Progress Legos - Scoping, Blocks, and other Critical Concepts

Michael Lonski
President - Allegro
AllegroConsultants.com

PUG Challenge Americas 2018
Who I Am...

- Started working with Progress® v3 (1986)
- Founded Allegro in 1993
- Internationally recognized speaker
- Author of “Coding Smart” book on ADM2
- Punster and Frequent Talker
- Ask me about Capoferro or Giganti
...And Why I Am Here

- A block by any other name
- What scope through yonder window
- Who started it, who controlled it
- Proving it with examples
- Are you sure?
- Nested buckets
What’s Next...

- A block by any other name
Blocks

- Definition: a sequence of one or more statements, including any nested blocks, that share a single context
- Typically closed with the “END.” statement
- Differing default properties depending on type of block
- Most blocks can be nested
Control Blocks

- FOR [ EACH | FIRST | LAST ]
- REPEAT
- DO
- EDITING
Procedure Blocks

- External Procedure
- TRIGGER
- PROCEDURE (internal)
- FUNCTION (user defined)
Block Properties

- Some examples are:
  - Transaction
  - Record scoping
  - Iteration
  - Undo

- Some blocks control some properties implicitly

- Control blocks can be modified to explicitly control properties
## Block Properties (Details)

<table>
<thead>
<tr>
<th>Property</th>
<th>REPEAT</th>
<th>FOR</th>
<th>DO</th>
<th>Proc/Trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Imp</td>
<td>Exp</td>
<td>Imp</td>
<td>Exp</td>
</tr>
<tr>
<td>Record Reading</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Record Phrase</td>
</tr>
<tr>
<td>Frame Scoping</td>
<td>Yes</td>
<td>WITH FRAME</td>
<td>Yes</td>
<td>WITH FRAME</td>
</tr>
<tr>
<td>Record Scoping</td>
<td>Yes</td>
<td>FOR</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>UNDO</td>
<td>Yes</td>
<td>TRANSACTION</td>
<td>Yes</td>
<td>TRANSACTION</td>
</tr>
<tr>
<td>Looping</td>
<td>Yes</td>
<td>WHILE TO/BY</td>
<td>Yes</td>
<td>WHILE TO/BY</td>
</tr>
<tr>
<td>Transaction</td>
<td>Yes</td>
<td>TRANSACTION</td>
<td>Yes</td>
<td>TRANSACTION</td>
</tr>
</tbody>
</table>
Some Examples

- **REPEAT:**
  - Implicitly scopes transactions, records and frames
  - Scoping is weak

- **REPEAT FOR customer WITH FRAME x:**
  - Explicitly scopes record and frame
  - Record scoping is strong
  - Transaction scoping still implicit (weak)
Working With Blocks

- Gives structure to execution flow
  - LEAVE
  - NEXT
  - RETURN
- Label blocks to remove doubt
  - Same rules as variable name
  - Immediately precedes block statement
  - Self-documentation
Impacting Transactions

- Explicitly manipulate transactions
  - UNDO, <action>
    - LEAVE
    - NEXT
    - RETRY
    - RETURN
    - THROW

- Use block labels to remove doubt
What’s Next...

- A block by any other name
- What scope through yonder window
Scope

- Definition: the duration that a resource is available to an application
- Prime component when discussing buffers and transactions
- Variables/Objects scoped where defined
Scope Types

- Record
- Transaction
- Frame
- Object
Seeing The Scope

- TRANSACTION function
  MESSAGE “Trans Active?:” TRANSACTION.

- COMPILER ... LISTING
  COMPILER demo1a.p LISTING demo1a.lst.
<table>
<thead>
<tr>
<th>File Name</th>
<th>Line Blk.</th>
<th>Type</th>
<th>Tran Blk.</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>demo1a.p</td>
<td>0</td>
<td>Procedure No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Buffers: sports.Customer</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Frames: Unnamed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>demo1a.p</td>
<td>19</td>
<td>Repeat</td>
<td>Yes order-rpt</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Buffers: sports.Order</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Frames: a-frm</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What’s Next...

- A block by any other name
- What scope through yonder window
- Who started it, who controlled it
Cause vs. Control

- Specific statements cause record or transaction scope
- Blocks control (or not) the scope of items started within them
- Scope can be explicitly increased and sometimes decreased programmatically
- Can both cause and control
Record Scope

- Smallest enclosing block that encompasses all references to the record
- Determines when...
  - The record buffer is active/available
  - Progress clears the record from the buffer
  - Writes the record to the database
  - How long a record lock is in effect
  - When to validate the record
Types of Record References

- **Strong-scoped reference** - buffers explicitly referenced in REPEAT FOR or DO FOR
- **Weak-scoped reference** - buffers implicitly referenced in FOR EACH or PRESELECT EACH block
- **Free reference** - All other references to records are free references
Record Reference Causes

- FIND
- FOR EACH
- REPEAT FOR
- INSERT/CREATE
- DEFINE QUERY
Transaction Scope

- Set of changes to the database that is either executed completely or leaves no modification to the database
- Transaction scope does *not* equal Record scope
Transaction Causes

- EXCLUSIVE-LOCK
- CREATE/INSERT
- DELETE
- ASSIGN
- UPDATE
- TRANSACTION keyword
Transaction GT Record

- Transaction started
- Reference to record is invalid

- If before record retrieval, record is NOT AVAILABLE
- Record has EXCLUSIVE-LOCK
- Record available until end of record scope

- Record returns to SHARE-LOCK
- Reference to record is invalid but lock *IS* held
- Transaction prevents full release of record
Record GT Transaction

- Record is NOT AVAILABLE until reference.
- Record is retrieved with SHARE-LOCK and now AVAILABLE

  - Transaction started
  - SHARE-LOCK upgraded to EXCLUSIVE-LOCK
  - Record updates occur
  - Nothing written out to database until end of transaction

- Record returns to SHARE-LOCK
- Transaction effectively bleeds out to record scope
Record EQ Transaction
What’s Next...

- A block by any other name
- What scope through yonder window
- Who started it, who controlled it
- Proving it with examples
What’s Next...

- A block by any other name
- What scope through yonder window
- Who started it, who controlled it
- Proving it with examples
- Are you sure?
The Prime Directives

- All scopes are defined in terms of their containing blocks
- Transaction scope and record scope are not automatically identical but do affect each other.
- Beware of bleeding locks
RELEASE Statement

- Probably one of the most misunderstood and misused Progress 4GL statements
- Too often used as a band-aid to “fix” scoping issues
Actual Effects

- Validates mandatory fields
- Validates unique index constraints
- Clears the record from the buffer
- If changed, writes back to the database
Misunderstandings

- Not a tool for releasing record locks
- Does *not* magically fix transaction or scoping issues
- Will not release a SHARE-LOCK if still inside of a transaction
What’s Next...

- A block by any other name
- What scope through yonder window
- Who started it, who controlled it
- Proving it with examples
- Are you sure?
- Nested buckets
Now that I’ve rambled on, are there any questions?
Final Notes

- Examples bundled with the presentation.
  - AllegroConsultants.com/about/downloads

- Thanks for attending!

- Please fill out your evaluations!