on Table 32, first 9 fields	64
Query on Table 32, first 12 fields	86
Record Table 32, all 72 fields	222

CurrPage.UPDATE(FALSE)

- Historically we have had a lot CurrForm.UPDATEs
- Most of them are not necessary in Pages.

Hunting the last 1%..

```
ItemLedgEntry.FINDFIRST;
ValueEntry.FINDFIRST;
FOR i := 1 TO 1000000 DO
  MyFunc(ItemLedgEntry,ValueEntry);
```

```
1) MyFunc(ItemLedgEntry;ValueEntry);                5.8s
2) MyFunc(VAR ItemLedgEntry;VAR ValueEntry);        0.3s
```

Background Sessions

[{Ok} :=] STARTSESSION(
  SessionId, CodeunitId [, Company] [, Record])

- + Great for off-loading user sessions
- - Performance cost of an extra session (a Windows handle, login time, memory etc.); approx. 10ms + ~20kB to start one session.
- Use JobQueue to process async. and to serialize execution.

Background Process Example...

- Towards f
  - Update
  - Update
- We did ac
  backgrou
  changed i
  than the g

```
OnRun(VAR Rec : Record "Analysis View")
IF Code <> '' THEN BEGIN
  InitLastEntryNo;
  LOCKTABLE;
  FIND;
  UpdateOne(Rec,2,"Last Entry No." < LastGLEntryNo - 1000);
END;
```

```
InitLastEntryNo()
UpdateAll(Which : 'Ledger Entries,Budget Entries,Both';DirectlyFromPosting : Boolean)
AnalysisView2.SETRANGE(Blocked,FALSE);
IF DirectlyFromPosting THEN
  AnalysisView2.SETRANGE("Update on Posting",TRUE);

IF AnalysisView2.ISEMPTY THEN
  EXIT;

InitLastEntryNo;

IF DirectlyFromPosting THEN
  AnalysisView2.SETFILTER("Last Entry No.",'<%1',LastGLEntryNo);

AnalysisView2.LOCKTABLE;
IF AnalysisView2.FINDSET THEN
  REPEAT
    // UpdateOne(AnalysisView2,Which,NOT DirectlyFromPosting AND (AnalysisView2."Last Ent
    STARTSESSION(SessionNo,CODEUNIT::"Update Analysis View",COMPANYNAME,AnalysisView2);
  UNTIL AnalysisView2.NEXT = 0;
```

Background examples

- REP84 - Update Analysis Views
- Start many sessions...

Recommendations

- SETFILTER/-RANGE instead of IF …
  - Loops (~NEXT) are evil!
- Use CALCSUM instead of loops
- Only use SETCURRENTKEY if you need sorting
- Use Queries for large data sets
- Clean out those left over CurrPage.UPDATEs

New DataStack

NAV techdays 2012

a mibuso.com conference

Demo

Utilizing a background session to get UI responsiveness
=> percieved performance

New Datastack

- Completely rewritten. NDBCS.DLL is no longer used by the server
- Based on ADO
  - ADO is maintained by the SQL team and supports new SQL Server functionality
- Better performing, more scalable
  - Vastly reduced managed/unmanaged transitions
  - Internal measurements show better performance than classic client and use far less memory

Connection pooling

- Use a single powerful account to connect with SQL Server
  - No need for setting up delegation
- Reduce memory consumption (Multiple users can share the same connection)
- Every user-session has a preferred connection

Cross user datacaching

- Data only read by the NST once
  - In NAV2009 caching was done for each user
- Affects all record API calls for retrieving data
- Query result-sets not cached in NAV 2013
- Cross NST cache synchronization
  - Caches up-to-date within one minute

One security model

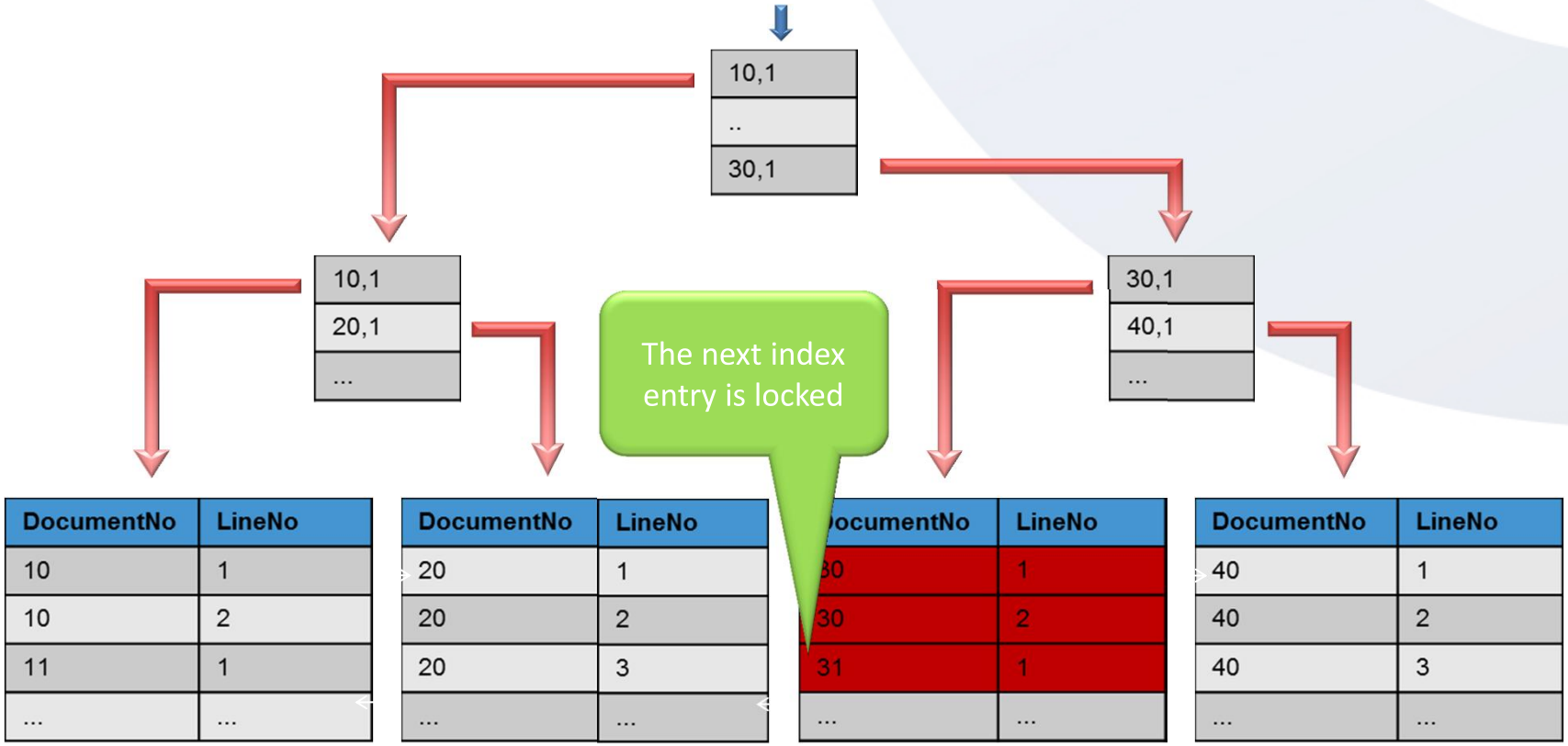
- Permission evaluation takes place in the NST only.
- No need for Synchronize All Logins
- No need for the XP_NDO extended stored procedure

Performance MARS vs Cursors

- Now using Multiple Active Result Set (MARS) instead of cursors
- Impact:
  - Less SQL Server roundtrips
  - SQL Server has increased freedom to choose the best access plan
  - FIND and FINDSET are equally fast
  - Supporting dynamic result-sets is more expensive (Consider using query)

The range-lock problem

```
SalesLine.SETCURRENTKEY(SalesLine.DocumentNo, SalesLine.LineNo);
SalesLine.SETRANGE(SalesLine.DocumentNo, 30);
SalesLine.LockTable;
IF (SalesLine.FindSet('-')) THEN ;
```



Changed isolation level

- Changed isolation level from SERIALIZABLE to REPEATABLE READ
  - Prevents blocking by examply when posting sales orders

```
SalesLine.LockTable;
SalesLine.SetFilter(“Document No.”, 10);
SalesLine.SetRange(“Line No.”, ‘1000..2000’);
IF (SalesLine.FINDSET) THEN
REPEAT

UNTIL (SalesLine.NEXT()=
```

With REPEATABLE READ only lines that exist are locked

Character data improvements

- Unicode support
- Only support for latest Windows Collations
  - Server leverages this knowledge and optimize SQL statements
- Conversion takes place during database conversion in one step

Unicode support

- Data from different codepages can be stored in the database with no loss
- Still only one sorting

Consistent sorting

- Temporary tables and normal tables sorted the same way
- Retrieval of marked records respects the chosen sorting
  - Retrieves marked records with one statement if possible
- @ operator use SQL COLLATE keyword

CalcSum/CalcFields improvements

- Can use SIFT even for Count and Average formulas
- SIFT indices are no longer a requirement for CalcSum / CalcField calls
- Use Min/Max SQL functions instead of multiple SQL statements
- No longer traverse data when Validated security filter mode is used

SetAutoCalcFields

- New command to calculate flow-fields automatically for find calls
  - Translated into a single SQL-Statement (Internally named ”Smart-SQL”)
- Used when filters are applied on flow-fields to only retrieve records that pass the filter from SQL Server
- Used behind the scenes by pages and reports

SetAutoCalcFields

```
IF (customers.FINDSET) THEN
BEGIN
  REPEAT
    customers.calcFields("Sales (LCY)");
    IF (customers."Sales (LCY)" <> 0) THEN
    BEGIN
      customers.calcFields("Balance (LCY)");
      IF (((customers."Balance (LCY)" / customers."Sales (LCY)") * 100) > 40) THEN
      BEGIN
        customers.LOCKTABLE;
        customers.Blocked := customers.Blocked::Invoice;
        customers.MODIFY;
      END;
    END;
  UNTIL (customers.NEXT()=0);
END;
```

Query implementation

- Query definitions always translated into a single SQL statement
- Underlying engine for ”Smart SQL”
  - Utilizing features like sub-queries, const column values and apply joins which are not yet available through the designer

Query and classic NAV constructs

- Utilizes SIFT
  - The query engine will use a SIFT index to cover retrieval of data for a data-item whenever possible
- Flow-Field support
  - Flow-Fields can be used in a query making it easier to keep business logic consistent.*
- Flow-Filter support
  - Flow-Filters can be used in join conditions to ensure consistency among joined data-items and flow-fields

Performance considerations for query

- Query results are not cached!
- Query is not guaranteed to deliver a dynamic result-set
  - On the other hand it is not guaranteed that it won't deliver a dynamic result-set either
- Ordering of tables in query matters!
  - Nested Loop and Force Ordering hints are applied

Buffered inserts

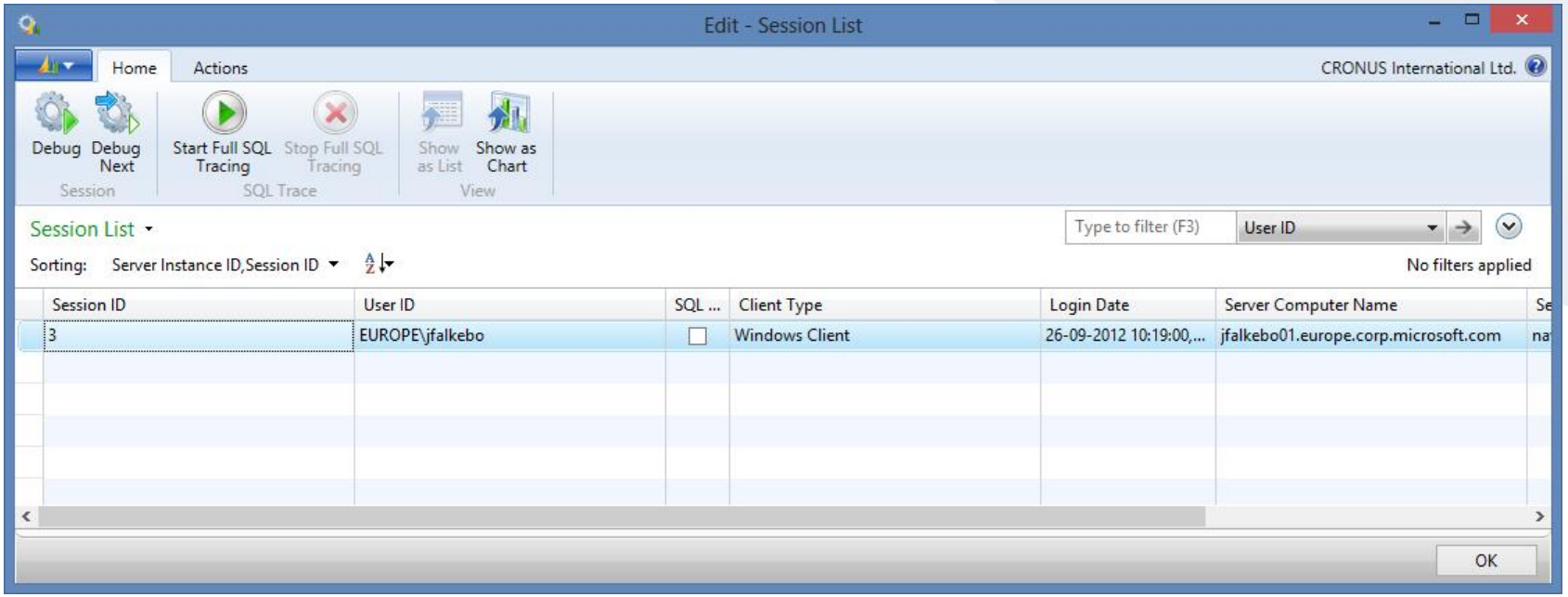
- Buffered inserts used more often:
  - Tables containing RecordId's and/or SqlVariant columns now supported

```
IF (JnlLine.Find('-')) THEN
BEGIN
  GLEntry.LockTable;
  IF (NOT GLEntry.FindLast) THEN
    GLEntry."Entry No.":=0;
  REPEAT
    GLEntry."Entry No.":= GLEntry."Entry No.“ +1;
    …
    GLEntry.Insert;
  UNTIL (JnlLine.Find('>') = 0)
END;
Commit;
```

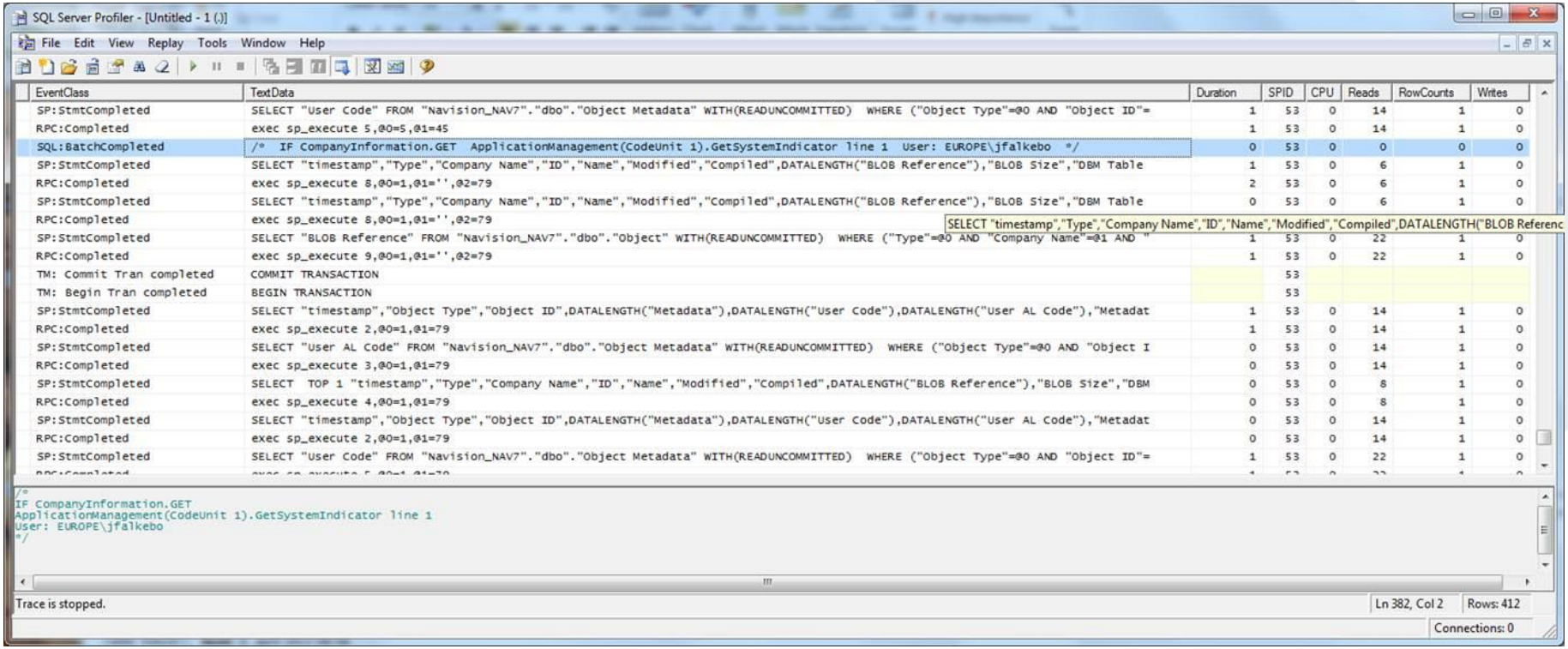
The records are sent to the database here

SQL Callstack trace

- Enabled through the debugger UI



SQL Callstack trace



NAV techdays 2012

a mibuso.com conference

Questions & Answers

NAV TechDays 2012 Aftercare

- Follow up questions & comments?
  - Post them in the NAV TechDays 2012 Sessions forum on mibuso.com
  - Will be reviewed by presenters from R&D
- Attend online meetings (organized by MDCC)
- Recorded sessions will be made available for download on mibuso.com

NAV techdays 2012

a mibuso.com conference

Thank you for your attention

NAV techdays 2012

a mibuso.com conference

Coffee break

See you back
in 30 minutes