Goals

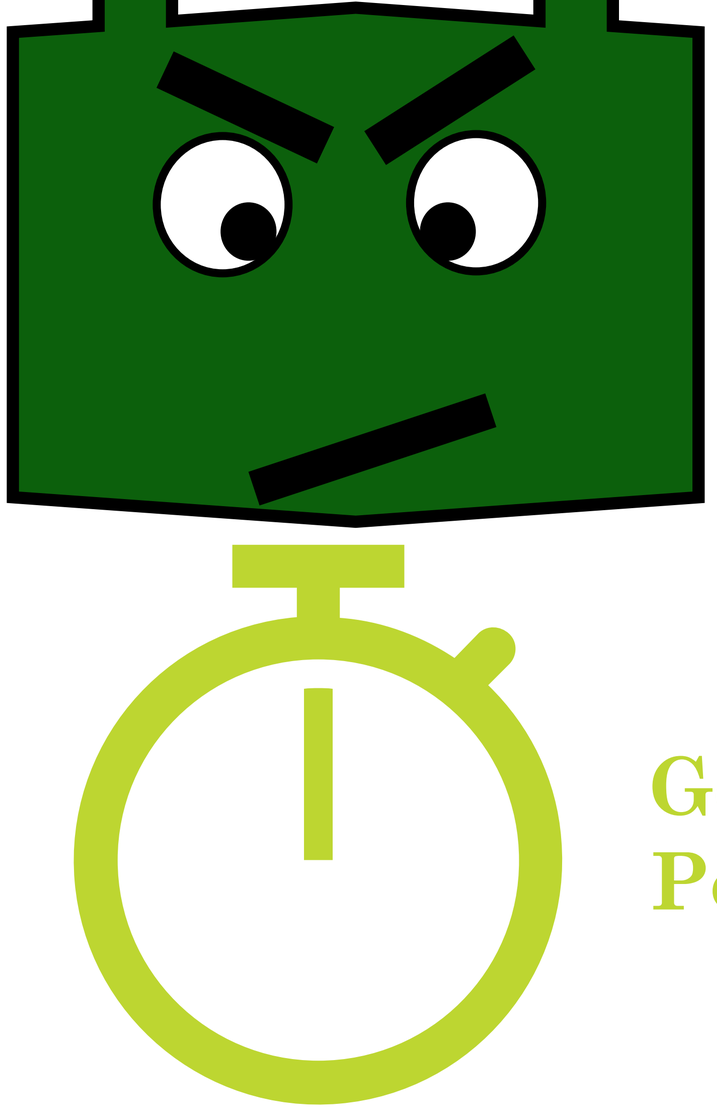
- Introduce ReadIsolation feature

Good Performance

And then someone wrote few lines of code...

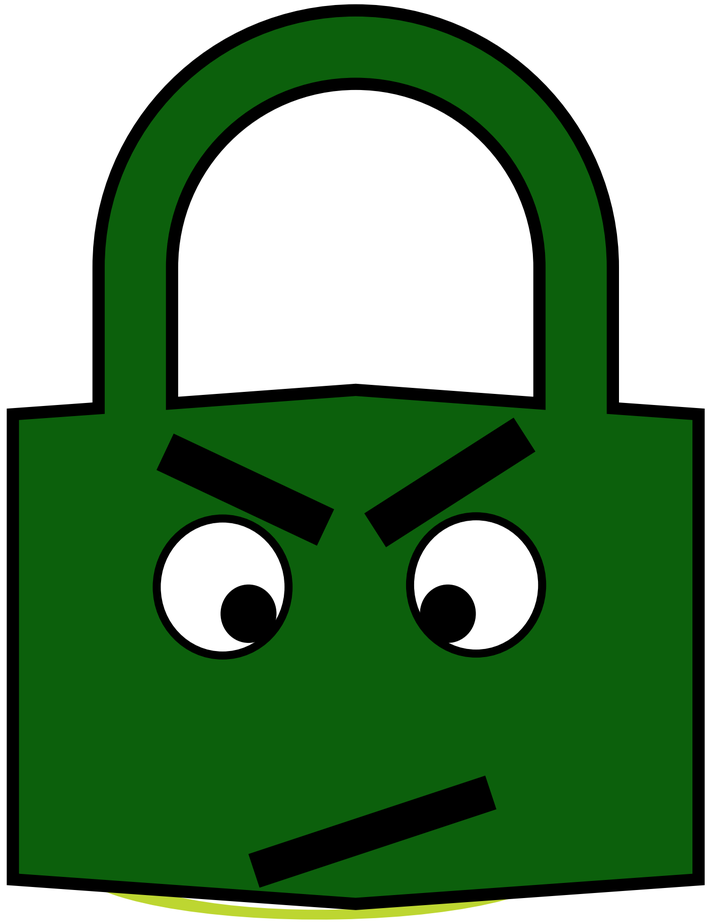
Good
Performance

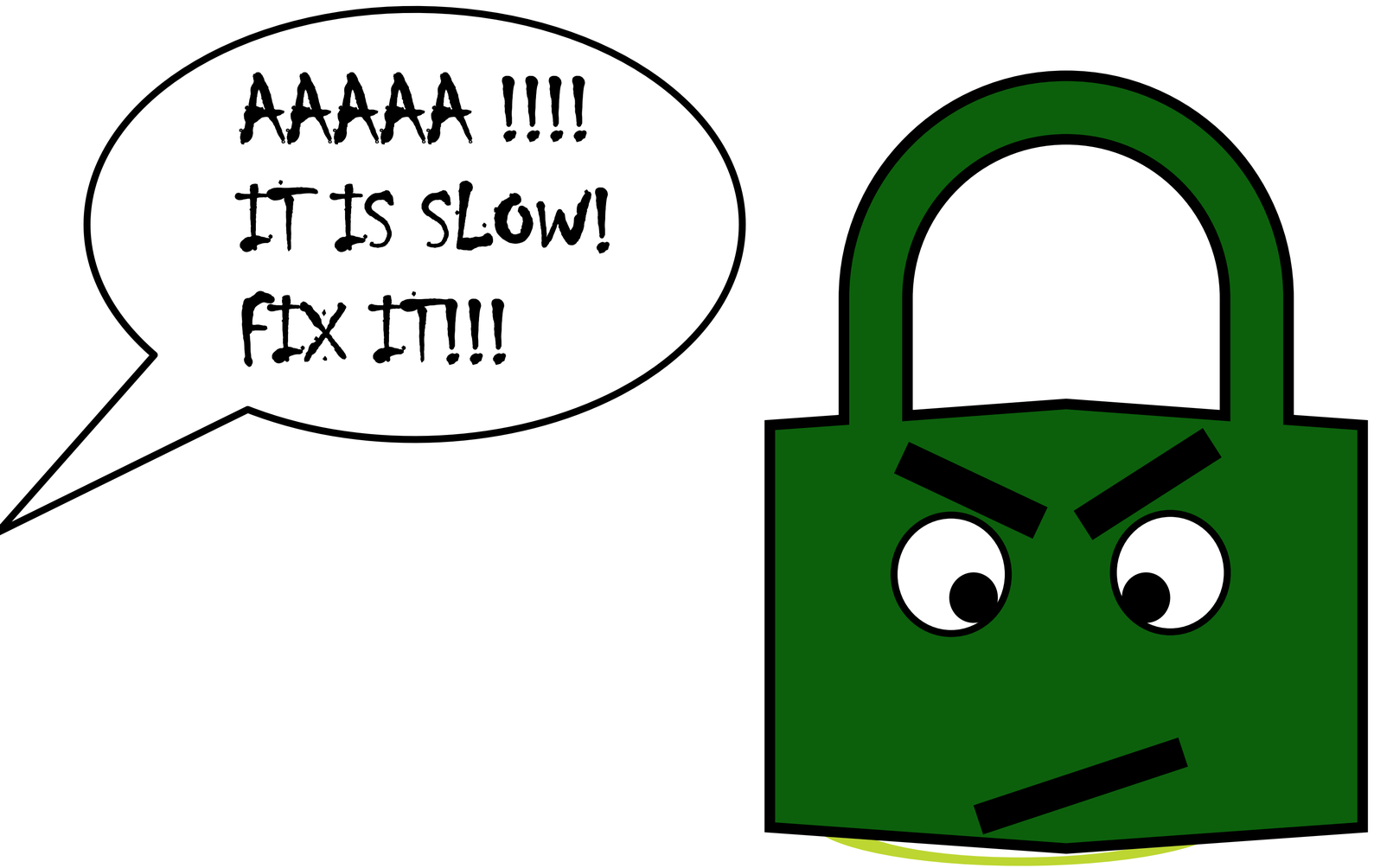
A Very Bad lock



Good Performance

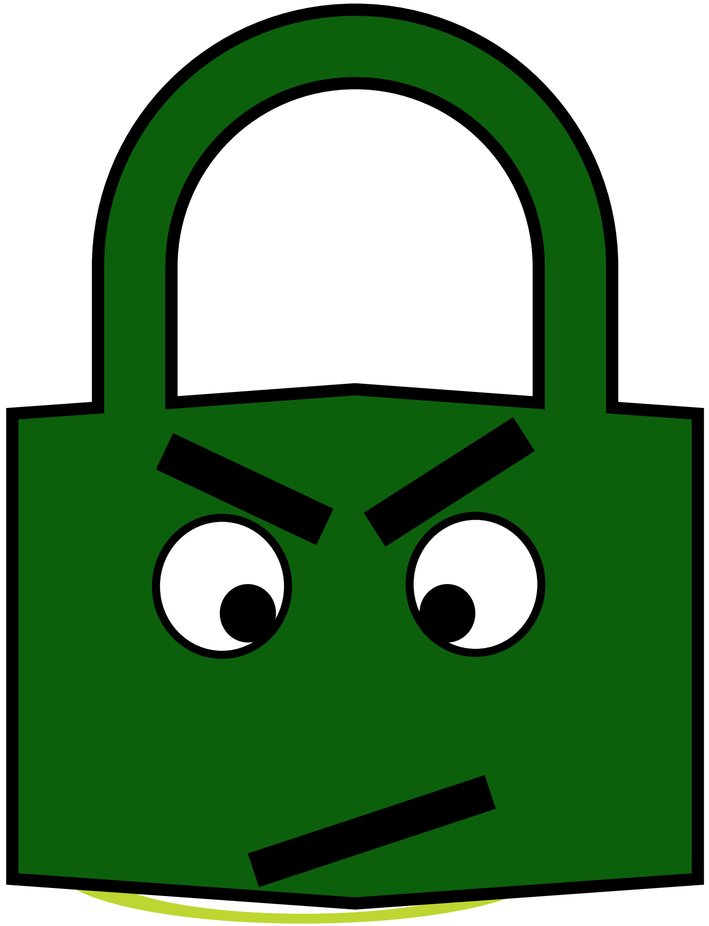
BC/NAV TECH DAYS 2023
mibuso.com



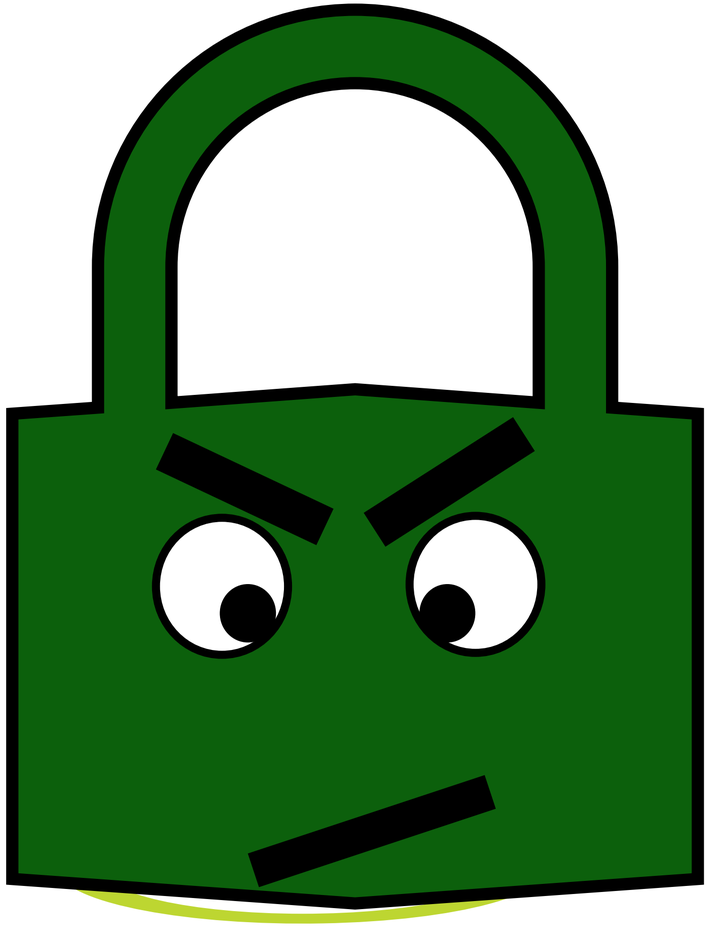

AAAAA !!!!
IT IS SLOW!
FIX IT!!!

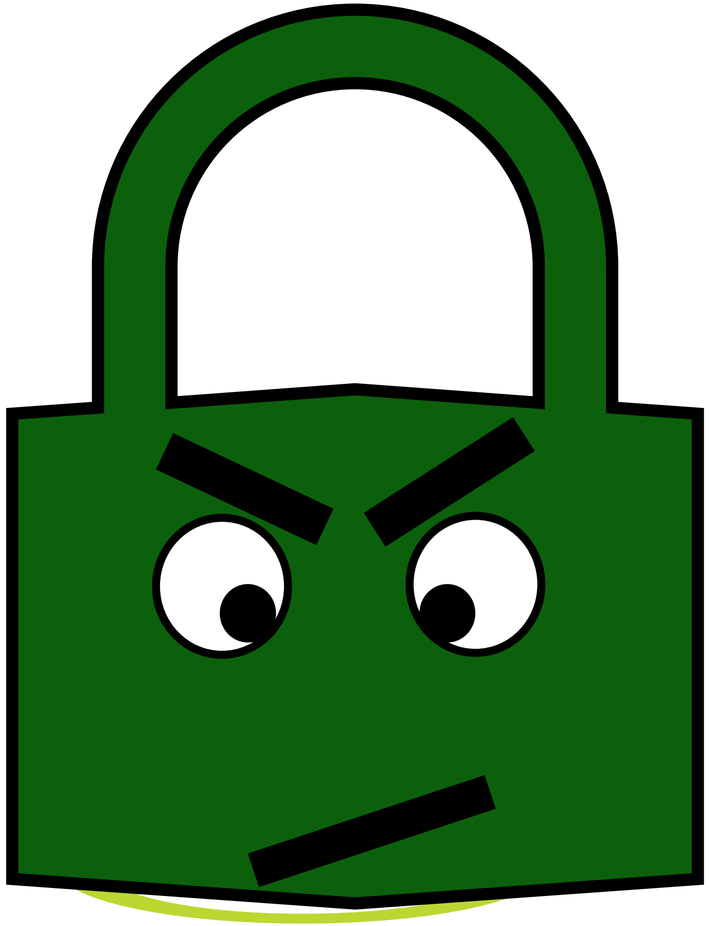
Throw in ReadIsolation

(writing few lines of code)

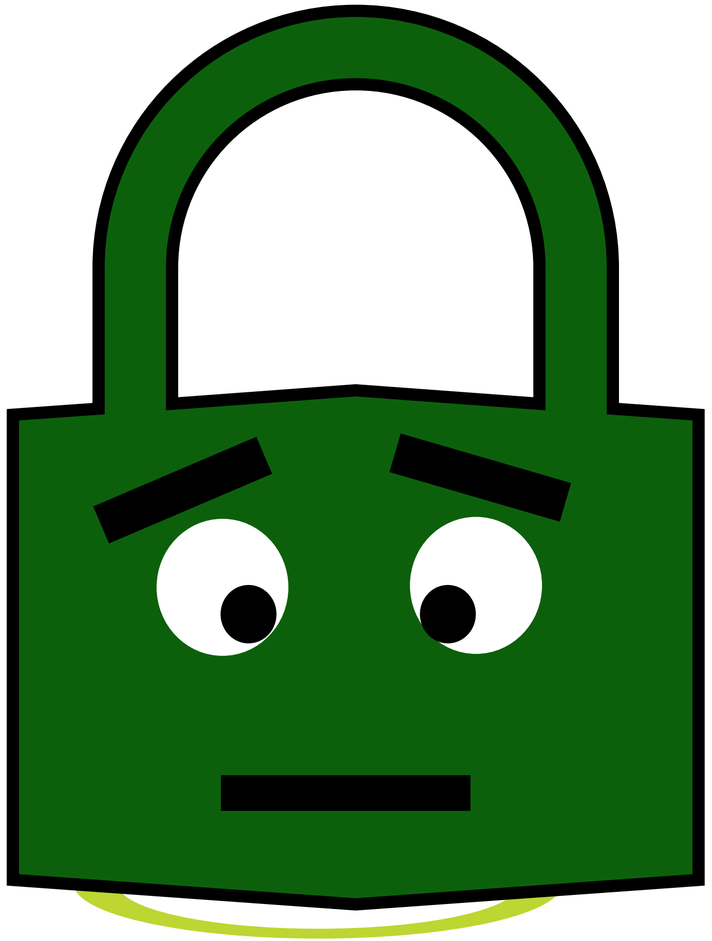


BC/NAV TECH DAYS 2023
mibuso.com





Read Isolation



Read Isolation

Good Performance

Goals

- Introduce ReadIsolation feature
- Increase your understanding on how the locking works
- Enable you to write a more performant code

In this session:

BC/NAV TECH DAYS 2023
mibuso.com

- Test your knowledge – Lock Quiz
- Theory
  - AL Interplay with DB
  - Transaction scope
  - How locking works
  - Caching implications
- Practical
  - ReadIsolation property
  - Patterns
  - Tooling

Test your knowledge

BC/NAV TECH DAYS 2023

mibuso.com

QUIZ

Will it lock or not?

Question 1. LockTable

BC/NAV TECH DAYS 2023
mibuso.com

```
internal procedure FindsetOnAnEmptyTable()
var
    Customer: Record Customer;
begin
    Customer.LockTable();
    Customer.FindSet();
end;
```

Question 1. LockTable

BC/NAV TECH DAYS 2023
mibuso.com

```
internal procedure FindsetOnAnEmptyTable()
var
    Customer: Record Customer;
begin
    Customer.LockTable();
    Customer.FindSet();
end;
```

Question 1. LockTable

BC/NAV TECH DAYS 2023
mibuso.com

```
internal procedure FindsetOnAnEmptyTable()
var
    Customer: Record Customer;
begin
    Customer.LockTable();
    Customer.FindSet();
end;
```

Locks

Question 2: Lock on a different variable

BC/NAV TECH DAYS 2023
mibuso.com

```
internal procedure FindsetOnAnEmptyTable()
var
    Customer1: Record Customer;
    Customer2: Record Customer;
begin
    Customer1.LockTable();
    Customer2.FindSet();
end;
```

Question 2: Lock on a different variable

BC/NAV TECH DAYS 2023
mibuso.com

```
internal procedure FindsetOnAnEmptyTable()
var
    Customer1: Record Customer;
    Customer2: Record Customer;
begin
    Customer1.LockTable();
    Customer2.FindSet();
end;
```

Question 2: Lock on a different variable

BC/NAV TECH DAYS 2023
mibuso.com

```
internal procedure FindsetOnAnEmptyTable()
var
    Customer1: Record Customer;
    Customer2: Record Customer;
begin
    Customer1.LockTable();
    Customer2.FindSet();
end;
```

LockTable is not connected to variable.

It is active on a table until transaction ends.

Locks

Lock table – Actual syntax

BC/NAV TECH DAYS 2023
mibuso.com

```
internal procedure FindsetOnAnEmptyTable()
var
    Customer1: Record Customer;
    Customer2: Record Customer;
begin
    Customer1.LockTable();
    Customer2.FindSet();
end;
```

Lock table – Actual syntax

BC/NAV TECH DAYS 2023
mibuso.com

```
internal procedure FindsetOnAnEmptyTable()
var
    Customer1: Record Customer;
    Customer2: Record Customer;
begin
    LockTable(Database::Customer);
    Customer2.FindSet();
end;
```

Question 3. Read after write

BC/NAV TECH DAYS 2023
mibuso.com

```
internal procedure WriteOperationExample()
var
    FirstCustomer: Record Customer;
    LastCustomer: Record Customer;
begin
    FirstCustomer.FindFirst();
    FirstCustomer.Name := FirstCustomer.Name + '2';
    FirstCustomer.Modify();

    LastCustomer.FindLast();
end;
```

Question 3. Read after write

BC/NAV TECH DAYS 2023
mibuso.com

```
internal procedure WriteOperationExample()
var
    FirstCustomer: Record Customer;
    LastCustomer: Record Customer;
begin
    FirstCustomer.FindFirst();
    FirstCustomer.Name := FirstCustomer.Name + '2';
    FirstCustomer.Modify();

    LastCustomer.FindLast();
end;
```

Question 3. Read after write

BC/NAV TECH DAYS 2023
mibuso.com

```
internal procedure WriteOperationExample()
var
    FirstCustomer: Record Customer;
    LastCustomer: Record Customer;
begin
    FirstCustomer.FindFirst();
    FirstCustomer.Name := FirstCustomer.Name + '2';
    FirstCustomer.Modify();

    LastCustomer.FindLast();
end;
```

Question 3. Read after write

BC/NAV TECH DAYS 2023
mibuso.com

```al
internal procedure WriteOperationExample()
var
    FirstCustomer: Record Customer;
    LastCustomer: Record Customer;
begin
    FirstCustomer.FindFirst();
    FirstCustomer.Name := FirstCustomer.Name + '2';
    FirstCustomer.Modify();

    LastCustomer.FindLast();
end;
```

If there is a write operation on a table any read will lock records

Locks

Question 4. Count after write

BC/NAV TECH DAYS 2023
mibuso.com

```
internal procedure WriteOperationExample()
var
    FirstCustomer: Record Customer;
    LastCustomer: Record Customer;
begin
    FirstCustomer.FindFirst();
    FirstCustomer.Name := FirstCustomer.Name + '2';
    FirstCustomer.Modify();

    LastCustomer.Count();
end;
```

Question 4. Count after write

BC/NAV TECH DAYS 2023
mibuso.com

```
internal procedure WriteOperationExample()
var
    FirstCustomer: Record Customer;
    LastCustomer: Record Customer;
begin
    FirstCustomer.FindFirst();
    FirstCustomer.Name := FirstCustomer.Name + '2';
    FirstCustomer.Modify();

    LastCustomer.Count();
end;
```

Locks

Question 5: Findset

BC/NAV TECH DAYS 2023
mibuso.com

```
internal procedure FindsetOnAnEmptyTable()
var
    MyTable: Record MyTable;
begin
    MyTable.DeleteAll();
    // some other code…
    if not MyTable.FindSet() then
        exit;
end;
```

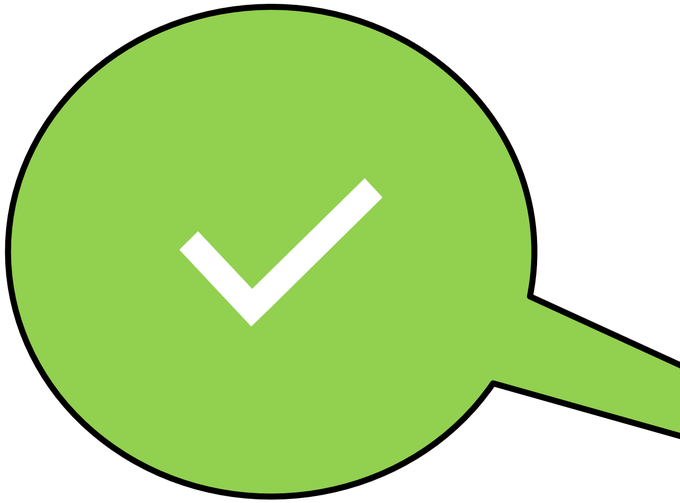
Question 5: Findset

BC/NAV TECH DAYS 2023
mibuso.com

```
internal procedure FindsetOnAnEmptyTable()
var
    MyTable: Record MyTable;
begin
    MyTable.DeleteAll();
    // some other code…
    if not MyTable.FindSet() then
        exit;
end;
```

If no rows are returned the entire table is locked for inserts until the transaction ends.

Locks



Question 5: Findset

```
internal procedure FindsetOnAnEmptyTable()
var
    MyTable: Record MyTable;
begin
    MyTable.DeleteAll();
    // some other code…
    if not MyTable.FindSet() then
        exit;
end;
```

Could lock entire table

Locks entire table

Question 6: Conditional Insert

BC/NAV TECH DAYS 2023
mibuso.com

```
internal procedure InsertOperationLock()
var
    NewCustomer: Record Customer;
begin
    NewCustomer."No." := 'New';
    if NewCustomer.Insert() then;
end;
```

Question 6: Catching errors on Insert

BC/NAV TECH DAYS 2023
mibuso.com

```
internal procedure InsertOperationLock()
var
    NewCustomer: Record Customer;
begin
    NewCustomer."No." := 'New';
    if NewCustomer.Insert() then;
end;
```

We are in a write transaction even if the insert fails

Locks

Question 7: FlowFields

BC/NAV TECH DAYS 2023
mibuso.com

```
trigger OnAction()
var
    dcle: Record "Detailed Cust. Ledg. Entry";
    cust: Record Customer;
begin
    dcle.LockTable();
    cust.FindFirst();
    cust.CalcFields(cust.Balance);
end;



field(58; Balance; Decimal)
{
    CalcFormula = Sum("Detailed Cust. Ledg. Entry".Amount
                      WHERE("Customer No." = FIELD("No.") …
    FieldClass = FlowField;
}
```

Question 7: FlowFields

BC/NAV TECH DAYS 2023
mibuso.com

```
trigger OnAction()
var
    dcle: Record "Detailed Cust. Ledg. Entry";
    cust: Record Customer;
begin
    dcle.LockTable();
    cust.FindFirst();
    cust.CalcFields(cust.Balance);
end;


field(58; Balance; Decimal)
{
    CalcFormula = Sum("Detailed Cust. Ledg. Entry".Amount
                      WHERE("Customer No." = FIELD("No.") …
    FieldClass = FlowField;
}
```

Question 7: FlowFields

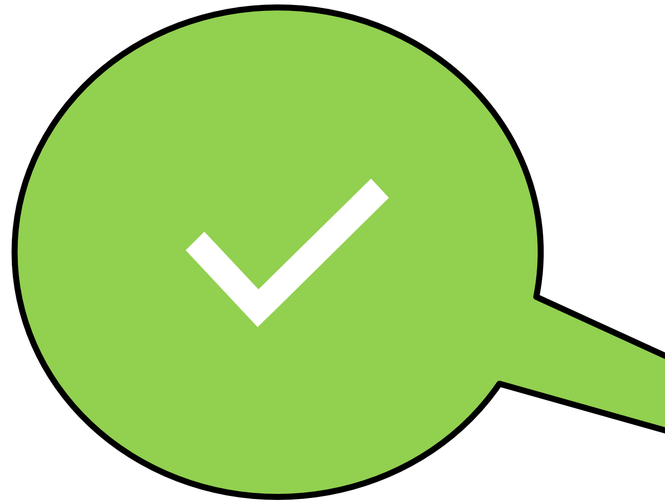
BC/NAV TECH DAYS 2023
mibuso.com

```
trigger OnAction()
var
    dcle: Record "Detailed Cust. Ledg. Entry";
    cust: Record Customer;
begin
    dcle.LockTable();
    cust.FindFirst();
    cust.CalcFields(cust.Balance);
end;


field(58; Balance; Decimal)
{
    CalcFormula = Sum("Detailed Cust. Ledg. Entry".Amount
                      WHERE("Customer No." = FIELD("No.") …
    FieldClass = FlowField;
}
```

Question 7: FlowFields

```
trigger OnAction()
var
    dcle: Record "Detailed Cust. Ledg. Entry";
    cust: Record Customer;
begin
    dcle.LockTable();
    cust.FindFirst();
    cust.CalcFields(cust.Balance);
end;
```

Locks

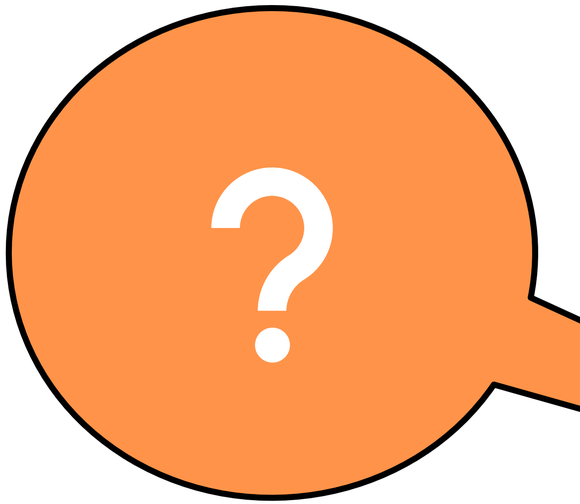


```
field(58; Balance; Decimal)
{
    CalcFormula = Sum("Detailed Cust. Ledg. Entry".Amount
                      WHERE("Customer No." = FIELD("No.") …
    FieldClass = FlowField;
}
```

Question 8: FlowFields

```
procedure local LockOnCustomerTable()
var
    cust: Record Customer;
begin
    cust.LockTable();
    cust.FindFirst();
    cust.CalcFields(cust.Balance);
end;
```

Maybe




```
field(58; Balance; Decimal)
{
    CalcFormula = Sum("Detailed Cust. Ledg. Entry".Amount
                      WHERE("Customer No." = FIELD("No.") …
    FieldClass = FlowField;
}
```

Quiz summary

BC/NAV TECH DAYS 2023
mibuso.com

7 Locking, 1 maybe

LockTable works on the table until transaction ends (not on variable)

Starting a write operation will lock all records read from that table
Count, CalcFields, CalcSums and other operations can lock records
FindSet, ModifyAll, DeleteAll can lock entire table (if empty)

Write operation – exclusive lock on records modified

Fundamentals

AL's interplay with the DB

BC/NAV TECH DAYS 2023
mibuso.com

AL table locking is done via the database.

Writes –> Lock done by SQL on rows modified

Can be escalated

Reads – We request via Hints (ReadUncommited, ReadCommited, RepeatableRead, UpdLock)

Transactions

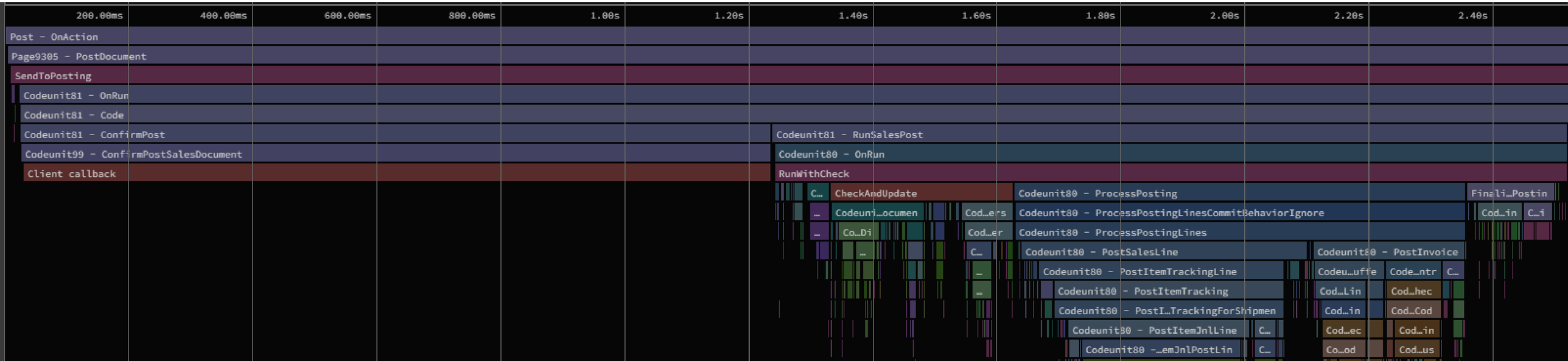
Transactionality in AL

BC/NAV TECH DAYS 2023
mibuso.com

Database operations in AL are transactional.

Update locks taken are also transactional, and the time held is bound to the transaction, meaning they are held for the same “length”.

Example: Post Sales Order



Example: Post Sales Order – Tables Locked

BC/NAV TECH DAYS 2023
mibuso.com

- Sales Line
- Item Charge Assignment (Sales)
- Purchase Line
- Purchase Header
- Sales Invoice Header
- Sales Invoice Entity Aggregate
- Sales Shipment Line
- Item Ledger Entry
- Value Entry
- Avg. Cost Adjmt. Entry Point
- Item Register
- Post Value Entry to G/L
- Item Application Entry
- Sales Invoice Line
- G/L Entry
- VAT Entry
- G/L Register
- G/L Entry - VAT Entry Link
- Detailed Cust. Ledg. Entry
- Cust. Ledger Entry
- Sales Cr. Memo Entity Buffer
- Sales Header
- Office Invoice
- Sales Order Entity Buffer
- Tracking Specification

Locks explained

SQL isolation explained

BC/NAV TECH DAYS 2023
mibuso.com

- READUNCOMMITTED (No Lock):
  - Allows for reading uncommitted data which may disappear after read (dirty reads).
  - No locks are taken, so no waiting for neither Update (U) or Exclusive (X) locks.

- UPDLOCK (U Lock):
  - No uncommitted data, since U locks are placed on read rows, guaranteeing they are valid for entire transaction.
  - Due to U locks being incompatible with U locks, readers will block each other, causing waits.
  - Are held for the remainder of the transaction.

- Exclusive (X Lock)

SQL server lock compatibility matrix

BC/NAV TECH DAYS 2023
mibuso.com

Requested Lock Type	No lock	Shared (S)	Update (U)	Exclusive (X)
No lock	N	N	N	N
Shared (S)	N	N	N	C
Update (U)	N	N	C	C
Exclusive (X)	N	C	C	C

- No Conflict: Both operations can happen concurrently.
- Conflict: The latter operation must wait till the former operation relinquish its held lock.

SQL server lock compatibility matrix

BC/NAV TECH DAYS 2023
mibuso.com

Requested Lock Type	No lock	Shared (S)	Update (U)	Exclusive (X)
No lock	N	N	N	N
Shared (S)	N	N	N	C
Update (U)	N	N	C	C
Exclusive (X)	N	C	C	C

- No Conflict: Both operations can happen concurrently.
- Conflict: The latter operation must wait till the former operation relinquish its held lock.

SQL server lock compatibility matrix

BC/NAV TECH DAYS 2023
mibuso.com

Requested Lock Type	No lock	Shared (S)	Update (U)	Exclusive (X)
No lock	N	N	N	N
Shared (S)	N	N	N	C
Update (U)	N	N	C	C
Exclusive (X)	N	C	C	C

- No Conflict: Both operations can happen concurrently.
- Conflict: The latter operation must wait till the former operation relinquish its held lock.

SQL server lock compatibility matrix

	NL	SCH-S	SCH-M	S	U	X	IS	IU	IX	SIU	SIX	UIX	BU	RS-S	RS-U	RI-N	RI-S	RI-U	RI-X	RX-S	RX-U	RX-X
NL	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
SCH-S	N	N	C	N	N	N	N	N	N	N	N	N	N	I	I	I	I	I	I	I	I	I
SCH-M	N	C	C	C	C	C	C	C	C	C	C	C	C	I	I	I	I	I	I	I	I	I
S	N	N	C	N	N	C	N	N	C	N	C	C	C	N	N	N	N	N	C	N	N	C
U	N	N	C	N	C	C	N	C	C	C	C	C	C	N	C	N	N	C	C	N	C	C
X	N	N	C	C	C	C	C	C	C	C	C	C	C	C	C	N	C	C	C	C	C	C
IS	N	N	C	N	N	C	N	N	N	N	N	N	C	I	I	I	I	I	I	I	I	I
IU	N	N	C	N	C	C	N	N	N	N	N	C	C	I	I	I	I	I	I	I	I	I
IX	N	N	C	C	C	C	N	N	N	C	C	C	C	I	I	I	I	I	I	I	I	I
SIU	N	N	C	N	C	C	N	N	C	N	C	C	C	I	I	I	I	I	I	I	I	I
SIX	N	N	C	C	C	C	N	N	C	C	C	C	C	I	I	I	I	I	I	I	I	I
UIX	N	N	C	C	C	C	N	C	C	C	C	C	C	I	I	I	I	I	I	I	I	I
BU	N	N	C	C	C	C	C	C	C	C	C	C	N	I	I	I	I	I	I	I	I	I
RS-S	N	I	I	N	N	C	I	I	I	I	I	I	I	N	N	C	C	C	C	C	C	C
RS-U	N	I	I	N	C	C	I	I	I	I	I	I	I	N	C	C	C	C	C	C	C	C
RI-N	N	I	I	N	N	N	I	I	I	I	I	I	I	C	C	N	N	N	N	C	C	C
RI-S	N	I	I	N	N	C	I	I	I	I	I	I	I	C	C	N	N	N	C	C	C	C
RI-U	N	I	I	N	C	C	I	I	I	I	I	I	I	C	C	N	N	C	C	C	C	C
RI-X	N	I	I	C	C	C	I	I	I	I	I	I	I	C	C	N	C	C	C	C	C	C
RX-S	N	I	I	N	N	C	I	I	I	I	I	I	I	C	C	C	C	C	C	C	C	C
RX-U	N	I	I	N	C	C	I	I	I	I	I	I	I	C	C	C	C	C	C	C	C	C
RX-X	N	I	I	C	C	C	I	I	I	I	I	I	I	C	C	C	C	C	C	C	C	C

Key

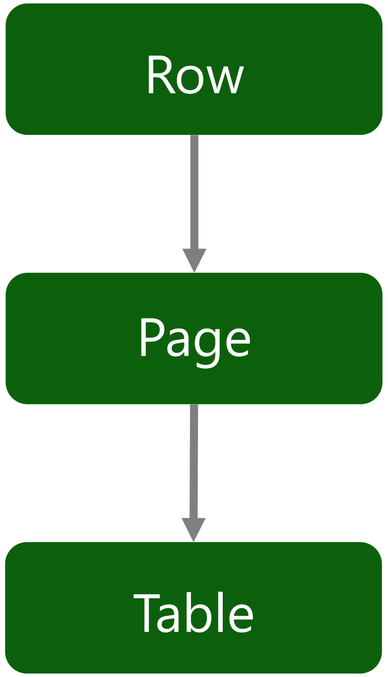
N	No Conflict	SIU	Share with Intent Update
I	Illegal	SIX	Shared with Intent Exclusive
C	Conflict	UIX	Update with Intent Exclusive
		BU	Bulk Update
NL	No Lock	RS-S	Shared Range-Shared
SCH-S	Schema Stability Locks	RS-U	Shared Range-Update
SCH-M	Schema Modification Locks	RI-N	Insert Range-Null
S	Shared	RI-S	Insert Range-Shared
U	Update	RI-U	Insert Range-Update
X	Exclusive	RI-X	Insert Range-Exclusive
IS	Intent Shared	RX-S	Exclusive Range-Shared
IU	Intent Update	RX-U	Exclusive Range-Update
IX	Intent Exclusive	RX-X	Exclusive Range-Exclusive

SQL lock escalation

Operations takes row level locks to start with.

Escalates to higher level based on SQL defined rules.
Main rule number of rows changed.

Means could lock the rows that were not read.



Runtime determined locking

BC/NAV TECH DAYS 2023
mibuso.com

1. Reads are done with READUNCOMMITTED as long as no writes (or LockTable) has been done to the table in the current transaction.

2. If writes has been done to the table (or LockTable called) in the current transaction, further reads will be done with UPDLOCK against the table.

Runtime determined locking

1. Reads are done with READUNCOMMITTED as long as no writes (or LockTable) has been done to the table in the current transaction.

2. If writes has been done to the table (or LockTable called) in the current transaction, further reads will be done with UPDLOCK against the table.

```
trigger OnAction()
var
    curr: Record Currency;
begin
    curr.FindLast();

    curr.Delete(); // Or LockTable,
    // Rename, Modify, DeleteAll, etc

    curr.FindFirst();

    ShowContinued();
end;

local procedure ShowContinued()
var
    otherCurr: Record Currency;
    cust: Record Customer;
begin
    otherCurr.FindLast();
    cust.FindFirst();
end;
```

BC/NAV TECH DAYS 2023
mibuso.com

Runtime determined locking

1. Reads are done with READUNCOMMITTED as long as no writes (or LockTable) has been done to the table in the current transaction.

2. If writes has been done to the table (or LockTable called) in the current transaction, further reads will be done with UPDLOCK against the table.

```
trigger OnAction()
var
    curr: Record Currency;
begin
    curr.FindLast(); // READUNCOMMITTED

    curr.Delete(); // Or LockTable,
    // Rename, Modify, DeleteAll, etc

    curr.FindFirst();

    ShowContinued();
end;

local procedure ShowContinued()
var
    otherCurr: Record Currency;
    cust: Record Customer;
begin
    otherCurr.FindLast();
    cust.FindFirst();
end;
```

BC/NAV TECH DAYS 2023
mibuso.com

Runtime determined locking

1. Reads are done with READUNCOMMITTED as long as no writes (or LockTable) has been done to the table in the current transaction.

2. If writes has been done to the table (or LockTable called) in the current transaction, further reads will be done with UPDLOCK against the table.

```
trigger OnAction()
var
    curr: Record Currency;
begin
    curr.FindLast(); // READUNCOMMITTED

    curr.Delete(); // Or LockTable,
    // Rename, Modify, DeleteAll, etc

    curr.FindFirst(); // UPDLOCK

    ShowContinued();
end;

local procedure ShowContinued()
var
    otherCurr: Record Currency;
    cust: Record Customer;
begin
    otherCurr.FindLast();
    cust.FindFirst();
end;
```

BC/NAV TECH DAYS 2023
mibuso.com

Runtime determined locking

1. Reads are done with READUNCOMMITTED as long as no writes (or LockTable) has been done to the table in the current transaction.

2. If writes has been done to the table (or LockTable called) in the current transaction, further reads will be done with UPDLOCK against the table.

```
trigger OnAction()
var
    curr: Record Currency;
begin
    curr.FindLast(); // READUNCOMMITTED

    curr.Delete(); // Or LockTable,
    // Rename, Modify, DeleteAll, etc

    curr.FindFirst(); // UPDLOCK

    ShowContinued();
end;

local procedure ShowContinued()
var
    otherCurr: Record Currency;
    cust: Record Customer;
begin
    otherCurr.FindLast(); // UPDLOCK
    cust.FindFirst();
end;
```

BC/NAV TECH DAYS 2023
mibuso.com

Runtime determined locking

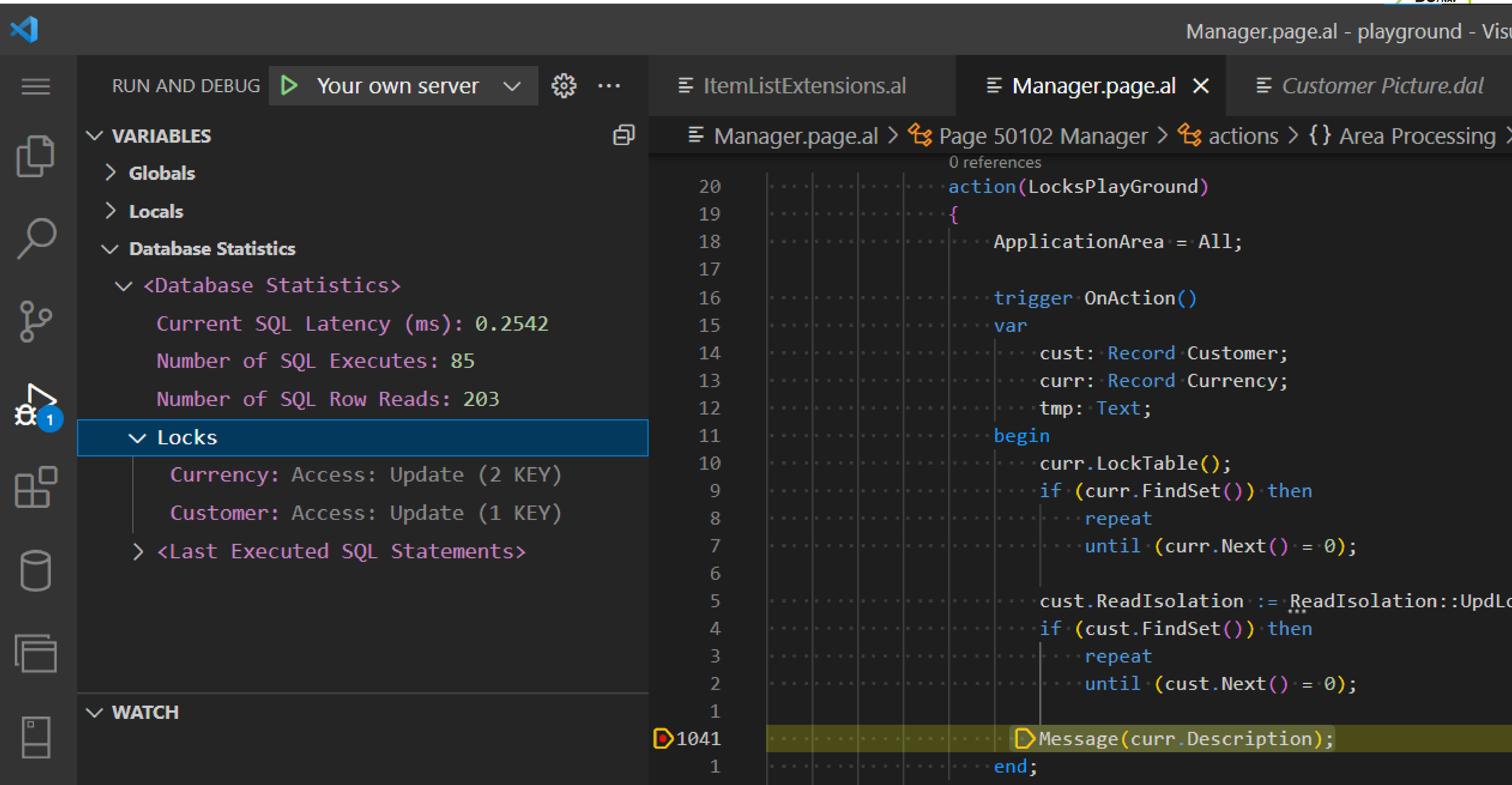
1. Reads are done with READUNCOMMITTED as long as no writes (or LockTable) has been done to the table in the current transaction.

2. If writes has been done to the table (or LockTable called) in the current transaction, further reads will be done with UPDLOCK against the table.

```
trigger OnAction()
var
    curr: Record Currency;
begin
    curr.FindLast(); // READUNCOMMITTED

    curr.Delete(); // Or LockTable,
    // Rename, Modify, DeleteAll, etc

    curr.FindFirst(); // UPDLOCK

    ShowContinued();
end;

local procedure ShowContinued()
var
    otherCurr: Record Currency;
    cust: Record Customer;
begin
    otherCurr.FindLast(); // UPDLOCK
    cust.FindFirst(); // READUNCOMMITTED
end;
```

BC/NAV TECH DAYS 2023
mibuso.com

Locks in VS Code debugger



Advanced analysis SQL query

```
BEGIN TRAN

-- Enter your query
SELECT * FROM [Navision_PlatformCore].[dbo].[CRONUS International Ltd_$Customer$437dbf0e-84ff-417a-965d-ed2bb9650972] WITH (UPDLOCK)
WHERE No_ > '0000000' and No_ < '3000000';

WITH locks AS (
   SELECT
      dm_tran_locks.request_owner_id,
      dm_tran_locks.request_session_id,
      dm_tran_locks.resource_type,
      dm_tran_locks.request_mode,
      dm_tran_locks.request_status,
      dm_tran_locks.resource_associated_entity_id
   FROM sys.dm_tran_locks WITH (NOLOCK)
   WHERE
   resource_associated_entity_id > 0
)
SELECT
   objects.name AS object_name,
   locks.request_owner_id,
   locks.request_session_id,
   locks.resource_type,
   locks.request_mode,
   locks.request_status
FROM locks
LEFT JOIN sys.partitions WITH (NOLOCK) ON partitions.hobt_id = locks.resource_associated_entity_id
JOIN sys.objects WITH (NOLOCK) ON sys.objects.object_id =
   (CASE WHEN resource_type = 'OBJECT' THEN locks.resource_associated_entity_id ELSE partitions.object_id END)


Rollback
```

Telemetry

- Slides 17 – 21
- BCTech/samples/AppInsights/Presentations/decks/telemetry-app-scenarios.pptx at master · microsoft/BCTech · GitHub

FlowFields?

- Lock decisions are based on target table, not defining table.

```
field(58; Balance; Decimal)
{
    CalcFormula = Sum("Detailed Cust. Ledg. Entry".Amount WHERE("Customer No." = FIELD("No.")
…
    FieldClass = FlowField;
}
```

Explicit control (ReadIsolation)

Controlling reads explicitly

From V22+ read behavior can be controlled on a per-record instance basis.

Lock state from table state is ignored.

Doesn’t influence other record instances.

Only read scenarios. Writes are not affected.

ReadIsolation values – Used by default

- READUNCOMMITTED:
  - Allows for reading uncommitted data which may disappear after read (dirty reads).
  - No locks are taken, so no waiting for neither Update (U) or Exclusive (X) locks.

- UPDLOCK:
  - No uncommitted data, since U locks are placed on read rows, guaranteeing they are valid for entire transaction.
  - Due to U locks being incompatible with U locks, readers will block each other, causing waits.
  - Are held for the remainder of the transaction.

ReadIsolation values

- READCOMMITTED:
  - Shared (S) locks are taken for the duration of the read, NOT duration of transaction.
  - No reading of uncommitted data, but data read might be removed since S locks are only held while reading.
  - **Recommended as default for reads**

- REPETABLEREAD:
  - Shared (S) locks are taken for the duration of the transaction.
  - No reading of uncommitted data and data is guaranteed to stay consistent for the duration of transaction.
  - RangeLock - With set range nobody will be able to insert the record in between
  - **Recommended if you want to ensure parallel reads and that nobody changes the data**

Question 2: Improved – Control example

BC/NAV TECH DAYS 2023
mibuso.com

```
internal procedure FindsetOnAnEmptyTable()
var
    Customer1: Record Customer;
    Customer2: Record Customer;
begin
    Customer1.LockTable();
    Customer1.FindLast();
    Customer2.Count();
end;
```

Question 2: Improved – Control example

BC/NAV TECH DAYS 2023
mibuso.com

```
internal procedure FindsetOnAnEmptyTable()
var
    Customer1: Record Customer;
    Customer2: Record Customer;
begin
    Customer1.ReadIsolation := IsolationLevel::UpdLock;
    Customer1.FindLast();

    Customer2.ReadIsolation := IsolationLevel::ReadCommited;
    Customer2.Count();
end;
```

Pattern - Temporary heightening

BC/NAV TECH DAYS 2023
mibuso.com

```
local procedure GetNextEntryNo(): Integer
var
    GLEntry: Record "G/L Entry";
begin
    GLEntry.ReadIsolation := IsolationLevel::UpdLock;
    GLEntry.FindLast();
    exit(GLEntry."Entry No." + 1)
end;
```

More controlled locking because it is defined on a variable.

ReadIsolation is better than Locktable for most (all) scenarios.

Pattern - Temporary lowering

```
[EventSubscriber(ObjectType::Table, Database::"Customer", 'OnBeforeInsertEvent', '', false, false)]
local procedure EventSubscriberCount(var Rec: Record Customer; RunTrigger: Boolean)
var
    Customer: Record Customer;
begin
    // Some code
    Customer.ReadIsolation := IsolationLevel::ReadCommitted;
    Customer.FindSet();
    // Some code
end;
```

Impossible to know the context called in, so be explicit.

Strongly recommended for event subscribers

ReadIsolation with FlowFieds

BC/NAV TECH DAYS 2023
mibuso.com

- Uses the ReadIsolation on the record over target table state.

```
trigger OnAction()
var
    cust: Record Customer;
    dcle: Record "Detailed Cust. Ledg. Entry";
begin
    cust.CalcFields(cust.Balance); // READUNCOMMITTED

    dcle.LockTable();
    cust.ReadIsolation := ReadIsolation::ReadCommitted;
    cust.CalcFields(cust.Balance); // READCOMMITTED
end;
```

Question 5: Avoid locking the entire table

BC/NAV TECH DAYS 2023
mibuso.com

```
internal procedure FindsetOnAnEmptyTable()
var
    MyTable: Record MyTable;
begin
    MyTable.DeleteAll();
    // some other code…
    if not MyTable.FindSet() then
        exit;
end;
```


Question 5: Avoid locking the entire table

BC/NAV TECH DAYS 2023
mibuso.com

```
internal procedure FindsetOnAnEmptyTable()
var
    MyTable: Record MyTable;
begin
    MyTable.ReadIsolation := ReadIsolation::ReadUncommited;

    if not MyTable.IsEmpty() then
        MyTable.DeleteAll();

    // some other code…
    if not MyTable.FindSet() then
        exit;
end;
```

Pattern – Transaction read lock

BC/NAV TECH DAYS 2023
mibuso.com

```
local procedure LockReadRecordsForTransaction(): Integer
var
    Customer: Record Customer;
begin
    Customer.ReadIsolation := IsolationLevel::RepeatableRead;
    Customer.FindSet();
end;
```

Records read will be locked for updates until the end of transaction

Think smart

Existing code – Refactor as you need, incrementally, eliminate bad locks

New code – Use ReadIsolation

- If possible, read before writing to the table
- Cache read results, don’t call too often
- Smaller short transactions
- Longer transactions - make code re-entrant + commit (avoid partial commits)

BC/NAV TECH DAYS 2023
mibuso.com

Q&A

Any Questions?

BC/NAV TECH DAYS 2023
mibuso.com

Thank You!